Relaxed syntax modeling in Transformers for future-proof license plate recognition

Florent Meyer, Laurent Guichard, Denis Coquenet, Guillaume Gravier, Yann Soullard, Bertrand Coüasnon

Relaxed syntax modeling in Transformers for future-proof license plate recognition

Florent Meyer[1,2] ✉ [0009−0001−7527−5215], Laurent Guichard[1] [0009−0008−7853−8704], Denis Coquenet[2] [0000−0001−5203−9423], Guillaume Gravier[2] [0000−0002−2266−5682], Yann Soullard[3] [0009−0001−8048−2489], and Bertrand Coüasnon[2] [0000−0002−7077−0751]

[1] ANTAI, Rennes, France
[2] Univ Rennes, CNRS, IRISA - UMR 6074, Rennes, France
[3] Univ Rennes, Université Rennes 2, CNRS, IRISA - UMR 6074, Rennes, France
florent.meyer@irisa.fr

Abstract. Effective license plate recognition systems are required to be resilient to constant change, as new license plates are released into traffic daily. While Transformer-based networks excel in their recognition at first sight, we observe significant performance drop over time which proves them unsuitable for tense production environments. Indeed, such systems obtain state-of-the-art results on plates whose syntax is seen during training. Yet, we show they perform similarly to random guessing on future plates where legible characters are wrongly recognized due to a shift in their syntax. After highlighting the flows of positional and contextual information in Transformer encoder-decoders, we identify several causes for their over-reliance on past syntax. Following, we devise architectural cut-offs and replacements which we integrate into SaLT, an attempt at a Syntax-Less Transformer for syntax-agnostic modeling of license plate representations. Experiments on both real and synthetic datasets show that our approach reaches top accuracy on past syntax and most importantly nearly maintains performance on future license plates. We further demonstrate the robustness of our architecture enhancements by way of various ablations.

Keywords: Architecture debiasing · Syntax bias · Transformer · License plate · Text recognition · Future-proofing.

1 Introduction

Vehicle License Plate Recognition (LPR) has been in use for several years in many countries with applications ranging from traffic surveillance to speed limit monitoring. In the context of traffic law enforcement notably, the tremendous volume of contraventions recorded daily needs efficient automatic recognition so as to minimize the amount of human workload. Consequently, LPR has benefitted from gradual advances in deep learning research for vision and state-of-the-art results are now obtained with Transformers [10,19,31].

Table 1. Exact match on real LP of TrOCR with diverse fine-tuning, pretraining and tokenization options. TrOCR consistently fails at decoding images of future LP starting with a `G` (target syntax), whether the whole model is fine-tuned or not. Fine-tuning only enables reaching decent whole-LP accuracy on plates starting with letters `A` to `F` (source syntax) when a character-level tokenizer replaces byte-pair-encoding (BPE) [23]. *stage1* denotes the first-stage pretrained checkpoint from [13].

Tokenizer	Pretraining		Fine-tuning	Target syntax	Source syntax
	Part	Checkpoint		`^G.*$`	`^[A-F].*$`
BPE	Encoder-decoder	stage1	✗	1.7	1.3
		printed		3.8	3.0
BPE	Encoder-decoder	stage1	✓	0.9 ± 1.7	99.4 ± 0.0
Character	Encoder			27.8 ± 32.8	99.5 ± 0.0

Most LPR systems are evaluated on test datasets whose distribution is practically identical to that of the training samples, thereby reaching satisfying performance and seeming ready to be run in production. Despite remarkable improvements in recognition accuracy, there remain challenges to be addressed. Indeed, unforeseen difficulties may arise from a misrepresentation of the syntax followed by license plates (LP) on the roads, for it is different from that learned by a deep network [11,22]. For instance, some countries use serials with characters increased by an increment from the right with each LP put in service. This specific scheme regularly leads to a tipping point, namely a *shift*, when the character updated the least often (e.g. in the leftmost position) is eventually increased as well. Put another way, a known character previously missing in a given position now appears in it. From then on, the content of the training dataset learned by the model will be less and less representative of the real-life LP in circulation as an increasing number of new-syntax LP make their way into traffic. Throughout this paper, we report the non-negligible impact this change in syntax has on cropped LP transcription accuracy of Transformer-based architectures. As showcased in Table 1, TrOCR [13] largely underperforms on target syntax beginning with letter `G` if LP starting with letters from `A` to `F` are seen during training. Note that off-the-shelf checkpoints are not specifically targeted at LPR. Fig. 1 depicts typical cases of success and failure in TrOCR's predictions. While we focus on LPR in this paper, similar syntax shift can be observed in regular expression-based use cases like ID, date, invoice or serial number recognition.

Starting from a Transformer encoder-decoder widely used for text recognition, we try to understand the root of its memorization of source syntax, i.e. that seen at training time. By studying the flows of contextual and positional information within the network, we identify several causes of inconvenient memorization both in the encoder and decoder parts. Based on these observations, we propose the Syntax-Less Transformer (SaLT), a light Transformer-based encoder-decoder equipped with elements devised to preserve performance over time without re-

training. Indeed, retraining is not satisfactory because the in-production behavior would be unpredictable and often erroneous on plates with new syntax until enough such images are collected, annotated and used for retraining. SaLT is derived from a *debiasing* framework with a convolutional encoder and enhancements regarding the input and cross-attention of the Transformer-decoder. Extensive evaluation reveals SaLT's robustness to a constantly evolving syntax, bringing a large gain on target syntax while strongly reducing performance variability. Meanwhile, performance on source syntax serials is fully retained. Additional experiments are conducted on LPR-MNIST, a new synthetic dataset which replicates LP syntax evolution through time, publicly released for future experiments on syntax evolution[1]. Finally, ablation studies reveal the combined effect of our modifications on robustness to syntax shift.

Fig. 1. Cropped photographs of real test LP with predictions by TrOCR. Left: Fine-tuned TrOCR successfully decodes images of LP starting with letters from `A` to `F`, (training-time syntax), despite degraded quality. **Right:** Yet, it fails on legible LP with a `G` in the leftmost position (future syntax). Errors (underlined) occur mainly in the first and second positions. LP are anonymised for RGPD compliance.

2 Related work

2.1 License plate recognition

With applications from traffic monitoring to parking fee payment, LPR systems have benefited from advances in deep learning research over the years. In this work, we consider images cropped by a localization module and resized around the LP beforehand, and thus focus on text recognition. Early multi-step architectures were generally composed of separate processing methods, typically segmenting plate characters before recognition of each individually [1]. End-to-end deep networks appeared later on, notably with object-detection methods derived from YOLO [18] detecting each character class separately. In our case however, neither character-level bounding box nor segmentation annotations are available due to their high labelling cost, we thus dismiss related methods. Indeed, we only have plate-level transcriptions. Moreover, the text on some LP is shared between two lines, e.g. on motorbikes. Thus, one-dimensional sequence alignement methods like connectionist temporal classification [6] are unsuited. Long short-term memory (LSTM)-based sequence modeling [34] was then applied to LPR. More

[1] Available at https://www-shadoc.irisa.fr/lpr-mnist-dataset/.

recently, Transformers have gained interest in research although not widely applied to LPR yet [5,9], outperforming recurrent networks like LSTM. For instance in [19,31], existing architectures are modified by inputting the image compression level to improve recognition under strong compression or by decomposing text recognition into two inter-connected tasks, namely image-to-character and character-to-word mapping. Among publicly available off-the-shelf models, the pure Transformer TrOCR [13] shows promising results for LPR [21,30].

2.2 Remediation to syntax shift

Diverse approaches attempt to increase robustness to a shifted target syntax, notably data augmentation, training strategy and additional model components.

Data augmentation One family of methods aims at obtaining a balanced training dataset, altering neither the model nor the learning process. The authors of [22] control the frequency of each character on LP by permuting them, swapping overrepresented letters with underrepresented ones to increase their ratio. Similarly in [32], infrequent characters are augmented by image synthesis with diverse backgrounds, font styles, and text shapes. Alternatively, in order to mitigate the imbalance between plates of diverse countries, Laroca et al. [11] opt for gathering and manually labeling more LP photographs from the Internet, which is a costly solution. To adapt to new languages and overcome an everchanging data distribution in multilingual scene text recognition (STR), an ensemble method of a language identifier followed by one module per language was devised in [37]. This requires retraining for each newly supported language. Conversely, we choose not to augment our dataset with artificial LP of all imaginable future character combinations. Rather than depend upon the data itself, we explore a more architecture-centered approach to debiasing without retraining.

Training strategy & additional components Another family of approaches deal with syntax shift in STR through sophisticated training procedures or additional network components. On the one hand, some methods suppose the availability of target syntax samples at training time. The authors of [20] explore an ensemble method to train on a long-tailed character distribution while preserving performance on a balanced dataset. Two separate experts learn on the unbalanced source dataset and a smaller target dataset with a balanced number of characters, respectively. So as to reduce bias towards high-frequency characters, Tran Tien et al. [26] use unlabeled target data to minimize the latent entropy of the predicted probability distribution over characters. By investigating the effect of unidirectional language priors, [8] propose an adaptive fusion module and add Kullback-Leibler divergence losses between left-to-right and right-to-left decoding passes. These methods are inapplicable to our use case on account of the temporal evolution of LP syntax, as no target syntax samples are available at training time. On the other hand, some contrastive or mutual learning approaches work without any real target syntax samples. In order to alleviate

over-fitting due to lexical dependencies in multiple STR tasks, it was proposed to randomly permute scene text image patches in a contrastive learning fashion, with each patch representing several characters [33,36]. Another way of reducing bias towards source vocabulary is a mutual learning strategy between two complementary models with additional Kullback-Leibler divergence loss between distributions predicted by final layers [29]. As opposed to these methods which increase model size with additional modules or complicate training, Liu et al. [16] reduce word order information for cross-lingual sequence labeling by replacing positional encoding with convolutions after Transformer attention. Liu et al. [14] disentangle positional information to improve zero-shot translation by removing residual connections in a Transformer-encoder. In this way, we propose architectural enhancements inspired by information flow analysis that leave the training strategy unchanged and do not involve any extra model parameters.

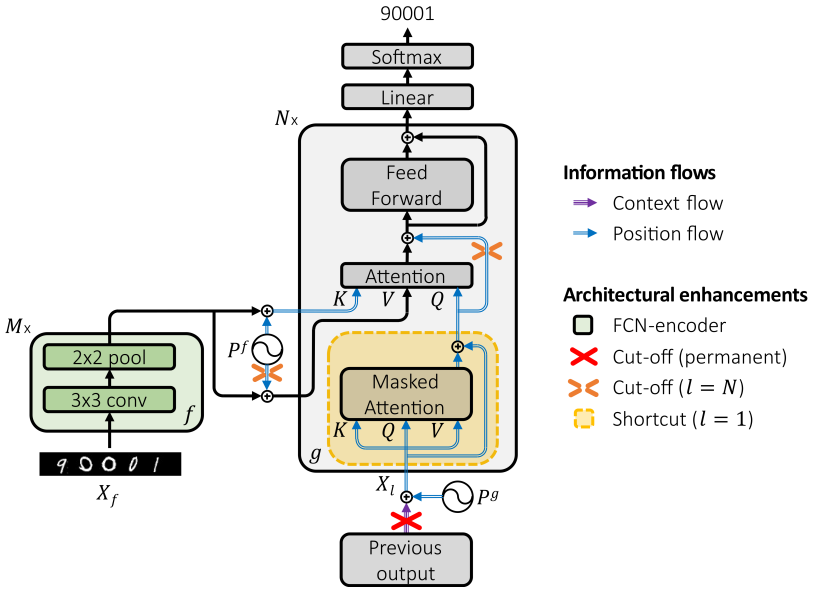

Fig. 2. Overall debiasing framework applied to a Transformer encoder-decoder network. The proposed architectural enhancements discard positional and contextual bias through an accurate control of information flows.

3 Debiasing framework

We start by formulating the syntax shift problem and the two resulting roots of bias in Transformers. Second, we formalize the advantage of preferring a fully convolutional encoder over a Transformer one for successful target syntax recognition. Third, we suggest syntax-debiasing enhancements to a vanilla

Transformer-decoder. We finally assemble these two parts into SaLT, a debiased encoder-decoder architecture. The overall framework is illustrated in Fig. 2.

3.1 Context & problem statement

Transformer encoder-decoders have demonstrated their ability to exceed the performance of previous architectures in LPR [19,30,31,35]. Notably, [21] achieves promising results using TrOCR [13]. We choose it as a baseline since it is a vanilla Transformer with minimal adaptations for STR. It can therefore be seen as a representative of Transformers and their bias. Albeit displaying excellent fine-tuned performance on a source syntax dataset, TrOCR fails to reliably recognize target syntax as shown in Table 1. The network obviously cannot read known characters in new positions on an LP. On the contrary, we argue it learns a biased strategy preventing it from achieving decent performance on future data.

Let us now consider the differences between source and target data which may lead to poor reading of target syntax. Let Ω be the set of all character classes. For each position, we define C_i^S and C_i^T as the sets of characters that can be found in position i for source and target syntax, respectively. One can define the source syntax $\mathcal{S}$ and the target syntax $\mathcal{T}$ as:

$$\begin{aligned} \mathcal{S} &= \{c_1, ..., c_L \mid c_i \in \mathcal{C}_i^{\mathrm{S}}, \mathcal{C}_i^{\mathrm{S}} \subseteq \Omega\} \\ \mathcal{T} &= \{c_1, ..., c_L \mid c_i \in \mathcal{C}_i^{\mathrm{T}}, \mathcal{C}_i^{\mathrm{T}} \subseteq \Omega\} \end{aligned} \tag{1}$$

where L is the maximum number of characters on a LP. If training and evaluation are performed on plates from the same time period, $T = S$. In our case however, $\exists\, i$, $\mathcal{C}_i^{\mathrm{T}} = \mathcal{C}_i^{\mathrm{S}} \cup \mathcal{N}_i$ where $\mathcal{N}_i$ is a non-empty set of new characters appeared after training, hence $T \neq S$. To correspond to LP evolution, we mainly focus on the case where the possible positions of characters in the sequence are the same for all but the leftmost one, i.e. $\mathcal{C}_1^{\mathrm{T}} = \mathcal{C}_1^{\mathrm{S}} \cup \mathcal{N}_1$. Visually, some letters or digits never appear in the leftmost position on the image, closest to the country indicator. Textually, this translates into a lack of examples of said characters as first in the sequence. Examples of this shift are illustrated in Fig. 1.

We assume the drop in performance of Transformer encoder-decoder architectures results from them memorizing aforementioned particularities in training data. Henceforth, we hypothesize such networks have several architectural mechanisms causing them to learn undesirable data bias which we respectively name *contextual* and *positional* bias. Yet, we suspect our real LP dataset to contain relevant information for creating robust latent representations of individual characters which could be leveraged for recognition in positions in which they never occurred at training time. Therefore, we now detail how the encoder and decoder can each learn both types of bias in their own way, and how to alleviate it.

3.2 Vision encoder

Our first proposal is to replace the vision Transformer-encoder with a fully convolutional network (FCN) denoted by f. Inspired by recent text recognition

encoder-decoder architectures [3,24], we use a convolutional, attention-less encoder in order to control its receptive field and discard positional information.

Visual positional bias Positional bias is intrinsically linked to the Transformer architecture, more particularly to the multi-head self-attention operation included in each encoder layer. Indeed, due to the permutation-invariance property of attention, it is necessary to inject explicit positional information to its input using positional encoding [28]. In this way, we argue Transformer-encoders bind absolute position information to character class. Let us recall that attention computes a weighted sum of values $\boldsymbol{V}$ according to attention weights dot-product from queries $\boldsymbol{Q}$ and keys $\boldsymbol{K}$. Specifically, if absolute position information is tied to encoded features sent as $\boldsymbol{V}$ for cross-attention, then it will impact the decoder's prediction. Thus, if some character is absent from a given zone in all training images then it can hardly be predicted in this zone. For instance, the model can learn that no `G` should be predicted near the left edge of a LP.

An FCN instead processes images and outputs features without position information.

Visual contextual bias On the one hand, except for attempts at benefiting from so-called *local inductive bias* [4,15], each vanilla Transformer layer with attention can *see the whole picture*. That is, its field of vision is not limited to a local region of the image, which allows it to learn correlations between elements located anywhere. When combined with above-mentioned mandatory position information, we suspect this leads to learning strong visual biases, among which inter-character relations, proximity with a country indicator, plate border, space or dash, etc. On the other hand, it is widely held that it is difficult for convolutional networks to capture long-range relationships, especially when using small filter kernels as in popular networks like ResNet [7]. Indeed, the receptive field of an FCN slowly increases along layers without necessarily covering the whole image at the end of the network.

As a countermeasure to visual context modeling, we thus plainly employ convolution blocks aiming for a final receptive field r of approximately the size of a character at the output of the encoder f. The main goal here is to disentangle the latent representations of successive characters so that features of a given character do not depend on the adjacent ones. It should not be large enough to cover large regions of the image and thereby absorb contextual bias, neither should it be too small, as feature modeling could otherwise be negatively impacted.

3.3 Text decoder

Sources of both contextual and positional bias are expected to be found in a Transformer-decoder too, denoted by g. However, we cannot totally get rid of this component as it is the best suited for multi-line text recognition. Hence, we bring changes to the cross-attention block and to the input of the decoder in order to eliminate syntax modeling and restrain the flow of positional information.

Textual contextual bias In autoregressive Transformer-decoders [28], left-to-right (LTR) language modeling emerges from the teacher forcing method. It consists in training the model to output the next text occurrence at each time step for a given sample using its left-side context while right-side tokens are masked. Let us recall that such samples begin with a special start-of-sentence token `<s>` and that absolute positional encoding is added to all token embeddings for the model to know the ordering of the input tokens. This suggests that a character, say `G`, appearing in a new position, say the leftmost one, would disrupt decoding for two reasons. First, $\mathbb{P}(\texttt{G}|\texttt{<s>})$ is poorly estimated because it goes against the language rules memorized during training since `<s>` is an unseen left context. Second, the same goes for $\mathbb{P}(x|\texttt{<s>G})$, where $x \in \mathcal{C}_2^{\mathrm{T}} = \mathcal{C}_2^{\mathrm{S}}$, assuming `<s>G` was correctly output. These two failure cases are illustrated in Fig. 1.

To prevent language modeling and move closer to syntactical independence, we entirely remove text token embeddings from the decoder input. More precisely, neither the embeddings of `<s>` nor those of previously predicted tokens are reinjected as decoder inputs as is usually done. Instead, we use a sequence of *position queries* $\mathbf{P}^{\mathrm{g}} = (\mathbf{p}_1, ..., \mathbf{p}_L)$ containing positional encoding only, where $\mathbf{p}_i \in R^d$ refers to a positional encoding vector from the original Transformer [28]. Decoding can thus be parallelized for greater efficiency [2], as the decoding process is no longer autoregressive. Also, since position queries are constant vectors, meaning that no relevant information can be extracted from them solely, we short-cut (i.e. remove) self-attention in the first decoder layer. In downstream layers, we keep the usual LTR mask in self-attention as an incentive to sequentially move its gaze on the image. The content of $\boldsymbol{Q}$ entering cross-attention can be summarized as follows:

$$\boldsymbol{Q}_l = \begin{cases} \mathbf{P}^{\mathrm{g}}, & \text{if } l = 1 \\ \mathrm{MultiHead}(\boldsymbol{X}_l) + \boldsymbol{X}_l, & \text{otherwise} \end{cases} \tag{2}$$

with $\boldsymbol{X}_l$ the input to the current decoder layer l, MultiHead the masked multi-head self-attention defined in [28].

Positional bias So far, we focused on removing inter-character-dependency modeling capabilities. We now interpret the way in which position information *flows* through a standard Transformer-decoder, i.e. how tensors progress along a forward pass in the network. As pictured in Fig. 2, position information enters any decoder layer from two spots: via image features (keys $\boldsymbol{K}$ and values $\boldsymbol{V}$ of the cross-attention) and position queries $\mathbf{P}^{\mathrm{g}}$ (decoder input). For the former, the intuition behind stopping position flow is to make cross-attention output a weighted sum of encoded features $\boldsymbol{V}$ which do not contain any information about the location inside the text sequence. For the latter, we focus on the residual connection to get rid of the position information before prediction takes place. We take action upon said spots in the last decoder layer only, as it is closest to the final decision making.

Textual bias One entry spot is the layer's upstream input. Indeed, we keep previously defined position queries $\mathbf{P}^{\mathrm{g}}$ because we want cross-attention to sequentially

focus on the successive characters in the image with respect to the current position of interest. $\mathbf{P}^{\mathrm{g}}$ then flows freely from previous layers by taking the residual path over successive cross-attention blocks.

However, we cut off said residual connection in the last layer. We implement this modification in the last layer only to prevent any vanishing gradient problem. Hence, the flow of sequence position information coming from the decoder input is stopped before the prediction layer.

Cross-modal bias The other entry of position information into a decoder layer is through the keys $\boldsymbol{K}$ and values $\boldsymbol{V}$ used in cross-attention. As detailed in Section 3.2, the FCN-encoder is guaranteed to provide features $f(\boldsymbol{X}_{\mathrm{f}})$ which do not contain any position information. Positional encoding must nonetheless be added to image features $f(\boldsymbol{X}_{\mathrm{f}})$ in order to compute $\boldsymbol{K}$. Indeed, cross-modal positioning between image and text is possible only if both modalities, namely $\boldsymbol{K}$ and $\boldsymbol{Q}$, contain positional information. Let us recall that $\boldsymbol{Q}$ already contains $\mathbf{P}^{\mathrm{g}}$ thanks to the residual path of previous layers.

Yet, as opposed to [3,24], we choose to leave $\boldsymbol{V}$ position-less in the last layer:

$$\boldsymbol{K} = f(\boldsymbol{X}_{\mathrm{f}}) + \mathbf{P}^{\mathrm{f}} \tag{3}$$

$$\boldsymbol{V}_l = \begin{cases} f(\boldsymbol{X}_{\mathrm{f}}), & \text{if } l = N \\ f(\boldsymbol{X}_{\mathrm{f}}) + \mathbf{P}^{\mathrm{f}}, & \text{otherwise} \end{cases} \tag{4}$$

where N is the number of decoder layers and $\mathbf{P}^{\mathrm{f}}$ represents image positional encoding. As such, the cross-attention in the last decoder layer outputs a weighted sum of position-less features $\boldsymbol{V}$.

3.4 SaLT

We now propose the Syntax-Less Transformer (SaLT), a custom encoder-decoder architecture depicted in Fig. 2 which integrates the key components we have discussed above. Intuitively, its FCN-encoder first provides disentangled character representations, independent of both their visual context and absolute position in the image. Then, regardless of other predicted characters, SaLT's decoder queries each pick among encoded features by searching for those seemingly containing a character and matching the position currently considered. Finally, the decoder prediction is computed from encoded features which do not contain any information about the location inside the image nor the text sequence.

Fully convolutional encoder The encoder takes as input a document image $\boldsymbol{X}_{\mathrm{f}} \in R^{H \times W \times C}$, with H, W and C being respectively the height, width, and number of channels. Images of any size can be processed thanks to the FCN-Transformer mix. SaLT's encoder consists in a stack of $M = 4$ convolution blocks. Each block comprises 4 components, namely (i) a 3×3 convolution with a stride of 2 which pads in such a way that the output has the same shape as the input,

(ii) instance normalization [27], (iii) 2×2 max pooling, and (iv) dropout [25] with a probability of 0.1. This results in a receptive field $r = 46$. In addition, time and memory complexity is greatly reduced compared to a multi-layer Transformer-encoder. The first convolution receives C_{in} channels depending on the color model of the image (grayscale or RGB) and outputs 32 channels. Then, the number of channels doubles at each convolution to reach a final $C_{out} = d = 256$ as expected by the decoder.

Debiased Transformer-decoder The decoding process of SaLT is handled by a debiased, N-layer Transformer decoder with $N = 2$. Its internal dimension is $d = 256$ and it uses 8 attention heads. Its position-wise 2-layer feed-forward network uses the same internal dimension d. SaLT and *debiased* model versions employ two of the three syntax-relaxing decoder modifications, as motivated by the ablation study in Section 5.4, namely position queries and residual cut-off. Decoding is done in a parallel fashion.

4 Datasets

In this section, we describe the two datasets at hand, namely a new synthetic LPR-MNIST and the real-life LP dataset. For both datasets, the test split containing both source and target syntax, while samples of target syntax are absent from the training and validation splits.

Fig. 3. Example LPR-MNIST target syntax samples with random padding.

LPR-MNIST Genuine LP photographs come in a variety of image qualities and weather conditions which make models need larger datasets to learn to ignore irrelevant variations and cloud our main topic of interest, namely syntax modeling. Moreover, the real LP dataset is private as it contains non-disclosable personal data. We thereby create a proxy dataset offering several key advantages over real photographs, (i) summarizing our key problem of syntax modeling, (ii) discarding recognition issues arising from image degradation, and (iii) accelerating experiments from using lower-resolution images and in lower numbers.

Let us introduce LPR-MNIST, a collection of 100,000 synthetic image-text pairs. It is easily generated by concatenating $L = 5$ black-and-white digits from MNIST [12], each padded to 32×32, giving $H = 32, W_{\mathcal{I}} = 160$. Here, $\Omega = \mathcal{D}$, the set of the 10 digits. Labels are drawn from `00000` to `99999` such that each possible combination is represented once. For each digit in a given label, the MNIST image is then randomly picked among all instances of this digit. Examples are

given in Fig. 3. No particular measure was taken to reproduce the alphanumeric syntax, country indicators or rivets peculiar to real LP, as this simple design is sufficient to reproduce syntax shift recognition issues. Yet, as detailed in Section 5, sim-to-real transfer succeeds as our architecture modifications improve debiasing on both LPR-MNIST and real LP. This proxy dataset is meant to be used on its own, not for pretraining or as data augmentation for real LP.

Creating a tipping point is as easy as purposely removing all samples starting with an arbitrary character at training time, here $\mathcal{N}_1 = \{\texttt{9}\}$. Training and validation splits respectively contain 72,081 and 8,937 samples. Among the test split, 8,982 samples are of source syntax and 1,018 of target syntax.

Real license plates The main goal of this paper is to reach production-ready text recognition accuracy on pre-localized cropped photographs of French LP captured in the wild. The corresponding dataset is composed of RGB photographs taken on the roads over a one-year period ranging from November 2020 to October 2021. For training purposes, annotations of the corresponding Latin alphanumeric letters and digits of length $L = 7$ are readily available. Here, the character set is $\Omega = \mathcal{A} \cup \mathcal{D}$, where $\mathcal{A}$ is the set of alphabetic uppercase letters.

As previously stated, the main difficulty which we address arises from the shift in the distribution of the characters on the LP as seen in Fig. 1. Due to the assignment of serial numbers in ascending order, the leftmost letter ultimately switches from one letter to the next, for example from an `F` to a `G`, thereby introducing a sudden mismatch between the trained model and the vehicle plates on the roads. Training and validation splits have their leftmost letter ranging from `A` to `F` and contain 797,469 and 97,725 samples, respectively. Some images starting with the letter `G` appeared around the end of 2021 are available in the test split, i.e. $\mathcal{N}_1 = \{\texttt{G}\}$, so that 96,909 samples are of source syntax and 3,097 of target syntax. Each split contains images from the whole image capture period in order to dismiss any visual shift unrelated to syntax shift.

5 Experiments

5.1 Evaluation protocol

Exact match To measure model performance, we compute the *exact match* (also known as subset accuracy), a stricter version of accuracy in which all characters in a predicted string have to match their label counterpart exactly for the sample to be counted as correct. It is reported as a percentage.

Seeds & standard deviation Despite being relevant for assessing the robustness of a method in bias reduction, standard deviation (SD) is rarely reported [8,22,26,29,32,33]. In this paper, all experiment are repeated on 5 random seeds so as to reliably report SD. Indeed, exact match must be analyzed through its mean and SD jointly since variability can be very high on target syntax,

meaning that some models totally block the prediction of known characters in a new position. When analyzing results in our controlled experimental setting, we therefore aim for the lowest possible SD.

5.2 Training details

When not otherwise specified, models follow the training procedure described below. All models are trained or fine-tuned on the dataset on which test metrics are reported, namely LPR-MNIST or else real LP. Real LP images are resized to $H = 32, W_{\mathcal{I}} = 288$ before being given as input to the model so as to match the aspect ratio of LPR-MNIST as closely as possible. 32 pixels of zero-padding are randomly shared between the left and right sides of the image, hence $W = W_{\mathcal{I}} + 32$. While this padding was first designed for LPR-MNIST to mimic the natural variability in the absolute position of characters on real LP, we found that the latter benefit from it too. Text is processed by a character-level tokenizer. We use the standard cross-entropy loss. The optimizer is AdamW [17] with weight decay equal to 10^{-4}. Models are trained with a budget of 30 epochs and early stopping with a patience of 4 epochs. SaLT is always trained from scratch with a batch size of 512 and a learning rate of 10^{-4}. $\text{TrOCR}_{\text{SMALL}}$ variant is used throughout this paper. For a fairer comparison, we replace TrOCR's byte-pair-encoding (BPE) [23] tokenizer with the character-level one used by SaLT. Indeed, as showcased in Table 1, a character-level tokenizer largely outperforms BPE on some seeds, which we argue is due to BPE being prone to model inter-character relationships. When fine-tuning TrOCR, pretrained weights are loaded into the encoder while the decoder is randomly initialized to adapt to its newly defined character-level tokenizer. We start from the first-stage checkpoint[2] as it is meant for further fine-tuning. TrOCR uses a batch size of 256 and a learning rate of 5×10^{-5}. Its vision encoder keeps its own square image resizing strategy.

Table 2. Performance comparison between vanilla and debiased model variants on real LP. Exact match on target syntax leaps from mediocre to near-training-time performance with SaLT. Conversely, TrOCR remains strongly biased towards source syntax regardless of the decoder.

Model	Encoder	Decoder	Target syntax `^G.*$`	Source syntax `^[A-F].*$`
TrOCR	Transformer	Transformer	27.8 ± 32.8	99.5 ± 0.0
TrOCR[d]		Transformer[d]	63.3 ± 34.1	99.5 ± 0.0
FCN+Transformer	FCN	Transformer	72.5 ± 40.2	99.3 ± 0.0
FCN+Transformer[d] (SaLT)		Transformer[d]	$\mathbf{97.3 \pm 0.7}$	99.3 ± 0.0

[d] debiased.

[2] Available at https://huggingface.co/microsoft.

5.3 Results on real license plates

SaLT is future-proof According to Table 2, SaLT outperforms the standard TrOCR baseline by a significant margin on the real LP dataset, gaining +69.5 points on target syntax to reach 97.3% mean exact match. Moreover, SD is reduced to a low 0.7%, making any seed suitable for production. Meanwhile, peak performance is preserved on source syntax, nearing that of TrOCR. SaLT is thus future-proof albeit only using the biased dataset of past serials, retraining it at each change in LP syntax is therefore unnecessary. Additionally, we show that debiasing the decoder is key to future-proofing by comparing the two FCN-encoder variants. Indeed, with an FCN-encoder, switching from a biased Transformer-decoder to SaLT's decoder stabilizes target syntax performance while bringing a +24.8 points gain in mean exact match.

Transformer-encoders are disrupted by syntax shift For completeness, we also try our decoder-debiasing modifications on TrOCR. As reported in Table 2, our attempt at debiasing TrOCR partially works with a significant improvement in mean exact match, although SD remains high for both the standard and debiased configurations. We argue it is impossible to debias TrOCR more efficiently because its Transformer-encoder can see the whole image and encode it with position information, hereby justifying the need for SaLT's position-less FCN-encoder with low receptive field r.

Table 3. Ablation study of SaLT components on LPR-MNIST. Here, a ✗ denotes component removal among previous token reinjection, residual connection and position in $\boldsymbol{V}$, thus making a step towards debiasing.

Model	Encoder	Decoder			Target syntax	Source syntax
		Reinject previous token	Last layer			
			Residual	Position in $\boldsymbol{V}$	`^9.*$`	`^[0-8].*$`
TrOCR	Transformer	✓	✓	✓	85.6 ± 7.3	98.8 ± 0.4
FCN+Transformer	FCN	✓	✓	✓	87.1 ± 8.4	99.3 ± 0.1
		✓	✓	✗	90.8 ± 6.7	99.3 ± 0.1
		✓	✗	✓	93.9 ± 2.6	99.4 ± 0.1
		✓	✗	✗	94.5 ± 1.9	99.3 ± 0.1
		✗	✓	✓	95.5 ± 0.7	99.3 ± 0.1
		✗	✓	✗	95.1 ± 1.0	99.3 ± 0.1
		✗	✗	✗	95.7 ± 1.5	99.4 ± 0.1
FCN+Transformer[d] (SaLT)	FCN	✗	✗	✓	$\mathbf{96.6 \pm 0.9}$	99.4 ± 0.1

[d] debiased.

5.4 Ablation study on LPR-MNIST

Joint effect of SaLT components In Table 3, we present results after ablating critical components of SaLT on LPR-MNIST, including TrOCR reaching a baseline of 85.6% mean exact match with 7.3% SD. We observe an overall trend of increase in mean exact match and decrease in SD, proving that our debiasing proposals tend to positively complement each other when combined. Meanwhile, exact match on source syntax is not altered.

Let us now consider each decoder modification separately, starting from an FCN-encoder with a biased decoder. Preferring position queries to previous token reinjection gives the highest gains on target syntax, with a biggest leap of +8.4 points mean and -7.7 points SD. Also, it consistently improves both the mean and SD compared to previous-token-reinjection counterparts. Cutting off the cross-attention residual is the second most consistent modification, with a biggest leap of +6.8 points mean and -5.8 points SD. When combined with position queries however, the gain in mean comes at the cost of a minor increase in instability between seeds. Lastly, while leaving $\boldsymbol{V}$ without positional encoding does bring a highest gain of +3.7 points mean and -1.7 points SD, combining it with position queries leads to small performance drops.

Consequently, the overall best configuration chosen for SaLT comprises only two of the three architectural modifications. We notice that removing previous token reinjection is the most direct way of cancelling language modeling. Cutting off the cross-attention residual further brings a small performance gain.

Table 4. Performance comparison between vanilla and debiased model variants on LPR-MNIST without encoder dropout. Decoder debiasing on LPR-MNIST also works when encoder dropout is disabled.

Model	Target syntax `^9.*$`	Source syntax `^[0-8].*$`
FCN+Transformer	89.1 ± 7.1	97.6 ± 0.6
FCN+Transformer[d]	$\mathbf{95.0 \pm 1.4}$	98.4 ± 0.2

[d] debiased.

Impact of dropout Assuming that dropout should help alleviating bias for being a regularization method, we further report debiasing performance without dropout in the FCN on LPR-MNIST. Table 4 demonstrates that decoder debiasing without encoder dropout brings a gain on target syntax close to that with it enabled (previously shown in Table 3). Yet, enabling dropout achieves better results both on source and target syntax.

Table 5. Performance comparison between vanilla and debiased model variants on LPR-MNIST with digit `9` missing in various positions. All missing character positions benefit from SaLT's decoder debiasing.

Model	Position	Target syntax `^9.*$`	Source syntax `^[0-8].*$`
FCN+Transformer	2	95.3 ± 1.3	99.2 ± 0.1
	3	95.4 ± 5.1	99.2 ± 0.1
	4	80.2 ± 35.2	99.2 ± 0.1
	5	93.2 ± 2.0	99.2 ± 0.1
FCN+Transformer[d] (SaLT)	2	$\mathbf{96.4 \pm 0.8}$	99.3 ± 0.1
	3	$\mathbf{97.6 \pm 0.9}$	99.3 ± 0.1
	4	$\mathbf{98.3 \pm 0.7}$	99.3 ± 0.1
	5	$\mathbf{96.7 \pm 1.0}$	99.3 ± 0.1

[d] debiased.

Generalization to other positions We investigate digit `9` missing in other positions in LPR-MNIST, i.e. $\mathcal{N}_i = \{\texttt{9}\}$, $i \in \{2, 3, 4, 5\}$. SaLT achieves debiasing on all positions. While position $i = 4$ gains +18.1 points mean and -34.5 points SD, others have decent initial exact match leaving little margin for improvement.

6 Conclusion & future work

This work investigates the robustness to syntax shift of Transformers for license plate recognition. After showing the role of contextual and positional information flows in syntax modeling, we introduce SaLT, a new syntax-less Transformer architecture. SaLT tackles training-time syntax absorption by using an FCN-encoder and cutting off key components in an enhanced Transformer-decoder. Distinct from popular attention-based networks hallucinating on legible images, it exhibits future-proof recognition performance on real-life vehicle plates and synthetic datasets. Indeed, extensive experiments show that SaLT achieves high recognition accuracy and low variability on future syntax while fully preserving performance on past syntax. Moreover, it provides competitive debiasing in any position on a license plate. We are therefore optimistic about its robustness to other characters or other countries with their own syntax. Better understanding the receptive field of FCN-encoded features and the impact of supplying them with positional information may be key to bridging the small performance gap separating target and source syntax. We also note the very goal of relaxing syntax modeling comes at the cost of not learning parts of the syntax which could be useful for prediction, e.g. the ordering between letters and digits in licenses, hence future work will concentrate on re-injecting desirable constraints.

Acknowledgments. This study was funded by the CIFRE ANRT grant No. 2023/0195.

References

1. Anagnostopoulos, C.N.E., Anagnostopoulos, I.E., Psoroulas, I.D., Loumos, V., Kayafas, E.: License Plate Recognition From Still Images and Video Sequences: A Survey. IEEE Transactions on Intelligent Transportation Systems **9**(3), 377–391 (Sep 2008). https://doi.org/10.1109/TITS.2008.922938
2. Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., Zagoruyko, S.: End-to-End Object Detection with Transformers. In: Vedaldi, A., Bischof, H., Brox, T., Frahm, J.M. (eds.) Computer Vision – ECCV 2020. pp. 213–229. Springer International Publishing, Cham (2020). https://doi.org/10.1007/978-3-030-58452-8_13
3. Coquenet, D., Chatelain, C., Paquet, T.: DAN: A Segmentation-free Document Attention Network for Handwritten Document Recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence pp. 1–17 (2023). https://doi.org/10.1109/TPAMI.2023.3235826
4. d'Ascoli, S., Touvron, H., Leavitt, M., Morcos, A., Biroli, G., Sagun, L.: ConViT: Improving Vision Transformers with Soft Convolutional Inductive Biases. Journal of Statistical Mechanics: Theory and Experiment **2022**(11), 114005 (Nov 2022). https://doi.org/10.1088/1742-5468/ac9830
5. Fu, Z.: Deep Learning for License Plate Number Recognition: A Survey. In: 2024 International Conference on Cyber-Physical Social Intelligence (ICCSI). pp. 1–6 (Nov 2024). https://doi.org/10.1109/ICCSI62669.2024.10799401
6. Graves, A., Fernández, S., Gomez, F., Schmidhuber, J.: Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. In: Proceedings of the 23rd International Conference on Machine Learning. pp. 369–376. ICML '06, Association for Computing Machinery, New York, NY, USA (Jun 2006). https://doi.org/10.1145/1143844.1143891
7. He, K., Zhang, X., Ren, S., Sun, J.: Deep Residual Learning for Image Recognition. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 770–778. IEEE, Las Vegas, NV, USA (Jun 2016). https://doi.org/10.1109/CVPR.2016.90
8. Hu, J., Liu, C., Yan, Q., Zhu, X., Wu, J., Du, J., Dai, L.: Vision-Language Adaptive Mutual Decoder for OOV-STR. In: Lu, H., Ouyang, W., Huang, H., Lu, J., Liu, R., Dong, J., Xu, M. (eds.) Image and Graphics. pp. 175–186. Springer Nature Switzerland, Cham (2023). https://doi.org/10.1007/978-3-031-46308-2_15
9. Khan, M.M., Ilyas, M.U., Khan, I.R., Alshomrani, S.M., Rahardja, S.: License Plate Recognition Methods Employing Neural Networks. IEEE Access **11**, 73613–73646 (2023). https://doi.org/10.1109/ACCESS.2023.3254365
10. Kumar, A., Shivakumara, P., Chowdhury, P.N., Pal, U., Liu, C.L.: DPAM: A New Deep Parallel Attention Model for Multiple License Plate Number Recognition. In: 2022 26th International Conference on Pattern Recognition (ICPR). pp. 1485–1491 (Aug 2022). https://doi.org/10.1109/ICPR56361.2022.9956285
11. Laroca, R., Zanlorensi, L.A., Gonçalves, G.R., Todt, E., Schwartz, W.R., Menotti, D.: An Efficient and Layout-Independent Automatic License Plate Recognition System Based on the YOLO detector. IET Intelligent Transport Systems **15**(4), 483–503 (Apr 2021). https://doi.org/10.1049/itr2.12030
12. Lecun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE **86**(11), 2278–2324 (Nov 1998). https://doi.org/10.1109/5.726791
13. Li, M., Lv, T., Chen, J., Cui, L., Lu, Y., Florencio, D., Zhang, C., Li, Z., Wei, F.: TrOCR: Transformer-based optical character recognition with pre-trained models.

In: Proceedings of the Thirty-Seventh AAAI Conference on Artificial Intelligence and Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence and Thirteenth Symposium on Educational Advances in Artificial Intelligence. AAAI'23/IAAI'23/EAAI'23, vol. 37, pp. 13094–13102. AAAI Press (Feb 2023). https://doi.org/10.1609/aaai.v37i11.26538

14. Liu, D., Niehues, J., Cross, J., Guzmán, F., Li, X.: Improving Zero-Shot Translation by Disentangling Positional Information. In: Zong, C., Xia, F., Li, W., Navigli, R. (eds.) Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers). pp. 1259–1273. Association for Computational Linguistics, Online (Aug 2021). https://doi.org/10.18653/v1/2021.acl-long.101
15. Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., Guo, B.: Swin Transformer: Hierarchical Vision Transformer using Shifted Windows. In: 2021 IEEE/CVF International Conference on Computer Vision (ICCV). pp. 9992–10002. IEEE, Montreal, QC, Canada (Oct 2021). https://doi.org/10.1109/ICCV48922.2021.00986
16. Liu, Z., Winata, G.I., Cahyawijaya, S., Madotto, A., Lin, Z., Fung, P.: On the Importance of Word Order Information in Cross-lingual Sequence Labeling. Proceedings of the AAAI Conference on Artificial Intelligence **35**(15), 13461–13469 (May 2021). https://doi.org/10.1609/aaai.v35i15.17588
17. Loshchilov, I., Hutter, F.: Decoupled Weight Decay Regularization. In: International Conference on Learning Representations (Sep 2018)
18. Montazzolli, S., Jung, C.: Real-Time Brazilian License Plate Detection and Recognition Using Deep Convolutional Neural Networks. In: 2017 30th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI). pp. 55–62 (Oct 2017). https://doi.org/10.1109/SIBGRAPI.2017.14
19. Moussa, D., Maier, A., Spruck, A., Seiler, J., Riess, C.: Forensic License Plate Recognition with Compression-Informed Transformers. In: 2022 IEEE International Conference on Image Processing (ICIP). pp. 406–410 (Oct 2022). https://doi.org/10.1109/ICIP46576.2022.9897178
20. Park, S., Chung, S., Lee, J., Choo, J.: Improving Scene Text Recognition for Character-Level Long-Tailed Distribution (Mar 2023). https://doi.org/10.48550/arXiv.2304.08592
21. Rai, V., Kumar, G., Kushwaha, A.K., Pandey, A., Kumar, V., Yadav, S.: TrALPR: Automatic License Plate Recognition using Transformers. In: 2023 International Conference on Modeling, Simulation & Intelligent Computing (MoSICom). pp. 144–149 (Dec 2023). https://doi.org/10.1109/MoSICom59118.2023.10458786
22. Resende Gonçalves, G., Alves Diniz, M., Laroca, R., Menotti, D., Robson Schwartz, W.: Real-Time Automatic License Plate Recognition through Deep Multi-Task Networks. In: 2018 31st SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI). pp. 110–117 (Oct 2018). https://doi.org/10.1109/SIBGRAPI.2018.00021
23. Sennrich, R., Haddow, B., Birch, A.: Neural Machine Translation of Rare Words with Subword Units. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 1715–1725. Association for Computational Linguistics, Berlin, Germany (2016). https://doi.org/10.18653/v1/P16-1162
24. Singh, S.S., Karayev, S.: Full Page Handwriting Recognition via Image to Sequence Extraction. In: Lladós, J., Lopresti, D., Uchida, S. (eds.) Document Analysis and

Recognition – ICDAR 2021. pp. 55–69. Springer International Publishing, Cham (2021). https://doi.org/10.1007/978-3-030-86334-0_4
25. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: A Simple Way to Prevent Neural Networks from Overfitting. Journal of Machine Learning Research **15**(56), 1929–1958 (2014)
26. Tran Tien, H., Ngo, T.D.: Unsupervised Domain Adaptation with Imbalanced Character Distribution for Scene Text Recognition. In: 2023 IEEE International Conference on Image Processing (ICIP). pp. 3493–3497 (Oct 2023). https://doi.org/10.1109/ICIP49359.2023.10222310
27. Ulyanov, D., Vedaldi, A., Lempitsky, V.: Improved Texture Networks: Maximizing Quality and Diversity in Feed-Forward Stylization and Texture Synthesis. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 4105–4113. IEEE, Honolulu, HI (Jul 2017). https://doi.org/10.1109/CVPR.2017.437
28. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. In: Proceedings of the 31st International Conference on Neural Information Processing Systems. pp. 6000–6010. NIPS'17, Curran Associates Inc., Red Hook, NY, USA (Dec 2017)
29. Wan, Z., Zhang, J., Zhang, L., Luo, J., Yao, C.: On Vocabulary Reliance in Scene Text Recognition. In: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 11422–11431. IEEE, Seattle, WA, USA (Jun 2020). https://doi.org/10.1109/CVPR42600.2020.01144
30. Wu, S., Sun, D.J., Qiu, G.: Emission analysis based on mixed traffic flow and license plate recognition model. Transportation Research Part D: Transport and Environment **134**, 104331 (Sep 2024). https://doi.org/10.1016/j.trd.2024.104331
31. Xue, C., Huang, J., Zhang, W., Lu, S., Wang, C., Bai, S.: Image-to-Character-to-Word Transformers for Accurate Scene Text Recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence **45**(11), 12908–12921 (Nov 2023). https://doi.org/10.1109/TPAMI.2022.3230962
32. Yim, M., Kim, Y., Cho, H.C., Park, S.: SynthTIGER: Synthetic Text Image GEneratoR Towards Better Text Recognition Models. In: Lladós, J., Lopresti, D., Uchida, S. (eds.) Document Analysis and Recognition – ICDAR 2021. pp. 109–124. Springer International Publishing, Cham (2021). https://doi.org/10.1007/978-3-030-86337-1_8
33. Zhang, J., Lin, T., Xu, Y., Chen, K., Zhang, R.: Relational Contrastive Learning for Scene Text Recognition. In: Proceedings of the 31st ACM International Conference on Multimedia. pp. 5764–5775. MM '23, Association for Computing Machinery, New York, NY, USA (Oct 2023). https://doi.org/10.1145/3581783.3612247
34. Zhang, L., Wang, P., Li, H., Li, Z., Shen, C., Zhang, Y.: A Robust Attentional Framework for License Plate Recognition in the Wild. IEEE Transactions on Intelligent Transportation Systems **22**(11), 6967–6976 (Nov 2021). https://doi.org/10.1109/TITS.2020.3000072
35. Zhang, T., Jia, W.: Automatic license plate recognition using transformer. In: Fourteenth International Conference on Graphics and Image Processing (ICGIP 2022). vol. 12705, pp. 129–138. SPIE (Jun 2023). https://doi.org/10.1117/12.2680529
36. Zhang, X., Zhu, B., Yao, X., Sun, Q., Li, R., Yu, B.: Context-Based Contrastive Learning for Scene Text Recognition. Proceedings of the AAAI Conference on Artificial Intelligence **36**(3), 3353–3361 (Jun 2022). https://doi.org/10.1609/aaai.v36i3.20245
37. Zheng, T., Chen, Z., Huang, B., Zhang, W., Jiang, Y.G.: MRN: Multiplexed Routing Network for Incremental Multilingual Text Recognition. In: 2023 IEEE/CVF

International Conference on Computer Vision (ICCV). pp. 18598–18607. IEEE, Paris, France (Oct 2023). https://doi.org/10.1109/ICCV51070.2023.01709