Transformer's hidden-size $d_{model}$, then a 2D positional encoding added and finally flattened into a 1D sequence. 2D positional encoding is a fixed sinusoidal encoding as in [28], but using the first $d_{model}/2$ channels to encode the Y coordinate and the rest to encode the X coordinate (Equation 1) (similar to [20]). Output $\boldsymbol{I}$ of the *Flatten* layer is made available to all Transformer decoder layers, as is standard.

$$\begin{aligned} PE(y, 2i) &= sin(y/10000^{2i/d_{model}}) \\ PE(y, 2i+1) &= cos(y/10000^{2i/d_{model}}) \\ PE(x, d_{model}/2 + 2i) &= sin(x/10000^{2i/d_{model}}) \\ PE(x, d_{model}/2 + 2i + 1) &= cos(x/10000^{2i/d_{model}}) \\ i &\in [0, d_{model}/4) \end{aligned} \tag{1}$$

**Decoder** The decoder is a Transformer stack with non-causual attention to the encoder output (its layers can attend to the encoder's entire output) and causal self-attention (it can only attend to past positions of its text input). As is standard, training is done with *teacher forcing*, which means that the ground truth text input is shifted one off from the output.

In total, the base configuration has 27.8 million parameters (6.3M decoder, 21.4M ResNet).

The input vectors are enhanced with 1D position encoding, as is standard. Additionally, we concatenate a *line number encoding* (*lne*) - the scaled text line number ($l$) that the token lies on - to it (Figure 3a). Assuming a maximum of 100 text lines, $l \in [1, 100]$ and $lne = l/100$. We added *lne* to the model in order to address line level errors i.e., missing or duplicated lines but it was applied only in Table 1. We haven't yet officially concluded on its impact on model performance and mention it here only for completeness sake.

In order to improve memory and computation requirements of our model, we implemented a localized form of causal self-attention by limiting the attention span to 50 (configurable) past positions. This is similar to Sliding Window Attention of [1]) or a 1D version of Local Self Attention of [20]). We hypothesized that a look back of 50 characters should be enough to satisfy the language modeling needs of our task, while the limited attention span should help training converge faster, both assumptions being validated by experiment. Final model performance however, was not impacted by it. That said, a thorough ablation study was not performed. Practically though, it allowed us to use larger mini-batches by about 12%.

**Objective Function** For each step $t$ the model outputs the token probability distribution $\boldsymbol{p}_t$ over the vocabulary set $\{1, \ldots, V\}$ (Equation 2). This distribution is conditioned upon the tokens generated thus far $\boldsymbol{y}_{<t}$ and $\boldsymbol{I}$ (Equation 3).