David helps Goliath: Inference-Time Collaboration Between Small Specialized and Large General Diffusion LMs

Xiaochuang Han[♠◇] **Sachin Kumar**[♣]
Yulia Tsvetkov[♠] **Marjan Ghazvininejad**[◇]
[♠]University of Washington [◇]Meta AI [♣]Carnegie Mellon University
{xhan77, yuliats}@cs.washington.edu[♠] sachink@cs.cmu.edu[♣] ghazvini@meta.com[◇]

Abstract

Diffusion-based language models are emerging as a promising alternative to autoregressive LMs: they approach the competence of autoregressive LMs while offering nuanced controllability at inference time. While autoregressive LMs have benefited immensely from scaling and instruction-based learning, existing studies of diffusion LMs have been conducted on a smaller scale. Starting with a recently proposed diffusion model SSD-LM, in this work we first explore methods to scale it from 0.4B to 13B parameters, proposing techniques to improve its training and inference efficiency, and to finetune the model to follow instructions. Armed with a more powerful, general purpose diffusion LM, we introduce the primary contribution of this work – SSD-2 – an approach to easily ensemble at inference time a large general-purpose diffusion LM with smaller, but specialized and contextualized diffusion LMs. We show that SSD-2 facilitates novel ensembles with 100x smaller models that can be customized and deployed by individual users. We find that compared to autoregressive models, the collaboration between diffusion LMs is more effective, leading to higher-quality model responses due to their ability to dynamically incorporate bi-directional contexts.[1]

1 Introduction

Following the footsteps of diffusion-based generative models for continuously valued data such as images, audio, and video (Ho et al., 2020; Kong et al., 2021; Ho et al., 2022), recent works have attempted to replicate these successes on discrete text data (Austin et al., 2021; Li et al., 2022c; Han et al., 2022; Strudel et al., 2022; Dieleman et al., 2022). Several studies have shown that diffusion-based language models (LMs) perform competitively to their autoregressive counterparts, and even surpass them at post-hoc controllable text generation (Li et al., 2022c; Han et al., 2022).

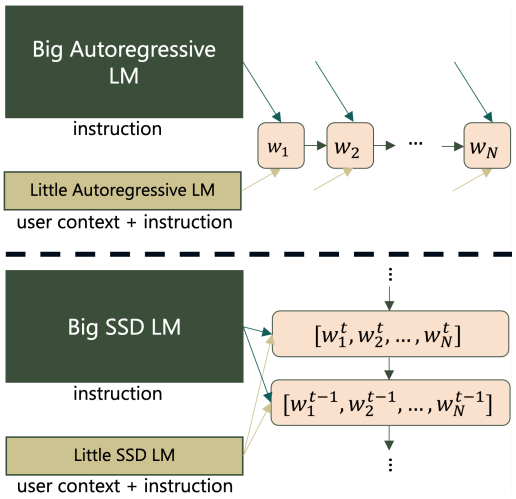

Figure 1: *Inference-time collaboration* between a large general model and a small user model that incorporates user-specified knowledge. The collaboration between autoregressive models performs decoding token-by-token, while the collaboration between diffusion models refines a block of generated tokens iteratively with bi-directional contexts (§3).

Meanwhile, autoregressive language models (Brown et al., 2020; Touvron et al., 2023) have emerged as general-purpose solutions capable of holding conversations with humans and solving tasks by following instructions (Ouyang et al., 2022; Wang et al., 2022; Longpre et al., 2023; Taori et al., 2023; Chiang et al., 2023; Xu et al., 2023). Their abilities are primarily due to two factors: scaling the model parameters as well as pretraining datasets, and instruction finetuning with carefully curated datasets (Ouyang et al., 2022).

However, as the models become increasingly immense and proprietary, it is difficult for individual users to customize the system with their own data (e.g., specialized knowledge) due to cost or privacy reasons (§3). A primary contribution of this work is to illustrate a novel setup of *inference-time collabo-*

[1] https://github.com/xhan77/ssd-2.

Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), pages 8385–8400
June 16-21, 2024

ration between LMs and show a unique advantage of diffusion LMs compared to autoregressive models in this scenario. With diffusion models' iterative generation design over a span of bi-directional contexts, multiple diffusion LMs with different capabilities can be easily ensembled at the sequence level at test time, leveraging advantages of each LM in the ensemble.

As a preliminary to our experiments, we first present an exploratory study to scale and incorporate instruction-following and conversational capabilities in diffusion-based LMs. We introduce SSD-2, an improved version of recently introduced simplex-based diffusion LM SSD-LM (Han et al., 2022) proposing several modifications to its training and inference procedures. We incorporate these improvements in scaling SSD-2 to 13B parameters, up from 0.4B in SSD-LM. We show that similarly to autoregressive LMs, by finetuning with curated instruction datasets, SSD-2 is well-suited to follow chat-style instructions.

We then present our main case study highlighting the setup of inference-time collaboration: we augment a general-purpose large SSD-2 model with 13B parameters with a 100x smaller, user-accessible model. This setup allows incorporating user-provided knowledge into the generation process without directly inputting it into the large model (which can be undesirable due to cost or privacy reasons, more details in §3). We show that SSD-2's instruction finetuned model is substantially more effective at this collaboration than the autoregressive baselines, leveraging bi-directional contexts in the ensemble.

2 Background

Semi-autoregressive simplex-based diffusion LM (SSD-LM) is trained to generate text in blocks of tokens by performing diffusion in the simplex space of the model vocabulary (Han et al., 2022). For text continuation tasks, it has shown competitive performance against autoregressive models (e.g., GPT-2; Radford et al., 2019) when trained with a similar number of model parameters and pretraining data. Furthermore, it naturally enables post-hoc control in generated text using off-the-shelf classifiers, outperforming prior approaches to controlling autoregressive models. Below we briefly overview the training and decoding algorithm of SSD-LM.

Training The core idea behind the training of diffusion models (Ho et al., 2020) is to add a series of progressive noise to the input data representations and learn a model to reverse this process, reconstructing the original data at different noise levels. Assume we have a sequence of tokens $\{w^0, \ldots, w^{c-1}, w^c, \ldots, w^{c+B-1}\}$, where we condition on a context of length c, $\{w^0, \ldots, w^{c-1}\}$ (or $\boldsymbol{w}^{<c}$), and learn to generate the subsequent block of text $\{w^c, \ldots, w^{c+B-1}\}$ (or $\boldsymbol{w}^{c:c+B}$ using a Python-style notation) containing B tokens. In SSD-LM, a progressive Gaussian noise is added to the block of text $\boldsymbol{w}^{c:c+B}$.

$$\tilde{\boldsymbol{w}}_0^{c:c+B} = \text{logits-initialization}(\boldsymbol{w}^{c:c+B})$$
$$\tilde{\boldsymbol{w}}_t^{c:c+B} = \sqrt{\bar{\alpha}_t}\tilde{\boldsymbol{w}}_0^{c:c+B} + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}$$

where $\text{logits-initialization}(\cdot)$ maps each discrete token to a base, almost-one-hot logits representation in the model's vocabulary space V, $\{-K, +K\}^{|V|}$. A noise schedule $\bar{\alpha}_t$ controls the level of noise $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, K^2\boldsymbol{I})$ added to the base representation, where timestep $t \sim \mathcal{U}(1, T)$ and larger t means a noisier representation.

SSD-LM's training loss on $\boldsymbol{w}^{c:c+B}$ is conditioned both on the noisy representation $\tilde{\boldsymbol{w}}_t^{c:c+B}$ and the prior context $\boldsymbol{w}^{<c}$ to the block.

$$\mathbb{E}_t[-\log p_\theta(\boldsymbol{w}^{c:c+B} \mid \tilde{\boldsymbol{w}}_t^{c:c+B}, \boldsymbol{w}^{<c})]$$
$$= \mathbb{E}_t\left[\sum_{j=c}^{j<c+B} -\log p_\theta(w^j \mid \tilde{\boldsymbol{w}}_t^{c:c+B}, \boldsymbol{w}^{<c})\right]$$

The model has access to a locally bi-directional context through the noisy representation. In contrast, the canonical autoregressive training loss for $\boldsymbol{w}^{c:c+B}$ would be $\sum_{j=c}^{j<c+B} -\log p_\theta(w^j \mid \boldsymbol{w}^{<j})$, conditioned on the uni-directional left context only.

Decoding At inference time, given a context $\boldsymbol{w}^{<c}$, SSD-LM generates a block $\boldsymbol{w}^{c:c+B}$ through an iterative denoising algorithm, backtracking the noise timesteps from $t = T$ to 1. Each iteration t consists of three main steps: (1) predict logits representation $\boldsymbol{w}_{\text{logits},t}^{c:c+B}$ for the decoding text block using the learned model, (2) project the logits to an almost-one-hot representation $\hat{\boldsymbol{w}}_t^{c:c+B}$ in the base space $\{-K, +K\}^{|V|}$ (with optional modifications), (3) add a Gaussian noise corresponding to timestep $t-1$ to the projected representation and proceed to the next iteration.

$$\boldsymbol{w}_{\text{logits},t}^{c:c+B} = \text{logits}_\theta(\boldsymbol{w}^{c:c+B} \mid \tilde{\boldsymbol{w}}_t^{c:c+B}, \boldsymbol{w}^{<c})$$
$$\hat{\boldsymbol{w}}_t^{c:c+B} = \text{logits-projection}(\boldsymbol{w}_{\text{logits},t}^{c:c+B})$$
$$\tilde{\boldsymbol{w}}_{t-1}^{c:c+B} = \sqrt{\bar{\alpha}_{t-1}}\hat{\boldsymbol{w}}_t^{c:c+B} + \sqrt{1-\bar{\alpha}_{t-1}}\boldsymbol{z}$$

where θ is the trained diffusion model and $\tilde{\boldsymbol{w}}_T^{c:c+B}$ is initialized with a Gaussian noise.

3 SSD-2

In the age of LLMs, individual users of NLP models may often face a dilemma when they wish to customize the system with their own data. On one hand, it is difficult for user-owned devices to fit very large models and smaller models are usually not powerful enough. On the other hand, uploading their data to a commercial host of large models for finetuning or long in-context learning is expensive and also may not be desirable due to privacy risks. We aim to address this dilemma in this work by proposing a collaborative inference-time algorithm between two diffusion models: a large general-purpose model (such as ones only accessible through an API) and a small model which a user can customize (§3.2).

We first present SSD-2 building on top of SSD-LM with several modifications to improve its training and decoding efficiency (§3.1). We train SSD-2 with a larger pretraining corpus and more parameters (ranging from 0.1B to 13B) than SSD-LM and fine-tune it to follow instructions (§4). Next, we present how different versions of SSD-2 (general-purpose large models and user-enhanced small models) can be effectively interpolated at inference time, outperforming their autoregressive counterparts (§5).

3.1 Algorithmic improvements over SSD-LM

Figure 2 describes the training and decoding algorithms of SSD-2. We highlight the changes in SSD-2 over SSD-LM below.

Self-conditioning The core idea behind self-conditioning (Chen et al., 2022) is that at iteration t, the model takes as input not just the noised sample $\tilde{\boldsymbol{w}}_t^{c:c+B}$, but also a clean output from the previous timestep $t+1$, $\boldsymbol{w}_{\text{logits},t+1}^{c:c+B}$. This allows the model to reuse useful information in the previous prediction and focus on refining it in the current timestep, allowing convergence in fewer iterations. That is, for $T > t \geq 1$:

$$\boldsymbol{w}_{\text{logits},t}^{c:c+B} = \text{logits}_\theta(\boldsymbol{w}^{c:c+B} \mid \tilde{\boldsymbol{w}}_t^{c:c+B}, \boldsymbol{w}_{\text{logits},t+1}^{c:c+B}, \boldsymbol{w}^{<c})$$

More specifically, the noisy representation $\tilde{\boldsymbol{w}}_t^{c:c+B}$ and the previous timestep prediction $\boldsymbol{w}_{\text{logits},t+1}^{c:c+B}$ are combined before the transformer blocks of θ, along with the positional embeddings and timestep embeddings as follows:[2]

$$\begin{aligned}
\tilde{\boldsymbol{h}} &= W_{\text{diff}}[\text{sm}(\tilde{\boldsymbol{w}}_t)] + W_{\text{pred}}[\text{sm}(\boldsymbol{w}_{\text{logits},t+1})] \\
&\quad + \text{Emb}_{\text{pos}}(c : c+B) + \text{Emb}_{\text{diff-time}}(t/T) \\
\boldsymbol{h}^{<c} &= \text{Emb}_{\text{ctx}}(\boldsymbol{w}^{<c}) + \text{Emb}_{\text{pos}}(< c) \\
&\quad + \text{Emb}_{\text{ctx-time}}(t/T) \\
\boldsymbol{w}_{\text{logits},t}^{c:c+B} &= \text{Transformer}[\text{concat}(\boldsymbol{h}^{<c}, \tilde{\boldsymbol{h}})]^{c:c+B}
\end{aligned}$$

To train the model to learn to reuse the predicted logits, we add an additional forward pass during the training phase, activated with a probability $p = 0.5$. We predict $\boldsymbol{w}_{\text{logits},t}^{c:c+B}$ disabling gradient back-propagation, and use it in the new cross entropy loss $-\log p_\theta(w^j \mid \tilde{\boldsymbol{w}}_t^{c:c+B}, \boldsymbol{w}_{\text{logits},t}^{c:c+B}, \boldsymbol{w}^{<c})$.

Removing context length sampling for efficiency The original training algorithm of SSD-LM first samples a context length $c \sim \mathcal{U}(1, |\boldsymbol{w}| - B)$ for each example, encodes the context bi-directionally and computes the diffusion loss for a block of B tokens following that context. The bi-directional encoding of the context $\boldsymbol{w}^{<c}$ cannot be shared across different context sizes c for the same example. Moreover, when the sequence length $|\boldsymbol{w}|$ is large, a high variance in the sampled c across devices in distributed training reduces the effective batch size, slowing down the training considerably. Therefore, in the pretraining and finetuning of SSD-2, we eliminate sampling different c's while equivalently modeling the same training loss as shown in Figure 2 for all $\frac{|\boldsymbol{w}|}{B}$ blocks in one data, by using a special attention mask. The transformer modules of SSD-2 encode the context $\boldsymbol{w}^{<c}$ uni-directionally while preserving the bi-directional attention for the diffusion generation block $\boldsymbol{w}^{c:c+B}$. This leads to a 2x speedup in our pilot pretraining. More details can be found in §A.

Sharded models across time-ranges and early stopping in decoding We observe that at test time SSD-2 often shows distinct behaviors at different timestep ranges. We empirically divide the number of iterations into five ranges of equal sizes. In the beginning of decoding ($t \in (0.8T, T]$), when the noise level is very high, there is no discernable pattern in which the model's intermediate predictions ($\text{argmax}\, \boldsymbol{w}_{\text{logits},t}^{c:c+B}$) in different iterations differ

[2] As a shorthand, we dropped the superscript for token positions c to $c+B$, and use sm for softmax, Emb for the embedding layer, and W_{diff} and W_{pred} for the embedding matrix for the noisy representation and self-conditioning prediction.

Algorithm 1 Training (at a given c)

$\tilde{\boldsymbol{w}}_0^{c:c+B} = \text{logits-initialization}(\boldsymbol{w}^{c:c+B})$
$t \sim \text{Uniform}(\{1, \dots, T\})$
$\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, K^2\mathbf{I})$
$\tilde{\boldsymbol{w}}_t^{c:c+B} = \sqrt{\bar{\alpha}_t}\tilde{\boldsymbol{w}}_0^{c:c+B} + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}$
$r \sim \text{Bernoulli}(0.5)$
if $r = 0$ **then**
  Take a gradient descent step on $\nabla_\theta[\sum_{j=c}^{j<c+B} - \log p_\theta(w^j \mid \tilde{\boldsymbol{w}}_t^{c:c+B}, \boldsymbol{w}^{<c})]$
else
  With gradient calculation disabled, calculate $\boldsymbol{w}_{\text{logits},t}^{c:c+B} = \text{logits}_\theta(\boldsymbol{w}^{c:c+B} \mid \tilde{\boldsymbol{w}}_t^{c:c+B}, \boldsymbol{w}^{<c})$
  Take a gradient descent step on $\nabla_\theta[\sum_{j=c}^{j<c+B} - \log p_\theta(w^j \mid \tilde{\boldsymbol{w}}_t^{c:c+B}, \boldsymbol{w}_{\text{logits},t}^{c:c+B}, \boldsymbol{w}^{<c})]$
end if

Algorithm 2 Decoding (at a given c)

$\tilde{\boldsymbol{w}}_T^{c:c+B} \sim \mathcal{N}(\mathbf{0}, K^2\mathbf{I})$
for $t = T, \dots, 1$ **do**
  if $t = T$ **then**
    $\boldsymbol{w}_{\text{logits},t}^{c:c+B} = \text{logits}_\theta(\boldsymbol{w}^{c:c+B} \mid \tilde{\boldsymbol{w}}_t^{c:c+B}, \boldsymbol{w}^{<c})$
  else
    $\boldsymbol{w}_{\text{logits},t}^{c:c+B} = \text{logits}_\theta(\boldsymbol{w}^{c:c+B} \mid \tilde{\boldsymbol{w}}_t^{c:c+B}, \boldsymbol{w}_{\text{logits},t+1}^{c:c+B}, \boldsymbol{w}^{<c})$
  end if
  Ensemble with other models if applicable, $\text{all-reduce}_{\Theta,\lambda}(\boldsymbol{w}_{\text{logits},t}^{c:c+B})$
  $\hat{\boldsymbol{w}}^{c:c+B} = \text{logits-projection}(\boldsymbol{w}_{\text{logits},t}^{c:c+B})$
  $\boldsymbol{z} \sim \mathcal{N}(\mathbf{0}, K^2\mathbf{I})$
  $\tilde{\boldsymbol{w}}_{t-1}^{c:c+B} = \sqrt{\bar{\alpha}_{t-1}}\hat{\boldsymbol{w}}^{c:c+B} + \sqrt{1-\bar{\alpha}_{t-1}}\boldsymbol{z}$
end for
return $\text{argmax}\, \tilde{\boldsymbol{w}}_0^{c:c+B}$

Figure 2: Training and decoding algorithms for SSD-2. The training algorithm describes the training objective at an arbitrary context length c. The decoding algorithm can be applied multiple rounds by appending the generation from one round to the context for the next. The decoding may stop after a fixed number of rounds or until a special end-of-sequence token is encountered.

from each other. Larger changes often happen at $t \in (0.6T, 0.8T]$ after which the majority of the content is in place, and for $t \in (0.4T, 0.6T]$ only minor changes happen sparsely to make a grammatical correction or settle down on an uncertain word choice. Finally, for $t \in (0, 0.4T]$, the sequence does not update at all in most cases. We hence hypothesize that the first three timestep ranges require different capabilities from the model. In SSD-2, we propose to optionally train three separate models $\theta_{(0.4,0.6)}$, $\theta_{(0.6,0.8)}$, and $\theta_{(0.8,1.0)}$ for the three ranges.[3] We still train a single model at pretraining to save resources and only perform this step during a final finetuning as described in §4.[4] We start the decoding at $t = T$ and stop at $t = 0.4T$, saving 40% of the inference computation.[5]

3.2 Inference-time collaboration

As shown in SSD-LM (Han et al., 2022) and prior work in other domains (Dhariwal and Nichol, 2021), diffusion models are naturally suited to allow for controlling the properties of the model outputs by interpolating the model outputs with gradients from a control function such as a classifier. Follow-up studies have extended this idea to classifier-free guidance where diffusion models with and without controlling attributes can be interpolated contrastively using a weighted sum of their outputs (Ho and Salimans, 2021). We explore a new setup of the latter idea for enabling collaboration between two versions of SSD-2 where we interpolate the output logits of the models. Intrinsic to the diffusion paradigm, this interpolation is sequence-level and through many iterations it leverages benefits of the bi-directional context.

Setup We first define a *core* model θ_{core} which is computationally expensive to train or deploy (e.g., a large model which can only be loaded on mutiple GPUs). We assume the model is good at general-domain instruction following. We then define a *user* model θ_{user} which is computationally friendly for a typical user to run on their personal device or a cloud device to their control. It allows incorporating data of their specific interest which they may not prefer to input to the large model. For both the core and user models, we also assume they do not have access to each others' model parameters.

We also assume a prompting instruction $\boldsymbol{w}_{\text{inst}}$ which both the models have access to, and expert data D_{user} that only the user model and not the core model has access to (see Figure 1). During inference,

- θ_{core} only takes in the prompt $\boldsymbol{w}_{\text{inst}}$, $f_{\theta_{\text{core}}}(\boldsymbol{w}_{\text{inst}})$.
- θ_{user} can be finetuned with D_{user}, or use D_{user} in in-context learning. In this work, we experiment with the latter setup, where the user

[3] A similar setup has also been explored in image diffusion as expert denoisers (Feng et al., 2022; Balaji et al., 2022).

[4] This setup could further be improved by considering models of different sizes for the three ranges where $\theta_{(0.4,0.6)}$ and $\theta_{(0.8,1.0)}$ could contain fewer parameters as they arguably perform simpler tasks to reduce the effective inference time. We leave it as future work.

[5] We report a comparison between the decoding speed of SSD-2 and the original SSD-LM in §D

model takes in both the user expert data and the instruction as input, $f_{\theta_{\text{user}}}(D_{\text{user}}, \boldsymbol{w}_{\text{inst}})$.
- Additionally, we assume the model size $|\theta_{\text{core}}| \gg |\theta_{\text{user}}|$ (the size difference is 100x in our experiments).

We will discuss the specific instantiation of the setup in §5. In the section below, we first introduce a prominent collaboration algorithm when θ_{core} and θ_{user} are autoregressive, and then propose a novel algorithm when the models are diffusion-based SSD-2.

Method The collaboration between θ_{core} and θ_{user} is essentially an ensemble of the model outputs. One prominent way of approaching it is through a weighted average of the models' logits at inference time.[6] For autoregressive LMs, this averaging can be performed at the token level where the logits are first combined and then transformed into probability distribution like a product-of-experts ensemble (e.g., Liu et al. (2021)).

$$\begin{aligned} w^c &\sim p_{\text{collab}}(w^c \mid D_{\text{user}}, \boldsymbol{w}_{\text{inst}}, \boldsymbol{w}^{<c}) \\ &= \text{softmax}[(1-\lambda_{\text{user}}) \text{ logits}_{\theta_{\text{core}}}(w^c \mid \boldsymbol{w}_{\text{inst}}, \boldsymbol{w}^{<c}) \\ &\quad + \lambda_{\text{user}} \text{ logits}_{\theta_{\text{user}}}(w^c \mid D_{\text{user}}, \boldsymbol{w}_{\text{inst}}, \boldsymbol{w}^{<c})] \end{aligned}$$

We also consider an extension of this setup where we add a contrastive term to θ_{user} without the input D_{user}, to promote the pointwise mutual information between the expert data and the generation conditioned on the instruction (Malkin et al., 2021).[7]

$$\begin{aligned} w^c \sim \text{softmax}[&(1-\lambda_{\text{user}}) \text{ logits}_{\theta_{\text{core}}}(w^c \mid \boldsymbol{w}_{\text{inst}}, \boldsymbol{w}^{<c}) \\ &+ \lambda_{\text{user}}(1+\alpha) \text{ logits}_{\theta_{\text{user}}}(w^c \mid D_{\text{user}}, \boldsymbol{w}_{\text{inst}}, \boldsymbol{w}^{<c}) \\ &- \lambda_{\text{user}}\alpha \text{ logits}_{\theta_{\text{user}}}(w^c \mid \boldsymbol{w}_{\text{inst}}, \boldsymbol{w}^{<c})] \end{aligned}$$

For SSD-2, the process of generating tokens is intrinsically different from autoregressive models. However, since it preserves the notion of logits in its iterative decoding procedure ($\boldsymbol{w}^{c:c+B}_{\text{logits},t}$), we propose a similar logits-averaging method for a diffusion θ_{core} and θ_{user}, performing an ensemble for a block of tokens at each diffusion timestep.

$$\begin{aligned} \boldsymbol{w}^{c:c+B}_{\text{core-logits},t} &= \text{logits}_{\theta_{\text{core}}}(\boldsymbol{w}^{c:c+B} \mid \boldsymbol{w}_{\text{inst}}, \boldsymbol{w}^{<c}, \tilde{\boldsymbol{w}}_t^{c:c+B}) \\ \boldsymbol{w}^{c:c+B}_{\text{user-logits},t} &= \text{logits}_{\theta_{\text{user}}}(\boldsymbol{w}^{c:c+B} \mid D_{\text{user}}, \boldsymbol{w}_{\text{inst}}, \boldsymbol{w}^{<c}, \tilde{\boldsymbol{w}}_t^{c:c+B}) \\ \boldsymbol{w}^{c:c+B}_{\neg\text{user-logits},t} &= \text{logits}_{\theta_{\text{user}}}(\boldsymbol{w}^{c:c+B} \mid \boldsymbol{w}_{\text{inst}}, \boldsymbol{w}^{<c}, \tilde{\boldsymbol{w}}_t^{c:c+B}) \\ \boldsymbol{w}^{c:c+B}_{\text{logits},t} &= (1-\lambda_{\text{user}})\boldsymbol{w}^{c:c+B}_{\text{core-logits},t} \\ &\quad + \lambda_{\text{user}}(1+\alpha)\boldsymbol{w}^{c:c+B}_{\text{user-logits},t} - \lambda_{\text{user}}\alpha\boldsymbol{w}^{c:c+B}_{\neg\text{user-logits},t} \end{aligned}$$

The above procedure is instantiated through the operation $\text{all-reduce}_{\Theta,\lambda}(\boldsymbol{w}^{c:c+B}_{\text{logits},t})$ in Figure 2. Figure 1 describes both the autoregressive and diffusion collaboration in our setup illustratively. It is noteworthy that for diffusion models, this manner of collaboration is only straightforward in a simplex-based model such as SSD-2. Diffusion variants proposed in the literature operating on token embeddings (§6) are not trivially suitable for it due to a mismatch in the models' embedding space.

4 Experimental Setup

Pretraining Existing work on diffusion LMs is limited to modest model sizes below the order of 1B parameters (Li et al., 2022c; Han et al., 2022; Dieleman et al., 2022). For example, SSD-LM has the same size as RoBERTA-large (Liu et al., 2019) with 0.4B parameters. It is unclear whether diffusions LMs have the ability to scale like autoregressive LMs.[8] To answer this question, we pretrain three versions of SSD-2 with 0.1B, 2.7B, and 13B parameters, on a subset of a large corpus C4 (Raffel et al., 2020). Instead of pretraining from scratch, we initialize these models using publicly available OPT models (Zhang et al., 2022).[9] We consider a maximum sequence length of 500 (up from 200 in SSD-LM), with a diffusion block size $B = 25$. On the 13B SSD-2 model for our main evaluation, we first do 50K warmup steps without self-conditioning and then start a 100K-step pretraining with the full algorithm. It uses approximately 38B tokens from the C4 data in total. Other pretraining hyperparameters can be found in §B. We show the pretraining losses of SSD-2 over time in §C. Based on the trend of pretraining losses and the scale of our pretraining data compared to recent work,[10] we conjecture that our SSD-2 models are

[6] Training-time ensemble can be achieved through methods like parameter-averaging (Li et al., 2022a). However, it is not the focus of this work since our models have drastically different shapes and we do not assume the models have access to the parameters of other models.

[7] We set the contrastive hyperparameter $\alpha = 1.0$ throughout the evalution, though the results with $\alpha = 0.0$ follow a similar trend.

[8] In fact, Strudel et al. (2022) show for embedding-based diffusion models, scaling up the embedding dimensions may hurt the performance in certain cases.

[9] Han et al. (2022) find initializing from pretrained non-diffusion models help the convergence of diffusion losses in SSD-LM.

[10] For example, the LLaMA 13B model (Touvron et al., 2023) uses 1T tokens from multiple corpora including C4, whereas we use 38B tokens from C4 only.

still considerably undertrained. Due to our computing budget, we leave to future work a potential continued pretraining over current SSD-2 models on larger and better curated data.

Instruction finetuning While Han et al. (2022) show the effectiveness of pretrained SSD-LM in general-domain text continuation, in this work, we primarily investigate the use of SSD-2 in downstream fine-tuning tasks, particularly on chat-style instruction following.[11] We finetune the models with the DOLLY dataset[12] containing 15K human-collected instructions and responses (Databricks, 2023). DOLLY covers categories like open/closed-QA, brainstorming, and creative writing, though it may still be less powerful than the distillation-based data in terms of size and quality.[13] We finetune on 95% of the DOLLY data and use the rest for held-out evaluation. We finetune with a batch size of 384 and for 500 or 1000 steps for the 0.1B/2.7B/13B models. As a baseline, we finetune the autoregressive model OPT (0.1B/2.7B/13B) on DOLLY with the same setup.

5 Experiments

5.1 Inference-time collaboration

As introduced in §3.2, a main focus of this work is to explore the advantages of a diffusion-based LM SSD-2 in a collaboration setup: interpolating the outputs of a large, general model θ_{core} and a small model θ_{user} enhanced by user expert data D_{user}.

We use the 13B-parameter SSD-2 finetuned with DOLLY as θ_{core} and the 0.1B finetuned SSD-2 as θ_{user}. We use OPT 13B and 0.1B finetuned with DOLLY under the same collaboration setup as the autoregressive baseline. DOLLY's held-out test prompts are used as $\boldsymbol{w}_{\text{inst}}$. A subset of DOLLY test examples is annotated with loosely related Wikipedia passages to support the output answers; we use these passages as a proxy for D_{user}. To avoid prompts with trivial answers, we additionally constrain the test instructions to those with an original annotated response of at least 50 tokens.

Inference-time collaboration is effective if the core model θ_{core} generates better responses after collaborating with the 100x smaller but user-enhanced θ_{user}. We investigate a range of weights λ_{user}, starting from 0 where the output of the collaboration solely depends on the large θ_{core}, and gradually increasing λ_{user} to incorporate more θ_{user}.

Automatic evaluation We first conduct an automatic evaluation, using state-of-the-art, production-level LMs to evaluate the quality of our models' generations, which have been shown to correlate highly with human judgments and are easier to scale (Liu et al., 2023). We use GPT-3.5-turbo to rate our models' responses to the test instructions on a scale of 10, towards the aspects of relevance, factuality, informativeness, coherence, and understandability. The specific prompting template we used is detailed in §G.

Table 1 summarizes the automatic evaluation results. We observe that when $\lambda_{\text{user}} = 0$ (θ_{core} only, no D_{user} incorporated), the OPT model finetuned with DOLLY consistently outperforms our finetuned SSD-2.[14] However, for OPT, collaborating with the small user model does not improve the core model's performance any further across all considered weights. Within the experimented weighting factors, λ_{user} of 0.1 to 0.3 is relatively optimal, though still leading to lower scores than without collaboration.

In contrast, the small user model θ_{user} improves the core model's performance in all tested attributes in SSD-2. With appropriate weight factors ($\lambda_{\text{user}} = 0.2, 0.3$), the collaborated SSD-2 system surpasses the best OPT performance in four of the five metrics and matches the fifth. We highlight in Table 1 the best absolute performance and the best

[11] We make an additional change while finetuning SSD-2 to address end of sequence (EOS) issues in variable length sequences in the downstream datasets. Since a sequence could terminate in the middle of a diffusion block, while training, we pad the sequence with the EOS token to the nearest boundary of a diffusion block of size B. We do not mask this padding while computing the loss. We use the standard padding token after the last diffusion block boundary. At inference, if the generated text block $\text{argmax}\, \tilde{\boldsymbol{w}}_0^{c:c+B}$ in the final iteration contains an EOS token, we prune the trailing tokens after the first EOS token in the block.

[12] https://huggingface.co/datasets/databricks/databricks-dolly-15k. We deliberately choose to finetune with DOLLY because as opposed to other similar datasets (e.g. the ones used to train models like Alpaca (Taori et al., 2023) and Vicuna (Chiang et al., 2023)), DOLLY has an open-source license and does not depend on distillations from OpenAI models.

[13] We did not explore other earlier instruction tuning data like Wang et al. (2022) and Longpre et al. (2023) since they align less with the chat scenario of our interest. Furthermore, such data can contain a considerable amount of questions that have a very short answer (e.g., multiple choice). We leave for future work to explore the applicability of diffusion on these datasets.

[14] We conjecture the reason is that SSD-2 is undertrained as discussed in §4 and can have a larger domain gap w.r.t. the DOLLY data. In §5.2, we compare the intrinsic instruction following ability of SSD-2 and OPT using one-shot in-context learning instead of finetuning.

	$\lambda_{\text{user}} = 0$	$\lambda_{\text{user}} = 0.1$	$\lambda_{\text{user}} = 0.2$	$\lambda_{\text{user}} = 0.3$	$\lambda_{\text{user}} = 0.4$	$\lambda_{\text{user}} = 0.5$	$\lambda_{\text{user}} = 1.0$
Relevance							
$\text{OPT}_{\{\text{core,user}\}}$	9.76	9.59	9.61	9.65	9.65	9.39	8.23
$\text{SSD-2}_{\{\text{core,user}\}}$	9.72	9.65	**9.91**	9.85	9.64	9.52	7.16
Δ_{collab} OPT		-0.17	-0.15	-0.11	-0.11	-0.37	-1.53
Δ_{collab} SSD-2		-0.07	**+0.19**	+0.13	-0.08	-0.20	-2.56
Factuality							
$\text{OPT}_{\{\text{core,user}\}}$	**9.64**	9.57	9.51	9.55	9.57	9.27	8.15
$\text{SSD-2}_{\{\text{core,user}\}}$	9.34	9.49	9.63	**9.64**	9.56	9.48	7.26
Δ_{collab} OPT		-0.06	-0.12	-0.08	-0.05	-0.29	-1.44
Δ_{collab} SSD-2		+0.14	+0.30	**+0.31**	+0.26	+0.15	-2.03
Informativeness							
$\text{OPT}_{\{\text{core,user}\}}$	9.30	9.20	9.12	9.27	9.06	8.95	7.41
$\text{SSD-2}_{\{\text{core,user}\}}$	8.97	9.02	9.33	**9.36**	9.06	8.97	6.38
Δ_{collab} OPT		-0.10	-0.18	-0.03	-0.24	-0.35	-1.89
Δ_{collab} SSD-2		+0.05	+0.36	**+0.39**	+0.09	0.00	-2.59
Coherence							
$\text{OPT}_{\{\text{core,user}\}}$	9.61	9.47	9.37	9.44	9.41	9.13	7.70
$\text{SSD-2}_{\{\text{core,user}\}}$	9.41	9.35	**9.65**	9.59	9.25	9.17	5.84
Δ_{collab} OPT		-0.14	-0.24	-0.17	-0.20	-0.48	-1.91
Δ_{collab} SSD-2		-0.06	**+0.24**	+0.18	-0.16	-0.24	-3.57
Understandability							
$\text{OPT}_{\{\text{core,user}\}}$	9.66	9.54	9.53	9.54	9.51	9.30	8.10
$\text{SSD-2}_{\{\text{core,user}\}}$	9.53	9.56	**9.72**	9.67	9.42	9.34	6.21
Δ_{collab} OPT		-0.12	-0.13	-0.12	-0.15	-0.36	-1.56
Δ_{collab} SSD-2		+0.03	**+0.19**	+0.14	-0.11	-0.19	-3.32

Table 1: Evaluation of the inference-time collaboration between the large core model θ_{core} and the small user model θ_{user}. A negative impact led by θ_{user} to θ_{core} is marked in red, and a positive impact in blue. SSD-2 is substantially more collaborative than the autoregressive OPT baseline.

performance gain due to the collaboration. We additionally show that when $\lambda_{\text{user}} = 1$, the small user model θ_{user} alone performs worse in SSD-2 than in OPT. This further indicates that the observed performance gain comes from an effective collaboration rather than a significantly better θ_{user}.

Human evaluation To corroborate our findings, we further perform a human evaluation comparing the outputs from SSD-2 and OPT under a collaborative setup. For each test prompt, we show SSD-2 and OPT responses with λ_{user} of 0.2 to the human annotators as a randomized pair. We asked the annotators to choose the preferred response while allowing for annotating equally good responses or equally bad responses. A total of 9 annotators (graduate and undergraduate researchers in NLP, not authoring this work) made 259 human preference annotations over 94 test prompts, with each response pair receiving 1-4 annotations. We show in Table 2 that the collaboration between SSD-2 θ_{core} and θ_{user} is overall more preferred by humans to the OPT models under the same setup. SSD-2 wins in 43 cases (45.7%) while loses only in 25 cases (26.6%). We additionally measure an average Cohen's kappa coefficient between all pairs of annotators who annotated the same subset of instances. We observe κ=0.31 indicating a fair agreement, especially that the task is highly subjective by nature.

Overall, through automatic and human evaluations, we show that SSD-2 offers unique benefits in an interesting case of inference-time collaboration, effectively fusing a general-purpose large model and a small model enhanced by some expert data.

5.2 Ablation study: SSD-2 as a standalone diffusion chat model

In this section, we divert from our main inference-time collaboration setup and investigate the capabil-

SSD-2_{collab} win	Draw	OPT_{collab} win
43 (45.7%)	26 (27.7%)	25 (26.6%)

Table 2: Human preference of the outputs from the inference-time collaboration experiments, comparing the diffusion-based SSD-2 and the autoregressive OPT.

	Win rate	PPL (GPT-Neo/GPT2)	BERTScore (Precision/F1)
$SSD\text{-}2_{13B}$	52.3%	7.58 / 9.62	85.9 / 85.2
OPT_{13B}	47.7%	8.44 / 10.08	85.3 / 84.9

Table 3: Original SSD-2 responding to Vicuna test instructions in an one-shot in-context learning setup. The win rate is computed between SSD-2 and OPT models using the original GPT-4 evaluation introduced in Chiang et al. (2023). BERTScore is computed for the model responses w.r.t. the generations from GPT-3.5.

ities of SSD-2 as a standalone language model. We are interested in the instruction following ability intrinsic to the vanilla SSD-2 *without* inference-time collaboration or any finetuning (like with DOLLY). We compare original SSD-2 and OPT 13B in responding to the prompts from the Vicuna test set (Chiang et al., 2023), which include problems of open-ended question answering, creative writing, etc.[15] We formulate the setup as a one-shot in-context learning problem. Before each Vicuna test prompt, we add one fixed, handcrafted in-context example from Zhou et al. (2023a) to help the models capture the format of the answers without changing the model parameters.

The main metric we report is the win rate from an automatic evaluation based on GPT-4 (OpenAI, 2023). We follow the original evaluation template as introduced in Chiang et al. (2023), prompting GPT-4 to rate SSD-2 and OPT responses along with explanations. As additional metrics, we also compute the conditional perplexity of the responses using external language models GPT-Neo-1.3B (Black et al., 2021) and GPT-2-large (Radford et al., 2019). While there are no gold answers to the Vicuna test prompts, we use GPT-3.5's responses as reference answers and subsequently compute a BERTScore w.r.t. them for the responses from SSD-2 and OPT. As shown in Table 3, we overall observe a higher win rate, lower perplexity, and higher BERTScore for our diffusion language model SSD-2 compared to the autoregressive OPT. We additionally evaluate SSD-2 finetuned with DOLLY and report results in §E. We show some qualitative examples of SSD-2's generations in §F.

6 Related work

Diffusion-based language models have been receiving increasing attention as a potential alternative to autoregressive language models. We identify three main categories of diffusion language models based on how they represent discrete data like text. Discrete diffusion language models represent language naturally as categorical data, while the diffusion or noising steps are often formulated as transition matrices (Hoogeboom et al., 2021; Austin et al., 2021; He et al., 2022; Reid et al., 2022; Zheng et al., 2023; Zhou et al., 2023b). Embedding-based diffusion language models often learn a mapping between the discrete language tokens and an embedding latent space, and the diffusion process is on the embedding space via a series of Gaussian noise (Li et al., 2022c; Gong et al., 2022; Dieleman et al., 2022; Gao et al., 2022; Lovelace et al., 2022; Yuan et al., 2022; Lin et al., 2022; Ye et al., 2023; Chen et al., 2023; Tang et al., 2023; Balagansky and Gavrilov, 2023). In this work, we focus on simplex-based diffusion language models that project discrete tokens to a simplex space and perform the diffusion process with a simple Gaussian noise (Han et al., 2022; Mahabadi et al., 2023). Our proposed inference-time collaboration setup is most straightforward to apply to simplex-based diffusion language models, since models with different sizes share the same simplex (vocabulary) space. Embedding-based models over different latent representation spaces are not suitable for a direct representation interpolation. Furthermore, to the best of our knowledge, SSD-2 is the first of this line of literature to pretrain and finetune a diffusion language model as a chat model, encouraging future work to compare and improve over our work.

With autoregressive language models, various efforts have been made towards building chat-style instruction following models based on open source language models (Touvron et al., 2023; Biderman et al., 2023) to replicate strong production-level closed source counterparts (Ouyang et al., 2022;

[15] Out of the 80 Vicuna test prompts, we empirically find both models constantly fail on prompts from the math and coding categories. We therefore filter them out and keep the rest 70 test cases for our experiments.

OpenAI, 2023). Many of such work are concurrent to ours and collect high-quality finetuning datasets by distilling prompts and responses from OpenAI models (Taori et al., 2023; Chiang et al., 2023; Xu et al., 2023). In this work, we deliberately seek fully open source data not depending on OpenAI models and adopt the DOLLY data (Databricks, 2023). We expect our models can be further improved with future releases of more curated chat-style instruction tuning datasets (Zhou et al., 2023a).

One novel setup we explored in this work is the inference-time collaboration between a large, general-purpose diffusion chat model and small, user-specific models. Inference-time collaboration has been generally explored in autoregressive models via ensembles of logits, either in an interpolation or contrastive manner (Liu et al., 2021; Malkin et al., 2021; Li et al., 2022a; Peng et al., 2022; Li et al., 2022b). In diffusion models, classifier-free guidance in image generation (Ho and Salimans, 2021) contrastively reconstruct representations with and without a controlling attribute using a single model, whereas our work collaboratively decode with models with different sizes and inputs. We show an unique advantage of simplex-based diffusion language models in such inference-time collaboration compared to autoregressive language models.

7 Conclusion

We present an exploratory step towards pretraining a large simplex-based diffusion language model SSD-2 and finetuning it with an open-source chat-style instruction dataset. In a motivated setup where large general models and small user models are to collaborate with each other at inference time, we find SSD-2 substantially more collaborative than its autoregressive counterparts. These findings show the promise of diffusion language models as an instruction-following chat model and a worthy alternative to autoregressive language models.

Limitations

In this work, we explore a novel setup of fusing large general diffusion language models and small customizable models enhanced with user expert data. One limitation of the proposed fusion algorithm is that it requires a search through a range of candidate balancing factors λ_{user}. Furthermore, a selected balancing factor remains the same across different diffusion timesteps, which is not necessarily optimal. Future work can explore and learn an optimal, dynamic schedule of the balancing factors. Another limitation of diffusion language models in general is a slow decoding speed compared to autoregressive models. Though our proposed SSD-2 model already includes improvements over the original SSD-LM leading to faster decoding speed (more details in §D), future work may further adapt methods from image diffusion models targeting specifically for efficient decoding (Song et al., 2021; Nichol and Dhariwal, 2021; Rombach et al., 2022; Meng et al., 2022).

Acknowledgements

The authors would like to thank Alisa Liu, Jiacheng Liu, Weijia Shi, Zihao Ye, members of TsvetShop, and the anonymous reviewers for their insightful discussions and feedback. We additionally thank Shangbin Feng, Tianxing He, Abe Hou, Yuhan Liu, Heng Wang, Jack Zhang, and Michael Zhang for their helpful evaluation. X.H. gratefully acknowledges funding from the UW-Meta AI Mentorship program. This research is supported in part by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA), via the HIATUS Program contract #2022-22072200004. This material is also funded in part by the DARPA Grant under Contract No. HR001120C0124. We also gratefully acknowledge support from NSF CAREER Grant No. IIS2142739, NSF Grants No. IIS2125201, IIS2203097, and the Alfred P. Sloan Foundation Fellowship. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of ODNI, IARPA, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright annotation therein.

References

Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. 2021. Structured denoising diffusion models in discrete state-spaces. In *Proc. NeurIPS*.

Nikita Balagansky and Daniil Gavrilov. 2023. Democratized diffusion language model. *ArXiv*, abs/2305.10818.

Yogesh Balaji, Seungjun Nah, Xun Huang, Arash Vahdat, Jiaming Song, Qinsheng Zhang, Karsten Kreis, Miika Aittala, Timo Aila, Samuli Laine, Bryan Catanzaro, Tero Karras, and Ming-Yu Liu. 2022. ediff-i: Text-to-image diffusion models with an ensemble of expert denoisers. *ArXiv*, abs/2211.01324.

Stella Biderman, Hailey Schoelkopf, Quentin Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, Aviya Skowron, Lintang Sutawika, and Oskar van der Wal. 2023. Pythia: A suite for analyzing large language models across training and scaling.

Sid Black, Gao Leo, Phil Wang, Connor Leahy, and Stella Biderman. 2021. GPT-Neo: Large Scale Autoregressive Language Modeling with Mesh-Tensorflow.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, T. J. Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeff Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. *ArXiv*, abs/2005.14165.

Jiaao Chen, Aston Zhang, Mu Li, Alexander J. Smola, and Diyi Yang. 2023. A cheaper and better diffusion language model with soft-masked noise. *ArXiv*, abs/2304.04746.

Ting Chen, Ruixiang Zhang, and Geo rey E. Hinton. 2022. Analog bits: Generating discrete data using diffusion models with self-conditioning. *ArXiv*, abs/2208.04202.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. 2023. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality.

Databricks. 2023. Databricks dolly 15k. https://huggingface.co/datasets/databricks/databricks-dolly-15k.

Prafulla Dhariwal and Alex Nichol. 2021. Diffusion models beat gans on image synthesis. *ArXiv*, abs/2105.05233.

Sander Dieleman, Laurent Sartran, Arman Roshannai, Nikolay Savinov, Yaroslav Ganin, Pierre H. Richemond, A. Doucet, Robin Strudel, Chris Dyer, Conor Durkan, Curtis Hawthorne, Rémi Leblond, Will Grathwohl, and Jonas Adler. 2022. Continuous diffusion for categorical data. *ArXiv*, abs/2211.15089.

Zhidan Feng, Zhenyu Zhang, Xintong Yu, Yewei Fang, Lanxin Li, Xuyi Chen, Yuxiang Lu, Jiaxiang Liu, Weichong Yin, Shi Feng, Yu Sun, Hao Tian, Hua Wu, and Haifeng Wang. 2022. Ernie-vilg 2.0: Improving text-to-image diffusion model with knowledge-enhanced mixture-of-denoising-experts. *ArXiv*, abs/2210.15257.

Zhujin Gao, Junliang Guo, Xuejiao Tan, Yongxin Zhu, Fang Zhang, Jiang Bian, and Linli Xu. 2022. Difformer: Empowering diffusion model on embedding space for text generation. *ArXiv*, abs/2212.09412.

Shansan Gong, Mukai Li, Jiangtao Feng, Zhiyong Wu, and Lingpeng Kong. 2022. Diffuseq: Sequence to sequence text generation with diffusion models. *ArXiv*, abs/2210.08933.

Xiaochuang Han, Sachin Kumar, and Yulia Tsvetkov. 2022. Ssd-lm: Semi-autoregressive simplex-based diffusion language model for text generation and modular control. *ArXiv*, abs/2210.17432.

Zhengfu He, Tianxiang Sun, Kuan Wang, Xuanjing Huang, and Xipeng Qiu. 2022. Diffusionbert: Improving generative masked language models with diffusion models. *ArXiv*, abs/2211.15029.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020. Denoising diffusion probabilistic models. In *Proc. NeurIPS*.

Jonathan Ho and Tim Salimans. 2021. Classifier-free diffusion guidance. In *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*.

Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J. Fleet. 2022. Video diffusion models. *ArXiv*, abs/2204.03458.

Emiel Hoogeboom, Didrik Nielsen, Priyank Jaini, Patrick Forré, and Max Welling. 2021. Argmax flows and multinomial diffusion: Learning categorical distributions. In *Proc. NeurIPS*.

Zhifeng Kong, Wei Ping, Jiaji Huang, Kexin Zhao, and Bryan Catanzaro. 2021. Diffwave: A versatile diffusion model for audio synthesis. In *Proc. ICLR*.

Margaret Li, Suchin Gururangan, Tim Dettmers, Mike Lewis, Tim Althoff, Noah A. Smith, and Luke Zettlemoyer. 2022a. Branch-train-merge: Embarrassingly parallel training of expert language models. *ArXiv*, abs/2208.03306.

Xiang Lisa Li, Ari Holtzman, Daniel Fried, Percy Liang, Jason Eisner, Tatsunori Hashimoto, Luke Zettlemoyer, and Mike Lewis. 2022b. Contrastive decoding: Open-ended text generation as optimization. *ArXiv*, abs/2210.15097.

Xiang Lisa Li, John Thickstun, Ishaan Gulrajani, Percy Liang, and Tatsunori Hashimoto. 2022c. Diffusion-lm improves controllable text generation. *ArXiv*, abs/2205.14217.

Zheng-Wen Lin, Yeyun Gong, Yelong Shen, Tong Wu, Zhihao Fan, Chen Lin, Weizhu Chen, and Nan Duan. 2022. Genie: Large scale pre-training for text generation with diffusion model. *ArXiv*, abs/2212.11685.

Alisa Liu, Maarten Sap, Ximing Lu, Swabha Swayamdipta, Chandra Bhagavatula, Noah A Smith, and Yejin Choi. 2021. Dexperts: Decoding-time controlled text generation with experts and anti-experts. In *Proc. ACL*.

Yang Liu, Dan Iter, Yichong Xu, Shuo Wang, Ruochen Xu, and Chenguang Zhu. 2023. G-eval: Nlg evaluation using gpt-4 with better human alignment. *ArXiv*, abs/2303.16634.

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

S. Longpre, Le Hou, Tu Vu, Albert Webson, Hyung Won Chung, Yi Tay, Denny Zhou, Quoc V. Le, Barret Zoph, Jason Wei, and Adam Roberts. 2023. The flan collection: Designing data and methods for effective instruction tuning. *ArXiv*, abs/2301.13688.

Justin Lovelace, Varsha Kishore, Chao gang Wan, Eliot Shekhtman, and Kilian Q. Weinberger. 2022. Latent diffusion for language generation. *ArXiv*, abs/2212.09462.

Rabeeh Karimi Mahabadi, Jaesung Tae, Hamish Ivison, James Henderson, Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2023. Tess: Text-to-text self-conditioned simplex diffusion. *ArXiv*, abs/2305.08379.

Nikolay Malkin, Zhen Wang, and Nebojsa Jojic. 2021. Coherence boosting: When your pretrained language model is not paying enough attention. In *Annual Meeting of the Association for Computational Linguistics*.

Chenlin Meng, Ruiqi Gao, Diederik P. Kingma, Stefano Ermon, Jonathan Ho, and Tim Salimans. 2022. On distillation of guided diffusion models. *ArXiv*, abs/2210.03142.

Alexander Quinn Nichol and Prafulla Dhariwal. 2021. Improved denoising diffusion probabilistic models. In *Proc. ICML*.

OpenAI. 2023. Gpt-4 technical report. *ArXiv*, abs/2303.08774.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke E. Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Francis Christiano, Jan Leike, and Ryan J. Lowe. 2022. Training language models to follow instructions with human feedback. *ArXiv*, abs/2203.02155.

Xiangyu Peng, Chen Xing, Prafulla Kumar Choubey, Chien-Sheng Wu, and Caiming Xiong. 2022. Model ensemble instead of prompt fusion: a sample-specific knowledge transfer method for few-shot prompt tuning. *ArXiv*, abs/2210.12587.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Colin Raffel, Noam M. Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *JMLR*.

Machel Reid, Vincent J. Hellendoorn, and Graham Neubig. 2022. Diffuser: Discrete diffusion via edit-based reconstruction. *ArXiv*, abs/2210.16886.

Robin Rombach, A. Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. 2022. High-resolution image synthesis with latent diffusion models. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10674–10685.

Jiaming Song, Chenlin Meng, and Stefano Ermon. 2021. Denoising diffusion implicit models. In *Proc. ICLR*.

Robin Strudel, Corentin Tallec, Florent Altch'e, Yilun Du, Yaroslav Ganin, Arthur Mensch, Will Grathwohl, Nikolay Savinov, Sander Dieleman, L. Sifre, and Rémi Leblond. 2022. Self-conditioned embedding diffusion for text generation. *ArXiv*, abs/2211.04236.

Zecheng Tang, Pinzheng Wang, Keyan Zhou, Juntao Li, Ziqiang Cao, and M. Zhang. 2023. Can diffusion model achieve better performance in text generation? bridging the gap between training and inference! *ArXiv*, abs/2305.04465.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Stanford alpaca: An instruction-following llama model. https://github.com/tatsu-lab/stanford_alpaca.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aur'elien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models. *ArXiv*, abs/2302.13971.

Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Anjana Arunkumar, Arjun Ashok, Arut Selvan Dhanasekaran, Atharva Naik, David Stap, Eshaan Pathak, Giannis Karamanolakis, Haizhi Gary Lai, Ishan Purohit, Ishani Mondal, Jacob Anderson, Kirby Kuznia, Krima Doshi, Maitreya Patel, Kuntal Kumar Pal, M. Moradshahi, Mihir Parmar, Mirali Purohit, Neeraj Varshney, Phani Rohitha Kaza, Pulkit Verma, Ravsehaj Singh Puri, Rushang Karia, Shailaja Keyur

Sampat, Savan Doshi, Siddharth Deepak Mishra, Sujan Reddy, Sumanta Patro, Tanay Dixit, Xudong Shen, Chitta Baral, Yejin Choi, Noah A. Smith, Hanna Hajishirzi, and Daniel Khashabi. 2022. Super-naturalinstructions: Generalization via declarative instructions on 1600+ nlp tasks. In *Conference on Empirical Methods in Natural Language Processing*.

Canwen Xu, Daya Guo, Nan Duan, and Julian McAuley. 2023. Baize: An open-source chat model with parameter-efficient tuning on self-chat data. *ArXiv*, abs/2304.01196.

Jiasheng Ye, Zaixiang Zheng, Yu Bao, Lihua Qian, and Mingxuan Wang. 2023. Dinoiser: Diffused conditional sequence learning by manipulating noises. *ArXiv*, abs/2302.10025.

Hongyi Yuan, Zheng Yuan, Chuanqi Tan, Fei Huang, and Songfang Huang. 2022. Seqdiffuseq: Text diffusion with encoder-decoder transformers. *ArXiv*, abs/2212.10325.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. 2022. Opt: Open pre-trained transformer language models. *ArXiv*, abs/2205.01068.

Lin Zheng, Jianbo Yuan, Lei Yu, and Lingpeng Kong. 2023. A reparameterized discrete diffusion model for text generation. *ArXiv*, abs/2302.05737.

Chunting Zhou, Pengfei Liu, Puxin Xu, Srini Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, Susan Zhang, Gargi Ghosh, Mike Lewis, Luke Zettlemoyer, and Omer Levy. 2023a. Lima: Less is more for alignment. *ArXiv*, abs/2305.11206.

Kun Zhou, Yifan Li, Wayne Xin Zhao, and Ji rong Wen. 2023b. Diffusion-nat: Self-prompting discrete diffusion for non-autoregressive text generation. *ArXiv*, abs/2305.04044.

A Eliminating the sampling of context size c in training

In the original training algorithm of SSD-LM (Han et al., 2022), they first sample a context length $c \sim \mathcal{U}(1, |\boldsymbol{w}| - B)$, and then compute the diffusion loss for reconstructing a block of length B following that context. When the sequence length $|\boldsymbol{w}|$ is large, this can lead to a drastic variance in the values of c. Implemented naively in a distribution training setup, this setup wastes computations, and reduces the effective batch size considerably slowing down training. We eliminate the sampling of the context length c in SSD-2 by processing multiple c's in parallel. To facilitate this, we encode the context $\boldsymbol{w}^{<c}$ uni-directionally while preserving the locally bi-directional attention for the diffusion generation block $\boldsymbol{w}^{c:c+B}$.

More specifically, assume we have a prompt $\boldsymbol{w}^{<c_0}$ and want to form the same training objective as in Figure 2 on all of the following n text blocks $\boldsymbol{w}^{c_0:c_0+nB}$. We prepare a context sequence $\boldsymbol{w}^{<c_0+(n-1)B}$ and obtain $\boldsymbol{h}^{<c_0+(n-1)B}$ as described previously in §3.1. We prepare a diffusion sequence $\tilde{\boldsymbol{w}}^{c_0:c_0+nB}$ and obtain $\tilde{\boldsymbol{h}}^{c_0:c_0+nB}$ as described previously. Then a forward pass of θ works as below.

$$\boldsymbol{o}^{<c_0+(2n-1)B} = \text{Transformer}[$$
$$\text{concat}(\boldsymbol{h}^{<c_0+(n-1)B}, \tilde{\boldsymbol{h}}^{c_0:c_0+nB}); \boldsymbol{\delta}(c_0, n, B)]$$
$$\boldsymbol{w}^{c_0:c_0+nB}_{\text{logits},t} = \boldsymbol{o}^{c_0+(n-1)B:c_0+(2n-1)B}$$

where $\boldsymbol{\delta}(c_0, n, B)$ is a special attention mask for the transformer model, allowing a reuse of the encoded contexts while preserving the original training loss:

$$\delta_{i,j} = \begin{cases} \mathbb{1}_{j \le i} \text{ if } i < c_0 + (n-1)B. \\ \mathbb{1}_{j \le c_0 + kB \text{ or } c_0+(n-1+k)B < j < c_0+(n+k)B} \\ \quad \text{if } c_0 + (n-1+k)B < i < c_0 + (n+k)B, \\ \quad \text{for } 0 \le k < n. \end{cases}$$

Row i of $\boldsymbol{\delta}$ indicates the attention-accessible positions for the i-th input token of the transformer. For example, assume the original context is $[a]$ and the target generation is in two blocks $[b, c]$ and $[d, e]$. The input sequence to the SSD-2 transformer model is $[a, b, c, \tilde{b}, \tilde{c}, \tilde{d}, \tilde{e}]$, and the attention mask is:

$$\boldsymbol{\delta}(1, 2, 2) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 1 \end{bmatrix}$$

Comparing to regular language models, SSD-2 has a uni-directional encoder and locally bi-directional decoder. In a pilot pretraining session, we observe this change leads to a twice as fast training speed compared to the original SSD-LM on a same amount of training tokens.

B Pretraining hyperparameters

For the SSD-2 model of each size (13B/2.7B/0.1B), we conduct two phases of training, a warmup phase

without self-conditioning and a formal phase with self-conditioning using the complete algorithm shown in Figure 2. Throughout all pretraining setups, we use a max sequence length of 500, a learning rate of 1e-4, and a weight decay of 0.01. For the 13B SSD-2, we train with a warmup batch size of 768 for 50,000 steps (19B tokens) and a formal batch size of 384 for 100,000 steps (19B tokens). For the 2.7B SSD-2, we train with a warmup batch size of 256 for 100,000 steps (13B tokens) and a formal batch size of 1024 for 100,000 steps (51B tokens). For the 0.1B SSD-2, we train with a warmup batch size of 2,048 for 200,000 steps (205B tokens) and a formal batch size of 2,048 for 100,000 steps (102B tokens). We use Nvidia V100 GPUs in distributed training, and the different batch size and number of warmup steps across different models are due to the models' memory footprint and the relative cluster traffic during our pilot pretraining. Future work with a dedicated group of computing resources can explore pretraining for longer to mitigate the undertraining issue mentioned in §4.

C Pretraining losses

Figure 3 shows the pretraining losses of SSD-2 over time. We report the losses after the warmup stage and average them across batches with a self-conditioning $p = 0.5$ as described in Figure 2. We see a sign of undertraining from the loss curves. Due to our computing budget, we leave to future work a potential continued pretraining over current SSD-2 models on larger and better curated data.

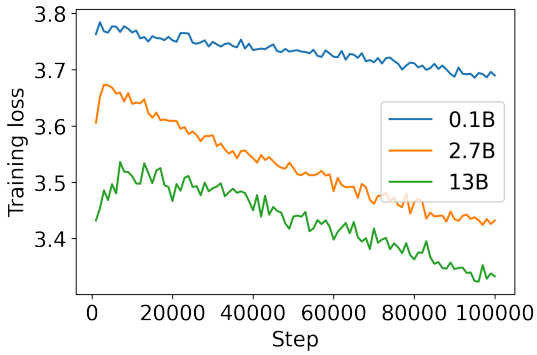

Figure 3: Pretraining losses across training steps (with self-conditioning, after the warmup stage). We conjecture that the models can benefit from more training given more resources.

D Decoding speed

Though the decoding of SSD-2 is still significantly slower than an autoregressive language model, it achieves a great speedup compared to the original SSD-LM. We use a same setup as the reported SSD-LM decoding in Han et al. (2022). Conditioning on 50 prompting tokens, we record the speed of generating the next 25 tokens with timestep $T = 1000$ on a Nvidia V100 GPU.

The 0.4B SSD-LM takes 25 seconds. By contrast, though our 2.7B SSD-2 and 13B SSD-2 are 7x and 33x larger than SSD-LM, they only take 22 seconds and 48 seconds respectively, indicating a significant speedup.[16]

E Standalone SSD-2 finetuned with DOLLY

Following §5.2, we evaluate the outputs from the finetuned models, SSD-2-DOLLY and OPT-DOLLY, on both DOLLY's held-out test set and Vicuna's test set. As shown in Table 4, we find that against very strong baselines pretrained on much larger datasets, our model still wins on a moderate percentage of test examples. Compared to LLaMA (which is trained on 1T tokens for much longer but not fine-tuned for chat), SSD-2 performs marginally better. It is overall mildly less preferred than the OPT-DOLLY model on both DOLLY's and Vicuna's test sets, and significantly less than the Alpaca model. We emphasize that compared to OPT and LLaMA-based models, SSD-2 is currently pretrained with a relatively small, single-corpus dataset,[17] and fine-tuned on an open-source dataset much smaller compared to its non-open-source licensed counterparts that Alpaca relies on. We believe if trained on similar datasets, SSD-2 can fill the current performance gap considerably.

F Qualitative examples

In Table 5, we show some qualitative examples of our finetuned diffusion language model, SSD-2-DOLLY's outputs. The outputs are generated by

[16] It is worth noting that we add a timestep embedding when encoding the context in SSD-2 (§3.1). During inference across timesteps, to reuse the encoded states of the context, we quantize the input timestep t to the context timestep embedding layer, $t_{\text{quantized}} = t - (t \mod Q)$. We use $Q = 50$ throughout the work and do not find the quantization to negatively impact the generation quality empirically.

[17] Despite that we initialize SSD-2 with the OPT weights at the beginning of pretraining, the diffusion objective is fundamentally different from a causal language modeling objective, and the encoded knowledge of OPT may not transfer well.

Comparing against	Win	Draw	Lose
On Dolly's test set			
OPT-Dolly-13B	41.4%	0%	58.6%
On Vicuna's test set			
LLaMa-13B	51.4%	0%	48.6%
OPT-Dolly-13B	37.1%	0%	62.9%
Alpaca-13B	20.0%	1.4%	78.6%

Table 4: Quantitative evaluation of the finetuned SSD-2's outputs against those of autogressive baselines based on ratings of GPT-4. We report the percentage our SSD-2 wins / draws / loses to the baseline models.

the 13B model to the queries from the Vicuna test set (Chiang et al., 2023), on open-ended question answering, creative writing, etc. We empirically find that the generations have an overall reasonable format and content.

G Evaluation templates

We show in Table 6 the templates we used to evaluate model responses via GPT-4 or GPT-3.5-turbo. For the inference-time collaboration results, apart from the five evaluated attributes described in §5, in the template we additionally queried for a rating of conciseness and human preference. However, we consider conciseness a less relevant attribute compared to the other attributes and GPT-3.5-turbo less qualified to give ratings of human preference. We filtered them out in our main results but they hold a similar trend as the results of the other five attributes.

Prompt: How can I improve my time management skills?

```
Time management is in short, an important skill to develop. A person needs to learn how to manage
 their time more effectively. Here are a few techniques that can help you do that:
1. Prioritizing: One of the first things that you should try in order to improve your time
management skills is to identify which tasks have high priority and need to be given immediate
attention, and those that have low priority but can be postponed until a later date.
2. Multi-tasking: Although it might feel productive to handle more than one task at the same time
, it's actually less efficient at managing our time. It's better to focus on one thing at a time,
 and fully dedicate 100% of our efforts to it.
3. Delegating: It's perfectly fine to say no to some tasks or even ask for help when needed. It's
 sometimes considered a sign of weakness, but delegation will allow us to free up our time so
that we can work on the things that really matter.
4. Being organized: Making a list will not only help us keep track of tasks that need to get done
 each day, but it will also help us avoid feeling overwhelmed by all the ideas and thoughts
running through our heads at any given moment.
By following these tips, you'll be able to significantly improve your time management skills!
```

Prompt: Write a symphony concert review, discussing the orchestra's performance and overall audience experience.

```
It was a magical night at Symphony Hall on Saturday, as the Boston Symphony Orchestra, under the
baton of Andin Mehta, performed Tchaikovsky's Symphonies No. 5 and No. 6. Tchaikovsky has always
been one of my favorite composers, and these two symphonies are among his most beautiful. They
are full of sweeping melodies, lyrical string writing, and lush orchestral textures that
transport you to a different world each time you listen.
And Mehta did an incredible job in leading the BSO through these masterpieces. He brought out
every nuance in the music - from the quiet tenderness of the first movement to the pulsing energy
 of the Allegroagio to the breathtaking crescendo of the Finale.
The entire experience was absolutely magical: I couldn't have asked for a better way to spend a
Saturday night! And I would highly recommend this performance for anyone who is a fan of
Tchaikovsky or classical music in general.
```

Table 5: Qualitative examples of SSD-2-DOLLY's responses to instructions. We show the first and last prompt from the Vicuna test set. The outputs of our diffusion chat model have an overall reasonable format and content, though being inaccurate in details like the conductor's name and the tempo terminology.

Evaluation template used to compare SSD-2's responses with baseline models' responses (§5.2), following Chiang et al. (2023).

```
[Question]
{test instruction}

[The Start of Assistant 1's Answer]
{baseline model's response}

[The End of Assistant 1's Answer]

[The Start of Assistant 2's Answer]
{SSD-2's response}

[The End of Assistant 2's Answer]

[System]
We would like to request your feedback on the
performance of two AI assistants in response to
the user question displayed above.
Please rate the helpfulness, relevance, accuracy,
 level of details of their responses. Each
assistant receives an overall score on a scale
of 1 to 10, where a higher score indicates
better overall performance.
Please first output a single line containing
only two values indicating the scores for
Assistant 1 and 2, respectively. The two scores
are separated by a space. In the subsequent line,
 please provide a comprehensive explanation of
your evaluation, avoiding any potential bias and
 ensuring that the order in which the responses
were presented does not affect your judgment.
```

Evaluation template used to rate responses from either the baseline models or SSD-2 w.r.t. different attributes (§5).

```
Rate the response below to an instruction, from
the aspects of relevance, factuality,
informativeness, conciseness, coherence,
understandability, and overall human preference,
 each on a scale of 10 (format: x/10).

========

Instruction: {test instruction}

Response: {model response}

========

Please give the ratings now.
```

Table 6: Evaluation templates used in §5.2 and §5. The first template was used with GPT-4 (temperature=0.2), whereas the second was used with GPT-3.5-turbo (greedy) since we need significantly more queries across different λ_{user}'s. In the comparative evaluation using the first template, flipping the order of the baseline model's response and SSD-2's response leads to a similar result.