Language Rectified Flow: Advancing Diffusion Language Generation with Probabilistic Flows

Shujian Zhang **Lemeng Wu** **Chengyue Gong** **Xingchao Liu**
The University of Texas at Austin
{szhang19, lm.wu, cygong, xcliu}@utexas.edu

Abstract

Recent works have demonstrated success in controlling sentence attributes ($e.g.$, sentiment) and structure ($e.g.$, syntactic structure) based on the diffusion language model. A key component that drives the impressive performance for generating high-quality samples from noise is iteratively denoise for thousands of steps. While beneficial, the complexity of starting from the noise and the learning steps has limited its implementation to many NLP real-world applications. This paper proposes Language Rectified Flow (LF). Our method is based on the reformulation of the standard probabilistic flow models. Language rectified flow learns (neural) ordinary differential equation models to transport between the source distribution and the target distribution, hence providing a unified and effective solution to generative modeling and domain transfer. From the source distribution, our language rectified flow yields fast simulation and effectively decreases the inference time. Experiments on three challenging fine-grained control tasks and multiple high-quality text editing show that our method consistently outperforms its baselines. Extensive experiments and ablation studies demonstrate that our method can be general, effective, and beneficial for many NLP tasks.

1 Introduction

Traditional pretrained large-scale language models (LM) can generate high-quality text for specific real-world applications (Radford et al., 2019; Brown et al., 2020; Chowdhery et al., 2022; Zhang et al., 2022c, 2023). However, updating the LM parameters or finding proper prompts for each control task can be expensive and unscalable given the combinatorially many possible compositions and the lack of supervised data.

Recent research thus has started to explore plug-and-play solutions. With a pretrained language model (LM), the plug-and-play approaches (Krause et al., 2020; Kumar et al., 2021; Yang and Klein, 2021; Zhang et al., 2022a; Mireshghallah et al., 2022) are served as the light-weight constrained guidance to the targeted text sequence generations (Dathathri et al., 2019). The approaches, however, typically rely on search or optimization in the complex text sequence space. The discrete nature of text makes the search/optimization extremely difficult. Though some recent work introduces continuous approximations to the discrete tokens (Kumar et al., 2021; Qin et al., 2022), the high dimensionality and complexity of the sequence space still render it inefficient to find accurate high quality text. The most recent approach Diffusion LM (Li et al., 2022) induces continuous latent representations, which enables efficient gradient-based methods for the controllable generation.

However, despite the advantage of Diffusion LM for separating the distribution map learning from a noise distribution to a meaningful shape distribution, it always requires thousands of steps. The transport trajectory learns from a noise distribution to a meaningful shape the distribution. This is a major efficiency bottleneck during inference since a standard diffusion model requires thousands of generation steps and a proper SDE solver to produce high-reconstruction and diverse text. In addition, the existing denoising diffusion techniques requires substantial hyper-parameter search in involved design space.

We propose language rectified flow, a surprisingly simple approach to the transport mapping problem, which unifiedly solves both generative modeling and domain transfer. The language rectified flow is an ordinary differential equation (ODE) model that transports distribution source distribution π_0 to target distribution π_1 by following straight line paths as much as possible. The straight paths are theoretically preferred because it is the shortest path between two end points, and computationally preferred because it can be precisely

Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), pages 3893–3905

simulated without time discretization. Hence, flows with straight paths bridge the gap between few-step and continuous-time models. Specifically, we formulate an ODE transport flow as the initial text generator with a simpler trajectory compared with the diffusion model formulated in SDE. Meanwhile, we optimize the transport flow cost for the initial flow model to significantly straighten the learning trajectory while maintaining the model's performance. Further, a VAE is utilized to connect the text sequence space and the flow latent space. This can be used to produce the initial values in the language flow for our controllable text generation.

To demonstrate control of language rectified flow, we consider three control targets ranging from span-anchored controls (e.g., length control and infilling) to complex structures (e.g., parse trees). Our LF can generate high-quality text, performing favorably relative to the diffusion-based LM with a 27x faster sampling on the controllable text generation task. In addition to these individual control tasks, we conduct experiments on three challenging settings, including sequential editing of text. Results show that composing operators within our method manage to generate or edit high-quality text, substantially improving over respective baselines in terms of quality and efficiency. Furthermore, we provide extensive ablation studies on different design choices for the proposed method, including the evidence with different generation steps, influence of the constrained optimization, flow in the latent space. Our analysis shows that LF contributes the performance improvement, helping the sampling efficiency and generation. With little modification, ours can be easily applied to other NLP tasks for better controllable text generation. Our contributions are summarized as follows:

- Present a language rectified flow for controllable text generation embracing domain transfer and fast simulation with the ordinary differential equation.
- Propose an efficient and effective way to train the language rectified flow, which can optimize the trade-off between representation learning and flow matching.
- Verify the effectiveness and general applicability of the proposed method in various NLP tasks, $e.g.$, fine-grained text generation and text editing benchmarks, and provide a rich analysis of our method with various design choices.

2 Method

We now introduce our method, Language Rectified Flow (LF), the transport flow for the controlled language generation. Generating text with transport flow can be viewed as transporting from source distributions to target distributions by following a learned trajectory. Specifically, we suggest a general recipe for the language rectified flow: 1) construct the continuous latent space, 2) learn a neural velocity flow network, construct an ODE process in the latent space with the shortest transport path, and utilize a neural network to fit this process, 3) propose the lexicographic (lexico) optimization strategy for the joint learning. In this section, we present in detail our proposed method, LF (Algorithm 1).

2.1 Encoding and Latent Space

As the text input is discrete, encoding the language as the latent vector serves as the higher-level and differentiable sentence representations (Dai et al., 2019; Li et al., 2020). Variational auto-encoders (VAEs) (Kingma and Welling, 2013; Mai et al., 2020) have been used to model text with a low-dimensional continuous latent space with certain regularities (Bowman et al., 2015). A VAE connects the text sequence space $\mathcal{X}$ and the latent space with an encoder $q(z|x)$ that maps text x into latent vector z, and a decoder $p(x|z)$ that maps a z into text. We learn text VAEs from scratch, optimizing the encoder and decoder parameters with the following objective:

$$\begin{aligned}\mathcal{L}_{\mathrm{VAE}}(\boldsymbol{h}) = &-\mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{x})}[\log p(\boldsymbol{x} \mid \boldsymbol{z})] \\ &+ \mathrm{KL}\left(q(\boldsymbol{z} \mid \boldsymbol{x}) \| p_{\mathrm{prior}}(\boldsymbol{z})\right),\end{aligned} \tag{1}$$

where $p_{prior}(z)$ is a standard Gaussian distribution as the prior, and $\mathrm{KL}(\cdot\|\cdot)$ is the Kullback-Leibler divergence that pushes q_{enc} to be close to the prior. The first term encourages z to encode relevant information for reconstructing the observed text x, while the second term adds regularity so that any $z \sim p_{prior}(z)$ can be decoded into high-quality text in the text sequence space $\mathcal{Z}$. Finally, VAE decoder $p(x|z)$ offers a way to map any given latent vector z into the corresponding text sequence.

2.2 Probability Flows

A stochastic differential equation (SDE) characterizes a diffusion process that maps real data to random noise in continuous time $t \in [0, T]$. Specifically, let z_t be the variable of the process following

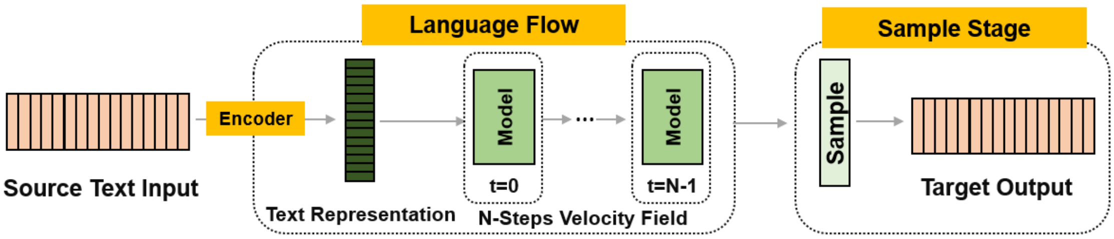

Figure 1: Overview of language rectified flow. Some notations are labeled along with corresponding components.

$z_t \sim \pi_t$, indexed by time t. At start time $t = 0$, $z_0 \sim \pi_0$ which is the data distribution, and at the end $t = T$, $z_T \sim \pi_T$ which is the noise distribution (*e.g.*, standard Gaussian). The reverse SDE instead generates a real sample from the noise by working backwards in time (from $t = T$ to $t = 0$). This diffusion process can be modeled as the SDE:

$$\mathrm{d}z_t = v_\theta(z_t, t)\,\mathrm{d}t + g_t dw, \quad z_0 \sim \pi_0, \tag{2}$$

where g_t is a scalar function known as the diffusion coefficient of z_t and SDE adds a stochastic term w (the standard Wiener process (a.k.a., Brownian motion)) in each update. The recent successful diffusion models (Song et al., 2020; Ho et al., 2020) can be understood as learning stochastic differential equations (SDEs) (Anderson, 1982), and lots of SDE techniques have been applied for developing better models (Karras et al., 2022). In practice, an SDE transport flow usually needs thousands of steps to reach the target distribution. In this work, instead of focusing on SDE, language rectified flow aims to build ODE with a simple training objective and fast sampling.

Our goal is to build a transport flow to push the latent text input from the source distribution to the target distribution. Specifically, given *i.i.d.* samples $\mathcal{D} = \left\{z^{(i)}\right\}_{i=1}^{N}$, we denote z_0 as the data samples from the source distribution π_0 and z_1 as the data samples from the complex target distribution π_1. A probability flow can be effectively learned by training a velocity field v_θ of an ordinary differential equation (ODE), which is indexed by a continuous time variable $t \in [0, T]$ as below:

$$\mathrm{d}z_t = v_\theta(z_t, t)\,\mathrm{d}t, \quad z_0 \sim \pi_0, \tag{3}$$

where z_t is the intermediate state representation at time t and it serves as the linear interpolation of z_0 and z_1. The velocity field $v_\theta(z_t, t)$ is a neural network parameterized by θ. The drift force $v : \mathbb{R}^d \to \mathbb{R}^d$ is set to drive the flow to follow the direction $(z_1 - z_0)$ of the linear path pointing from z_0 to z_1 at any given time t as much as possible. To optimize the velocity, with $t \in [0, 1]$, the flow is followed the ODE process $\mathrm{d}x_t = (z_1 - z_0)\mathrm{d}t$ by solving a least squares regression problem as $\min_\theta \int_0^1 \mathbb{E}_{z\sim\mathcal{D}}\left[\|(v_\theta(z_t, t) - (z_1 - z_0)\|^2\right]\mathrm{d}t$ with $z_t = tz_1 + (1-t)z_0$. During the training, we discretize the above as

$$\mathcal{L}_{\text{FLOW}} = \min_\theta \mathbb{E}_{z\sim\mathcal{D}, t\sim[0,1]}\left[\|(v_\theta(z_t, t) - (z_1 - z_0)\|^2\right]. \tag{4}$$

By fitting the drift v_θ with $z_1 - z_0$, the language flow causalizes the paths of linear interpolation z_t, yielding an ODE flow that can be simulated. After we get v, we solve the ODE starting from $z_0 \sim \pi_0$ to transfer π_0 to π_1, backwardly starting from $z_1 \sim \pi_1$ to transfer π_1 to π_0. The forward and backward sampling are equally favored by the training algorithm, because the objective in Eqn (4) is time-symmetric in that it yields the equivalent problem if we exchange z_0 and z_1 and flip the sign of v.

2.3 Efficiency with Language Rectified Flows

After the neural velocity field v_θ is well-trained, samples can be generated from the ODE to get z_1 as draws from the target distribution π_1 by discretizing the ODE process with Euler solver in Eqn (3) into N steps,

$$z_{(k+1)/N} \leftarrow z_{k/N} + \frac{1}{N} v_\theta(z_{k/N}, k/N), \tag{5}$$

where the integer time step $\hat{t}$ is defined as $\hat{t} \in \{0, 1, \cdots, N-1\}$. k is defined as $k \in \{0, 1, \ldots, N-1\}$. Here $z_0 = z_{0/N}$ is a random sample from π_0 and $z_1 = z_{N/N}$ is the generated data. The number of discretization steps, N, determines the closeness of the simulated trajectory Eqn (5) and the learned continuous ODE trajectory. When $N \to \infty$, the simulated trajectory

Eqn (5) has approached the same endpoint as the continuous one. Intuitively, the solver will be more accurate with a larger N. The language ODE flow follows straight line paths as much as possible. Compared to SDE ($e.g.$, diffusion language model usually requires 1000 to 2000 steps), the straight paths are preferred both theoretically because it is the shortest path between two end points (Lipman et al., 2022; Liu et al., 2022b, 2023a,b), and computationally because it can be exactly simulated without time discretization. In practice, it is often found that an appropriate choice of N for existing probability flow ODEs ranges from 10 to 20.

2.4 Constrained Optimization

Trade-off is a Problem. The encoder-decoder network aims to encode and decode the representation of input text and the flow network aims to transfer from the source distribution to the target distribution. We consider this problem with a bi-level optimization perspective: we want to identify the ideal flow path within the optimized latent representation of the input text. We solve the problem with a simple gradient descent-like approach that iteratively updates the flow network and the encoder-decoder network in a guaranteed fashion. The key feature of our method is that it ensures to optimize the reconstruction loss as a typical optimization method while adding transfer from the source distribution to the target distribution as an extra constrained loss,

$$\min \mathcal{L}_{\text{VAE}} \ \text{ s.t. } \ \mathcal{L}_{\text{FLOW}} < \text{Constraint}, \qquad (6)$$

where $\mathcal{L}_{\text{VAE}}$ refers to the reconstruction loss, $\mathcal{L}_{\text{FLOW}}$ is referred to the Eqn (4), and Constraint is a constraint threshold. The linear combination of multiple objectives is the most widely used approach. However, the coefficient of the combination requires manual tuning, and it is theoretically unsuitable for non-convex functions. This work considers constrained optimization on trading off two objectives, with a special emphasis on lexicographic (lexico) optimization.

Our Equation. To optimize the trade-off between flow optimization and representation construction in Eqn (6), we use lexicographic optimization, in which the parameters are iteratively updated to obtain such a goal:

$$\begin{aligned}\theta_{s+1} \leftarrow \theta_s - \gamma_s (\nabla \mathcal{L}_{\text{VAE}} \\ + \lambda_s \nabla \mathcal{L}_{\text{FLOW}}(\theta_s)),\end{aligned} \qquad (7)$$

where $\gamma_s \geq 0$ is an adaptive step size, $\nabla \mathcal{L}_{\text{FLOW}}$ and $\nabla \mathcal{L}_{\text{VAE}}$ are estimated by score function, and the λ_s can be computed as $\lambda_s = \max\left(\frac{\phi(\theta_t) - \nabla \mathcal{L}_{\text{VAE}}(\theta_s)^{\top} \nabla \mathcal{L}_{\text{FLOW}}(\theta_s)}{\|\nabla \mathcal{L}_{\text{FLOW}}(\theta_s)\|^2}, \quad 0\right)$. $\phi(\theta_s)$ equals to $q(\theta_s) - c$ and the c represents the minimal loss and s is the number of optimization iterations.

The Proposed Algorithm. Our text flow with sampling efficiency and lexico optimization is shown in Algorithm 1. We iteratively update the language rectified flow and the latent representation network in a single-loop manner. Overall, our algorithm gives a ten-step text generation approach. After these three training stages, one can sample a language rectified flow in a few steps starting from a source domain by following Eqn (5).

3 Experimental Settings

We evaluate our method on control tasks and text editing tasks. Table 1 shows the experimental data configuration.

3.1 Control Tasks and Evaluation Metrics

Dataset. We consider three control tasks shown in Table 1: the first two tasks (parts-of-speech and length) rely on the E2E dataset (Novikova et al., 2017), and the last task (infill) is based on Abductive NLG dataset (Bhagavatula et al., 2019). For E2E, the dataset provides information about restaurants and consists of more than 50K combinations of dialogue-act-based meaning representations. Abductive NLG (aNLG) is a conditional generation task for explaining given observations in natural language. It is based on the ART (Bhagavatula et al., 2019) that consists of over 20k commonsense narrative contexts and 200k explanations.

Setting and Metrics. Following Li et al. (2022), for each control task, we sample 200 control targets from the validation splits, and we generate 50 samples for each control target. To evaluate the fluency of the generated text, following the prior works (Dathathri et al., 2019; Yang and Klein, 2021; Li et al., 2022), we report the perplexity (PPL) of generated text. In prior works (Li et al., 2022), this metric is named as fluency score. A lower perplexity score indicates better sample quality.

We define success metrics (SR) for each control task as follows: ❶ For Parts-of-speech, given a sequence of parts-of-speech (POS) tags (e.g., Pronoun Verb Determiner Noun), generate a sequence

Algorithm 1: Language Rectified Flow

Input: Source text $x_0 \in \mathcal{X}_0$ and target text $x_1 \in \mathcal{X}_1$. The initial velocity field v_θ parameterized by θ.
Training Stage
for s iterations **do**
    Randomly sample $x_0 \sim \mathcal{X}_0$ and $x_1 \sim \mathcal{X}_1$.
    Encode the x_0, x_1 as latent vector $z_0 \in \pi_0$, $z_1 \in \pi_1$.
    Train v_θ follows the objective function Eqn (4) with t uniformly sampling from $[0, 1]$.
    Update θ with the lexicographic optimization in Eqn (7).
end for
Sample Stage
Given a $x_0 \in \mathcal{X}_0$, encode it to $z_0 \in \pi_0$, transfer it to domain π_1 using Eqn (5) and well-trained v_θ. Then, decode it to target text domain $\mathcal{X}_1$.

of words of the same length whose POS tags (under an oracle POS tagger) match the target (e.g., I ate an apple). We quantify success via word-level exact match. ❷ For length, given a target length 10, $\cdots$,40, generate a sequence with a length within ± 2 of the target. ❸ For infilling, given a left context (O_1) and a right context (O_2) from the aNLG dataset, and the goal is to generate a sentence that logically connects O_1 and O_2. For evaluation, we report automatic metrics (BLEU (Papineni et al., 2002), ROUGE (Lin and Hovy, 2003), and BertScore (Zhang et al., 2019)).

Baselines. For POS and length, we compare our method with FT (Radford et al., 2019), FUDGE (Yang and Klein, 2021), and Diffusion LM (DLM) (Li et al., 2022). FT is a fine-tune GPT-2 on text pairs, yielding an oracle conditional language model (Li et al., 2022). FUDGE is controllable generation approach based on an autoregressive LM. Diffusion LM is a diffusion based language model. For infilling, three baseline methods are compared including Delorean (Qin et al., 2020), Cold (Qin et al., 2022), and Diffusion LM (Li et al., 2022).

3.2 Text Editing Task and Evaluation Metrics

Dataset. We evaluate in two domains, including the Yelp review (Shen et al., 2017) preprocessed by Li et al. (2018) and the Amazon comment corpus (He and McAuley, 2016). For Yelp, each example is a sentence from a business review on Yelp, and is labeled as having either positive or negative sentiment. Amazon dataset is similar to Yelp. Each example is a sentence from a product review on Amazon, and is labeled as having either positive or negative sentiment (He and McAuley, 2016).

Setting and Metrics. We conduct the sequential editing whose goal is to edit the given text by changing an attribute each time and keep the main content consistent. For example, we consider the input sentence as the source distribution with negative sentiment and the goal is to transfer the input sentence from the negative sentiment to positive sentiment. Generation accuracy is given by a BERT classifier to evaluate the success rate (Zhang et al., 2019). Perplexity (PPL) is calculated on the corresponding domain to measure fluency. To further evaluate the ability of content preservation, we measure BLEU (Papineni et al., 2002) between human-annotated ground truth and output. For each case, we sample 100 examples to evaluate.

Task	Dataset	Train	Val	Test
Control Task	E2E	42.1K	4.7K	4.7K
	Abductive NLG	256.6K	4.6K	9.2K
Text Editing	Yelp	450K	4K	1K
	Amazon	555K	2K	1K

Table 1: Dataset Configuration. The top block is for the control task and the bottom block is for the text editing.

Baselines. Following the prior work (Liu et al., 2022a), we compare with FUDGE (Yang and Klein, 2021) , Style Transformer (Dai et al., 2019), and LatentOps (Liu et al., 2022a) models to sequentially edit the source sentences as a baseline of sequential editing. ① For FUDGE, it has a discriminator that takes in a prefix sequence and predicts whether the generated sequence would meet the conditions. ② Style transformer makes no assumption about the latent representation of source sentence and takes the proven self-attention network. The Transformer at here serves as a basic module to train a style transfer system. ③ For LatentOps, following Liu et al. (2022a), it permits plugging in attribute classifiers applied on text latent vectors in the latent space and utilize an ordinary differential equations sampler to draw latent vector samples.

3.3 Implementation Details

Following the Li et al. (2022) and Ho et al. (2020), for language flow, we set the generation time steps

Model	Parts of Speech		Length			Infill		
	SR ↑	PPL ↓	SR ↑	PPL ↓		BLEU ↑	ROUGE-L ↑	BERTScore ↑
FUDGE	27.0	7.96	46.9	3.11	DELOREAN	1.6	19.1	41.7
FT	89.5	4.72	98.1	3.84	COLD	1.8	19.5	42.7
DLM	90.0	5.16	**99.9**	2.16	DLM	7.1	28.3	89.0
Ours	**94.2**	**4.65**	99.3	**1.74**	**Ours**	**8.2**	**32.7**	**92.1**

Table 2: Comparison to models on Parts of Speech, Length, and Infill. 'SR' and ' PPL' refer to the success metric and the fluency of the generated text, in Section 3.1.

to be 10 and the sequence length to be 64. A U-Net (Ho et al., 2020) backbone is utilized. We use self-attention at the 32 feature map resolution (Wang et al., 2018). All models have two convolutional residual blocks per resolution level and self-attention blocks at the 16 resolution between the convolutional blocks (Chen et al., 2018). The generation time t is specified by adding the Transformer sinusoidal position embedding into each residual block. We train language flow using Adam optimizer and a learning rate at 1×10^{-5}, dropout of 0.1, batch size of 64, and the total number of training iteration is 20K for control tasks, and 30K for text editing tasks. For the details about the latent space structures, we include the details in Appendix A.

4 Experiments

We evaluate the performance of our language rectified flow and learning framework in this section. We bold the best result within each column block. The results of our method are obtained with three independent runs to determine the variance. See Appendix A for full results with error bars.

4.1 Control Tasks Results

Table 2 reports our results on three control-oriented text generation tasks. ① The top block displays the performance of baselines and the LF on the parts of speech and length, and the bottom block shows the results of infill task. We report the results with success metric (SR) and PPL. LF shows consistent performance gains and better model generalization on both complex controls task (Parts-of-speech) and precise future planning tasks (Length) (*e.g.*, 90.0 $\rightarrow$ 94.2 on success rate, 5.16 $\rightarrow$ 4.65 on fluency). ② Our language rectified flow continually outperforms the baseline methods for infilling. Our method achieves better performance in automatic evaluation. These results suggest that LF can solve many types of controllable generation tasks that depend on generation order or lexical constraints (such as infilling) without specialized training. ③ Through these results, it further confirms that LF can work as an effective method to be incorporated into different-type models on the challenging fine-grained text generations.

Generation Efficiency. We provide the running time of generating 50 examples for the parts of speech task. Experiments in this part are performed on a single GPU during generation. The results of our method are tested with three independent runs and the average results are reported in Table 3. Our language flow with the domain transfer flow and straight through sampling is 26.7× faster than Diffusion LM and 16.7× faster than FUDGE. It shows that LF gives the best performance outperforming plug-and-play LM (FUDGE) and Diffusion LM (DLM), while keeping the lowest generation time.

Model	FUDGE	DLM	Ours
Time (s)	50	80	3

Table 3: Results of the generation time for each method. 's' represents the second.

4.2 Text Editing

We further show the experimental results on the text editing in Table 4. We adopt several baselines from the existing literature. Following Liu et al. (2022a), we compare our method with FUDGE (Yang and Klein, 2021), Style Transformer (Dai et al., 2019) models, LatentOps (Liu et al., 2022a). In Table 4: ❶ We observe sizable gains over all baselines with a clear margin (from LatentOps' 24.1 to Ours 25.8). ❷ LF demonstrates the strong capability of controllable editing during the training and allowing the transport from the source distribution to the target distribution. Thus, it comes to the best performance in most of the settings.

Model	BLEU ↑	Accuracy ↑	PPL ↓
STrans (Dai et al., 2019)	25.6	0.89	41.4
FUDGE (Yang and Klein, 2021)	17.2	0.36	38.9
LatentOps (Liu et al., 2022a)	24.1	0.93	26.1
Ours	**25.8**	**0.95**	**24.2**
FUDGE (Yang and Klein, 2021)	35.3	0.25	50.2
LatentOps (Liu et al., 2022a)	28.7	0.77	44.8
Ours	**39.9**	**0.86**	**40.1**

Table 4: Comparison to models on Yelp (top) and Amazon (bottom) dataset. Automatic evaluations (BLEU, accuracy, and PPL) are reported for each model. 'STrans' represents style transformer.

5 Analysis

What is the influence of the constrained optimization vs. manually tuning coefficient? Here we verify whether the constrained optimization in LF is better than the manually tuning strategies. With the designed trade-off, LF targets optimizing between the flow and representation construction. Instead of automatically searching the trade-off between the the flow optimization and representation construction, we manually set a constant λ in Eqn (7) as 0.1, 1.0, 2.0. ❶ Table 5 shows that the constrained optimization of our method brings clear benefits. ❷ We find that without CO, '- CO' with different manually tuned λ value shows an trade-off between the BLEU, ROUGE, and BERTScore across all λ values, indicating that manually tuned λ value can not bring the optimized flow and representation together. It demonstrates the necessity and effectiveness of the constrained optimization for the switchable candidate set in LF structure. We also study the impact of jointly training VAE and Unet vs. separately training them. More results are included in the Appendix A.

	Infill		
	BLEU ↑	ROUGE-L ↑	BERTScore ↑
DLM	7.1	28.3	89.0
Ours	8.2	32.7	92.1
- CO, $\lambda = 0.1$	7.5	29.8	90.5
- CO, $\lambda = 1.0$	6.5	28.8	88.3
- CO, $\lambda = 2.0$	7.8	30.6	90.8

Table 5: Comparison of different λ values for the manually tuned trade-off between the flow and the representation vs. Ours. 'CO' denotes constrained optimization.

Ablations on the components of language rectified flow. Our language rectified flow focuses on latent diffusion. The latent space is constructed by VAE. Thereforce, the purpose of the latent flow is to learn the transport from the Gaussian distribution to the Gaussian distribution. Under this setting, UNet (Ho et al., 2020) and transformer (Vaswani et al., 2017) are two eligible considerations. Following the diffusion language model hyperparameter and code base (Liu et al., 2022a), we obtain the below result (Table 6) for the VAE + UNet (V+U) vs. VAE + transformer (V+T) on the infill task for our language rectified flow. It is clear that our LF can effectively utilizes VAE + UNet or VAE + transformer to achieve comparable performance. It indicates our method is insensitive to different learning structures. This confirms our discussion in Section 2 that LF can serve as an efficient probability flow for language generations.

Model	BLEU ↑	ROUGE-L ↑	BERTScore ↑
V+T	7.9	32.9	92.0
V+U	8.2	32.7	92.1

Table 6: Results of the impact of the latent flow with different model structures on the infill task.

Studies on the role of latent LF. We conduct the ablation study to exam the role of latent LF in the latent space. With the designed flow transport, LF targets the fine-grained text generation. We compare LF with and w/o latent LF settings. W/o latent LF here represents mapping the sentence in one domain to the latent space and directly mapping this latent code into a sentence in another domain without transporting by LF. As shown in Table 7, the w/o latent LF strategy can not generate the high quality text while LF yields better results with a clear margin. It demonstrates the necessity and effectiveness of incorporating the flow with the domain transfer and faster sampling for text generation in LF structure, and a possible reason is that it is too hard to optimize the discrete space.

Model	BLEU ↑	ROUGE-L ↑	BERTScore ↑
W/o Latent Flow	2.1	20.5	45.8
Ours (w. latent Flow)	8.2	32.7	92.1

Table 7: Ablation of the impact of without the latent flow on the infilling controllable text generation.

More evidence for faster simulations with different generation steps. As discussed in Section 2.3, the propose language flow demonstrate the faster sampling with very few generation steps. In Section 3.3, we consolidate the generation steps as ten by default. We select multiple generation steps and study LF's abilities in text generation with different schedule. We follow the same training settings in Section 3 and present the results in Figure 2. We notice that for our case the difference between different generation steps is small. Our method already converges well with just ten steps. The sample quality is further improved with additional update steps.

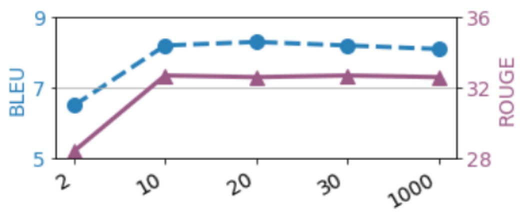

Figure 2: Results of the study on different generation steps for ours on the infill text generation.

Qualitative Analysis. We provide some generated examples in Table 8 to raise a more direct comparison. More examples are included in Appendix. Consistent with the quantitative results, it is difficult for FUDGE to control all the desired length while generate the the semantic informative sentence successfully. For example, the sentence's length of FUDGE is 13. Although Diffusion LM generate the informative sentence, it does not come with the perfect length as our target is 12 and the length of the generated sentence is 11. However, ours is inclined to generate the informative and meaningful sentences with the perfect the length. In conclusion, there seem be a trade-off informative and the length. It can be handled well by ours, but for the baselines, they either lose informative or the accurate length.

Model	Length: 12
FUDGE	The Phoenix is an average Japanese restaurant that is in the City Centre. (13)
Diffusion LM	The Twenty Two serves Chinese food and is not family friendly. (11)
LF	The sky glowed with a vibrant sunset, painting nature's masterpiece across the horizon. (12)

Table 8: Qualitative output of the length control tasks, where all the generated texts tried to match the target length exactly. We mark the length at the end of the sentences.

6 Related Work

Probabilistic Generative Models for Text Probabilistic models for text have shown promising performance improvements. Previous methods (Mueller et al., 2017; Liu et al., 2020; Fan et al., 2020, 2021) utilize the probabilistic variational auto encoders to encode the input sequence into the latent space and use networks for the jointly training. Mai et al. (2020) proposes to train an additional MLP in the latent space of an auto-encoder. Because of the recent success of diffusion models, Diffusion LM (Li et al., 2022) and LDEBM (Yu et al., 2022) utilize the diffusion process in the latent space for text generation. For controllable text generation, Yang and Klein (2021) learns an attribute predictor operating on a partial sequence. and (Xue et al., 2023) uses parameter-efficient tuning (e.g., prompting tuning and low-rank adaptation) to optimize control tokens for LLLM such as GPT and then fine-tune models for controllable generations. Our language rectified flow focuses on the domain transfer by utilizing the ordinary differential equation (ODE) for the faster sampling, and we regard combining multiple constraints as a promising future direction. Our approach is built on the reformulation of the probabilistic flow. Similar to Mai et al. (2020), we build up a flow in the latent space constructed by an auto-encoder, and further transports the text input from the source distribution to the target distribution for the fine-grained and high-quality text generation.

Comparisons of Diffusion Flow and Flow-based Models. Existing flow-based text generation models are normalizing flow-based methods (Ma et al., 2019; Ding and Gimpel, 2021; Tang et al., 2021). Normalizing flows often require explicit, invertible transformations, often resulting in triangular or diagonal Jacobian matrices to ensure efficient computation of determinants. Thus, it needs careful design to ensure invertibility and tractable Jacobians making it hard to train. It requires designing an invertible layer in the neural network. Our language flow is also an ODE / SDE probability flow (Song et al., 2021; Lipman et al., 2022). It has more flexibility to choose the model structures, such as the latest Unet or Transformer in our work. It is usually easy to train as we model the trajectory from the source distribution to the target distribution. Therefore our language flow has no constraints on both architecture and invertibility. In addition, LF and diffusion models can be viewed as members of the probability flow family.

7 Conclusion

Our work demonstrates the benefits of introducing the language rectified flow. The proposed flow can learn the neural ordinary differential equation model to transport between the source distribution and the target distribution, providing the unified and effective solution to generative modeling and domain transfer. Our LF produces the fine-grained and high quality controllable text with fast simulation. The proposed strategy shows noticeable gains in performance across controllable text generation (parts-of speech, length and infill) and text editing (Yelp and Amazon). We further conduct the detailed study with the LF in different settings, *e.g.*, comparing between constrained optimization vs. manually tuning coefficient, providing more evidence for faster simulation with different generation steps, and verifying the impact of different components. To summarize, the flow method is effective and general, with the potential to be incorporated into existing models for various NLP tasks.

References

Brian DO Anderson. 1982. Reverse-time diffusion equation models. *Stochastic Processes and their Applications*, 12(3):313–326.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Chandra Bhagavatula, Ronan Le Bras, Chaitanya Malaviya, Keisuke Sakaguchi, Ari Holtzman, Hannah Rashkin, Doug Downey, Scott Wen-tau Yih, and Yejin Choi. 2019. Abductive commonsense reasoning. *arXiv preprint arXiv:1908.05739*.

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

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. 2018. Neural ordinary differential equations. *Advances in neural information processing systems*, 31.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2022. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*.

Ning Dai, Jianze Liang, Xipeng Qiu, and Xuanjing Huang. 2019. Style transformer: Unpaired text style transfer without disentangled latent representation. *arXiv preprint arXiv:1905.05621*.

Sumanth Dathathri, Andrea Madotto, Janice Lan, Jane Hung, Eric Frank, Piero Molino, Jason Yosinski, and Rosanne Liu. 2019. Plug and play language models: A simple approach to controlled text generation. *arXiv preprint arXiv:1912.02164*.

Rahul Dey and Fathi M Salem. 2017. Gate-variants of gated recurrent unit (gru) neural networks. In *2017 IEEE 60th international midwest symposium on circuits and systems (MWSCAS)*, pages 1597–1600. IEEE.

Xiaoan Ding and Kevin Gimpel. 2021. Flowprior: Learning expressive priors for latent variable sentence models. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3242–3258.

Xinjie Fan, Shujian Zhang, Bo Chen, and Mingyuan Zhou. 2020. Bayesian attention modules. *Advances in Neural Information Processing Systems*, 33:16362–16376.

Xinjie Fan, Shujian Zhang, Korawat Tanwisuth, Xiaoning Qian, and Mingyuan Zhou. 2021. Contextual dropout: An efficient sample-dependent dropout module. *arXiv preprint arXiv:2103.04181*.

Ian Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron Courville, and Yoshua Bengio. 2013. Maxout networks. In *International conference on machine learning*, pages 1319–1327. PMLR.

Ruining He and Julian McAuley. 2016. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *proceedings of the 25th international conference on world wide web*, pages 507–517.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851.

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. 2022. Elucidating the design space of diffusion-based generative models. *arXiv preprint arXiv:2206.00364*.

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

Ben Krause, Akhilesh Deepak Gotmare, Bryan McCann, Nitish Shirish Keskar, Shafiq Joty, Richard Socher, and Nazneen Fatema Rajani. 2020. Gedi: Generative discriminator guided sequence generation. *arXiv preprint arXiv:2009.06367*.

Sachin Kumar, Eric Malmi, Aliaksei Severyn, and Yulia Tsvetkov. 2021. Controlled text generation as continuous optimization with multiple constraints. *Advances in Neural Information Processing Systems*, 34:14542–14554.

Chunyuan Li, Xiang Gao, Yuan Li, Baolin Peng, Xiujun Li, Yizhe Zhang, and Jianfeng Gao. 2020. Optimus: Organizing sentences via pre-trained modeling of a latent space. *arXiv preprint arXiv:2004.04092*.

Juncen Li, Robin Jia, He He, and Percy Liang. 2018. Delete, retrieve, generate: a simple approach to sentiment and style transfer. *arXiv preprint arXiv:1804.06437*.

Xiang Lisa Li, John Thickstun, Ishaan Gulrajani, Percy Liang, and Tatsunori B Hashimoto. 2022. Diffusion-lm improves controllable text generation. *arXiv preprint arXiv:2205.14217*.

Chin-Yew Lin and Eduard Hovy. 2003. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *Proceedings of the 2003 human language technology conference of the North American chapter of the association for computational linguistics*, pages 150–157.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. 2022. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*.

Dayiheng Liu, Jie Fu, Yidan Zhang, Chris Pal, and Jiancheng Lv. 2020. Revision in continuous space: Unsupervised text style transfer without adversarial learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8376–8383.

Guangyi Liu, Zeyu Feng, Yuan Gao, Zichao Yang, Xiaodan Liang, Junwei Bao, Xiaodong He, Shuguang Cui, Zhen Li, and Zhiting Hu. 2022a. Composable text control operations in latent space with ordinary differential equations. *arXiv preprint arXiv:2208.00638*.

Xingchao Liu, Chengyue Gong, and Qiang Liu. 2022b. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*.

Xingchao Liu, Lemeng Wu, Shujian Zhang, Chengyue Gong, Wei Ping, and Qiang Liu. 2023a. Flowgrad: Controlling the output of generative odes with gradients. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 24335–24344.

Xingchao Liu, Xiwen Zhang, Jianzhu Ma, Jian Peng, et al. 2023b. Instaflow: One step is enough for high-quality diffusion-based text-to-image generation. In *The Twelfth International Conference on Learning Representations*.

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

Florian Mai, Nikolaos Pappas, Ivan Montero, Noah A Smith, and James Henderson. 2020. Plug and play autoencoders for conditional text generation. *arXiv preprint arXiv:2010.02983*.

Fatemehsadat Mireshghallah, Kartik Goyal, and Taylor Berg-Kirkpatrick. 2022. Mix and match: Learning-free controllable text generation using energy language models. *arXiv preprint arXiv:2203.13299*.

Jonas Mueller, David Gifford, and Tommi Jaakkola. 2017. Sequence to better sequence: continuous revision of combinatorial structures. In *International Conference on Machine Learning*, pages 2536–2544. PMLR.

Jekaterina Novikova, Ondřej Dušek, and Verena Rieser. 2017. The e2e dataset: New challenges for end-to-end generation. *arXiv preprint arXiv:1706.09254*.

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

Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. 2013. How to construct deep recurrent neural networks. *arXiv preprint arXiv:1312.6026*.

Lianhui Qin, Vered Shwartz, Peter West, Chandra Bhagavatula, Jena Hwang, Ronan Le Bras, Antoine Bosselut, and Yejin Choi. 2020. Back to the future: Unsupervised backprop-based decoding for counterfactual and abductive commonsense reasoning. *arXiv preprint arXiv:2010.05906*.

Lianhui Qin, Sean Welleck, Daniel Khashabi, and Yejin Choi. 2022. Cold decoding: Energy-based constrained text generation with langevin dynamics. *arXiv preprint arXiv:2202.11705*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. 2022. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10684–10695.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pages 234–241. Springer.

Tianxiao Shen, Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2017. Style transfer from non-parallel text by cross-alignment. *Advances in neural information processing systems*, 30.

Yang Song, Liyue Shen, Lei Xing, and Stefano Ermon. 2021. Solving inverse problems in medical imaging with score-based generative models. *arXiv preprint arXiv:2111.08005*.

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

Zineng Tang, Shiyue Zhang, Hyounghun Kim, and Mohit Bansal. 2021. Continuous language generative flow. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4609–4622.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. 2018. Non-local neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7794–7803.

Tianci Xue, Ziqi Wang, and Heng Ji. 2023. Parameter-efficient tuning helps language model alignment. *arXiv preprint arXiv:2310.00819*.

Kevin Yang and Dan Klein. 2021. Fudge: Controlled text generation with future discriminators. *arXiv preprint arXiv:2104.05218*.

Peiyu Yu, Sirui Xie, Xiaojian Ma, Baoxiong Jia, Bo Pang, Ruigi Gao, Yixin Zhu, Song-Chun Zhu, and Ying Nian Wu. 2022. Latent diffusion energy-based model for interpretable text modeling. *arXiv preprint arXiv:2206.05895*.

Shujian Zhang, Chengyue Gong, and Eunsol Choi. 2021a. Knowing more about questions can help: Improving calibration in question answering. *arXiv preprint arXiv:2106.01494*.

Shujian Zhang, Chengyue Gong, and Eunsol Choi. 2021b. Learning with different amounts of annotation: From zero to many labels. *arXiv preprint arXiv:2109.04408*.

Shujian Zhang, Chengyue Gong, and Xingchao Liu. 2022a. Passage-mask: A learnable regularization strategy for retriever-reader models. *arXiv preprint arXiv:2211.00915*.

Shujian Zhang, Chengyue Gong, Xingchao Liu, Pengcheng He, Weizhu Chen, and Mingyuan Zhou. 2022b. Allsh: Active learning guided by local sensitivity and hardness. *arXiv preprint arXiv:2205.04980*.

Shujian Zhang, Chengyue Gong, Lemeng Wu, Xingchao Liu, and Mingyuan Zhou. 2023. Automl-gpt: Automatic machine learning with gpt. *arXiv preprint arXiv:2305.02499*.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. 2022c. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2019. Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*.

A Experimental details

A.1 Full Results With Error Bar

We report the full results of our method with the error bar for control tasks and text editing tasks in Table 9 and 10, respectively.

Model	Parts of Speech		Length	
	SR ↑	PPL ↓	SR ↑	PPL ↓
FUDGE	27.0	7.96	46.9	3.11
FT	89.5	4.72	98.1	3.84
DLM	90.0	5.16	99.9	2.16
Ours	94.2±0.4	4.65±0.1	99.3±0.2	1.74±0.2

	Infill		
	BLEU ↑	ROUGE-L ↑	BERTScore ↑
DELOREAN	1.6	19.1	41.7
COLD	1.8	19.5	42.7
DLM	7.1	28.3	89.0
Ours	8.2±0.3	32.7±0.4	92.1±0.2

Table 9: Comparison to models on Parts of Speech, Length, and Infill. ‘SR’ and ‘ PPL’ refer to the success metric and the fluency of the generated text, in Section 3.1.

Model	BLEU ↑	Accuracy ↑	PPL ↓
STrans (Dai et al., 2019)	25.6	0.89	41.4
FUDGE (Yang and Klein, 2021)	17.2	0.36	38.9
LatentOps (Liu et al., 2022a)	24.1	0.93	26.1
Ours	25.8±0.2	0.95±0.1	24.2±0.3
FUDGE (Yang and Klein, 2021)	35.3	0.25	50.2
LatentOps (Liu et al., 2022a)	28.7	0.77	44.8
Ours	39.9±0.5	0.86±0.05	40.1±0.3

Table 10: Comparison to models on Yelp (top) and Amazon (bottom) dataset. Automatic evaluations (BLEU, accuracy, and PPL) are reported for each model. ‘STrans’ represents style transformer.

A.2 Experimental Datasets

For parts-of-speech and length in control tasks, we rely on the E2E dataset (Novikova et al., 2017). The infill task is based on Abductive NLG dataset (Bhagavatula et al., 2019). For text editing tasks, we utilize the Yelp review (Shen et al., 2017) and Amazon comment corpus (He and McAuley, 2016).

E2E dataset (Novikova et al., 2017) was assembled using the CrowdFlower platform and underwent quality control according to Novikova et al. (2017). This dataset contains information about restaurants and comprises over 50k combinations of dialogue-act-based MRs with an average of 8.1 references each. The dataset is divided into training, validation, and testing sets (at a 76.5-8.5-15 ratio), maintaining a similar distribution of MR and reference text lengths, and ensuring that MRs in different sets are unique. Each MR features 3-8 attributes (slots), such as name, food, or area, along with their corresponding values. In line with Novikova et al. (2017), the E2E data was collected using images as stimuli, which were found to produce significantly more natural, informative, and well-phrased human references than textual MRs. **Abductive NLG (aNLG)** is a task focused on generating natural language explanations for given observations. It is built on the ART dataset (Bhagavatula et al., 2019), which contains more than 20k commonsense narrative contexts and 200k explanations. The training set comprises both plausible and implausible hypotheses gathered through crowdsourcing. In contrast, the development and test sets include hypotheses chosen using the Adversarial Filtering algorithm. **Yelp review** dataset is sourced from the 2018 Yelp Challenge, which focuses on local businesses such as restaurants and bars, treating them as items. To maintain data quality, the same 10-core setting is employed. The preprocessed Yelp review data utilized in this study is provided by Li et al. (2018). The **Amazon dataset** shares similarities with Yelp. Each example consists of a sentence from an Amazon product review and is labeled as expressing either positive or negative sentiment (He and McAuley, 2016).

A.3 Experimental Settings

We use a U-Net (Ho et al., 2020) backbone similar to an unmasked PixelCNN++ (Ronneberger et al., 2015). We set gradient clipping to 1.0. For control tasks, the fine-tune GPT-2 (Radford et al., 2019; Zhang et al., 2022b) on (control, text) pair. We report the sampling (with temperature 1.0) of the fine-tuned models denoted as FT. For FUDGE (Yang and Klein, 2021), it incorporates a discriminator that examines a prefix sequence and anticipates whether the generated sequence will comply with the conditions. It can steer text generation by adjusting the probabilities of the pretrained language model based on the discriminator's output. Following Li et al. (2022), we adopt FUDGE's architecture, training a discriminator for each attribute. A three-layer LSTM followed by a Linear layer is served as the discriminator. For the latent space encoder and decoder, we employ an RNN encoder–decoder setup, as suggested by Cho et al. (2014); Bahdanau et al. (2014). The RNN Encoder–Decoder used in our experiment contains 1K hidden units, featuring the proposed gates in both the encoder and decoder.

The encoder of the our latent construction comprises forward and backward RNNs, each with 1K hidden units. The decoder also contains 1K hidden units. In both instances, we utilize a multilayer network with a single maxout hidden layer, as per Goodfellow et al. (2013), to calculate the conditional probability of each target word, following the method outlined by Pascanu et al. (2013). We have implemented the GRU as the activation function, as suggested by Dey and Salem (2017). Attention is integrated into the decoder, allowing it to determine the sections of the source sentence that should be focused on, in accordance with Bahdanau et al. (2014). We maintain the same optimizer and learning rate in this setup. The language rectified flow experiments are carried out in an end-to-end fashion, as described by Rombach et al. (2022).

For the latent space encoder and decoder, we train an RNN encoder–decoder (Cho et al., 2014; Bahdanau et al., 2014). The RNN Encoder–Decoder used in the experiment had 1K hidden units with the proposed gates at the encoder and at the decoder. The encoder of the RNN consists of forward and backward recurrent neural networks (RNN) each having 1K hidden units. Its decoder has 1K hidden units. In both cases, we use a multilayer network with a single maxout (Goodfellow et al., 2013) hidden layer to compute the conditional probability of each target word (Pascanu et al., 2013). The GRU (Dey and Salem, 2017) is implemented as the activation function. The attention is incorporated in the decoder. The decoder decides parts of the source sentence to pay attention to (Bahdanau et al., 2014). We use the same optimizer and learning rate here. The language flow experiments are conducted in an end-to-end manner (Zhang et al., 2021a,b; Rombach et al., 2022). More detailed experimental settings are included in Appendix A.

A.4 Jointly Train vs. Separately Train.

Training the language rectified flow and VAE separately requires more training time. In Table 11, We obtain the below result for the infill task. It is clear that our LF effectively utilizes constrained optimization to achieve slightly better performance while still maintaining the lower training cost (training iterations). From experimental results, we can successfully train them together, and we hypothesize that our VAE is easier to train under these settings.

Model	BLEU ↑	ROUGE-L ↑	BERTScore ↑	Number of Training iterations ↓
Separately	7.9	32.5	92.2	VAE: 10K + LF 20K
Ours	8.2	32.7	92.1	20K

Table 11: Reported results of Training LF and VAE.

A.5 More Qualitative Examples

We provide some generated examples in Table 12 to raise a more direct comparison. Aligned with prior quantitative findings, there appears to be a trade-off between informativeness and sentence length. Our model manages this well, but the baseline models tend to sacrifice either informativeness or precise length.

Model	Length: 6
FUDGE	Climate change affects global weather patterns. (6)
FUDGE	Recent virtual reality technology is advancing rapidly. (7)
FUDGE	Blockchain is revolutionizing financial systems. (5)
Diffusion LM	Artificial intelligence transforms modern healthcare. (5)
Diffusion LM	Plant-based diets gain popularity worldwide. (6)
Diffusion LM	Remote work reshapes traditional office culture. (6)
LF	Technology enhances learning experiences in schools. (6)
LF	The global economy navigates unprecedented challenges. (6)
LF	Sustainability remains crucial in modern architecture. (6)

Table 12: More qualitative output of the length control tasks, where all the generated texts try to match the target length exactly. We mark the length at the end of the sentences.