A. Proof of Theorem 1

Assumption 1 (A1) We assume the linguistic information functions as a common-cause of the input visual information and the prediction outputs at all timestamps (See Fig. 3). We model the sample image as a "rendered" result of the label $\boldsymbol{y}$. Also, we assume the label (words) is generated according to linguistic information c, making the label $\boldsymbol{y}$ a causal result of c. Hence, linguistic context c and the character-level visual information $x_{[t]}$ are the only two direct factors affecting the probability of $y_{[t]}$ at timestamp t,

$$\begin{aligned}
&P(y_{[t]}|x_{[t]},\boldsymbol{x},y_{[t-1]}...y_{[0]},l,c)\\
\overset{\text{(a)}}{:=}&P(y_{[t]}|x_{[t]},\boldsymbol{x},Pre_{[t]},l,c)\\
=&P(y_{[t]}|x_{[t]},c),
\end{aligned} \tag{16}$$

where we denote prefix of $y_{[t]}$ as $Pre_{[t]}$ in (a).

Proof.

$$\begin{aligned}
&P(\boldsymbol{y}|\boldsymbol{x},l)\\
=&\prod_{t=1}^{l}P(y_{[t]}|\boldsymbol{x},l,y_{[t-1]}...y_{[0]})\\
=&\prod_{t=1}^{l}\int^{c\in C_{[t]}}P(y_{[t]}|\boldsymbol{x},Pre_{[t]},l,c)P(c|\boldsymbol{x},l)\\
\overset{\text{(a)}}{=}&\prod_{t=1}^{l}\int^{c\in C_{[t]}}P(y_{[t]}|x_{[t]},c)P(c|\boldsymbol{x},l).
\end{aligned} \tag{17}$$

□

Here, (a) is derived by applying Eq. 16 of Assumption A1. The integral term can be interpreted as an ensemble of "anchored prediction" $P(y_{[t]}|x_{[t]},c)$ over all possible contexts c, which is similar to the hidden anchor mechanism [15]. Hence, we call this theorem the anchor property of context.

B. Proof of Theorem 2

Assumption 2 (A2) The shape (character visual information) of a character and its context (linguistic information) are independent given the character $y_{[t]}$, i.e.,

$$\begin{aligned}
&P(x_{[t]}|y_{[t]},c)=P(x_{[t]}|y_{[t]})\\
\iff &P(x_{[t]},c|y_{[t]})=P(x_{[t]}|y_{[t]})P((c|y_{[t]}))
\end{aligned} \tag{18}$$

Theorem 2: The Separable Property of Linguistic Information and Character Visual Information

Given assumption A2 holds, the effect of character visual information over label $P(y_{[t]}|x_{[t]})$ and the effect of $P(y_{[t]}|c)$ can be separated from contextual prediction $P(y_{[t]}|x_{[t]},c)$:

$$\begin{aligned}
&P(y_{[t]}|x_{[t]},c)\\
\propto&\frac{P(y_{[t]}|x_{[t]})P(y_{[t]}|c)}{P(y_{[t]})}\\
:=&Pr(y_{[t]},x_{[t]},c)
\end{aligned} \tag{19}$$

Proof.

$$\begin{aligned}
&P(y_{[t]}|x_{[t]},c)\\
=&\frac{P(x_{[t]},y_{[t]},c)}{P(x_{[t]},c)}\\
=&\frac{P(x_{[t]},c|y_{[t]})P(y_{[t]})}{P(x_{[t]},c)}\\
\overset{\text{(a)}}{=}&\frac{P(x_{[t]}|y_{[t]})P(c|y_{[t]})P(y_{[t]})}{P(x_{[t]},c)}\\
=&\frac{P(y_{[t]})}{P(x_{[t]},c)}P(x_{[t]}|y_{[t]})P(c|y_{[t]})\\
=&\frac{P(y_{[t]})}{P(x_{[t]},c)}\ \frac{P(y_{[t]}|x_{[t]})P(x_{[t]})}{P(y_{[t]})}\ \frac{P(y_{[t]}|c)P(c)}{P(y_{[t]})}\\
=&P(y_{[t]}|c)P(y_{[t]}|x_{[t]})\frac{P(x_{[t]})P(c)}{P(x_{[t]},c)P(y_{[t]})}\\
=&\frac{P(y_{[t]}|x_{[t]})P(y_{[t]}|c)}{P(y_{[t]})}\ \frac{P(x_{[t]})}{P(x_{[t]}|c)}\\
=&Pr(y_{[t]},x_{[t]},c)\frac{P(x_{[t]})}{P(x_{[t]}|c)}\\
=&Pr(y_{[t]},x_{[t]},c)\frac{P(x_{[t]})}{\sum_{y'_{[t]}}^{\mathcal{Y}}P(x_{[t]}|y'_{[t]},c)P(y'_{[t]}|c)}\\
\overset{\text{(b)}}{=}&Pr(y_{[t]},x_{[t]},c)\frac{P(x_{[t]})}{\sum_{y'_{[t]}}^{\mathcal{Y}}P(x_{[t]}|y'_{[t]})P(y'_{[t]}|c)}\\
\overset{\text{(c)}}{=}&\frac{Pr(y_{[t]},x_{[t]},c)}{\sum_{y'_{[t]}}^{\mathcal{Y}}Pr(y'_{[t]},x_{[t]},c)}\\
\overset{\text{(d)}}{\propto}&Pr(y_{[t]},x_{[t]},c).
\end{aligned} \tag{20}$$

□

Here, $\mathcal{Y}$ is the character set. Step (a) and (b) is derived using Eq. 18 in assumption A2. Step (c) is derived by applying Bayesian rule over $P(x_{[t]}|y'_{[t]})$ and canceling $x_{[t]}$. Although $\sum_{y'_{[t]}}^{\mathcal{Y}}Pr(y'_{[t]},x_{[t]},c)$ is not a constant number and may vary with timestamp t, but it is the same for all label $y'_{[t]}$ at a certain timestamp t, hence step (d) holds, despite the constant factor can change.

C. Proof of Theorem 3

Combining Theorem 1 and Theorem 2, we have the Decoupled Context Anchor mechanism,

$$\begin{aligned} &P(\boldsymbol{y}|\boldsymbol{x},l) \\ =&\prod_{t=1}^{l} P(y_{[t]}|x_{[t]}) \prod_{t=1}^{l} \int^{c\in C_{[t]}} P(y_{[t]}|c)P(c|\boldsymbol{x},l). \end{aligned} \tag{21}$$

Proof.

$$\begin{aligned} &P(\boldsymbol{y}|\boldsymbol{x},l) = \prod_{t=1}^{l} \int^{c\in C_{[t]}} P(y_{[t]}|x_{[t]},c)P(c|\boldsymbol{x},l) \\ \propto&\prod_{t=1}^{l} \int^{c\in C_{[t]}} \frac{P(y_{[t]}|x_{[t]})P(y_{[t]}|c)}{P(y_{[t]})} P(c|\boldsymbol{x},l) \\ =&\alpha \prod_{t=1}^{l} \frac{P(y_{[t]}|x_{[t]})}{P(y_{[t]})} \prod_{t=1}^{l} \int^{c\in C_{[t]}} P(y_{[t]}|c)P(c|\boldsymbol{x},l), \\ =&\beta(\mathbf{y}) \prod_{t=1}^{l} P(y_{[t]}|x_{[t]}) \prod_{t=1}^{l} \int^{c\in C_{[t]}} P(y_{[t]}|c)P(c|\boldsymbol{x},l), \\ :=&\prod_{t=1}^{l} P(y_{[t]}|x_{[t]}) \prod_{t=1}^{l} \int^{c\in C_{[t]}} P(y_{[t]}|c)P(c|\boldsymbol{x},l), \end{aligned} \tag{22}$$

□

The visual prediction can be taken out of the integral as it is not affected by the linguistic information, i.e., c. Here,

$$\beta(\mathbf{y}) = \frac{\alpha}{\prod_1^l P(y_t)}, \tag{23}$$

and it is a character frequency term related to the word. During training, $\beta(\mathbf{y}^*)$ only associates with the label, hence would be constant for a certain label and won't produce gradient. During the evaluation, as the dictionary and character frequency are unknown, character frequency would be assumed as uniform, resulting in $\beta(\mathbf{y})$ being a constant number $\frac{\alpha}{|C_{eval}|^l}$ for all words with length l. Hence, despite varying from word to word, treating it as a constant does not affect either training or evaluation. As a result, β is omitted for writing convenience.

D. An Engineering Perspective of OpenCCD

While the main paper puts more stress on the theoretical part of the framework, we present an engineering perspective of the OpenCCD framework which focuses on how things are implemented over what each module does.

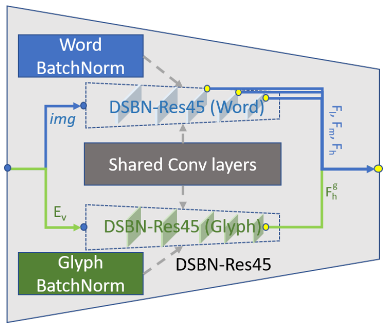

Figure 9. The DSBN-Res45 backbone.

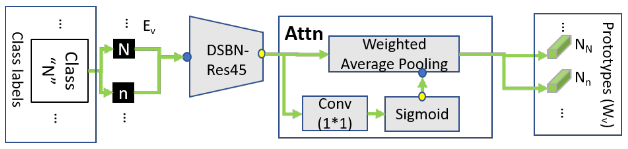

Figure 10. The prototype generation process.

D.1. DSBN-Res45

In this work, the DSBN-Res45 (Fig. 9) is used to encode word images and glyphs into corresponding visual features. The DSBN-Res45 backbone is a modified version of the 45-layer ResNet used in [44]. Here, we replaced all its batch norm layers with re-implemented DSBN [5] layers. This adaption is made to alleviate the impact of the bias between the word image domain and the glyph domain. Specifically, the network uses the same set of convolution kernels for both word-level images and glyph images, while using the domain-specific batch statistics for normalization of each specific domain. The layout of the regular model is similar to the original DAN implementation [44], and the large model simply adds more latent channels to the backbone, and further details like specific network layout can be found in the released code. This module is a part of the base model and is used in **all** models in ablative studies.

D.2. Prototype Generation

The prototypes generation process is shown in Fig. 10 for each class, the framework first extracts corresponding visual features of each glyph with the backbone. Then spatial attention is applied to reduce the feature map to a single feature via the Attn Module. Specifically, the module first estimates the foreground/background attention mask with a convolution layer, then reduces and normalizes the feature map into corresponding prototypes. Same to [23], a label may possess more than one prototype as each character can have different "cases", e.g., 'N' and 'n'.The prototypes are normalized to alleviate character-frequency related bias. This module is also a part of the base model and is used in **all** models in ablative studies.

Table 5. Important notations in the paper with first occurrence and their brief explanations.

Notation	Occurrence	Explanation
N	-	Number of glyphs (‘N’ and ‘n’ has the same label)
M	-	Number of characters (labels)
E	Fig.2	The collection of the glyph (E_v) and semantic embedding (E_c)
E_c	Fig.2	The collection of the glyph (E_v) for characters. Each character can have several glyphs according to how many cases it has.
E_v	Fig.2	The semantic embedding (E_c) for characters. Each character only has one embedding in our framework.
W_v	Fig.2	Prototypes generated from E_v. $W_v : R^{N\times D}$, $N = \|E_v\|$ and $D = 512$
$\hat{\boldsymbol{y}}$	Eq. 1	Predicted word label, consisted of a ordered sequence of predicted character labels $\hat{y}_{[t]}$.
$\boldsymbol{y}$	Eq. 1	Any word label, consisted of a ordered sequence of character labels $y_{[t]}$.
$\hat{y}_{[t]}$	Eq. 1	t^{th} predicted character.
θ	Eq. 1	The trainable parameters of the framework.
$\boldsymbol{x}$	Eq. 1	character visual information of all characters in a word
l	Eq. 2	Length of a sequence.
$x_{[t]}$	Eq. 3	character visual information of the t^{th} character in the word
c	Eq. 3	Linguistic information.
$\boldsymbol{y^*}$	Eq. 4	Ground truth word label, consisted of a ordered sequence of ground truth character labels $y^*_{[t]}$.
l^*	Eq. 4	Length of the ground truth word.
$C_{[t]}$	Eq. 4	All possible Linguistic information.
L_{len}	Eq. 4	Minus likelihood of the correct length being predicted: $-logP(l^*\|\boldsymbol{x})$
L_{vis}	Eq. 4	Minus likelihood of the correct word being predicted according to character visual information: $-\sum_{t=1}^{l^*}(logP(y^*_{[t]}\|x_{[t]}))$
L_{vis}	Eq. 4	Minus likelihood of the correct word being predicted according to linguistic information: $-\sum_{t=1}^{l} log(\int^{c\in C_{[t]}} P(y^*_{[t]}\|c)P(c\|\boldsymbol{x}, l^*))$
$y_{[t]}$	Eq. 5	Any t^{th} character, applies to any possible character in the character set (means it applies to predicted and ground truth as well.)
ψ	Ch3.3.3	Function indexes all corresponding prototypes of the input label $y_{[t]}$.
w_v	Ch3.3.3	A row in W_v.
$\hat{c}$	Ch3.3.4	Context predicted via transformer. $\hat{c} \in R^{l\times D}$
$\mathbf{Y}_{[t]}$	Ch3.3.4	The probability distribution at time stamp t: $P(\mathbf{Y}_{[t]}\|x_{[t]}) : (P(y^0_{[t]}\|x_{[t]}), ..., P(y^M_{[t]}\|x_{[t]}))$
Y	Ch3.3.4	The probability distribution at all time stamps: $Y \in (0, 1)^{l\times M} : (P(\mathbf{Y}_{[0]}\|x_{[0]}), ..., P(\mathbf{Y}_{[l]}\|x_{[l]}))$

D.3. Data, Training, and Evaluation

The training and evaluation and models for most experiments (all except dictionary-based close-set experiments) are now released to Kaggle[4]. For the open-set task, the training dataset is built by aggregating the following datasets: RCTW [36], Chinese and Latin subset of the MLT-2019 dataset [28], LSVT [38], ART [10], and CTW [50]. The training character set contains 3755 Tire-1 Simplified Chinese characters, 52 Latin characters, and 10 digits. Vertical Samples and samples containing characters outside of the training character set are removed from the training set (samples with Tradition Chinese Characters are removed as well). The evaluation dataset contains 4009 horizontal images from the Japanese subset of the MLT-2019 dataset and the testing character set includes 1460 characters appearing in the evaluation set, making a total of 1461 different classes adding the “unknown” class. For the close-set model, we use exactly the same datasets as DAN [44], which adopts the most used MJ [17]-ST [14] combo as the training set. For the zero-shot Character recognition tasks, we reuse the split from OSOCR [23], which follows HCCR’s protocol [4]. Note that like HCCR [4], few methods made the exact split of seen and novel characters public.

During training, a label sampler is used to sample a sub-

[4]https://www.kaggle.com/vsdf2898kaggle/osocrtraining

set of characters during each iteration like OSOCR. For word-level tasks, the model processes data is similar to DAN [44]. Specifically, the model takes 32*128 RGB clips, where the images are resized keeping aspect ratio and center-padded into 32*128 with zeros for both training and testing. The common dictionary-free protocols are used for all word-level evaluations. For character recognition tasks, the model treats character images like word images, despite the clip size being set as 32*64 to speed up. For all three tasks, we use Notofont as the glyph provider, where each character is rendered and centered to 32*32 binary patches. The training processes are mostly the same with DAN [44] except for adding a prototype generation process.

For evaluation, the most popular evaluation protocol, the Line Accuracy is used to measure word recognition performance following the community. The Character Accuracy (1-NED) is also used as compensation for open-set word recognition tasks to give an intuitive insight on the recognition quality per character. For the zero-shot Chinese Character recognition task, Character Accuracy and Line Accuracy is the same number, simply called Accuracy by the community.

E. Extra Details

E.1. Notations

We made a notation table (Table 5) to include all used notations in this paper. In most cases, hat $(\hat{.})$ indicate the max probable prediction, and asterisk $(.^{*})$ indicates the ground truth. **Bold** notations indicate vectors and capital alphabets indicate matrices, sets, or distributions.

E.2. About Related Work

Despite the proposed Decoupled Context Anchor mechanism (DCA) uses a transformer to model contextual information, this work is not directly related to the transformer-based methods [8, 12, 55]. Structure-wise, the transformer in our method is a BERT-style transformer encoder, also used in [11, 49], while [8, 12, 55] uses GPT-style transformer decoders. Purpose-wise, the transformer in this work is used as a regularization term (which is the direct reason we backpropagate to the feature encoder). Instead, [11, 49] use the transformer as a post-process module by cutting the gradient flow (Note [49] did not clarify this in their paper, but you can see it from their officially released code). Conventional transformer-based methods directly decode the CNN features into prediction sequences. Therefore, the DCA module is different in terms of structure and motivation.