CLIP-DIFFUSIONLM: APPLY DIFFUSION MODEL ON IMAGE CAPTIONING

Shitong Xu
Imperial College London
`shitong.xu19@imperial.ac.uk`

ABSTRACT

Image captioning task has been extensively researched on by previous work. However, limited experiments focus on generating captions based on non-autoregressive text decoder. Inspired by the recent success of denoising diffusion model on image synthesize tasks, we apply denoising diffusion probabilistic models to text generation in image captioning tasks. We show that our CLIP-diffusionLM capable of in significantly less inference steps than autoregressive models. On the Flickr8k dataset, the model achieves 0.1876 BLEU-4 score. By combining samples from Flickr8k and Flickr30k dataset, our model achieves .... BLEU-4 score. Our code is available at ...

***K*eywords** Diffusion model · CLIP · Non-autoregressive text generation

1 Introduction

Image captioning has being a focus of research over the recent years. Among the previous proposed works, the text encoder used can be classified to 2 general classes, i.e. autoregressive and non-autoregressive class. Most of the sota models falls in the autoregressive class[**?**, **?**, **?**, **?**, **?**]. However, autoregressive generation suffer from 1) the slow generation speed due to the generation step is token by token; and 2) not capable of refining prefix of sentences based on the later generated tokens. Multiple attempts have experimented using a non-autoregressive model in the text generation steps[**?**, **?**, **?**]. The closest work to ours is Masked Non-Autoregressive Image Captioning by Gao et al. [**?**], which uses a BERT model as generator and involves 2 steps-refinement on the generated sequence. Masked Language Model (MLM) is used in their work to supervise the generation of captions.

In contrast to MLM, which is a language model based on discrete tokens embedding prediction, diffusion models based on continuous latent embedding has been thriving in image and audio generation tasks[**?**, **?**, **?**, **?**, **?**]. To the best of our knowledge, there has not been previous work on generating caption embedding based on diffusion language model. Our work aim at employing a model to refine generated token continuously on sequence embedding, and provide empirical insight on useful tricks to improve the generated captions. In particular, we use pretrained CLIP[**?**] model for extracting image and text features, and DistilBert[**?**] model based on diffusion-lm[**?**] for text sequence generation. Our contribution consists of proposing a diffusion based image captioning model Sec.4, and experiments on effectiveness of multiple model design and hyperparameter settings, including classification free guidance Sec.5.1, weight assign in loss function terms Sec.5.2, learning rate Sec.5.3, x_0-prediction Sec.5.4 and feature fusion method Sec.5.5.

2 Related Work

2.1 Autoregressive image captioning

Mao et al. proposed the mRNN model[**?**], which use CNN for image extraction and RNN for text generation. Xu et al. [**?**] applied the LSTM to text generation and experimented on soft and hard attention for early fusion between image feature and text feature. Based on this early fusion method, Lu et al. [**?**] experimented with the late fusion of image and text features, allowing model to attend on either image or text modality in late phase of generation. Wu and Hu [**?**]

experimented on reversing the generated caption, allowing its backend model to refine the former tokens based on later caption tokens. Le et al. [?] used attention model to combine local and global feature from images, so that captions can more accurately identify occluded objects. Similarly, Wei et al. [?] also used image features from both high and low generality, and combined them using cross attention. Their work also involves multi-step refining of the generated text caption. Feng et al. [?] trained image caption in a GAN framework, with a LSTM discriminator reproducing the original image feature from generated text sequence. Similrarly, Guo et al. [?] proposed GAN based method to train model predicting stylized text. Multiple discriminators are used to supervise if generated text captured image related feature, is in the desired style, and is similar to a caption made by human. Kim et al. [?] used variational auto encoder for extracting image information, their model allows multi caption generation by sampling from the learned image feature distribution, thus produce various captions for a single image. He et al. [?] used POS tagging to help the generation of text. The image feature is used as additional input when the model is predicting tokens related to image-specific information, i.e. object, colour, relative position of objects. Mokady, Hertz and Bermano [?] experimented on using pretrained CLIP image feature for sequence generation. The CLIP features are transformed to a sequence of token and used as prefix for a GPT-2 model in generation. Li et al. [?] introduced skipped connections between transformer layers to address the information asymmetry between vision and language modality. The model achieves state of the art performance and strong zero shot ability on various tasks. Nguyen et al. [?] experimented changing the cross attention part of transformer decoder to use both Regional feature from Faster-RCNN[?] and Grid features from Swin transformer[?].

2.2 Non autoregressive image captioning

In contrast, non-autoregressive models benefit from the attention models' ability to pass textural information in both directions during generation. The text generated in former timesteps could adjust based on text in later timesteps, thus is expected to achieve better performance. Gao et al.[?] used BERT[?] as text decoder and employed a 2 step generation method. Based on this work, Partially Non-Autoregressive Image Captioning by Fei [?] and semi Non-Autoregressive Image Captioning by Xu et al. [?] partitioned the generated text in subgroups. Words in the same group are generated non-autoregressively and different groups are generated autoregressively. Our model falls in non-autoregressive category and is most closed to the Masked Non-Autoregressive Image Captioning[?]. The difference is we choose to use diffusion model as the non-autoregressive generation model.

2.3 Diffusion models

Diffusion model aims at training a model that denoise Gaussial noise incrementally to reproduce original features. Ho, Jain and Abbeel[?] proposed the Denoising Diffusion Probabilistic Model (DDPM) to simplify the loss function by only letting models predict the noise in generation steps, and proposed an alternative loss function by removing the weight coefficients. In the following explanation, we refer to diffusion model as DDPM for simplicity. Nichol and Dhariwal[?] proposed several improvements based on DDPM, including setting variance to be learn-able parameters, apply cosine instead of linear noise schedule, and speed up forward process by reducing forward steps. Song, Meng and Ermon[?] experimented on reducing the variance in forward process. The result shows that by reducing variance to 0, the deterministic model achieves higher FID score in image generation on both CIFAR10 and CelebA. DiffusionLM by Li et al. [?] is a recent work on applying continuous diffusion model on text generation. Their work explored various techniques to improve the performance of continuous diffusion model on text generation.

Dhariwal and Nichol [?] proposed classifier guidance for improving generated image FID score. In a classifier guided diffusion model, a classifier model is pretrained to predict noised images' object class. During training, the classifier provides gradient on which direction to optimise the generated image, so that the generate image resembles an object closer to the target class.

To avoid training classifier for guiding model, Jonathan and Tim [?] proposed classifier free guidance model. In classifier free guidance, the difference between outputs of generative model when provided with either guided and unguided context information is used as implicit guidance. By using classifier free diffusion model as text-to-image generator, DALL-E2[?], GLIDE[?] and High-Resolution Image Synthesis With Latent Diffusion Models[?] model achieves significant image generation performance. In particular, DALL-E2 use CLIP model for extracting feature from text, predict the corresponding image CLIP feature through prior network, then use predicted image CLIP feature for final image generation. The model achieves significant novelty in generated images and also inspired us to train a image-to-text model with diffusion model in generation step.

2.4 CLIP model

CLIP model is a contrastive-learning-based model trained on WebImageText dataset. The WebImageText consists of 400 million image-text pairs collected from publicly available source on Internet. CLIP model demonstrates strong zero shot performance in their evaluation on ImageNet[?] dataset. Further work on CLIP model also shows its transferability to other tasks, including image segmentation[?, ?], objection detection[?, ?], action recognition[?] and video-text retrieval[?].

3 Background

3.1 Diffusion models

The training of denoise diffusion probabilistic model involves generation of noised samples (forward process), and model denoising Gaussian noise back to the original feature (backward process). Let x_t represent the intermediate representation between the diffusion generation stages. In our context, x_t is the sequence of word embedding of the caption. The forward process incrementally add noise to the ground truth embedding x_0 to generates T noised features $[x_1, ...x_T]$. Each x_t at step t is sampled from probability distribution $q(x_t|x_{t-1}) = N(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t I)$ and the final step feature x_T is aiming at approximate a Gaussian noise. From reparameterization trick, the x_t at any step t could be directly sampled from x_0: $x_t = \sqrt{\bar{\alpha}_t}x_0 + (1-\bar{\alpha}_t)\epsilon$, where $\bar{\alpha}_t = \prod_{s=1}^{t}(1-\beta_s)$ and ϵ follows a Multivariate Gaussian Normal distribution. The backward process involves training model with parameter θ to denoise the samples generated in the forward process. The training objective is to minimize the negative log-likelihood of generating x_0 from arbitrary Gaussian noise x_T as generated by q, that is to minimize

$$E_q[-\log(p_\theta(x_0))] = E_q[-\log(\int p_\theta(x_{0:T})d(x_{1:T}))]$$

By optimizing the variational lower bound of $p_\theta(x_0)$ instead, and modeling the backward process as a Markov Process gives:

$$\begin{aligned} E_q[-\log(p_\theta(x_0))] &\le E_q[\log(\frac{p_\theta(x_{0:T})}{q(x_{1:T}|x_0)})] \\ &= E_q[\log(p(x_T)) + \sum_{t=1}^{T}\log(\frac{p_\theta(x_{t-1}|x_t)}{q(x_t|x_{t-1})})] \end{aligned}$$

where $p_\theta(x_{t-1}|x_t) = N(x_{t-1}; \mu_\theta(x_t,t), \sum_\theta(x_t,t))$ and $\mu_\theta(x_t,t)$ is model's prediction on mean of x_{t-1} conditioned on x_t. From DDPM[?], expanding and reweighting each term of the negative log-likelihood gives a concise loss function

$$L_{simple} = \sum_{t=1}^{T} E_{q(x_t|x_0)}\|\mu_\theta(x_t,t) - \hat{\mu}(x_t,x_0)\|^2$$

where $\hat{\mu}(x_t,x_0)$ is the mean of posterior $q(x_{t-1}|x_t,x_0)$.

Due to the large generation step number (T = 1000 as proposed in [?]), and the generation step being autoregressive on the denoised feature in the previous step, the reverse diffusion is significantly slower than the other generative models[?, ?]. Multiple strategies were proposed to accelerate the generation process. Improved DDPM proposed by Nichol and Dhariwal[?] uses a subset of generation steps $\{s_0, ..., s_N | s_t < s_{t-1}\} \in (0, T]$. Model is trained to predict $x_{s_{t-1}}$ based on x_{s_t}. DiffusionLM proposed by Li et al.[?] is trained directly to predict the x_0 instead of an denoised intermediate steps x_{t-1}.

In addition, we add a rounding term and a x_1 restoring loss term to the loss function as propose by Li et al.[?]. x_1 restoring loss $\|\mu_\theta(x_1,1) - \hat{\mu}(x_1,x_0)\|^2$ evaluates the performance of model on restoring x_1. For simplicity, in the following explanation and experiments, we evaluate the x_1 restoring loss together with L_{simple} and refer to their sum as L'_{simple}. Rounding term L_R is parameterized by $E_q[-\log(p_\theta(w|\hat{x}))] = E_q[-\log(\prod_{i=1}^{l} p(w_i|\hat{x}_i))]$, where l represents the generated sequence length, w represent the ground truth sentence and $\hat{x}$ is the predicted sequence embedding from the input x_t. $p_\theta(w_i|\hat{x}_i)$ follows the Softmax distribution. In our experiments, we find the relative importance between rounding term L_R and restoring embedding loss L'_{simple} significantly influence the model's performance. The relative importance is addressed by the hyperparameter λ as rounding term coefficient and discussed in detail Sec.5.2. Based on the above 2 modifications, our training objective is as follow:

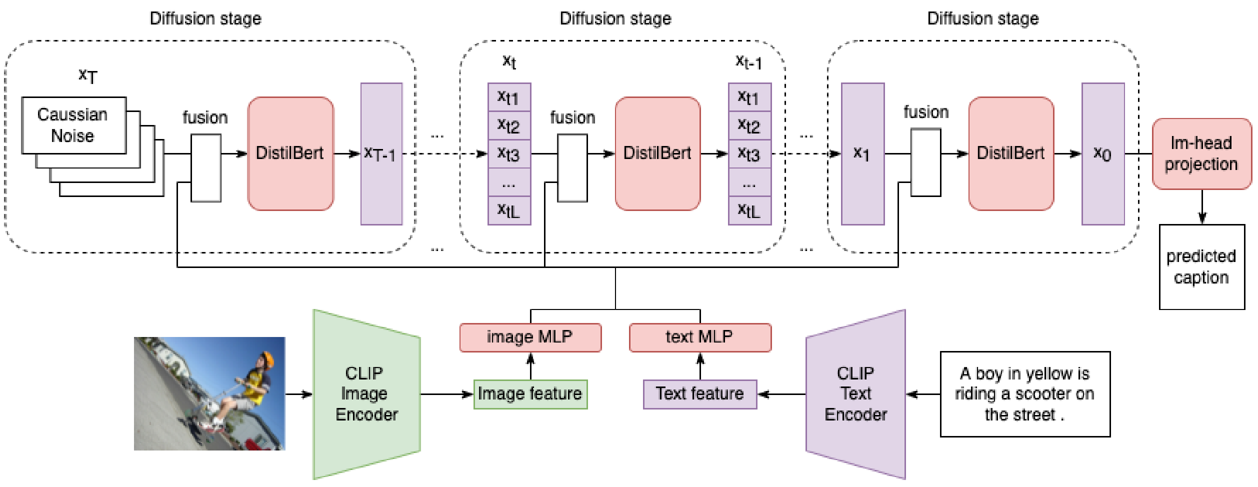

Figure 1: CLIP-DiffuseLM model

$$L_{diffuseLM} = L'_{simple} + \lambda L_R = \sum_{t=1}^{N} E_{q(x_{s_t}|x_0)}[||\mu_\theta(x_{s_t}, s_t) - \hat{\mu}(x_{s_t}, x_0)||^2 - \lambda \log(p_\theta(w|\hat{x}))]$$

4 CLIP-DiffusionLM

Our CLIP-DiffusionLM model (Figure 1) consists of 2 parts. A pretrained CLIP (ViT-B/32) model[?] is used for image and ground truth caption label feature extraction, and a DistilBert-base-uncased model[?] for performing denoising diffusion in each diffusion stage conditioned on CLIP image feature.

In a forward process, the CLIP model first extract image features using a ViT-Base/32 architecture[?]. In this model, source RGB image is partitioned into 32 patches and prefixed with a <CLS> token to form the input of the 12 ViT layer model. Output feature at the <CLS> token position is treated as the global feature for the whole image. For classification free experiments Sec.5.1, CLIP text feature is also extracted as guidance context to improve the performance. Model used in CLIP caption feature extraction is a modified transformer encoder model [?, ?]. Activation at the highest transformer layer in <EOS> token is used as the global caption feature. In our following experiments, both CLIP text and image branches parameters are not optimized in the backward propagation.

The generation of caption for a given image consists of a sequence of diffusion stages conditioned on CLIP feature extracted. In each stage, CLIP-DiffusionLM receives input of $L \times D_{word}$ dimension vectors x_t as embedding of a L length caption, and two D_{CLIP} dimension vector for CLIP text and image features. Input caption embedding is either the output embedding x_t in the previous diffusion stages, or the Gaussian noise x_T in case of the first diffusion stages. Each of CLIP features is projected by a MLP layer to D_{word} space, before fuse Sec.5.5 with the sequence embedding x_t. The fused sequence is the input of the DistilBert model, and output of the DistilBert model is the output of one single diffusion stage, which consists of the prediction for x_{t-1}. After the final diffusion stage, the model's prediction for x_0 is linear projected by weight matrix `lm-head` and taken the Softmax value to get the probability of the predicted word in each L length caption sequence position. Finally, each position's word with maximum probability are concatenated to form the output text caption sequence. Unless otherwise specified, both the embedding layer for extracting ground truth caption embedding and the `lm-head` use pretrained DistilBert model embedding layer parameter and are not optimized in model training.

5 Experiments

We train our baseline model on Flickr8k[?] dataset, which consists of 8k images and 5 captions for each image. After adjusting learning rate Sec.5.3 and rounding coefficient Sec.5.2, we train our final model on combined dataset of Flickr30k[?] and Flickr8k[?]. The combined dataset, which we refer to as Flickr30+8k dataset, consists of 38k images and 190k captions. Both models early stop when validation loss is higher than training loss, and result in 15 epochs training used for both models.

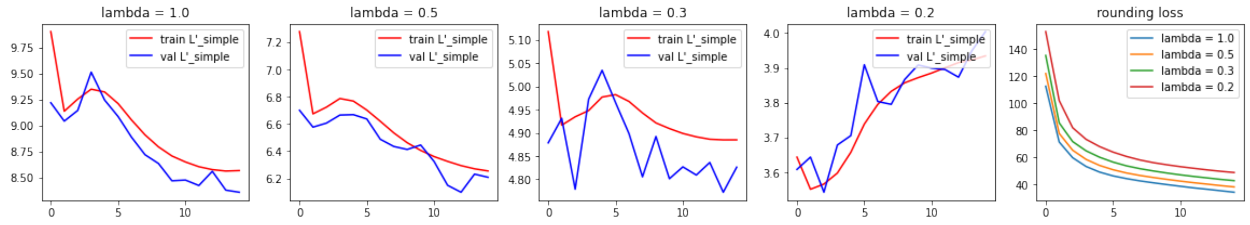

Figure 2: constant L_R coefficient loss

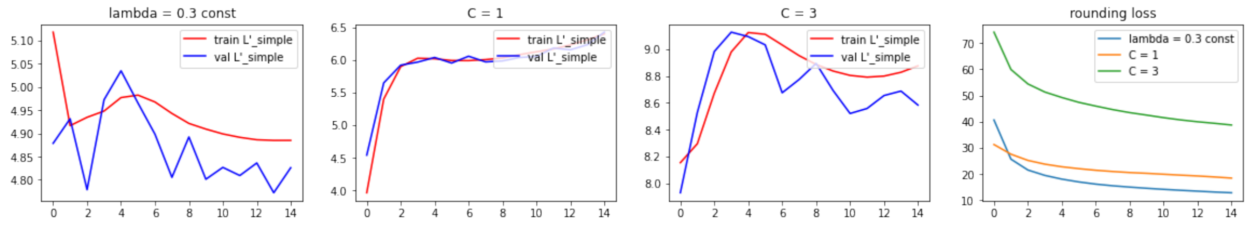

Figure 3: dynamic L_R coefficient scheduling loss

The baseline model uses configuration of batch size of 8, $\lambda = 0.3$, $lr = 5e-5$, concatenation fusion and non-classification-free guidance. It takes around 5 hours to train the baseline model for 15 epochs using AdamW optimizer on a single Nvidia A30 GPU. During training, a subset of 100 x_t noise embeddings are randomly sampled from the total 1000 intermediate representations. The model is trained to predict mean x_0 directly as suggested in diffusion-lm[**?**]. In final inference step, the baseline model refines its output autoregressively for 5 iterations and all but the first element from every consecutive group of repeated words are removed to improve the BLEU-4 performance. The final model uses same configuration as the baseline model, apart from decaying learning rate linearly from $1e-4$ to $5e-5$ and setting λ to 0.5. The final model takes around 18 hours to finish 15 epochs training on Flickr30+8k dataset, and achieves ... BLEU score on validation set of Flickr30+8k dataset.

5.1 classification free guidance

The effectiveness of applying classification free guidance is examined on our model. The guidance provided is the CLIP text feature of ground truth label caption. Table 1. shows the comparison between the baseline and classification free guidance trained model. In contrast to previous success of applying classification free guidance, our model fail to improve significantly than none classification guided baseline. We further test the 2 classification-free-guided models' performance by evaluating BLEU score on validation set. The model trained using classifier free guidance hardly outperform the simpler baseline model, with the parameter suggested by Jonathan and Tim [**?**] ($w = 0.3$, $p_{uncond} = 0.2$) even result in slight decrease in BLEU-4 score. As a result, classification free guidance is not used in our final model.

w	p_{uncond}	L'_{simple}	L_R	BLEU-4
no classifier-free guidance	-	4.89	12.87	0.1549
0.3	0.2	4.92	13.49	0.1539
1.0	0.2	4.89	13.06	0.1558

Table 1: classifier free guidance comparison

λ	BLEU-4
1.0	0.1599
0.5	0.1613
0.3	0.1549
0.2	0.1550

Table 2: constant L_R coefficient BLEU

5.2 Relative importance between Rounding term L_R and Embedding-restoring loss term L_{simple}

In this session we explain our choice of rounding coefficient hyperparameter λ. We introduce the coefficient in the expectancy of controlling model to optimize primarily based on either the L'_{simple} or the L_R loss. $\lambda \in \{1.0, 0.5, 0.3, 0.2\}$ are experimented on as shown in Figure 2. When λ decrease, the model shows a decreasing trend in its L'_{simple} loss (from 8.56 to 4.88) at the cost of increasing L_R loss (from 34.4 to 48.9) as expected. In addition, setting λ to lower than 0.3 shows increase in L'_{simple} loss, which means model is increasing the probability of output the ground truth

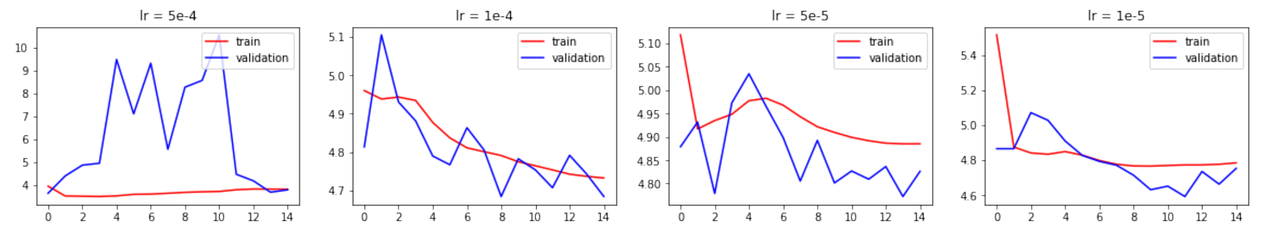

Figure 4: constant learning rate L'_{simple} loss

sequence while not captioning the similarity between the predicted word and ground truth words. Though we find the final L'_{simple} of $\lambda = 0.2$ (3.93) lower that of $\lambda = 0.3$ (4.88), BLEU score of the model shows the opposite in Table 2.

Furthermore, we explore if a dynamic λ scheduling method help the model adjust λ based on ratio between L'_{simple} and L_R. The dynamic scheduling method update λ value after each gradient descend step as $\lambda = L'_{simple}/L_R * C$, where C is a hyperparameter defining the relative weight kept between L'_{simple} and L_R throughout the training. To choose a reasonable C value, we examine the $\lambda = 0.3$ experiment final epoch's $\frac{L'_{simple}}{L_R}$ value, which is around 2.80. Based on this observation, we take $C = 1$ and expect both L'_{simple} and L_R loss decrease at the same rate and reach $\lambda \approx 0.3$ in the final epoch. Same reasoning applies to the choice of $C = 3$, which aims at reaching $\lambda \approx 1.0$ in the final epoch.

We plot the dynamic scheduled loss in Figure 3. The relative ratio between L'_{simple} L_R does not follow the dynamic ratio defined as expected. A possible explanation is that the loss is taken before gradient decent is performed, and evaluation of loss is based on the next batch sample, result in the logged $\frac{L'_{simple}}{L_R}$ value not following the expected relative loss ratio. In both experiments, the model shows worse performance with dynamic λ scheduling enabled. In our final model, the non-dynamic scheduled $\lambda = 0.5$ is used.

lr	BLEU-4
5e-5	0.1549
1e-4	0.1699
log schedule	0.1648
linear schedule	0.1876
cosine annealing	0.1848

Table 3: learning rate BLEU-4 scores

fusion method	Flickr8k	Flickr30+8k	final model
addition	0.1033	0.1948	-
concatenation	0.1549	0.2337	?

Table 4: fusion method BLEU-4 scores

5.3 learning rate

We experiment on constant learning rate of $lr \in \{5e-4, 1e-4, 5e-5, 1e-5\}$ (Figure 4 and Table 3). Among these learning rate choices, $lr = 1e-4$ achieve the highest BLEU-4 score (0.1699), and the lowest L'_{simple} loss (4.69). To our surprise, model fail to converge to both a lower L'_{simple} and L_R loss before over-fitting when learning rate is set below $1e-4$. A probable cause is training with a low learning rate prevents the model exploring the parameter space and converging to a local minimum.

To experiment on if model can refine on the result achieved by larger learning rate, we apply linear, log and cosine annealing scheduling between the 2 best-performing learning rate values ($1e-4$ and $5e-5$). Among the 3 scheduling method, linear scheduling achieve the highest BLEU-4 score (0.1876), followed by cosine annealing (0.1848). The log scheduling, which decrease learning rate significantly more rapidly to a low value, fail to achieve comparable performance to other 2 scheduling method. This failed experiment also supports our guessing above. Linear decay of learning rate from $1e-4$ to $5e-5$ is chosen as our final model configuration.

5.4 x_0 prediction or x_{s_t-n} prediction

In DiffusionLM proposed by Li et al.[**?**], model is trained to predict x_0. In this experiment, we examine the difference between x_0-prediction and x_{s_t-n}-prediction, where x_{s_t-n}-prediction trains the model to predict a constant $n = 100$ steps after the sampled latent embedding. In our experiments, the x_0-prediction method shows better BLEU-4 score (0.1549) and $L_{diffuseLM}$ loss (17.8) performance compared with $x_{s_{t-1}}$-prediction (0.1474 BLEU-4 score and 22.6

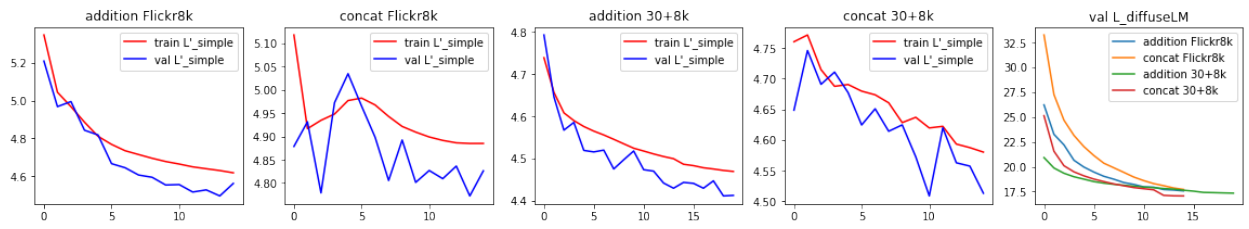

Figure 5: concatenation or addition fusion L'_{simple} loss

$L_{diffuseLM}$ loss). In addition, by following the x_0-prediction method, our model converges to a reasonable output in 5 diffusion stages. This inference step number is significantly less than that of an autoregressive encoder-decoder structure, which involves generation step number proportional to the output sequence length.

5.5 fusion: concatenation or element-wise addition

We experiment with 2 fusion methods: concatenation or element-wise addition. Concatenation append the CLIP features as additional tokens to the end of the caption embedding x_t. An additional segment embedding layer is used to distinguish CLIP feature tokens with caption embedding. In contrast, element-wise addition method use CLIP image feature as position embedding. CLIP image feature is element-wise added to caption word embedding in each timestep. In case of Guidance free training, the CLIP text feature is also element-wise added. In later explanation, element-wise addition is referred to as addition method for simplicity.

We conduct experiments by running both concatenation and addition fusion. Figure 5 and Table 4 show their performance on Flickr8k and Flickr 30+8k dataset. The addition method experience less violent fluctuation in L'_{simple} loss, and converge to a lower loss before over-fitting. Extending the training dataset to use Flickr30+8k shows similar result. However, BLEU-4 scores of addition method (0.1033 on Flickr8k and 0.1948 on Flickr30+8k) are significantly lower than that of concatenation method (0.1549 on Flickr8k and 0.2337 on Flickr30+8k) in both datasets' training. As a result, concatenation method is chosen despite its higher converging loss and instability.

By modify our baseline model to train with linear learning rate decay and $\lambda = 0.5$ rounding coefficient, the final model achieved ... on the Flickr30+8k dataset.

6 Conclusion

We present the application of diffusion in image caption task, and prove its validity on Flickr8k and Flickr30k dataset. Particularly, we identify the importance of rounding term in loss function to help model converge, and introduce the adaptive ratio adjustment to balance its importance with other terms. There are various possible improvements to the model and training process:

- In various cases, the model fail to identify the correct object colour. For example, wrongly tag the object colour to be the background colour. We suggest 1) performing image data argumentation, which has been proven to improve the performance for VLP tasks [**?**], and 2) investigate attention graph of the wrongly tagged object in image for possible causes.
- The output text sequence from the model contains apparent grammar mistakes occasionally (e.g. missing subject and repeated words). Providing the model additional guidance on the output text grammar might help reduce the grammar mistakes.
- We limit the sequence generated by DistilBert to be under 16 words due to the computation resource limit. As a result, the BLEU score suffers from the Brevity Penalty in BLEU metrics. Training model with relaxed length constrain might further improve the BLEU score.
- In various cases models used in experiments with lower loss does not have higher BLEU score, suggesting better loss function is required to define model's performance.

We believe that analysing and improving based on the above observations, CLIP-DiffusionLM as text generation step will achieve comparable or better performance than auto-regressive models in image captioning tasks.

Acknowledgments

We thank Mu Li and Yi Zhu for sharing their insight in various models in vision and NLP field publicly online, Boyang Gu for providing advice in early stage of the research. The computation resource was supported by Imperial College London.