STEP-UNROLLED DENOISING AUTOENCODERS FOR TEXT GENERATION

Anonymous authors
Paper under double-blind review

ABSTRACT

In this paper we propose a new generative model of text, *Step-unrolled Denoising Autoencoder* (SUNDAE), that does not rely on autoregressive models. Similarly to denoising diffusion techniques, SUNDAE is repeatedly applied on a sequence of tokens, starting from random inputs and improving it each time until convergence. We present a simple new improvement operator that converges in fewer iterations than diffusion methods, while qualitatively producing better samples on natural language datasets. SUNDAE achieves state-of-the-art results (among non-autoregressive methods) on the WMT'14 English-to-German translation task and good qualitative results on unconditional language modeling on the Colossal Cleaned Common Crawl dataset and a dataset of Python code from GitHub. The non-autoregressive nature of SUNDAE opens up the possibilities beyond prompted generation, by filling in arbitrary templates.

1 INTRODUCTION

Autoregressive (AR) models have shown excellent results in generating text (e.g., GPT-3, Brown et al., 2020). However, while their training scales very well, sampling is prohibitively slow for many practical applications. Moreover, there are limitations to the kinds of conditioning AR models can seamlessly handle: the left-to-right restriction makes it hard to "fill in the gaps" in a partially written text draft. Even more importantly, this prohibits iterative refinement of complete text drafts to make them more self-consistent, which is a common task for human writers. Finally, AR models require network architectures to be causal, severely limiting the kinds of neural network architectures that can be used for text-modeling. All of these motivated the machine learning community to make extensive efforts to propose alternatives to AR models.

Perhaps one of the first text generation tasks for which it has been shown that an AR approach can have serious rivals was machine translation (MT): methods like CMLM (Ghazvininejad et al., 2019) and DisCo (Kasai et al., 2020) show promising results and their decoding speed is excellent compared to AR. However, while their performance is competitive, they are still behind the AR benchmark and actually require distillation of a larger AR model — without which, performance drops considerably.

Non-AR methods have proved hard to apply to the general unconditional language modeling (LM) task. When there is no conditioning, the multi-modality problem becomes paramount, as shown by Gu et al. (2017), which likely makes it problematic to use methods like CMLM and DisCo because their decoding mechanism is deterministic and does not model uncertainty. Yet, recently the community has seen promising results from non-AR models like Multinomial Diffusion (Hoogeboom et al., 2021) and D3PM (Austin et al., 2021). These methods optimize a lower bound (ELBO) on likelihoods and have shown negative log-likelihood (NLL) results approaching AR models on several benchmarks like text8 (Mahoney, 2011) and LM1B (Chelba et al., 2013). However, a major gap in NLL persists, and samples from those models lack coherence.

In this paper we propose a novel non-autoregressive method which shows state-of-the-art results in machine translation on WMT'14 EN→DE raw data (without distillation from AR) amongst non-AR methods and good qualitative results on unconditional language modeling on the Colossal Clean Common Crawl (C4) dataset (Raffel et al., 2019) and a dataset of Python code from GitHub. Our model operates as a time-homogeneous Markov Chain similar to that of Lee et al. (2018): condi-

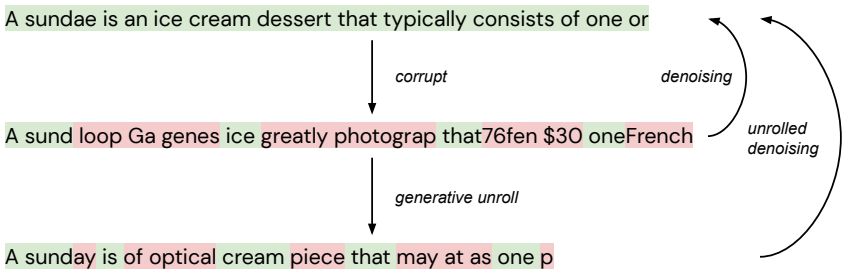


Figure 1: The difference between denoising and *unrolled denoising*. Original text (top) is randomly corrupted, producing a text (middle) where some tokens are original (green) and others are corrupted (red). This text is denoised by sampling from the generative model to produce another noisy text (bottom). While usual denoising autoencoders only learn a mapping from middle text to top text, *Step-unrolled Denoising Autoencoder* learns a mapping from bottom to top (including middle as a special case of zero unroll steps). This has an intuitive meaning: during generation time, the network will mostly (right after the first step) encounter texts like the bottom one, not like the middle one — so unrolls prepare the model during training to what it will get as the input at generation time.

tioned on the corrupted data, it tries to approximate the original uncorrupted samples by a per-token factorised distribution. During generation, we unroll the chain by sampling from the transition distribution and feeding samples back into the input. We propose a new training mechanism called *unrolled denoising*, which uses unrolls of the chain during training as well, and which we empirically show to be crucial for practical performance of the algorithm in ablation studies. This feature motivates the name for the model, *Step-unrolled Denoising Autoencoder* (SUNDAE). The difference between usual denoising and *unrolled denoising* is illustrated in Figure 1.

To summarize our contributions:

- We present *SUNDAE*, a new generative model of text that unrolls the denoising process during training.
- SUNDAE achieves state-of-the-art results on WMT'14 English-to-German translation task among non-AR methods.
- We demonstrate good qualitative results for unconditional generation and inpainting on Colossal Clean Common Crawl dataset and a dataset of Python code from GitHub.
- We carefully ablate and analyze the properties of the proposed method, and show that unrolls during training are crucial for the model's performance.

2 METHOD

We approach the problem of generative modelling of discrete sequences by bringing together the framework of denoising autoencoders and Markov chain models. In this section, we first discuss the definition of the generative model, and then describe the training process, which includes our main contribution, *unrolled denoising*.

For a fixed prior distribution p_0 on some space X consider a process $\mathbf{x}_t \sim f_\theta(\cdot|\mathbf{x}_{t-1})$, where f_θ is a parametric transition function. Then $\{\mathbf{x}_t\}_t$ is a time-homogeneous Markov chain, and transition t-steps ahead has the following form[1]

$$p_t(\boldsymbol{x}_t|\boldsymbol{x}_0) = \sum_{\boldsymbol{x}_1,\boldsymbol{x}_2,\ldots,\boldsymbol{x}_{t-1}\in X} \prod_{s=1}^{t} f_\theta(\boldsymbol{x}_s|\boldsymbol{x}_{s-1}). \tag{1}$$

For a fixed number of steps T, the prior p_0 and decoder p_T together determine our model distribution $p_T(\mathbf{x}_T) = p_T(\mathbf{x}_T|\mathbf{x}_0)p_0(\mathbf{x}_0)$.

We assume $X = \{1, \ldots, v\}^N$ to be the space of sequences of length N with values at all positions coming from a vocabulary of size v, and the prior p_0 to be uniform over X. Let p_{data} be a distribution

[1] for formal derivation, see Section A in supplementary material.

of the data and assume that f_θ induces a conditional probability distribution $f_\theta(\cdot|\boldsymbol{x}')$ that can be factorised into a conditionally-indepedent product as follows:

$$f_\theta(\boldsymbol{x}|\boldsymbol{x}') = f_\theta^{(1)}(\boldsymbol{x}^{(1)}|\boldsymbol{x}')f_\theta^{(2)}(\boldsymbol{x}^{(2)}|\boldsymbol{x}')\cdots f_\theta^{(N)}(\boldsymbol{x}^{(N)}|\boldsymbol{x}'), \qquad \text{for } \boldsymbol{x}, \boldsymbol{x}' \in X. \tag{2}$$

Note that although for $t = 1$ the distribution $p_1(\cdot|\boldsymbol{x}_0)$ is limited by product structure (Eq. 2), the subsequent p_ts are not restricted in the same way, and after each step they belong to potentially more and more expressive family.

2.1 Training with Unrolled Denoising

Due to intractability of the likelihood of the model p_T as well as computational requirements of optimising the whole chain $\{p_t\}_t$, we propose an efficient two-step training method which we term *unrolled denosing* and visualise in Fig. 1. We consider a smaller number of steps than T that we intend to use at sampling, but to compensate that, we unroll the chain starting from corrupted data samples, rather than the prior p_0. This way, the model learns to denoise the samples it is likely to encounter during the full unroll used at sample time. While using a single step would resemble the training strategy of BERT (Devlin et al., 2018), using at least two steps is essential for the performance of our model.

Consider a *corruption* distribution $q(\boldsymbol{x}^c|\mathbf{x})$ that replaces a random proportion of tokens in a sentence $\mathbf{x}$ with ones randomly sampled from a uniform distribution. Our objective is given by $L^{(1:2)} = \frac{1}{2}\left(L^{(1)} + L^{(2)}\right)$ where

$$L^{(t)}(\theta) := -\mathbb{E}_{\substack{\mathbf{x}\sim p_{\text{data}} \\ \mathbf{x}_0\sim q(\cdot|\mathbf{x}) \\ \mathbf{x}_1\sim f_\theta(\cdot|\mathbf{x}_0) \\ \cdots \\ \mathbf{x}_{t-1}\sim f_\theta(\cdot|\mathbf{x}_{t-2})}}[\log f_\theta(\mathbf{x}|\mathbf{x}_{t-1})], \tag{3}$$

i.e. the reconstruction loss of the chain after t steps, starting from a corrupted sample $\mathbf{x}_0$. In supplementary material (Section A.1) we show that $L^{(t)}$ is an upper bound on the actual negative log-likelihood from the model distribution p_t:

$$\widetilde{L^{(t)}}(\theta) := -\mathbb{E}_{\substack{\mathbf{x}\sim p_{\text{data}} \\ \mathbf{x}^c\sim q(\cdot|\mathbf{x})}}[\log p_t(\mathbf{x}|\mathbf{x}^c)]. \tag{4}$$

Note that $\widetilde{L^{(T)}}$ is the actual term we would like to minimise if p_T was tractable, and in fact is closely related to evidence lower bound (ELBO) via a fixed encoder $q(\mathbf{x}^c|\mathbf{x})$ (Section A.2). However, due to inherent non-differentiable nature of Markov chain models, optimisation of $\widetilde{L^{(T)}}$ would not only be costly, but also limit the gradient flow to the last step of the chain. We found instead that optimising several steps together is sufficient for obtaining good results. Averaging more $L^{(t)}$ terms can lead to minor improvements in performance (See Fig. 3d in supplementary material), but it considerably slows down the training speed. We discuss these and other loss variants further in Section 3.1.

Since function f_θ is modelled as a neural network, the logarithm under expectation on the right hand side of Eq. 3 is obtained from predicted logits, thus we term $L^{(1)}(\theta)$ and $L^{(2)}(\theta)$ *logits* loss and *unrolled logits* loss, respectively.

Corruption function. To corrupt texts, we first sample a proportion of tokens uniformly from $[0, 1]$, randomly select positions according to this proportion and then change tokens at those positions to random tokens sampled uniformly from the vocabulary. This way, we want to ensure that samples seen at various steps during generative unroll (including the first one, sampled from the uniform prior) are well-represented in corruptions during training. The detailed construction as well as relation to the forward discrete diffusion process (Hoogeboom et al., 2021) is presented in Section A.3 in supplementary material .

2.2 Sampling

At sampling time we follow the structure of the Markov process and sample sequentially $\boldsymbol{x}_t \sim f_\theta(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})$ for some fixed number of steps T, beginning from a random sequence $\boldsymbol{x}_0$ (potentially starting with a prompt or a template for conditional generation). To control the speed of convergence, we propose three improved strategies which allow to use of a much smaller number of steps:

- **Low-temperature sampling**. For temperature τ and trained network f_θ we consider modified function f_θ^τ such that $\log f_\theta^\tau(\cdot|\boldsymbol{x}) \propto \frac{1}{\tau} \log f_\theta(\cdot|\boldsymbol{x})$. We found that sampling with temperatures lower than 1 leads to much faster convergence than with the original logits, and allowed generation of high-quality samples in a matter of 10-32 steps.
- **Argmax-unrolled decoding**. At the limit $\tau \to 0$ sampling with temperature reduces to deterministic argmax decoding where the most probable token is chosen at each step. Relatedly to our *unrolled logits* loss, we modify this strategy by resampling the low-certainty tokens in accordance with *unrolled logits*. This heuristic allowed further improvements on the sampling speed while maintaining high quality of the samples. We discuss it in detail in Section 3.1.
- **Updating fewer tokens**. In tasks where diversity is paramount (like unconditional text generation), we found that updating a random subset of tokens at each decoding step leads to faster convergence. This likely happens because independent sampling of all tokens might create uncoordinated changes which could take some time to fix in the subsequent steps. We use this strategy in Section 3.2.

3 EXPERIMENTS

3.1 MACHINE TRANSLATION

We first evaluate SUNDAE on Machine Translation (MT) benchmarks. We compare SUNDAE against AR and non-AR models in terms of the translation quality using BLEU (Papineni et al., 2002) as the metric. We demonstrate that without using techniques like sequence-level knowledge distillation (Kim & Rush, 2016), SUNDAE can perform almost as well as the AR baseline and outperform all other models that do not rely on AR models.

Target Length Prediction. Unlike their AR counterparts, non-AR models do not explicitly learn to predict an end of sequence, but instead have been shown to benefit from auxiliary target length prediction (Lee et al., 2018; Ghazvininejad et al., 2019; Kasai et al., 2020). A common approach is to treat it as a classification task, predicting either the exact length or the difference between the source and target lengths. The decoding process has to commit to the predicted length, and often multiple length beams are used to get the best translation results. In our models, we use a separate network that receives source encodings as input and predicts corresponding target length. Target length embedding vector is prepended to source embeddings so that decoder can attend to it. During training, we use the reference target length, while at sampling, a predicted one, although the model does not need to fully commit to the predicted length like some previous non-AR models (Saharia et al., 2020). Note that the length classification loss does not affect the encoder parameters. Our models are also trained to predict the padding tokens at the end of the text. For more details, see Section B in supplementary material .

Experiment Settings. We conduct experiments on WMT'14 parallel corpora using EN↔DE (4.5M pairs) and EN→FR (36M pairs) translation tasks. The raw texts are encoded using BPE (Sennrich et al., 2015) as the subword units, and we use the same preprocessed data as in Vaswani et al. (2017) for fair comparisons. We evaluate the performance by measuring BLEU (Papineni et al., 2002; Post, 2018) on the test split of each translation task[2]. We use the encoder-decoder Transformer architecture for MT (Vaswani et al., 2017), but remove the causality masking in the decoder. There are 6 attention layers for both encoder and decoder, 8 attentions heads, 512 model size and 2048 hidden size. The total number of parameters is 63M including the target length prediction module described in 3.1. We use dropout ($p = 0.1$) and label smoothing ($\epsilon = 0.1$) during training. The training batch size is 4096, and we use Adam (Kingma & Ba, 2014) with $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-6}$ and weight decay of 0.1 (Loshchilov & Hutter, 2017). We warm up the the learning-rate from 10^{-7} to 10^{-4} in the first 5K steps and decay it to 10^{-5} using cosine annealing (Loshchilov & Hutter, 2016). Our models were trained for 10^6 steps using 16 TPU accelerators using bf16 precision. We crop sequences that have more than 128 tokens (this occurs less than 0.2% in training data). Finally, we average the last 10 checkpoints to obtain a single model for the evaluation.

Decoding. We decode translations from SUNDAE using two different types of heuristic decoding methods in MT experiments. The decoding process always begins with an array of random integers

[2]Due to the limited space, we report BLEU using SacreBLEU in Section F in supplementary material .

Model	Steps (T)	Raw BLEU		AR-distilled BLEU	
		EN→DE	DE→EN	EN→DE	DE→EN
AR Models					
Transformer Base (65M) (Vaswani et al., 2017) ($n=4$)	-	27.3	31.78*	-	-
Non-AR Models					
NAT (Gu et al., 2017) ($n=100$)	1	-	-	19.17	23.20
LVM-DAE (Lee et al., 2018)	-	-	-	21.54	25.43
NAT-REG (Wang et al., 2019) ($n=9$)	1	-	-	24.61	28.90
LV-NAR (Shu et al., 2020) ($n=50$)	1	11.8	-	25.10	-
NART w/ hints (Li et al., 2019)($n=9$)	1	-	-	25.20	29.52
FlowSeq (Ma et al., 2019) ($n=30$)	1	23.64	28.29	25.31	30.68
ReorderNAT (Ran et al., 2019)	1	-	-	26.49	31.13
NART (Sun et al., 2019) ($n=19$)	1	-	-	26.80	30.04
CMLM (Ghazvininejad et al., 2019) + Mask-Predict ($n=5$)	4	22.25	-	25.94	29.90
CMLM (Ghazvininejad et al., 2019) + Mask-Predict ($n=5$)	10	24.61	-	27.03	30.53
DisCo (Kasai et al., 2020) + Mask-Predict ($n=5$)	4	-	-	25.83	30.15
DisCo (Kasai et al., 2020) + Mask-Predict ($n=5$)	10	-	-	27.06	30.89
DisCo (Kasai et al., 2020) + Easy-First ($n=5$)	4-5†	24.8	-	27.34	31.31
NARLVM (Lee et al., 2020) ($n=25$)	4	-	-	27.40	-
JM-NAT (Guo et al., 2020) ($n=3$)	4	-	-	27.05	31.51
JM-NAT (Guo et al., 2020) ($n=3$)	10	-	-	27.69	**32.54**
SMART (Ghazvininejad et al., 2020) ($n=5$)	4	-	-	27.03	30.87
SMART (Ghazvininejad et al., 2020) ($n=5$)	10	-	-	27.65	31.27
Imputer (Saharia et al., 2020) ($n=1$)	4	24.70	-	27.90	30.90
Imputer (Saharia et al., 2020) ($n=1$)	8	25.20	-	27.90	31.10
SUNDAE (ours 63M)					
Deterministic ($n=16$)	4	25.01	29.53	28.00	31.82
Deterministic ($n=16$)	10	25.54	30.11	28.01	31.84
Stochastic ($n=16$)	4	23.05	28.13	27.78	31.42
Stochastic ($n=16$)	10	26.25	**30.80**	**28.11**	31.87
Stochastic ($n=16$)	32	**26.57**	30.74	28.02	31.92

Table 1: Test BLEU scores of AR and non-AR systems on the WMT'14 English-to-German (EN→DE) and German-to-English (DE→EN) translation tasks. The number of reranked candidates is denoted n. SUNDAE does not use an AR model for re-ranking. We highlight BLEU scores of the best non-AR systems in bold font. All the entries are ordered based on the EN→DE BLEU score. *indicates the results of the Transformer baselines implemented by the authors of DisCo. †DisCo has a dynamic inference mode for which we report the average number of steps.

sampled from the discrete uniform prior $\boldsymbol{y}_0 \sim p_0$, while the encoder input - the source sentence - remains the same throughout the process. The first decoding method is the *low-temperature sampling*, where we iteratively sample $\boldsymbol{y}_t \sim f_\theta^\tau(\cdot|\boldsymbol{y}_{t-1})$ for T iterations (with the decoder's output logits divided by τ, see Section 2.2), with $T \leq 128$ and $\tau \in [0.1, 0.6]$ determined based on the validation performance. The second method is *argmax-unrolled decoding*, which requires a smaller number of iterations compared to its stochastic counterpart. In argmax-unrolled decoding, we first compute the logits $\lambda_1 = f_\theta(\cdot|\boldsymbol{y}_0)$ of the initial sample array $\boldsymbol{y}_0$ and obtain the samples $\boldsymbol{y}_1$, pass a tuple $(\boldsymbol{y}_1, \lambda_1)$ to the next iteration. At each step $t \geq 2$, the method finds top-ρ share of the tokens that are sorted by the log-probability in descending order (i.e. uncertain tokens) from λ_{t-1}, where $\rho \in [0.1, 0.6]$ is another hyperparameter searched for in the validation stage. We compute *unrolled logits* for those top-ρ tokens and then apply $\arg\max$ to obtain *unrolled samples*. For the rest of the tokens of $\boldsymbol{y}_{t-1}$ we compute logits once and obtain $\lambda_t = f_\theta(\cdot|\boldsymbol{y}_{t-1})$ and then apply $\arg\max$ to obtain *samples*. Finally, we combine both *unrolled samples* and *samples* to obtain $\boldsymbol{y}_t$, and the tuple $(\boldsymbol{y}_t, \lambda_t)$ is provided as the input to the next iteration. This procedure is repeated over a fixed number of iterations T. We always decode n samples in parallel and rerank them based on the model score. We show the relative speed gain of SUNDAE in Table 3, the Transformer base is used as the AR baseline, where we perform incremental sampling by caching the previous attention states.

Baselines. We compare SUNDAE with Transformer base (Vaswani et al., 2017) as the AR baseline and several non-AR models including CMLM (Ghazvininejad et al., 2019) and DisCo (Kasai et al., 2020). The last two are both iterative non-AR models that share a substantial amount of common ground with SUNDAE. However, their best results come from the variants distilled from Transformer large (Vaswani et al., 2017), with only few test results available without distillation. As we aim to remove any kinds of dependencies on AR approaches, we would like to make a dis-

Index	Sample from our model
1	I had to go back and forth and start over to see what had happened, what was working, and all the pieces were in the right places.
2	tossed in a delicious sauce. Beef brisket is a hearty cut of meat – thick and tender. When you make a lot of meat
3	can expect to see more of the same in 'Wild Herring', although not very often enough to achieve the same effect. In 'Herring',
4	, she was considered to be a fake. Dr. Deborah Chienna has 93 in the last 365 days., headaches, stomach pain, sore
5	as an artist. printing photo cards, artwork, and posters. email me for some ideas. "Rebecca's studio
6	Jersey, Philadelphia and Washington DC. This is a conference attended by residents, physicians, graduate medical students, and others in the field of public health.
7	's location. This research was supported by a grant from the National Association for the Advancement of Science through the U.S. Department of Energy
8	different artists, we would love to show you the piece we have in our shop! Keep an eye on our Gallery – all you have to do is
10	to writing with the Autumns. Together, some of the class will spend time in the classroom to read a poem from the original book. From there we

Table 2: Unconditional samples from our model trained on C4 (without cherry-picking). Since C4 was crawled from the web, newline symbols are abundant both in the training data and the samples.

Index	Type	Text
1	Prompt	Seeing * * * * * * for any visitor to a national park. While it is an exciting moment, a bear can * * * * * * *
	Completion	Seeing a big bear is a must for any visitor to a national park. While it is an exciting moment, a bear can be intimidating as it will come
2	Prompt	Seeing * * * * * * for any visitor to a national park. While it is an exciting moment, a moose can * * * * * *
	Completion	Seeing moose is a unique experience for any visitor to a national park. While it is an exciting moment, a moose can navigate its way through a forest

Table 4: Inpainting from our model trained on C4 (cherry-picked).

tinction between raw, fully-non-AR models, and *AR-distilled* ones, i.e. those trained via distillation from large AR teachers (typically perform better than the student models), or via AR reranking.

Results and Ablation Studies. In Table 1, we show test BLEU scores from SUNDAE and other baselines. On EN→DE, SUNDAE achieved 26.57 BLEU, and it performed the best among the raw non-AR models. To the best of our knowledge, SUNDAE achieved the closest result ($\triangle = 0.73$) to the score of Transformer base (Vaswani et al., 2017) without the aid of AR models. In the raw setting, argmax-unrolled decoding outperforms mask-predict (Ghazvininejad et al., 2019) and easy-first (Kasai et al., 2020) by 0.74 BLEU when using the same amount of the worst-case compute budget. At $T = 10$, SUNDAE performs better than Imputer (Saharia et al., 2020), which is a strong baseline that can generate high-quality translation within few (4-8) iterations. We discovered that as T increases, low-temperature sampling outperforms argmax-unrolled decoding, and at $T = 10$, low-temperature sampling outperforms *Mask-Predict* (Ghazvininejad et al., 2019) and *Easy-First* (Kasai et al., 2020) by 1.5 BLEU score. SUNDAE achieved 30.80 BLEU on DE→EN, where there are not that many raw baselines to compare with. The difference between SUNDAE and the AR baseline on DE→EN is 0.98. Finally, SUNDAE achieved 37.77 BLEU on EN→FR, where the Transformer base (Vaswani et al., 2017) reported 38.1 BLEU on this task. SUNDAE is only 0.33 BLEU behind the standard AR baseline on EN→FR, once again without distillation.

We ablate the number of steps of *unrolled denoising* on EN→DE test split. We observe that having at least one *unrolled denoising* step is crucial to obtain a good performance in translation. With only $L^{(1)}$ loss, SUNDAE achieves 11.19 BLEU, whereas using $L^{(1:2)}$, the score is improved to 26.57 BLEU. Using $L^{(1:3)}$, i.e. averaging more *unrolled denoising* loss terms, did not improve the performance further, scoring 26.25 BLEU. The target length prediction also turned to be an important component of our model: without it, the translation performance degrades on average by 2 BLEU score points on EN→DE (see Section C in supplementary material).

3.2 Text Generation

Unconditional Text Generation (Qualitative). We train our method on a large high-quality publicly available Colossal Clean Common Crawl (C4) dataset (Raffel et al., 2019) to demonstrate

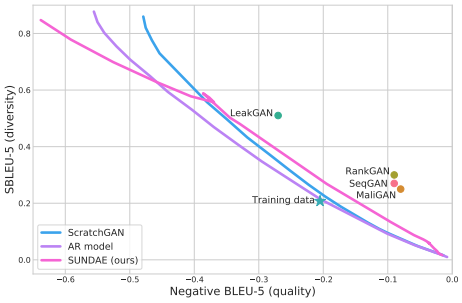


Figure 2: BLEU scores on EMNLP2017 News. Left is better, lower is better. Quality/diversity is controlled by varying the temperature.

Steps (T)	Relative Speed Improvement
4	4.7x
8	2.6x
16	1.4x
32	0.7x

Table 3: Relative speed gain of SUNDAE over AR Transformer base (greedy decoding) on WMT'14 EN→DE validation set.

samples. We tokenize the data using SentencePiece (Kudo & Richardson, 2018) [3] with vocabulary size 32K and train on randomly cropped sequences of length 32. The network is the same as the Transformer decoder part for MT in Section 3.1 but is much larger — 335M parameters: 24 layers, 1024 embedding size, 4096 hidden size, 16 attention heads. Same as in the MT setup, we remove the causal mask since our model is non-autoregressive. We train up to 400K steps with Adam optimizer with batch size 4096 and use cosine annealing for scheduling the learning-rate.

The trained model is then used for unconditional sampling. Starting with 32 random tokens, we iteratively perform up to 1K steps from our model, stopping earlier if samples do not change anymore. To make convergence faster, we use temperature 0.8 and update only 30% of all tokens at each step, chosen randomly — as explained in Section 2.2. We show 10 non-cherry-picked samples from our model in Table 2. All of them resemble reasonable-quality internet texts, perhaps except number 4.

Unconditional Text Generation (Quantitative). While C4 is one of the largest high-quality datasets publicly available and it is good for showing interesting samples, it does not have an established benchmark among non-AR methods. To quantitatively evaluate our algorithm, we train the same architecture on the EMNLP2017 News[4] dataset, which has numerous text GAN baselines, including a strong ScratchGAN method (d'Autume et al., 2019). We also compare to several other baselines reported in (d'Autume et al., 2019; Caccia et al., 2020): SeqGAN (Yu et al., 2017), LeakGAN (Guo et al., 2017), RankGAN (Lin et al., 2017), MaliGAN (Che et al., 2017) and autoregressive (AR) language model.

The results are shown in Figure 2. SUNDAE demonstrates good quality/diversity trade-off, as measured by BLEU/self-BLEU metrics, outperforming 4 out of 5 GAN baselines, and competing with ScratchGAN and AR model. In the higher quality region, SUNDAE outperforms ScratchGAN, while in higher diversity region it slightly underperforms. The advantage of ScratchGAN in higher diversity region could be explained by its autoregressive decoding (it is a hybrid between AR and GAN). Interestingly, the curve for SUNDAE has a more complicated shape than the baselines — possibly because of its Markov Chain sampling, which can empirically exhibit something akin to "phase transitions".

Text In-painting. The non-autoregressive nature of our model opens up the possibilities beyond prompted generation, such as filling in arbitrary templates. In this section, we qualitatively investigate this new capability, using the C4-trained model described in the previous section, and another one, trained on a dataset of Python code from GitHub. All the architecture/training settings remain the same, except we train twice longer: up to 800K steps. Sampling settings are also the same, except we update all the tokens at each step as diversity is less of an issue with stronger conditioning. We construct the code dataset by extracting files ending in `.py` from open source GitHub repositories with licenses that are one of `apache-2.0, mit, bsd-3-clause, bsd-2-clause, unlicense, cc0-1.0, isc, artistic-2.0`. We perform document level de-duplication checking for exact matches only.

[3] https://github.com/google/sentencepiece
[4] http://www.statmt.org/wmt17/

Index	Type	Text
1	Prompt	`def trunc_len(x, t):  * * * * * * * * * * * * * * * * return min(n, t)`

	Completion	`def trunc_len(x, t):`
		`  n = len(x)`
		`  return min(n, t)`

2	Prompt	`is_even = lambda x:  * * * * * *` `is_odd = lambda x:  not * * * * * * # reuse`
	Completion	`is_even = lambda x:  x % 2 == 0` `is_odd = lambda x:  not is_even(x) # reuse`

Table 5: Inpainting from our model trained on GitHub (cherry-picked).

For the C4 dataset, we present a model with a particular type of prompt `[CTX1][MSK1][CTX2][MSK2]`, where `[CTX]` stands for "context" and `[MSK]` for "masked", in Table 4, which shows how information can be "teleported" from arbitrary location to another arbitrary location, while taking into account bidirectional context. We change a species of the animal in `[CTX2]` and observe that the model actually uses the right word while in-painting `[MSK1]` and plausibly continues `[MSK2]` according to the species. This kind of task would be impossible to perform for an AR model: with left-to-right causality it would be unable to correctly guess the name of the animal, while with a right-to-left one it would have to start filling `[MSK2]` tokens first, so it would just ignore the whole conditioning. By contrast, our model can easily perform this task.

For GitHub Python data, we use two different kinds of prompts for our model in Table 5: `[CTX1][MSK1][CTX2]` and `[CTX1][MSK1][CTX2][MSK2][CTX3]`. In the first example, our model guesses that `n` should be the length of the input sequence and correctly in-paints `[MSK1]` region. This would again be impossible to do for AR models. Looking from left-to-right, it is impossible to guess that `n` will be the name of the length variable and `t` is meant to be a threshold. Looking from right-to-left, it is unclear what variables `n` and `t` represent and what is the function being computed. Only a bidirectional model can perform well in this task. In the second example, our model correctly follows the suggestion to reuse the function which it also has to in-paint.

4 Related Work

In this section we provide an overview of generative models, focusing on applications in text domain.

4.1 Autoregressive models

Early attempts to perform autoregressive modelling of text using neural networks began with Bengio et al. (2003), later Sutskever et al. (2011) has extended this approach with RNNs. Since then, improvements in neural network architecture, such as the Transformer (Vaswani et al., 2017), and scaling up as with GPT-3 (Brown et al., 2020) dramatically improved the performance of AR models. While the architectures evolved, the loss remained the same with minor modifications.

Insertion (Stern et al., 2019) and Levenshtein (Gu et al., 2019) Transformers are deviations from the standard paradigm of training and could potentially overcome some of AR problems. However, in the parallel decoding regime, they have to rely on AR distillation for obtaining competitive results.

4.2 Non-autoregressive models in general

Non-autoregressive models evolved in parallel with autoregressive ones but have so far struggled to address the same spectrum of tasks with the same quality as AR models.

Diffusion for generative models was originally introduced by Sohl-Dickstein et al. (2015). Recently Ho et al. (2020) proposed a different parametrization: instead of modeling one-step transitions, one could instead model $p(x_0|x_t)$ and later convert those to $p(x_{t-1}|x_t)$ by probabilistic reasoning. This was recently up-scaled for image modeling by Nichol & Dhariwal (2021) and extended to continuous time by Song et al. (2020). In terms of applications to text generation, two recent works addressed this: Hoogeboom et al. (2021) and Austin et al. (2021). Likelihoods obtained in those works are promising, but still behind AR and lacking sample quality as well.

Some diffusion works like Mittal et al. (2021) also approached modeling discrete sequences in another way: first encode sequences with a VAE into a continuous space and then apply continuous diffusion approaches commonly used for image generation.

Variational Autoencoders (VAEs) were proposed by Rezende et al. (2014) and Kingma & Welling (2013) as a likelihood-based extension of autoencoders. Application of such methods to

text has been problematic empirically: as it was first noted in Bowman et al. (2015), good results come from using a strong AR decoder instead of a simple factorised one, but the latents are often ignored — which was named "posterior collapse problem". This problem was later tackled by Delta-VAEs (Razavi et al., 2019), but with limited success in the text domain, as results of Bosc & Vincent (2020) show.

Normalising Flows (Dinh et al., 2014; Rezende & Mohamed, 2015; Dinh et al., 2016) are another prominent family of generative models, originally proposed for real-valued data and recently revisited for text generation by Hoogeboom et al. (2021). While conceptually interesting, the text samples of such methods lack in fidelity.

Generative Adversarial Networks (GANs) were originally introduced by Goodfellow et al. (2014) and remain one of the dominant methods for image generation. They were later adapted to text generation in (Yu et al., 2017; Guo et al., 2017; Lin et al., 2017; Che et al., 2017). However, text generation with such methods is still a challenge, with a more recent ScratchGAN (d'Autume et al., 2019) showing relatively low-quality samples. At least part of the problem with applying GANs to text generation comes from non-differentiability of discrete text samples, which requires usage of zeroth order optimization methods.

Energy-Based models have a long history dating back to Hopfield (1982); Hinton (2002); LeCun et al. (2006); Ranzato et al. (2007). Recently, Deng et al. (2020) shows promising results for text.

4.3 Non-autoregressive models for machine translation

There have been many attempts to apply non-autoregressive methods to machine translation. Latent transformer (Kaiser et al., 2018) generated latent variables autoregressively and then decoded feed-forwardly. NAT (Gu et al., 2017) was the first to characterize a problem with non-autoregressive generation — "multi-modality". FlowSeq (Ma et al., 2019) applied Normalising Flows to machine translation. LVM-DAEs (Lee et al., 2018) are perhaps most related to our method, but these models do not have *unrolled denoising* and their decoding method is deterministic. The leading methods in this area are CMLM (Ghazvininejad et al., 2019) and DisCo (Kasai et al., 2020). While conceptually similar to LVM-DAEs (Lee et al., 2018), these works have introduced significant improvements to the training and decoding procedure, and, as a result, shown strong competition to AR methods. Both of them also do not have *unrolled denoising* and sampling like we do, but they are still powerful baselines. Later, there were attempts to combine CMLM with local AR (Kong et al., 2020). Perhaps the most related to our work is the SMART (Ghazvininejad et al., 2020) follow-up of the CMLM. However, this method is not probabilistic and it does not show unconditional generation capabilities.

4.4 Denoising objective in general

With the advent of BERT (Devlin et al., 2018), the denoising objective became very popular for text representation learning. The original idea was later simplified in RoBERTa (Liu et al., 2019) by removing unnecessary next-sentence-prediction task and only retaining denoising task (the cloze task). This is similar to our work, however, we do not apply masking to corrupt the input tokens, and instead only switch them to random ones. Another work, Electra (Clark et al., 2020a), deals with the same kind of corruption that we use, but instead of predicting the original token before the corruption, Electra predicts whether it was corrupted or not. Electric (Clark et al., 2020b) does not perform masking, but instead uses noise-contrastive loss to learn a representation of text. BART (Lewis et al., 2019) uses both BERT-style denoising and autoregressive modeling for learning representations. All of these methods were originally motivated by improving representation learning of text. There were a few attempts to heuristically decode models like BERT and turn them into text generative models such as in Wang & Cho (2019), however, samples from these models lack coherence.

5 Conclusion

In this paper we have proposed a novel non-autoregressive method that operates within the framework of denoising autoencoders and, crucially, unrolls the denoising process during training. *Unrolled denoising* allows us to achieve state-of-the art results in WMT'14 English-to-German translation task amongst non-AR methods (without distillation from large AR models) and good qualitative results in unconditional text modeling on C4 dataset. We have also qualitatively demonstrated new inpainting capabilities on C4 and GitHub Python data, which opens up new avenues for creative text editing where a human could more naturally collaborate with a machine on writing and even programming.

REFERENCES

Jacob Austin, Daniel Johnson, Jonathan Ho, Danny Tarlow, and Rianne van den Berg. Structured denoising diffusion models in discrete state-spaces. *arXiv preprint arXiv:2107.03006*, 2021.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. A neural probabilistic language model. *The journal of machine learning research*, 3:1137–1155, 2003.

Tom Bosc and Pascal Vincent. Do sequence-to-sequence vaes learn global features of sentences? *arXiv preprint arXiv:2004.07683*, 2020.

Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. Generating sentences from a continuous space. *arXiv preprint arXiv:1511.06349*, 2015.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

Massimo Caccia, Lucas Caccia, William Fedus, Hugo Larochelle, Joelle Pineau, and Laurent Charlin. Language gans falling short. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=BJgza6VtPB.

Tong Che, Yanran Li, Ruixiang Zhang, R Devon Hjelm, Wenjie Li, Yangqiu Song, and Yoshua Bengio. Maximum-likelihood augmented discrete generative adversarial networks. *arXiv preprint arXiv:1702.07983*, 2017.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, and Phillipp Koehn. One billion word benchmark for measuring progress in statistical language modeling. *CoRR*, abs/1312.3005, 2013. URL http://arxiv.org/abs/1312.3005.

Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. Electra: Pre-training text encoders as discriminators rather than generators. *arXiv preprint arXiv:2003.10555*, 2020a.

Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. Pre-training transformers as energy-based cloze models. *arXiv preprint arXiv:2012.08561*, 2020b.

Cyprien de Masson d'Autume, Mihaela Rosca, Jack Rae, and Shakir Mohamed. Training language gans from scratch. *arXiv preprint arXiv:1905.09922*, 2019.

Yuntian Deng, Anton Bakhtin, Myle Ott, Arthur Szlam, and Marc'Aurelio Ranzato. Residual energy-based models for text generation. *arXiv preprint arXiv:2004.11714*, 2020.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Laurent Dinh, David Krueger, and Yoshua Bengio. Nice: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*, 2014.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2016.

Marjan Ghazvininejad, Omer Levy, Yinhan Liu, and Luke Zettlemoyer. Mask-predict: Parallel decoding of conditional masked language models. *arXiv preprint arXiv:1904.09324*, 2019.

Marjan Ghazvininejad, Omer Levy, and Luke Zettlemoyer. Semi-autoregressive training improves mask-predict decoding. *arXiv preprint arXiv:2001.08785*, 2020.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

Jiatao Gu, James Bradbury, Caiming Xiong, Victor OK Li, and Richard Socher. Non-autoregressive neural machine translation. *arXiv preprint arXiv:1711.02281*, 2017.

Jiatao Gu, Changhan Wang, and Jake Zhao. Levenshtein transformer. *arXiv preprint arXiv:1905.11006*, 2019.

Jiaxian Guo, Sidi Lu, Han Cai, Weinan Zhang, Yong Yu, and Jun Wang. Long text generation via adversarial training with leaked information. arxiv e-prints, art. *arXiv preprint arXiv:1709.08624*, 2017.

Junliang Guo, Linli Xu, and Enhong Chen. Jointly masked sequence-to-sequence model for non-autoregressive neural machine translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 376–385, 2020.

Geoffrey E Hinton. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800, 2002.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *arXiv preprint arXiv:2006.11239*, 2020.

Emiel Hoogeboom, Didrik Nielsen, Priyank Jaini, Patrick Forré, and Max Welling. Argmax flows and multinomial diffusion: Towards non-autoregressive language models. *arXiv preprint arXiv:2102.05379*, 2021.

John J Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8):2554–2558, 1982.

Lukasz Kaiser, Samy Bengio, Aurko Roy, Ashish Vaswani, Niki Parmar, Jakob Uszkoreit, and Noam Shazeer. Fast decoding in sequence models using discrete latent variables. In *International Conference on Machine Learning*, pp. 2390–2399. PMLR, 2018.

Jungo Kasai, James Cross, Marjan Ghazvininejad, and Jiatao Gu. Non-autoregressive machine translation with disentangled context transformer. In *International Conference on Machine Learning*, pp. 5144–5155. PMLR, 2020.

Yoon Kim and Alexander M Rush. Sequence-level knowledge distillation. *arXiv preprint arXiv:1606.07947*, 2016.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Xiang Kong, Zhisong Zhang, and Eduard Hovy. Incorporating a local translation mechanism into non-autoregressive translation. *arXiv preprint arXiv:2011.06132*, 2020.

Taku Kudo and John Richardson. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *arXiv preprint arXiv:1808.06226*, 2018.

Yann LeCun, Sumit Chopra, Raia Hadsell, M Ranzato, and F Huang. A tutorial on energy-based learning. *Predicting structured data*, 1(0), 2006.

Jason Lee, Elman Mansimov, and Kyunghyun Cho. Deterministic non-autoregressive neural sequence modeling by iterative refinement. *arXiv preprint arXiv:1802.06901*, 2018.

Jason Lee, Raphael Shu, and Kyunghyun Cho. Iterative refinement in the continuous space for non-autoregressive neural machine translation. *arXiv preprint arXiv:2009.07177*, 2020.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*, 2019.

Zhuohan Li, Zi Lin, Di He, Fei Tian, Tao Qin, Liwei Wang, and Tie-Yan Liu. Hint-based training for non-autoregressive machine translation. *arXiv preprint arXiv:1909.06708*, 2019.

Kevin Lin, Dianqi Li, Xiaodong He, Zhengyou Zhang, and Ming-Ting Sun. Adversarial ranking for language generation. *arXiv preprint arXiv:1705.11001*, 2017.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Xuezhe Ma, Chunting Zhou, Xian Li, Graham Neubig, and Eduard Hovy. Flowseq: Non-autoregressive conditional sequence generation with generative flow. *arXiv preprint arXiv:1909.02480*, 2019.

Matt Mahoney. Text 8 dataset. http://mattmahoney.net/dc/textdata, 2011. Accessed: 2021-09-29.

Gautam Mittal, Jesse Engel, Curtis Hawthorne, and Ian Simon. Symbolic music generation with diffusion models. *arXiv preprint arXiv:2103.16091*, 2021.

Alex Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. *arXiv preprint arXiv:2102.09672*, 2021.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pp. 311–318, 2002.

Matt Post. A call for clarity in reporting bleu scores. *arXiv preprint arXiv:1804.08771*, 2018.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv e-prints*, 2019.

Qiu Ran, Yankai Lin, Peng Li, and Jie Zhou. Guiding non-autoregressive neural machine translation decoding with reordering information. *arXiv preprint arXiv:1911.02215*, 2019.

Marc'Aurelio Ranzato, Y-Lan Boureau, Sumit Chopra, and Yann LeCun. A unified energy-based framework for unsupervised learning. In *Artificial Intelligence and Statistics*, pp. 371–379. PMLR, 2007.

Ali Razavi, Aäron van den Oord, Ben Poole, and Oriol Vinyals. Preventing posterior collapse with delta-vaes. *arXiv preprint arXiv:1901.03416*, 2019.

Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International conference on machine learning*, pp. 1530–1538. PMLR, 2015.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International conference on machine learning*, pp. 1278–1286. PMLR, 2014.

Chitwan Saharia, William Chan, Saurabh Saxena, and Mohammad Norouzi. Non-autoregressive machine translation with latent alignments. *arXiv preprint arXiv:2004.07437*, 2020.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*, 2015.

Raphael Shu, Jason Lee, Hideki Nakayama, and Kyunghyun Cho. Latent-variable non-autoregressive neural machine translation with deterministic inference using a delta posterior. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 8846–8853, 2020.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, pp. 2256–2265. PMLR, 2015.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

Mitchell Stern, William Chan, Jamie Kiros, and Jakob Uszkoreit. Insertion transformer: Flexible sequence generation via insertion operations. In *International Conference on Machine Learning*, pp. 5976–5985. PMLR, 2019.

Zhiqing Sun, Zhuohan Li, Haoqing Wang, Zi Lin, Di He, and Zhi-Hong Deng. Fast structured decoding for sequence models. *arXiv preprint arXiv:1910.11555*, 2019.

Ilya Sutskever, James Martens, and Geoffrey E Hinton. Generating text with recurrent neural networks. In *ICML*, 2011.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pp. 5998–6008, 2017.

Alex Wang and Kyunghyun Cho. Bert has a mouth, and it must speak: Bert as a markov random field language model. *arXiv preprint arXiv:1902.04094*, 2019.

Yiren Wang, Fei Tian, Di He, Tao Qin, ChengXiang Zhai, and Tie-Yan Liu. Non-autoregressive machine translation with auxiliary regularization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 5377–5384, 2019.

Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. Seqgan: Sequence generative adversarial nets with policy gradient. In *Proceedings of the AAAI conference on artificial intelligence*, volume 31, 2017.

A METHOD DETAILS

Using Markov property and the form of $p_t = f_\theta(\cdot|\mathbf{x}_{t-1})$, for any $t > 0$ we obtain

$$p_t(\mathbf{x}_t|\mathbf{x}_0) = \sum_{\boldsymbol{x}_{t-1} \in X} p_{t-1}(\boldsymbol{x}_{t-1}|\mathbf{x}_0) f_\theta(\mathbf{x}_t|\boldsymbol{x}_{t-1}) \tag{5}$$

$$= \mathbb{E}_{\mathbf{x}_{t-1} \sim p_{t-1}(\cdot|\mathbf{x}_0)} f_\theta(\mathbf{x}_t|\mathbf{x}_{t-1}). \tag{6}$$

By induction, we can keep unrolling the chain for all steps $s = t-1, t-2, \ldots, 1$, eventually obtaining

$$p_t(\boldsymbol{x}_t|\boldsymbol{x}_0) = \sum_{\substack{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{t-1} \\ \in X}} \prod_{s=1}^{t} f_\theta(\boldsymbol{x}_s|\boldsymbol{x}_{s-1})$$

$$= \mathbb{E}_{\substack{\mathbf{x}_1 \sim p_1(\cdot|\boldsymbol{x}_0) \\ \cdots \\ \mathbf{x}_{t-1} \sim p_{t-1}(\cdot|\mathbf{x}_{t-2})}} \left[f_\theta(\boldsymbol{x}_t|\mathbf{x}_{t-1}) \right], \tag{7}$$

which proves the from of the model distribution (Eq. 1).

A.1 UPPER BOUND ON THE LOSS.

Using Jensen's inequality we can swap log and expectation in Eq. 7 to obtain

$$-\log p_t(\mathbf{x}|\mathbf{x}_0) \leq -\mathbb{E}_{\substack{\mathbf{x}_1 \sim p_1(\cdot|\boldsymbol{x}_0) \\ \cdots \\ \mathbf{x}_{t-1} \sim p(\cdot|\mathbf{x}_{t-2})}} \left[\log f_\theta(\mathbf{x}|\mathbf{x}_{t-1}) \right], \tag{8}$$

hence

$$\widetilde{L^{(t)}}(\theta) = -\mathbb{E}_{\substack{\mathbf{x} \sim p_{\text{data}} \\ \mathbf{x}_0 \sim q(\cdot|\mathbf{x})}} \left[\log p_t(\mathbf{x}|\mathbf{x}_0) \right] \leq -\mathbb{E}_{\substack{\mathbf{x} \sim p_{\text{data}} \\ \mathbf{x}_0 \sim q(\cdot|\mathbf{x}) \\ \mathbf{x}_1 \sim f_\theta(\cdot|\mathbf{x}_0) \\ \cdots \\ \mathbf{x}_{t-1} \sim f_\theta(\cdot|\mathbf{x}_{t-2})}} \left[\log f_\theta(\mathbf{x}|\mathbf{x}_{t-1}) \right] = L^{(t)}(\theta). \tag{9}$$

A.2 RELATION TO VAE

Variational Autoencoders (Kingma & Welling, 2013; Rezende et al., 2014) optimise the expected evidence lower bound (ELBO) on the marginal likelihood

$$\log p(\mathbf{x}) \geq -D_{\text{KL}}(q(\mathbf{z}|\mathbf{x}) \| p_0(\mathbf{z})) + \mathbb{E}_{\mathbf{z} \sim q(\cdot|\mathbf{x})} \left[\log p(\mathbf{x}|\mathbf{z}) \right], \tag{10}$$

with respect to parameters of both the variational distribution q and generative distribution p, where p_0 remains the prior for for the latent variables. Denosing autoencoders limit the optimisation to the second term, assuming a fixed encoder q, which is also the case with SUNDAE: identifying encoder with our corruption distribution and generative distribution p with the distribution of our chain at the last step p_T, expected ELBO becomes

$$\mathbb{E}_{\mathbf{x} \sim p_{\text{data}}} [\log p_T(\mathbf{x})] \geq -\mathbb{E}_{\mathbf{x} \sim p_{\text{data}}} \left[D_{\text{KL}}(q(\mathbf{z}|\mathbf{x}) \| p_0(\mathbf{z})) \right] + \mathbb{E}_{\substack{\mathbf{x} \sim p_{\text{data}} \\ \mathbf{x}^c \sim q(\cdot|\mathbf{x})}} [\log p_t(\mathbf{x}|\mathbf{x}^c)]$$

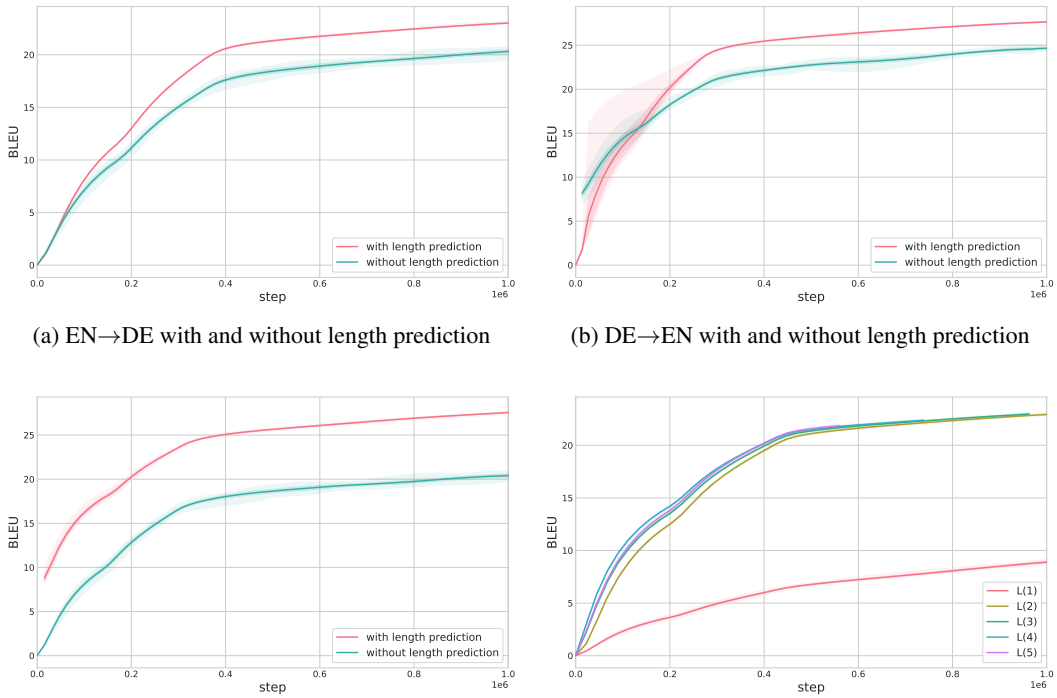
$$\geq -\mathbb{E}_{\mathbf{x} \sim p_{\text{data}}} \left[D_{\text{KL}}(q(\mathbf{z}|\mathbf{x}) \| p_0(\mathbf{z})) \right] - L^{(T)}(\theta). \tag{11}$$

where the second inequality is a consequence of the upper bound 3. Hence, minimizing $L^{(T)}$ would lead to maximisation of the lower bound on the model likelihood.

A.3 CORRUPTION FUNCTION

Our corruption function is obtained in stages:

- sample uniform expected corruption proportion $\alpha \sim \mathcal{U}([0, 1])$,
- sample Bernoulli mask, independently for each token $\boldsymbol{m} \sim (\text{Bernoulli}(\alpha))^N$,
- sample noise $\eta \sim (\mathcal{U}(\{1, 2, \ldots, v\})^N$,



(a) EN→DE with and without length prediction

(b) DE→EN with and without length prediction

(c) EN→FR with and without length prediction

(d) EN→DE with increasing number of unrolls

Figure 3: Validation BLEU curves during training with and without target length prediction on WMT'14 EN→DE, DE→EN and EN→FR tasks shown in Figure 3a, 3b and 3c, respectively. Figure 3d shows the effect of *unrolled denoising* terms on translation quality, here $L(k) = L^{(1:k)}$. All models are trained with a batch size of 256 using 20 different random seeds to display uncertainty.

- compute corruption $\boldsymbol{x}^c = c_{\alpha,\boldsymbol{m},\eta}(\boldsymbol{x}) = (1-\boldsymbol{m}) \cdot \boldsymbol{x} + \boldsymbol{m} \cdot \eta$ (with multiplications taken element-wise).

The distribution $p^c(\cdot|\boldsymbol{x})p_{\text{data}}(\boldsymbol{x})$ obtained in this process covers the entire input space X, with probability skewed towards samples whose parts come from p_{data}. Note that except for corruption proportion, all steps are independent for all individual tokens. Given α, $p^{c_\alpha}(\boldsymbol{y}|\boldsymbol{x})$ can be factorised as

$$\prod_n p^{c_\alpha}(\boldsymbol{y}^{(n)}|\boldsymbol{x}^{(n)}) = \prod_n h(\boldsymbol{y}^{(n)})^T \boldsymbol{Q}_\alpha h(\boldsymbol{x}^{(n)}),$$

where $\boldsymbol{Q}_\alpha = \alpha \boldsymbol{I} + (1-\alpha)\boldsymbol{V}$ is a transition matrix, $\boldsymbol{V}$ is a $v \times v$ matrix with all entries equal to $\frac{1}{v}$, and $h(a)$ denotes one-hot vector representation of $a \in \{1, 2, \ldots, v\}$. In this context, corruption with rate α corresponds to a *multinomial-diffusion forward process* with probability α of remaining in the same state, as proposed by (Hoogeboom et al., 2021).

B Details of Target Length Prediction

The target length prediction module consists of 6 residual blocks. The length prediction module takes the source encodings $\boldsymbol{h}_{\boldsymbol{x}} = encoder(\boldsymbol{x})$ as the input, where $\boldsymbol{x}$ is a sequence of source tokens. We first project $\boldsymbol{h}_{\boldsymbol{x}}$ into a vector $\boldsymbol{v}_{\boldsymbol{x}} \in \mathbb{R}^{d_{LP}}$, where $d_{LP} = 128$ is the hidden size of the target length prediction module. We add a source length embedding vector $\boldsymbol{v}_{len} = S_{(N_{source} \times d_{LP})}(l_{\boldsymbol{x}})$ to $\boldsymbol{v}_{\boldsymbol{x}}$ and obtain the final encoder state representation $\boldsymbol{v}'_{\boldsymbol{x}} = \boldsymbol{v}_{\boldsymbol{x}} + \boldsymbol{v}_{len}$, where $l_{\boldsymbol{x}}$ is the number of the source tokens, and $N_{source} = 128$ is the maximum number of the source tokens. The target length prediction module then takes $\boldsymbol{v}_{\boldsymbol{x}}$ as the input and predicts the (downsampled) target length $\tilde{l}_d$. We found beneficial downsampling the target lengths for the length prediction by a factor of 2, i.e. $l_d = \lceil l/2 \rceil$ as compared to exact prediction of l; since the maximum number of target tokens is $N = 128$, the maximum size of the outcome of length prediction is $N_d = 64$.

Source	Ich habe dort Dinge gesehen, aus denen ich gestärkt hervorgegangen bin.
Initialization	ThirVerschlMarta moderatopposed frameworks solidierweitert [...]
Step 1	I saw things there that which I I stronger..
Step 2	I saw things there from which I me stronger..
Step 3	I saw things there from which I emerged stronger.
Step 4	I saw things there from which I emerged stronger.
Reference	I saw some things there and came out stronger.

Table 6: German-to-English translation process. Since initialization is quite long, we substitute the trailing tokens with [...]. Tokens changed from previous step are highlighted in gray. The process converges after 3 steps, while AR would take 10 steps (one for each token in the translation).

We prepend an embedding $\boldsymbol{h}_{len} \in \mathbb{R}^{d_{enc}}$ of target length to the source encodings $\boldsymbol{h}_{\boldsymbol{x}} \in \mathbb{R}^{N_{source} \times d_{enc}}$ for each source example $\boldsymbol{x}$ and allow the decoder to attend to it (d_{enc} here denotes the dimensionality of the encoder). $\boldsymbol{h}_{len}$ is obtained by applying an embedding matrix $V_{(N_d \times d_{enc})}$ to ground truth (downsampled) length l_d at training time, or to predicted one $\tilde{l}_d$ during sampling. Note that optimisation of the length classification loss does not affect the encoder parameters.

C Effects of Target Length Prediction

It is shown empirically that the length of the translated results can affect BLEU score. Approximate search methods such as beam search usually incorporate a length normalization term in order to prevent the search algorithm to prefer shorter sentences. This is due to the fact that the scoring function is usually chosen as the likelihood modelled by the translation system, and beams containing shorter sequences tend to score higher likelihoods. In non-AR models, the end of sequence is not explicitly learned by the models, thus they can benefit by knowing the target sequence length in advance to generating the translation. However, the length of the target sequence cannot be known at inference time, therefore, a model should predict it instead. We show how BLEU scores can differ in WMT'14 translation tasks with and without the target length prediction in Figure 3a-3c.

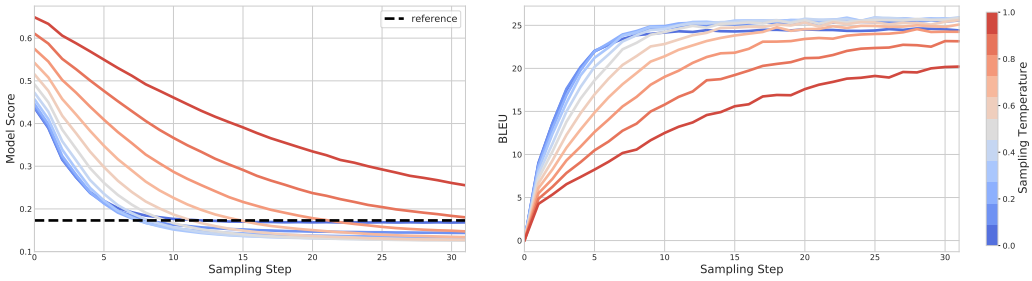
D Effects of Unrolled Logits Losses

We show how varying the number of *unrolled logits* can affect the MT performance in Figure 3d.

E Translation analysis

We show how our algorithm works step-by-step for translation in Table 6. It is interesting to observe how multimodality problem (Gu et al., 2017) gets resolved with more steps. Multimodality usually manifests in repeated tokens in translation, which intrinsically comes from inability to coordinate decisions when independently sampling multiple tokens at once. However, after a few steps all repetitions are corrected, because the model conditions on the previous step and hence can better coordinate further changes.

Fig. 4 shows how the model and the BLEU scores evolve during sampling for various temperatures. The model score here refers to the the cross entropy between translation and model logits produced by giving this translation as the input. While for very low temperatures the scores initially improve faster, they are eventually outperformed by scores for higher temperatures, possibly due to lack of diversity induced by near-deterministic sampling. For very high temperatures, on the other hand, the scores improve slowly, and hence using such temperatures is rather impractical.



(a) Model score evolution throughout sampling.

(b) BLEU evolution throughout sampling.

Figure 4: Model Score (left) and BLEU (right) evolution throughout sampling for various temperatures. The reference score on the left hand side denotes the model score obtained for ground truth inputs. Overall, medium-low temperatures give the best results.

F SACREBLEU SCORES ON WMT'14 EXPERIMENTS

SacreBLEU (Post, 2018) is a library for computing BLEU score[5] that does not require the user to manually tokenize the reference and candidate translations. We report BLEU scores measured with SacreBLEU library in order to provide a reference for future studies. We show all the BLEU scores in Table 7.

		EN→DE		DE→EN		EN→FR	
Model	Steps (T)	BLEU	BLEU*	BLEU	BLEU*	BLEU	BLEU*
Deterministic ($n=16$)	2	20.29	20.13	25.28	24.83	-	-
Deterministic ($n=16$)	3	24.07	23.84	28.64	28.12	-	-
Deterministic ($n=16$)	4	25.01	24.75	29.53	28.99	-	-
Deterministic ($n=16$)	10	25.54	25.25	30.11	29.54	-	-
Stochastic ($n=16$)	4	23.05	22.75	28.13	27.62	35.27	34.03
Stochastic ($n=16$)	10	26.25	25.99	30.80	30.24	37.53	36.23
Stochastic ($n=16$)	32	26.57	26.31	30.74	30.11	37.60	36.20
Stochastic ($n=8$)	128	26.53	26.17	30.95	30.31	37.77	36.37

Table 7: Test BLEU: scores computed with SacreBLEU library are denoted as BLEU*.

G PSEUDOCODE OF SUNDAE

We provide a Python-like pseudocode for our method in Listing 1. The main functions are `build_loss_fn`, which should be used for training, and `sampling_fn` which should be used for sampling. For simplicity, we consider decoder-only setup as in unconditional generation experiments (Section 3.2).

```
def get_random_text(shape, vocab_size):
  random_text = rand_int(shape, minval=0, maxval=vocab_size)
  return random_text


def corrupt_text(batched_text, vocab_size):
  corruption_prob_per_sequence = rand_uniform([batched_text.shape[0], 1])
  rand = rand_uniform(batched_text.shape)
  mask = rand < corruption_prob_per_sequence
  random_text = get_random_text(batched_text.shape, vocab_size)
  corrupted = mask * random_text + (1 - mask) * batched_text
  return corrupted
```

[5] https://github.com/mjpost/sacrebleu

```


def build_logits_fn(vocab_size, n_unrolled_steps, enable_sampling):

  def logits_fn(batched_text):
    model = Transformer(vocab_size, use_causal_mask=False)

    def fn(input_batched_text):
      logits = model(input_batched_text)
      return logits

    def unrolled_fn(input_batched_text):
      samples = corrupt_text(input_batched_text, vocab_size)
      all_logits = []
      for _ in range(n_unrolled_steps):
        logits = fn(samples)
        samples = stop_grad(rand_categorical(logits))
        all_logits += [logits]
      final_logits = concatenate(all_logits, axis=0)
      return final_logits

    if enable_sampling:
      return fn(batched_text)
    else:
      return unrolled_fn(batched_text)

  return logits_fn


def build_loss_fn(vocab_size, n_unrolled_steps=2):
  logits_fn = build_logits_fn(
      vocab_size, n_unrolled_steps, enable_sampling=False)

  def loss_fn(batched_text):
    logits = logits_fn(batched_text)
    # repeat batched_text to fit unrolled logits
    targets = concatenate([batched_text] * n_unrolled_steps, axis=0)
    one_hot_targets = one_hot(targets, vocab_size)
    loss_per_token = -sum(
        one_hot_targets * log_softmax(logits), axis=-1)
    loss = mean(loss_per_token)
    return loss

  return loss_fn


def sampling_fn(logits_fn, steps, temperature, batch_size,
                sequence_length, vocab_size):
  batched_text = get_random_text(
      shape=[batch_size, sequence_length], vocab_size)
  for _ in range(steps):
    logits = logits_fn(batched_text)
    samples = rand_categorical(logits / temperature)
    batched_text = samples
  return batched_text
```

Listing 1: Pseudocode of SUNDAE.