# Victor Project AI: Image Description Generation with Latent Visual Clues

Iacer Calixto and Wilker Aziz

August 21, 2018

**Abstract**

We present a generative model of image description that mixes two components: a language model component (agnostic of visual context) and a description component that maps from visual input to words without relying on textual context. In order to better disentangle the two components we employ modelling techniques inspired by topic modelling literature. Furthermore, the mixture model treatment allows for a principled integration of text-only resources such as large unannotated text corpora, for example, by pre-training of the language model component.

**Notation guidelines** We use capital Roman letters (e.g. $X$) for random variables and their lowercase counterparts for assignments (e.g. $x$), $x_1^n$ is shorthand for a sequence $\langle x_1, \ldots, x_n \rangle$ of length $n$, and $x_{<i}$ is shorthand for a prefix sequence of length $i-1$ (when $i = 1$ the prefix $x_{<i}$ denotes an empty sequence), we use boldface letters (e.g. $\mathbf{w}$, $\mathbf{W}$) for deterministic vectors and matrices.

## 1 Image description generation

Image description generation is typically approached as a conditional language modelling task where an image provides additional context for an otherwise standard application of chain rule of probabilities. Figure 1 depicts the conditional independences of the model where a random word observation $x$ depends directly on a deterministic image $\mathbf{v}$ and a random observed history $x_<$, moreover, $\theta$ collectively denotes the deterministic parameters of the model. In this framework, the probability $P_\theta(x_1^n|\mathbf{v})$ of a description $x_1^n$ given an image $\mathbf{v}$ factorises without Markov assumptions as shown below.

$$P_\theta(x_1^n|\mathbf{v}) = \prod_{i=1}^{n} P_\theta(x_i|\mathbf{v}, x_1^m) = \prod_{i=1}^{n} \mathrm{Cat}(x_i|f(\mathbf{v}, x_{<i}; \theta)) \ . \tag{1}$$

At each step $i$, the information available (i.e. image $\mathbf{v}$ and prefix $x_{<i}$) is mapped to a categorical probability distribution over the vocabulary of the language

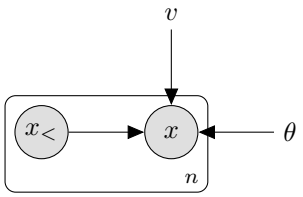


Figure 1: A fully supervised model of image description.

using a parametrised function $f(\cdot;\theta)$—typically a differentiable neural network architecture. Given a dataset of observations, the parameters $\theta$ of the model are point-estimated to attain a local optimum of the likelihood function using a stochastic gradient-based optimiser.

## 2 Latent visual clues

Our model is a hierarchical generative model where we start by sampling topic embeddings $\beta_k$ for $k = 1, \ldots, K$

$$\beta_k \sim \mathcal{N}(0_d, I_d) \ . \tag{2}$$

We then proceed by sampling a sentence-level topic distribution,

$$\pi_0|\mathbf{v} \sim \mathrm{Dir}(\alpha_0(\mathbf{v})) \tag{3}$$

that is, a $K$-dimensional probability vector. Then, for each word position $i = 1, \ldots, n$, we sample a switch

$$B_i|\mathbf{v}, \beta_1^K, \pi_0, x_{<i} \sim \mathrm{Bern}(s(\mathbf{v}, z_0, x_{<i})) \tag{4a}$$

$$z_0 = \sum_{k=1}^{K} \pi_{0k}\beta_k \tag{4b}$$

that alternates between inserting words from language context

$$X_i|B_i = 1, x_{<i} \sim \mathrm{Cat}(f(x_{<i})) \tag{5}$$

and mapping words from visual clues

$$\pi_i|B_i = 0, \mathbf{v}, \beta_1^K, \pi_0, x_{<i} \sim \mathrm{Dir}(\alpha(\mathbf{v}, z_0, x_{<i})) \tag{6a}$$

$$X_i|B_i = 0, \beta_1^K, \pi_0, \pi_i \sim \mathrm{Cat}(g(z_0, z_i)) \tag{6b}$$

$$z_i = \sum_{k=1}^{K} \pi_{ik}\beta_k \tag{6c}$$

expressed in $\mathbf{v}$ and also captured by $z_0$ and $z_i$. Note that the observed image $\mathbf{v}$ only affects the description component (6b) through the topic distributions $\pi_0$ and $\pi_i$ and topic embeddings $\beta_1^K$.

# 3 Architecture

Function $\alpha_0$ used to compute the sentence-level topic distributions (Equation (3)) is a two-layer feed-forward neural network with non-linear activation (e.g. ReLU), as in (7):

$$\begin{aligned}\mathbf{r}_1^1 &= ReLU(\mathbf{W}_1^1\mathbf{v} + \mathbf{b}_1^1),\\ \mathbf{r}_2^1 &= ReLU(\mathbf{W}_2^1\mathbf{r}_1^1 + \mathbf{b}_2^1).\end{aligned} \tag{7}$$

Function $s$ used to compute the stochastic switch $B_i$ (Equation (4a)) depends on whether image features $\mathbf{v}$ are global ($\mathbf{v} \in \mathbb{R}^v$) or local ($\mathbf{v} \in \mathbb{R}^{l \times v}$), where $v$ is the feature dimensionality for each image patch, and $l = 49$ is the number of patches in the image and in practice depends on the actual pre-trained CNN being used.

If image features are global, we concatenate all available features together $[\mathbf{v}; z_0; h_i]$ (where $h_i$ is the hidden state of the decoder for the current timestep) and use it as input to a two-layer feed-forward neural network with non-linear activation (e.g. ReLU), as in (8):

$$\begin{aligned}\mathbf{r}_1^2 &= ReLU(\mathbf{W}_1^2[\mathbf{v}; z_0; h_i] + \mathbf{b}_1^2),\\ \mathbf{r}_2^2 &= ReLU(\mathbf{W}_2^2\mathbf{r}_1^2 + \mathbf{b}_2^2).\end{aligned} \tag{8}$$

If image features are local, we concatenate the hidden state of the decoder for the current timestep with the global topic embedding $[z_0; h_i]$ and use it as a query to select $\mathbf{v}$ (implemented as an attention mechanism). Assuming local image features are vectorised as $\{\mathbf{v}_1, \mathbf{v}_2 \cdots, \mathbf{v}_l\}$, the attention mechanism can be computed as in (9):

$$\begin{aligned}\mathbf{q}_i &= [\mathbf{z}_0; \mathbf{h}_i],\\ e_{il} &= \text{score}(\mathbf{q}_i, \{\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_l\}),\\ \mathbf{v}_{att} &= \text{softmax}(e_{il}),\end{aligned} \tag{9}$$

where the score function can be implemented as the bilinear attention as in (10):

$$\text{score}(\mathbf{x}_i, \mathbf{y}_i) = \mathbf{x}_i\mathbf{W}_{xy}^1\mathbf{y}_i, \tag{10}$$

where $W_{xy}^1$ is a trained model parameter.

**Language model component** The language model component is a standard LSTM-RNN.

**Description model component** Similarly to function $s$ described above (Equation (4a)), function $\alpha$ used to compute the mixing coefficients $\pi_i$ (Equation (6a)) depends on whether image features $\mathbf{v}$ are global ($\mathbf{v} \in \mathbb{R}^v$) or local ($\mathbf{v} \in \mathbb{R}^{l \times v}$), where $v$ is the feature dimensionality for each image patch, and $l = 49$ is the number of patches in the image and in practice depends on the actual pre-trained CNN being used.

If image features are global, we concatenate all available features together $[\mathbf{v}; z_0; h_i]$ (where $h_i$ is the hidden state of the decoder for the current timestep) and use it as input to a two-layer feed-forward neural network with non-linear activation (e.g. ReLU), as in (11):

$$\begin{aligned}\mathbf{r}_1^3 &= ReLU(\mathbf{W}_1^3[\mathbf{v}; z_0; h_i] + \mathbf{b}_1^3),\\ \mathbf{r}_2^3 &= ReLU(\mathbf{W}_2^3\mathbf{r}_1^3 + \mathbf{b}_2^3).\end{aligned} \tag{11}$$

If image features are local, we concatenate the hidden state of the decoder for the current timestep with the global topic embedding $[z_0; h_i]$ and use it as a query to select $\mathbf{v}$ (implemented as an attention mechanism). Assuming local image features are vectorised as $\{\mathbf{v}_1, \mathbf{v}_2 \cdots, \mathbf{v}_l\}$, the attention mechanism can be computed as in (12):

$$\begin{aligned}\mathbf{q}_i &= [\mathbf{z}_0; \mathbf{h}_i],\\ e_{il} &= \text{score}(\mathbf{q}_i, \{\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_l\}),\\ \mathbf{v}_{att} &= \text{softmax}(e_{il}),\end{aligned} \tag{12}$$

where the score function can be implemented as the bilinear attention as described in (10), with distinct learned parameters $W_{xy}^2$. Finally, we either use $\mathbf{r}_2^3$ (global $\mathbf{v}$) or $\mathbf{v}_{att}$ (local $\mathbf{v}$) as input to yet another two-layer feed-forward neural network with non-linear activation (e.g. ReLU), as in (13):

$$\begin{aligned}\mathbf{q} &= \begin{cases}\mathbf{r}_2^3, & \text{if } \mathbf{v} \text{ are global features}\\ \mathbf{v}_{att}, & \text{otherwise}\end{cases}\\ \mathbf{r}_1^4 &= ReLU(\mathbf{W}_1^4\mathbf{q} + \mathbf{b}_1^4),\\ \mathbf{r}_2^4 &= ReLU(\mathbf{W}_2^4\mathbf{r}_1^4 + \mathbf{b}_2^4),\end{aligned} \tag{13}$$

Finally, function $g()$ in Equation (6b) can be implemented as a two-layer feed-forward neural network with non-linear activation (e.g. ReLU) as in (14):

$$\begin{aligned}\mathbf{r}_1^5 &= ReLU(\mathbf{W}_1^5[z_0; z_i] + \mathbf{b}_1^5),\\ \mathbf{r}_2^5 &= ReLU(\mathbf{W}_2^5\mathbf{r}_1^5 + \mathbf{b}_2^5),\end{aligned} \tag{14}$$

which output can be fed into a softmax activation to obtain the next word probabilities.

# References