YOUR AUTOREGRESSIVE GENERATIVE MODEL CAN BE BETTER IF YOU TREAT IT AS AN ENERGY-BASED ONE

Anonymous authors
Paper under double-blind review

ABSTRACT

Autoregressive generative models are commonly used, especially for those tasks involving sequential data. They have, however, been plagued by a slew of inherent flaws due to the intrinsic characteristics of chain-style conditional modeling (e.g., exposure bias or lack of long-range coherence), severely limiting their ability to model distributions properly. In this paper, we propose a unique method for training the autoregressive generative model that takes advantage of a well-designed energy-based learning objective. We show that our method is capable of alleviating the exposure bias problem and increase temporal coherence by imposing a constraint which fits joint distributions at each time step. Besides, unlike former energy-based models, we estimate energy scores based on the underlying autoregressive network itself, which does not require any extra network. Finally, thanks to importance sampling, we can train the entire model efficiently without requiring an MCMC process. Extensive empirical results, covering benchmarks like language modeling, neural machine translation, and image generation, demonstrate the effectiveness of the proposed approach.

1 INTRODUCTION

By factorizing the joint distribution into the product of a series of conditional distributions, autoregressive generative models (abbr. ARGMs) (Vaswani et al., 2017; Dai et al., 2019; van den Oord et al., 2016a;b; Salimans et al., 2017; Chen et al., 2018) simplify the difficult challenge of modeling high-dimensional joint distributions. They can be trained efficiently via maximum likelihood and generate samples of exceptional quality, making this technique popular for modeling distributions, especially for sequential data. Nonetheless, despite their potency and flexibility, ARGMs still have inherent weaknesses due to the intrinsic characteristics of chain-style conditional modeling. For example, ARGMs usually suffer from a discrepancy of the input context distributions between the training and inference stages, which causes consequent error propagation (*i.e.*, Exposure Bias (Ranzato et al., 2016; Bengio et al., 2015)). Besides, due to the nature of greedy selection of beam search approximations, the decoded results from ARGMs may also lack in long-range coherence. We consider one approach by which ARGMs could be adapted to reduce these concerns.

Earlier work, both heuristic and theoretical, has already been proposed with those goals. For instance, the exposure bias problem of ARGMs can be alleviated to some extent with scheduled sampling (Bengio et al., 2015; Mihaylova & Martins, 2019), by mixing input contexts from both real data and autoregressive generation, during the training stage. However, this scheme suffers from an over-correcting problem (Zhang et al., 2019). In addition, at the inference stage, beam search makes it possible to choose more diverse candidates, improving the quality of generated sequences. Nevertheless, this results in only marginal improvements in temporal coherence, since ARGMs can only leverage previous decoded contexts without consideration of the whole sequence information. Moreover, setting aside the difficulty in training them, energy-based models (EBMs) have demonstrated their effectiveness in modeling high-dimensional distributions in a variety of machine learning applications (Zhao et al., 2017; Arbel et al., 2021; Gao et al., 2021), without requiring the transformation of the target distribution into a product of conditional distributions. As a result, several studies (Deng et al., 2020; Bakhtin et al., 2021; Durkan & Nash, 2019) attempt to combine EBMs with ARGMs, expecting to benefit from the strengths of both approaches. However, though some positive results were obtained, the existing works preferred a two-stage optimization, which first obtains a well-trained ARGM and then trains an additional EBM based on it. Such an optimiza-

tion strategy does not enable ARGMs to benefit from the properties of EBM in modeling the joint distribution in a temporally more coherent way.

In this paper, we present a novel design for seamlessly integrating **E**nergy**-**based models into **A**uto**R**egressive **M**odels (E-ARM). Our training is based on an energy-based learning objective, which forces ARGMs training to fit the joint distribution along with the conditional one at each time step. Thanks to our well-designed energy function, the two involved models can share a single base network without additional parameters, that is, the base network not only serves as a generator that provides fake data to facilitate the training of EBMs like previous works (Che et al., 2020; Xiao et al., 2021; Durkan & Nash, 2019; Deng et al., 2020), but also plays the role of modeling the energy surface. This property makes it easy to plug E-ARM into the training of any autoregressive generative models.

Intuitively, the exposure bias in ARGMs is caused by the fact that the model is trained on real data rather than data generated by the model. On the other hand, in the EBM's optimization process for modeling joint densities, the negative phase of wake-sleep algorithms (Hinton, 2002; Kim & Bengio, 2016) requires sampling data from the EBM itself. Along with the fact that our method combines the EBM and the ARGM seamlessly as a whole, E-ARM can reduce the discrepancy between input data of the training and inference stage, which mitigates the exposure bias problem of the ARGM. On top of it, unlike ARGMs, which factor the joint distribution into a product of conditional distributions, EBMs are able to model the joint distribution directly and score each input at the sequence level instead of at the token level, which makes them capable of modeling long-range coherence. Additionally, in order to optimize the proposed energy-based learning objective efficiently via gradient-based wake-sleep algorithms (Kim & Bengio, 2016), we present a way to estimate the negative phase gradient (which is a necessary component in the gradient-based wake-sleep algorithms) through those samples generated with the autoregressive view instead of the EBM view, which would require an expensive Markov Chain Monte Carlo (MCMC) process. This allows us to sidestep extremely time-consuming MCMCs, thus accelerating training.

In summary, the following contributions are made with this paper: i) We introduce a novel scheme, E-ARM, to integrate the EBM view into autoregressive generative models seamlessly; ii) with the help of a proposed energy-based learning objective, we compel the ARGM to fit not only the conditional distributions but also the joint distribution at each time step, which can address ARGMs inherent issues, such as exposure bias and weak temporal coherence; iii) We demonstrate how to efficiently optimize our model constructed from a single network, using wake-sleep algorithms without MCMC; iv) In a number of applications, such as language modeling, neural machine translation, and image generation, our model can achieve better results in comparison with relevant baselines.

2 BACKGROUND

2.1 ENERGY-BASED MODELS

Energy-based models (LeCun et al., 2006) can express any probability density $p(\mathbf{x})$ for $\mathbf{x} \in \mathbb{R}^D$ as

$$p_\theta(\mathbf{x}) = \frac{\exp(-\mathbf{E}_\theta(\mathbf{x}))}{\mathbf{Z}_\theta}, \tag{1}$$

where $E_\theta : \mathbb{R}^D \to \mathbb{R}$ denotes an energy function which aims to map a D-dimensional datapoint to a scalar, and $\mathbf{Z}(\theta) = \sum_{\mathbf{x}} \exp(-\mathbf{E}_\theta(\mathbf{x}))$ denotes the normalizing constant, also known as the partition function. Any function can be used as an energy function to represent an EBM as long as it can generate a single scalar given some input $\mathbf{x}$ and the normalizing constant is finite[1]. Wake-sleep algorithms are commonly used to optimize EBMs (Hinton, 2002; Kim & Bengio, 2016; Grathwohl et al., 2020) via gradient-based approximate maximum likelihood. Specifically, the gradient of the log-likelihood, which needs to be maximized, with respect to θ can be expressed as

$$\mathbb{E}_{p_d(\mathbf{x})}\Big[\frac{\partial}{\partial\theta}\log p_\theta(\mathbf{x})\Big] = \mathbb{E}_{p_\theta(\mathbf{x})}\Big[\frac{\partial}{\partial\theta}\mathbf{E}_\theta(\mathbf{x})\Big] - \mathbb{E}_{p_d(\mathbf{x})}\Big[\frac{\partial}{\partial\theta}\mathbf{E}_\theta(\mathbf{x})\Big]. \tag{2}$$

[1]Without constraining the parametrization of $\mathbf{E}_\theta$, this can be achieved by bounding the region of space in which x takes its allowed values.

The first term in the right hand side of Eq. 2 is the negative phase term while the second term is called the positive phase term. MCMC methods have been used (Hinton, 2002; Welling & Teh, 2011) to approximately sample from $p_\theta(\mathbf{x})$, for estimating the negative phase term.

2.2 MODELING DISTRIBUTIONS AUTOREGRESSIVELY

Autoregressive generative models (ARGM)[2] can decompose any joint distribution $p(\mathbf{x})$ into a product of conditional distributions using the product rule of probability by ordering those random variables within the joint distribution and characterizing each random variable given all variables preceding it in that order. Formally, we use $\mathbf{x}_{<k}$ to denote the vector variable covering all random variables before the time step k and use x_k denote the random variable at time step k. Then we have

$$p(\mathbf{x}) = \prod_{k=1}^{K} p(x_k|\mathbf{x}_{<k}). \tag{3}$$

In general, modeling distributions autoregressively has achieved remarkable accomplishments in numerous areas (Vaswani et al., 2017; Radford et al., 2019; van den Oord et al., 2016c;b; Salimans et al., 2017) thanks to its ability to avoid the challenging target of modeling joint high-dimensional distributions directly. We primarily focus on autoregressive language models in this paper, but we also conduct experiments on image generation to validate the generality of our method.

2.3 EXPOSURE BIAS AND INCOHERENCE PROBLEMS IN AUTOREGRESSIVE MODELS

In the discussion about the defects of sequential autoregressive generative models, the exposure bias problem (Bengio et al., 2015; Ranzato et al., 2016) is an important issue, which greatly affects the model's deployment performance. During the training stage, the autoregressive model is always conditioned on ground truth token sequences. In generation stage however, the model has to rely on its own previously generated tokens to predict the next token, when the model is deployed. If an incorrect token is selected, this error can be amplified in following steps because the next prediction will be made using an unusual input (one unlike those in the training set). Besides, autoregressive decoding typically greedily selects the most probable token at each time step, given the ones previously selected. Such a scheme assumes the largest joint probability of the whole sequence can be achieved by separately choosing the most probable next token (given its previous context) over all time steps, which is de facto unreliable. Correspondingly, the chosen sequence is not always the model's optimum result.

3 INTEGRATE EBMS INTO AUTOREGRESSIVE MODELS SEAMLESSLY

For a long time, as a result of compromises for improving training stability and efficiency (*e.g.*, modeling a joint distribution by decomposing it and using a teacher-forcing training strategy), conventional autoregressive generative models have suffered from flaws such as the exposure bias and the lack of long-range coherence. To tackle these issues, we attempt to seamlessly integrate **E**nergy-based models into **A**uto**R**egressive **M**odels (E-ARM), which can be regarded as a variant of autoregressive generative models blending with an energy-based learning objective. Given a joint sequential distribution, E-ARM also addresses it autoregressively, that is, tackling tokens step by step under a specific order. However, what differs from conventional ARGMs is that we attempt to model both the conditional and the joint distributions simultaneously at each time step. In this way, E-ARM can model distributions conveniently in an autoregressive manner while avoiding those potential problems brought by ARGMs.

Formally, given a sequence of random variables $(x_1, x_2, \ldots, x_K)$ with length K, we introduce a parametric autoregressive model $q_\theta(\mathbf{x}_{<k}) = \prod_{l=1}^{k-1} q_\theta(x_l|\mathbf{x}_{<l})$ (k denotes the time step) with parameters θ. Particularly, we define $\tilde{q}_\theta(\mathbf{x}_{<k}) = \prod_{l=m}^{k-1} q_\theta(x_l|\mathbf{x}_{<l}) \prod_{n=1}^{m-1} q(x_n|\mathbf{x}_{<n})$, which means only those conditional distributions $q_\theta(x_l|\mathbf{x}_{<l})$ of the most recent $k-m$ time steps are involved in the current update of parameters θ while those distant conditional distributions $q(x_n|\mathbf{x}_{<n})$ are

[2]In this paper, the term "autoregressive model" is sometimes used to denote the autoregressive generative model for convenience.

treated as fixed (The rationale behind such a design will be elaborated in Sec.4). Then, we define $p_\theta(x_k, \mathbf{x}_{<k})$ as a product of the autoregressive model and an EBM as follows,

$$p_\theta(x_k, \mathbf{x}_{<k}) = \tilde{q}_\theta(\mathbf{x}_{<k}) \cdot \frac{e^{-\phi_\theta(x_k, \mathbf{x}_{<k})}}{\mathbf{Z}_\theta}, \tag{4}$$

where the energy function $\phi_\theta(x_k, \mathbf{x}_{<k})$ is defined as the x_k's negative corresponding component of the base network's output logit with the input prefix context $\mathbf{x}_{<k} = (x_1, x_2, \ldots, x_{k-1})$ (*e.g.*, given a sequence "This is Friday." and assuming the corresponding index of the token "Friday" in the vocabulary is i, then the value of $-\phi_\theta$("Friday", "This is") is the i-th component of the output logit, which is the straight input tensor of the final softmax layer), and the normalization term $\mathbf{Z}_\theta = \mathbb{E}_{\mathbf{x}'_{<k} \sim \tilde{q}_\theta(\mathbf{x}_{<k})}[\sum_{x_k} e^{-\phi_\theta(x_k, \mathbf{x}'_{<k})}]$.

Our primary goal is to make the distribution $q_\theta(x_k|\mathbf{x}_{<k})$ to approach the real conditional $p_d(x_k|\mathbf{x}_{<k})$ whilst maintaining $p_\theta(x_k, \mathbf{x}_{<k})$ as close to the real joint $p_d(x_k, \mathbf{x}_{<k})$ as possible at each time step, which can be achieved by minimizing the Kullback-Leibler (KL) divergence between the distributions,

$$\theta^* = \arg\min_\theta \Big[\mathbf{D}_{KL}\Big(p_d(x_k|\mathbf{x}_{<k})||q_\theta(x_k|\mathbf{x}_{<k})\Big) + \lambda \mathbf{D}_{KL}\Big(p_d(x_k, \mathbf{x}_{<k})||p_\theta(x_k, \mathbf{x}_{<k})\Big)\Big], \tag{5}$$

where λ adjusts the ratio between the two objectives. In Eq. 5, the first objective can be easily optimized by cross entropy while the second objective is optimized in the sense of EBMs by wake-sleep algorithms (Hinton et al., 1995; Kim & Bengio, 2016), which minimizes the objective by descending the following gradient of θ according to Eq. 2[3]

$$\underbrace{\mathbb{E}_{x_k, \mathbf{x}_{<k} \sim p_d(x_k, \mathbf{x}_{<k})}\left[\frac{\partial}{\partial\theta}\mathbf{E}_\theta(x_k, \mathbf{x}_{<k})\right]}_{\textbf{Positive Phase}} - \underbrace{\mathbb{E}_{x_k, \mathbf{x}_{<k} \sim p_\theta(x_k, \mathbf{x}_{<k})}\left[\frac{\partial}{\partial\theta}\mathbf{E}_\theta(x_k, \mathbf{x}_{<k})\right]}_{\textbf{Negative Phase}}, \tag{6}$$

where we have $\mathbf{E}_\theta(x_k, \mathbf{x}_{<k}) = \phi_\theta(x_k, \mathbf{x}_{<k}) - \log \tilde{q}_\theta(\mathbf{x}_{<k})$. Optimization via Eq. 2 or 6 involves sampling data from the model and can thus lead to the discovery of non-data-like samples, whose likelihood is then explicitly reduced by the energy function. E-ARM is therefore not plagued by the exposure bias problem. Besides, because we model the joint distribution throughout the training process, E-ARM can assess the entire sequence as a whole and generate more coherent data using energy sampling (Deng et al., 2020).

4 OPTIMIZATION

In this section, we present how to efficiently optimize E-ARM. To begin with, we optimize the first objective in Eq. 5 as with conventional autoregressive models by reducing the per time-step cross-entropy loss. As for the second objective, we resort to descend the estimated gradient as shown in Eq. 6. Thanks to the importance sampling technique and our well-defined energy function, we now show that an improved version of Eq. 6 has a simple and symmetric form that can be easily estimated whilst not requiring an expensive MCMC.

Specifically, by replacing $\mathbf{E}_\theta(x_k, \mathbf{x}_{<k})$ with the specific form $\phi_\theta(x_k, \mathbf{x}_{<k}) - \log \tilde{q}_\theta(\mathbf{x}_{<k})$, the gradient w.r.t. θ in the positive phase of Eq. 6 can be written into

$$-\mathbb{E}_{\mathbf{x}_{<k} \sim p_d}[\frac{\partial}{\partial\theta} \log \tilde{q}_\theta(\mathbf{x}_{<k})] + \mathbb{E}_{x_k, \mathbf{x}_{<k} \sim p_d}[\frac{\partial}{\partial\theta}\phi_\theta(x_k, \mathbf{x}_{<k})]. \tag{7}$$

Similarly, we can get the negative phase gradient as

$$-\mathbb{E}_{\mathbf{x}_{<k} \sim p_\theta}[\frac{\partial}{\partial\theta} \log \tilde{q}_\theta(\mathbf{x}_{<k})] + \mathbb{E}_{x_k, \mathbf{x}_{<k} \sim p_\theta}[\frac{\partial}{\partial\theta}\phi_\theta(x_k, \mathbf{x}_{<k})]. \tag{8}$$

The first term $-\mathbb{E}_{\mathbf{x}_{<k} \sim p_\theta}[\frac{\partial}{\partial\theta} \log \tilde{q}_\theta(\mathbf{x}_{<k})]$ in Eq. 7 is equivalent to the log-likelihood gradient of $\tilde{q}_\theta(\mathbf{x}_{<k})$, which means improvements in this direction will be automatically taken care of as a result of steps arising from the gradient of the first KL-divergence in Eq. 5, albeit at the expense of

[3]here, we take a minimization version of the Eq. 2. Thus the sign before each phase is converse.

changing the weight given to the second vs. the first KL, λ. Besides, because the estimation of the expectation operator over the data distribution p_d is easy, and the score $\phi_\theta(x_k, \mathbf{x}_{<k})$ can be acquired simply accessing the output logit of ARGM (see the definition of ϕ_θ in Sec. 3), the second term can likewise be readily estimated and optimized. As a result, the positive phase optimization is both feasible and efficient.

The negative phase gradient estimation, on the other hand, is more involved. In Eq. 8, sampling data from p_θ is required for estimating the expectation $\mathbb{E}_{p_\theta}$, whereas p_θ is a parametric joint density involving an energy-based unnormalized density estimator that may require time-consuming MCMC methods to generate data. However, thanks to importance sampling, we can substitute the troublesome computation of the expectation over the distribution p_θ with the expectation over the distribution q_θ, which can generate samples autoregressively without MCMC. Formally, the negative phase gradient $\mathbb{E}_{x_k, \mathbf{x}_{<k} \sim p_\theta}[\frac{\partial}{\partial \theta} \mathbf{E}_\theta(x_k, \mathbf{x}_{<k})]$ is equivalent to the following formulation (See the detailed derivation in Appendix A),

$$-\mathbb{E}_{\mathbf{x}_{<k} \sim \tilde{q}_\theta(\mathbf{x}_{<k})}[\mathbf{w}(\mathbf{x}_{<k}) \frac{\partial}{\partial \theta} \log \tilde{q}_\theta(\mathbf{x}_{<k})] + \mathbb{E}_{x_k, \mathbf{x}_{<k} \sim \tilde{q}_\theta(x_k, \mathbf{x}_{<k})}[\mathbf{w}(\mathbf{x}_{<k}) \frac{\partial}{\partial \theta} \phi_\theta(x_k, \mathbf{x}_{<k})], \quad (9)$$

$$\textbf{where} \quad \mathbf{w}(\mathbf{x}_{<k}) = \frac{\sum_{x_k} e^{-\phi(x_k, \mathbf{x}_{<k})}}{\mathbb{E}_{\mathbf{x}'_{<k} \sim \tilde{q}_\theta(\mathbf{x}_{<k})}[\sum_{x_k} e^{-\phi_\theta(x_k, \mathbf{x}'_{<k})}]}. \quad (10)$$

According to Eq. 9, all the estimated expectations only need sampling from the autoregressive model rather than the joint distribution, and the reweighing weight $\mathbf{w}$ in Eq. 10 also does not involve expectation computation over distribution p_θ. Generally, producing data from an autoregressive model is a very simple ancerstral sampling process, as compared with sampling straight from an EBM, which needs MCMC approaches (Durkan & Nash, 2019). On account of that, the optimization process can be much more efficient.

Besides, the term $\mathbb{E}_{\mathbf{x}_{<k} \sim \tilde{q}_\theta(\mathbf{x}_{<k})}[\mathbf{w}(\mathbf{x}_{<k}) \frac{\partial}{\partial \theta} \log \tilde{q}_\theta(\mathbf{x}_{<k})]$ in Eq. 9 is equivalent to a re-weighted version of the gradient of q_θ's information entropy with respect to θ. This term can be optimized similarly to the teacher-forcing training of autoregressive model with the "teacher" sequence generated autoregressively by the model itself. Actually, the scheduled sampling methods (Bengio et al., 2015; Ranzato et al., 2016; Mihaylova & Martins, 2019) are similar to this term but without the reweighting factor. Furthermore, it is worth noting that for a sequence with total length K, since we add a constraint to fit the joint distribution p_θ at each time step k, Eq. 9 actually has K counterparts with different time steps. If we use the $q_\theta(\mathbf{x}_{<k})$ directly instead of $\tilde{q}_\theta(\mathbf{x}_{<k})$ in the Eq. 4 to define $p_\theta(x_k, \mathbf{x}_{<k})$, due to the fact that the distribution $q_\theta(\mathbf{x}_{<k})$ modeled by an autoregressive model can be naturally broken up into pieces, simply summing up these K gradients results in the term

$$\sum_{k=1}^{K} \mathbb{E}_{q_\theta(\mathbf{x}_{<k})}[\mathbf{w}(\mathbf{x}_{<k}) \frac{\partial}{\partial \theta} \log q_\theta(\mathbf{x}_{<k})] = \sum_{l=1}^{K} \sum_{k=1}^{K+1-l} \mathbb{E}_{q_\theta(\mathbf{x}_{<k})}[\mathbf{w}(\mathbf{x}_{<k}) \frac{\partial}{\partial \theta} \log q_\theta(x_l | \mathbf{x}_{<l})], \quad (11)$$

where l indicates the specific index of the current token in the entire sequence. As a result, earlier time steps (smaller l) will get stronger training signals (larger $K + 1 - l$, indicating more gradient terms), giving rise to imbalanced training for different time steps. To solve this, we introduce $\tilde{q}_\theta(\mathbf{x}_{<k})$ as $\prod_{l=m}^{k-1} q_\theta(x_l | \mathbf{x}_{<l}) \prod_{n=1}^{m-1} q(x_n | \mathbf{x}_{<n})$ to define $p_\theta(x_k, \mathbf{x}_{<k})$ shown in Sec. 3, allowing gradients only back propagate through conditional distributions w.r.t. a few recent tokens[4]. This explains our proposal of using $\tilde{q}_\theta(\mathbf{x}_{<k})$ to define $p_\theta(x_k, \mathbf{x}_{<k})$.

Ultimately, combining Eq. 7 and Eq. 9, we can optimize $p_\theta(x_k, \mathbf{x}_{<k})$ via descending the estimated gradient of θ as follows,

$$\underbrace{\begin{pmatrix} -\ \mathbb{E}_{\mathbf{x}_{<k} \sim p_d}[\frac{\partial}{\partial \theta} \log \tilde{q}_\theta(\mathbf{x}_{<k})] \\ +\ \mathbb{E}_{x_k, \mathbf{x}_{<k} \sim p_d}[\frac{\partial}{\partial \theta} \phi_\theta(x_k, \mathbf{x}_{<k})] \end{pmatrix}}_{\textbf{Positive Phase}} - \underbrace{\begin{pmatrix} -\ \mathbb{E}_{\mathbf{x}_{<k} \sim \tilde{q}_\theta(\mathbf{x}_{<k})}[\mathbf{w}(\mathbf{x}_{<k}) \frac{\partial}{\partial \theta} \log \tilde{q}_\theta(\mathbf{x}_{<k})] \\ +\ \mathbb{E}_{x_k, \mathbf{x}_{<k} \sim \tilde{q}_\theta(x_k, \mathbf{x}_{<k})}[\mathbf{w}(\mathbf{x}_{<k}) \frac{\partial}{\partial \theta} \phi_\theta(x_k, \mathbf{x}_{<k})] \end{pmatrix}}_{\textbf{Negative Phase}}. \quad (12)$$

[4]In practice, we find that using recent 2 tokens worked best.

From Eq. 12, we can see that the only difference between two phases is that in the negative phase, the expectation over $\tilde{q}_\theta$ have a reweighing weight $\mathbf{w}$ for each sample. The reweighing weight $\mathbf{w}$ in Eq. 10 and Eq. 12 can be further refined (see the derivation in Appendix B) and we can observe that

$$\mathbf{w}(\mathbf{x}_{<k}) = \frac{\mu(\mathbf{x}_{<k})}{\mathbb{E}_{\mathbf{x}'_{<k}} \mu(\mathbf{x}_{<k})}, \tag{13}$$

where $\mu(\mathbf{x}_{<k}) = \frac{p_\theta(\mathbf{x}_{<k})}{\tilde{q}_\theta(\mathbf{x}_{<k})}$ indicates the possibility of which distribution the prefix context $\mathbf{x}_{<k}$ is most likely to come from, the distribution p_θ or the distribution $\tilde{q}_\theta$. Correspondingly, $\mathbf{w}(\mathbf{x}_{<k})$ reflects the context $\mathbf{x}_{<k}$'s relative magnitude of $\mu(\mathbf{x}_{<k})$ compared with the average among all potential contexts—the larger the value of $\mathbf{w}(\mathbf{x}_{<k})$, the more likely the context $\mathbf{x}_{<k}$ in the data space coming from p_θ, which is modeled by the product of autoregressive models and EBMs. During training, those input sequences with contexts more likely under p_θ than q_θ will be assigned larger weights $\mathbf{w}$ while those input sequences with contexts that are less likely under p_θ than q_θ will be assigned smaller weights $\mathbf{w}$.

In general, E-ARM ought to be viewed as a regularizer that ensures our base autoregressive network stays close to the real distribution p_d. We found that training from scratch with the E-ARM objective alone did not work well because of variance in the importance weights when $\tilde{q}_\theta$ does not closely match p_θ. The expectation estimation is then dominated by the examples with the larger weight. Moreover, importance weights lead to underestimating the gradient w.r.t. θ in the negative phase when $\tilde{q}_\theta$ does not fully cover regions of high density under p_θ (Salakhutdinov & Murray, 2008). As a result, in order to make the optimization more stable, we must maintain the cross-entropy loss low throughout training and pre-train as a pure ARGM for a few epochs before introducing the E-ARM objective. Actually, The starting epoch of E-ARM is a hyper-parameter, and we discuss it in the Sec. 5.2.

Following the excellent work of Deng et al. (2020); Bakhtin et al. (2021), we also adopt Top-K energy re-sampling in the inference stage, which means that in the generative process, we first gather multiple candidate sequences generated autoregressively, and then re-sample from them based on their energy scores estimated by the network's logit at the last time step where the entire sequence has been processed. Since we employ the EBM to model the joint distribution at each time step, such a re-sampling strategy can mitigate the undesirable impact of the greedy selection of one token at a time, and we found this variation to increase the coherence of generated samples.

5 EXPERIMENTS

To empirically corroborate the effectiveness of E-ARM and show its broad applicability, we conduct extensive experiments covering three machine learning applications, which are neural machine translation (NMT), language modeling, and image generation. In this section, we will introduce the three corresponding experimental setups, followed by an analysis of the obtained results. We will release the source code once upon acceptance.

5.1 APPLICATION TO NEURAL MACHINE TRANSLATION

E-ARM is first evaluated in the context of neural machine translation (NMT), which is a conditional generation task and is important in the natural language processing (NLP) field. We first analyze E-ARM on the IWSLT14 dataset, which includes six different language pairs ({German, Spanish, Italian} $\rightarrow$ English and English $\rightarrow$ {German, Spanish, Italian}). In addition, we test E-ARM on the WMT16 (English $\rightarrow$ German) benchmark to make sure we evaluating E-ARM on a larger dataset. Hereafter we abbreviate English, German, Spanish, Italian as "En", "De", "Es", "It". The weight λ in Eq. 5 is set as 0.05 for all translation tasks. We use one size of transformer ("Base-IWSLT") for the IWSLT14 benchmark and two sizes of transformer ("Base-WMT", "Large-WMT") for the WMT16 benchmark [5]. Scheduled Sampling is carried out following Mihaylova & Martins (2019).

The results of IWSLT14 tasks are shown in Table 1. We test not only the pure performance of E-ARM but also the compatibility with other techniques. Specifically, we can observe that (1) without any particular engineering, E-ARM outperforms the base autoregressive translation model

[5]The implementation is developed on Fairseq (Ott et al., 2019).

Model	Label Smoothing	Scheduled Sampling	Beam Searching	BLEU ↑						Avg.
				DE→ EN	EN→ DE	EN→ IT	IT→ EN	ES→ EN	EN→ ES	
Base	-	-	-	32.44±0.06	26.64±0.10	27.92±0.03	30.48±0.08	38.61±0.11	35.42±0.09	31.92
			5 B	33.62±0.07	27.41±0.08	28.72±0.04	31.39±0.05	39.55±0.12	36.38±0.07	32.85
	✔	-	-	33.68±0.03	27.62±0.04	28.81±0.07	31.42±0.07	39.85±0.13	36.71±0.09	33.02
			5 B	34.61±0.08	28.46±0.06	29.72±0.10	32.29±0.03	40.64±0.07	37.48±0.05	33.87
	✔	✔	-	34.23±0.06	27.96±0.03	29.26±0.11	31.93±0.08	40.16±0.03	37.21±0.04	33.46
			5 B	35.10±0.04	28.73±0.04	29.97±0.07	32.64±0.12	40.91±0.06	37.93±0.10	34.21
E-ARM	-	-	-	32.99±0.10	27.15±0.03	28.33±0.12	31.13±0.04	39.56±0.01	36.07±0.02	32.54
			5 B	34.06±0.06	27.97±0.08	29.26±0.09	31.90 ±0.13	40.30 ±0.03	36.92 ±0.09	33.40
	✔	-	-	33.97 ±0.08	28.03 ±0.04	29.13 ±0.02	31.84 ±0.11	40.32 ±0.03	36.96 ±0.07	33.38
			5 B	34.93 ±0.05	28.91 ±0.12	30.04 ±0.11	32.56 ±0.04	41.01 ±0.06	37.73 ±0.12	34.20
	✔	✔	-	34.58 ±0.09	28.38 ±0.12	29.56 ±0.10	32.11 ±0.03	40.93 ±0.03	37.56 ±0.07	33.85
			5 B	**35.36** ±0.05	**29.11** ±0.04	**30.25** ±0.09	**32.82** ±0.11	**41.58** ±0.07	**38.19** ±0.03	**34.55**

Table 1: Comparison of BLEU scores between our approach E-ARM and the base ARGM trained just with cross-entropy loss on six translation pairs of IWSLT14 datasets. We use "-" to denote that the training trick is not used while "✔" indicates we use it. "**5 B**" represents we use beam searching with 5 beams.

trained with cross-entropy singly by 0.62 (31.92 → 32.54) BLEU points in average, especially on three translation pairs—38.61 → 39.56 on Spanish-to-English, 30.48 → 31.13 on Italian-to-English, 35.42 → 36.07 on English-to-Spanish. (2) E-ARM is compatible with other techniques like scheduled sampling, which can help alleviate the exposure bias problem to some extent. They are not mutually exclusive and can work together to further improve the performance of the base ARGM. (3) However, since scheduled sampling can reduce exposure bias and beam search can somewhat alleviate the flaws caused by greedy selection at each time step, the performance gain of E-ARM when all these tactics are combined is only 0.34 (34.21 → 34.55), which is lower than the 0.62 (31.92 → 32.54) obtained when the model is purely trained without these other techniques.

Model	L.S.	S.S.	w/E-ARM	BLEU ↑
Base-WMT	-	-	-	27.56
	✔	-	-	28.04
	✔	✔	-	28.36
	✔	✔	✔	**28.62**
Large-WMT	-	-	-	28.70
	✔	-	-	29.05
	✔	✔	-	29.23
	✔	✔	✔	**29.44**

Table 2: Translation performance of proposed E-ARM on WMT16 English→German, evaluated with BLEU. We uniformly use 5 beams when applying beam search. "**L.S.**" denotes Label Smoothing and "**S.S.**" denotes Scheduled Sampling.

Additionally, Table 2 shows the performance of E-ARM on the WMT16 English → German task. For two different model sizes, enabling label smoothing (L.S.) improves model performance by 0.52 and 0.35, respectively. The performance of the base transformer model further increases to 28.36 BLEU points when scheduled sampling (S.S.) is used, while the larger model improves to 29.23 points. E-ARM paired with label smoothing and scheduled sampling yields the highest scores of 28.62 and 29.44, respectively. Overall, our training strategy outperforms ARGM's vanilla teacher-forcing training and can have uniformly favorable impacts across different models and dataset sizes.

5.2 Application to Language Modeling

To further demonstrate E-ARM's consistency in reducing flaws of autoregressive generative models, we also conduct language modeling experiments. The WikiText-103 dataset (Merity et al., 2017), which is the largest word-level language modeling benchmark with long-term dependency, was chosen as the testbed. It comprises 103 million

Model	#Params	PPL ↓
Tr-Base	156M	30.56
Tr-Base (w/E-ARM)	156M	**29.89**
Standard Tr-XL	151M	24.20
Standard Tr-XL (w/E-ARM)	151M	**23.81**

Table 3: Language modeling performance of different models on WikiText103. Evaluation is conducted using perplexity (PPL).

training tokens from 28 thousand articles, with an average length of 3.6 thousand tokens per article, which allows model to evaluate the ability of modeling long-term dependency. Two network structures are mainly tested, which are Transformer-Base (Vaswani et al., 2017) and Transformer-XL (Dai et al., 2019) (Tr-Base and Tr-XL for short respectively hereafter).

The final results are reported in Table 3. We can see from the results that E-ARM outperforms baselines with clear margins for different types of models. Specifically, the Transformer-Base improves performance by 0.67 PPL points (from 30.56 to 29.89), while the Transformer-XL improves model by 0.20 PPL points (from 24.20 to 23.81). Our strategy does not change the structure of the base network nor introduces any additional module or learnable parameters, therefore we can conclude that the performance boost is solely from the introduced energy-based learning objective.

		Start Epoch		
		5	**15**	**25**
$\boldsymbol{\lambda}$	**0.00**	30.56	30.56	30.56
	0.01	30.48	30.12	30.22
	0.05	30.43	**29.89**	30.16
	0.1	30.60	30.03	30.14
	0.5	30.71	30.36	30.47

Table 4: How different λ and the E-ARM start epoch (when we introduce the E-ARM into the training on WikiText103) affect performance evaluated by perplexity (PPL). The Tr-Base model structure is used and is train 40 epochs in total.

In addition, we study the effect of hyper-parameter settings on the performance of language modeling, which can be seen in Table 4. From this, we may deduce that starting E-ARM training at the 15-th epoch yields the best results, whereas starting earlier or later yields a performance decline. It is reasonable because, if E-ARM was introduced too early, the autoregressive model may not have been optimized well at that moment. As a result, generative quality would be terrible, and make energy-based training unstable. On the other hand, the underlying autoregressive model can be modified only marginally if E-ARM is introduced when the ARGM training is virtually complete. Besides, from the vertical perspective which presents the impact of different λ, we can observe that the best λ in Eq. 5 is 0.05. The first line of the table indicates the baseline of training the autoregressive model with pure cross-entropy loss.

5.3 Application to Image Generation

In order to illustrate the effectiveness and generality of our method in processing different modality tasks, we further show the results of applying E-ARM to image generation in this section. We apply E-ARM to Pixel-CNN (Van Oord et al., 2016) and its variant Gated Pixel-CNN (Oord et al., 2016). Experiments are carried out on the MNIST and CIFAR-10 datasets.

Model	**Test (Train) NLL ↓**	
	MNIST	**CIFAR-10**
Pixel-CNN	0.17 (0.13)	3.14 (3.08)
Pixel-CNN (w/E-ARM)	**0.15 (0.12)**	**3.07 (2.98)**
Gated Pixel-CNN	0.14 (0.11)	3.03 (2.90)
Gated Pixel-CNN (w/E-ARM)	**0.12 (0.10)**	**2.97 (2.91)**

Table 5: Performance of E-ARM with different base networks on MNIST and CIFAR-10 in bits/dim (lower is better), training performance in brackets.

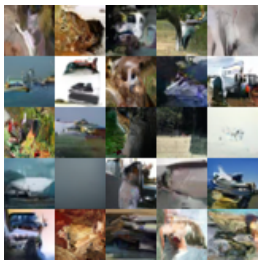

Figure 1: Samples of CIFAR-10 from Gated Pixel-CNN (w/E-ARM).

Table 5 summarizes the quantitative results measured by per-pixel negative log-likelihood (NLL), while Figure 1 depicts some of the generated samples. We can see that with the help of our E-ARM, both the Pixel-CNN and the Gated Pixel-CNN can obtain improvements in all datasets (0.17 $\rightarrow$ 0.15 and 3.14 $\rightarrow$ 3.07 for Pixel-CNN on MNIST and CIFAR10 respectively and 0.14 $\rightarrow$ 0.12 and 3.03 $\rightarrow$ 2.97 for Gated Pixel-CNN on MNIST and CIFAR10 respectively). This is further evidence in favour of the energy-based learning objective for improving autoregressive models.

6 RELATED WORKS

6.1 AUTOREGRESSIVE GENERATIVE MODELS

Modeling high-dimensional data distributions directly is usually a rather challenging task due to "the curse of dimensionality" (Bellman, 1954). One alternative method is to sequentialize the random variables and then factorize the joint probability distribution into the product of conditionals based on the sequence structure, which is exactly the core idea of autoregressive generative models (ARGMs).

ARGMs have been very successful, in particular for sequential data. For example, ARGMs have been widely used in language modeling (Vaswani et al., 2017; Dai et al., 2019; Radford et al., 2019), audio synthesis (van den Oord et al., 2016a), and even image generation (van den Oord et al., 2016c;b; Salimans et al., 2017). The advantages of ARGMs are however balanced by issues of (1) exposure bias (Ranzato et al., 2016; Bengio et al., 2015; Song et al., 2020), due to the discrepancy in input context distributions between the training and inference stages, and (2) weak long-range coherence, due to the inherent greedy selection of one token at a time without look-ahead.

6.2 ENERGY-BASED MODELS

In the field of generative modeling, energy-based models (EBMs) have been widely used (Zhao et al., 2017; Arbel et al., 2021; Gao et al., 2021). The primary idea behind EBMs is to decompose the dependencies between variables (*e.g.* images and labels) through different terms of an energy function, assigning low energies to proper configurations found in the dataset, while assigning high energies to incorrect or unseen ones (LeCun et al., 2006).

Due to the challenge of sampling from EBMs, training EBMs by wake-sleep algorithms (Hinton, 2002; Kim & Bengio, 2016; Grathwohl et al., 2021), which require expensive MCMC approaches, has been notoriously difficult, especially on high-dimensional data like images or texts. Stochastic Gradient Langevin Dynamics (SGLD) (Welling & Teh, 2011) is a frequently used gradient-based MCMC approach that injects noise into parameter updates and anneals the step size during the course of training, and which has been adopted in numerous prior works (Nijkamp et al., 2019; Du & Mordatch, 2019; Grathwohl et al., 2020). However, these gradient-based MCMC methods require enormous extra computing overheads and are not applicable when the input is discrete like for text sequences (Deng et al., 2020).

As a result, a variety of recent works attempt to explore the strategy of training an EBM without MCMC. In particular, Bakhtin et al. (2021); Xu et al. (2021) optimize the EBMs by using noise contrastive estimation (NCE) (Gutmann & Hyvärinen, 2010; Ma & Collins, 2018). Durkan & Nash (2019) estimate the intractable normalization component by utilizing ARGMs and importance sampling. Che et al. (2020); Wang et al. (2021) skirt the challenge of collecting data in the high-dimensional data space by producing data in the lower-dimensional feature space, which improves sampling efficiency.

7 CONCLUSIONS AND FUTURE WORK

In this paper, we propose a novel method dubbed E-ARM to integrate energy-based models into autoregressive generative models seamlessly, with an energy-based training objective that exploits an underlying autoregressive model. This is achieved by defining the energy function from the output logits of the base autoregressive network, to model the unnormalized joint distribution of the subsequence up to each time step. We also found ways to improve training of E-ARM using importance sampling, avoiding the requirement of MCMC for the energy-based training. Experimental results on two language tasks and one vision task demonstrate the effectiveness of E-ARM to alleviate exposure bias and incoherence problems of ARGMs. In the future, we expect to extend E-ARM on other sequential generation tasks (*e.g.* text summarization, audio generation), and incorporate the proposed methodology into other advanced autoregressive architectures.

REFERENCES

Michael Arbel, Liang Zhou, and Arthur Gretton. Generalized energy based models. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*, 2021.

Anton Bakhtin, Yuntian Deng, Sam Gross, Myle Ott, Marc'Aurelio Ranzato, and Arthur Szlam. Residual energy-based models for text. *J. Mach. Learn. Res.*, 22:40:1–40:41, 2021.

Richard Ernest Bellman. *The Theory of Dynamic Programming*. RAND Corporation, Santa Monica, CA, 1954.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pp. 1171–1179, 2015.

Tong Che, Ruixiang Zhang, Jascha Sohl-Dickstein, Hugo Larochelle, Liam Paull, Yuan Cao, and Yoshua Bengio. Your GAN is secretly an energy-based model and you should use discriminator driven latent sampling. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

Xi Chen, Nikhil Mishra, Mostafa Rohaninejad, and Pieter Abbeel. Pixelsnail: An improved autoregressive generative model. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pp. 863–871, 2018.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime G. Carbonell, Quoc Viet Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pp. 2978–2988, 2019.

Yuntian Deng, Anton Bakhtin, Myle Ott, Arthur Szlam, and Marc'Aurelio Ranzato. Residual energy-based models for text generation. In *8th International Conference on Learning Representations, ICLR 2020*. OpenReview.net, 2020.

Yilun Du and Igor Mordatch. Implicit generation and modeling with energy based models. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 3603–3613, 2019.

Conor Durkan and Charlie Nash. Autoregressive energy machines. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning, ICML 2019*, volume 97, pp. 1735–1744. PMLR, 2019.

Ruiqi Gao, Yang Song, Ben Poole, Ying Nian Wu, and Diederik P. Kingma. Learning energy-based models by diffusion recovery likelihood. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*, 2021.

Will Grathwohl, Kuan-Chieh Wang, Jörn-Henrik Jacobsen, David Duvenaud, Mohammad Norouzi, and Kevin Swersky. Your classifier is secretly an energy based model and you should treat it like one. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*, 2020.

Will Sussman Grathwohl, Jacob Jin Kelly, Milad Hashemi, Mohammad Norouzi, Kevin Swersky, and David Duvenaud. No MCMC for me: Amortized sampling for fast and stable training of energy-based models. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*, 2021.

Michael Gutmann and Aapo Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2010, Chia Laguna Resort, Sardinia, Italy, May 13-15, 2010*, volume 9 of *JMLR Proceedings*, pp. 297–304, 2010.

G. Hinton, P. Dayan, B. Frey, and R. Neal. The "wake-sleep" algorithm for unsupervised neural networks. *Science*, 268 5214:1158–61, 1995.

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

Taesup Kim and Yoshua Bengio. Deep directed generative models with energy-based probability estimation. *CoRR*, abs/1606.03439, 2016.

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

Zhuang Ma and Michael Collins. Noise contrastive estimation and negative sampling for conditional models: Consistency and statistical efficiency. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pp. 3698–3707, 2018.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.

Tsvetomila Mihaylova and André F. T. Martins. Scheduled sampling for transformers. In Fernando Emilio Alva-Manchego, Eunsol Choi, and Daniel Khashabi (eds.), *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28 - August 2, 2019, Volume 2*, pp. 351–356, 2019.

Erik Nijkamp, Mitch Hill, Song-Chun Zhu, and Ying Nian Wu. Learning non-convergent non-persistent short-run MCMC toward energy-based model. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 5233–5243, 2019.

Aaron van den Oord, Nal Kalchbrenner, Oriol Vinyals, Lasse Espeholt, Alex Graves, and Koray Kavukcuoglu. Conditional image generation with pixelcnn decoders. *arXiv preprint arXiv:1606.05328*, 2016.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*, 2019.

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

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. Sequence level training with recurrent neural networks. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016.

Ruslan Salakhutdinov and Iain Murray. On the quantitative analysis of deep belief networks. In *Machine Learning, Proceedings of the Twenty-Fifth International Conference (ICML 2008), Helsinki, Finland, June 5-9, 2008*, volume 307 of *ACM International Conference Proceeding Series*, pp. 872–879. ACM, 2008.

Tim Salimans, Andrej Karpathy, Xi Chen, and Diederik P. Kingma. Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.

Kaitao Song, Xu Tan, and Jianfeng Lu. Neural machine translation with error correction. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, pp. 3891–3897, 2020.

Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew W. Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. In *The 9th ISCA Speech Synthesis Workshop, Sunnyvale, CA, USA, 13-15 September 2016*, pp. 125, 2016a.

Aäron van den Oord, Nal Kalchbrenner, Lasse Espeholt, Koray Kavukcuoglu, Oriol Vinyals, and Alex Graves. Conditional image generation with pixelcnn decoders. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pp. 4790–4798, 2016b.

Aäron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pp. 1747–1756. JMLR.org, 2016c.

Aaron Van Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. In *International Conference on Machine Learning*, pp. 1747–1756. PMLR, 2016.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 5998–6008, 2017.

Yezhen Wang, Bo Li, Tong Che, Kaiyang Zhou, Ziwei Liu, and Dongsheng Li. Energy-based open-world uncertainty modeling for confidence calibration. *CoRR*, abs/2107.12628, 2021.

Max Welling and Yee Whye Teh. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th International Conference on Machine Learning, ICML 2011, Bellevue, Washington, USA, June 28 - July 2, 2011*, pp. 681–688, 2011.

Zhisheng Xiao, Karsten Kreis, Jan Kautz, and Arash Vahdat. VAEBM: A symbiosis between variational autoencoders and energy-based models. In *9th International Conference on Learning Representations, ICLR 2021*. OpenReview.net, 2021.

Minkai Xu, Shitong Luo, Yoshua Bengio, Jian Peng, and Jian Tang. Learning neural generative dynamics for molecular conformation generation. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*, 2021.

Wen Zhang, Yang Feng, Fandong Meng, Di You, and Qun Liu. Bridging the gap between training and inference for neural machine translation. In Anna Korhonen, David R. Traum, and Lluís Màrquez (eds.), *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1*, pp. 4334–4343, 2019.

Junbo Jake Zhao, Michaël Mathieu, and Yann LeCun. Energy-based generative adversarial networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.

A THE DERIVATION OF THE NEGATIVE PHASE GRADIENT

In this section, we show the detailed derivation of Eq. 9. Formally, as shown in Sec. 3, given an autoregressive model $q_\theta(\mathbf{x}_{<k}) = \prod_{l=1}^{k-1} q_\theta(x_l|\mathbf{x}_{<l})$ (k denotes the time step) with parameters θ, we define a product of the autoregressive model and an EBM as follows

$$p_\theta(x_k, \mathbf{x}_{<k}) = \tilde{q}_\theta(\mathbf{x}_{<k}) \cdot \frac{e^{-\phi_\theta(x_k, \mathbf{x}_{<k})}}{\mathbf{Z}_\theta}, \tag{14}$$

where $\tilde{q}_\theta(\mathbf{x}_{<k}) = \prod_{l=m}^{k-1} q_\theta(x_l|\mathbf{x}_{<l}) \prod_{n=1}^{m-1} q(x_n|\mathbf{x}_{<n})$. Under such definition, only those conditional distributions $q_\theta(x_l|\mathbf{x}_{<l})$ of the most recent $k-m$ time steps are involved in the current update of parameters θ while those distant conditional distributions $q(x_n|\mathbf{x}_{<n})$ are treated as fixed. We have explained the rationale and intuition in Sec.4. $\mathbf{Z}_\theta$ is the normalization term and equal to $\mathbb{E}_{\mathbf{x}'_{<k} \sim \tilde{q}_\theta(\mathbf{x}_{<k})}[\sum_{x_k} e^{-\phi_\theta(x_k, \mathbf{x}'_{<k})}]$. The optimization of $p_\theta(x_k, \mathbf{x}_{<k})$ includes two phases, and the gradient w.r.t θ of negative phase is

$$-\mathbb{E}_{\mathbf{x}_{<k} \sim p_\theta}[\frac{\partial}{\partial \theta} \log \tilde{q}_\theta(\mathbf{x}_{<k})] + \mathbb{E}_{x_k, \mathbf{x}_{<k} \sim p_\theta}[\frac{\partial}{\partial \theta} \phi_\theta(x_k, \mathbf{x}_{<k})]. \tag{15}$$

Next, we will show the specific derivation of these two terms in Eq. 15 so that the entire Eq. 15 can be transformed into Eq. 9.

A.1 THE DERIVATION OF THE FIRST TERM

The first term $\mathbb{E}_{\mathbf{x}_{<k} \sim p_\theta}[\frac{\partial}{\partial \theta} \log \tilde{q}_\theta(\mathbf{x}_{<k})]$ can be processed as follows

$$\begin{aligned}
\mathbb{E}_{\mathbf{x}_{<k} \sim p_\theta}[\frac{\partial}{\partial \theta} \log \tilde{q}_\theta(\mathbf{x}_{<k})] &= \sum_{\mathbf{x}_{<k}} p_\theta(\mathbf{x}_{<k}) \frac{\partial}{\partial \theta} \log \tilde{q}_\theta(\mathbf{x}_{<k}) \\
&= \sum_{\mathbf{x}_{<k}} \sum_{x_k} p_\theta(x_k, \mathbf{x}_{<k}) \frac{\partial}{\partial \theta} \log \tilde{q}_\theta(\mathbf{x}_{<k}) \\
&= \sum_{\mathbf{x}_{<k}} \tilde{q}_\theta(\mathbf{x}_{<k}) \frac{\sum_{x_k} e^{-\phi_\theta(x_k, \mathbf{x}_{<k})}}{\mathbf{Z}_\theta} \frac{\partial}{\partial \theta} \log \tilde{q}_\theta(\mathbf{x}_{<k}) \\
&= \mathbb{E}_{\mathbf{x}_{<k} \sim \tilde{q}_\theta(\mathbf{x}_{<k})}[\mathbf{w}(\mathbf{x}_{<k}) \frac{\partial}{\partial \theta} \log \tilde{q}_\theta(\mathbf{x}_{<k})],
\end{aligned} \tag{16}$$

where we have $\mathbf{w}(\mathbf{x}_{<k}) = \frac{\sum_{x_k} e^{-\phi(x_k, \mathbf{x}_{<k})}}{\mathbb{E}_{\mathbf{x}'_{<k} \sim \tilde{q}_\theta(\mathbf{x}_{<k})}[\sum_{x_k} e^{-\phi_\theta(x_k, \mathbf{x}'_{<k})}]}$ because

$$\begin{aligned}
\mathbf{w}(\mathbf{x}_{<k}) &= \frac{\sum_{x_k} e^{-\phi(x_k, \mathbf{x}_{<k})}}{\mathbf{Z}_\theta} = \frac{\sum_{x_k} e^{-\phi(x_k, \mathbf{x}_{<k})}}{\sum_{\mathbf{x}_{<k}} \sum_{x_k} \tilde{q}_\theta(\mathbf{x}_{<k}) e^{-\phi_\theta(x_k, \mathbf{x}_{<k})}} \\
&= \frac{\sum_{x_k} e^{-\phi(x_k, \mathbf{x}_{<k})}}{\sum_{\mathbf{x}_{<k}} \tilde{q}_\theta(\mathbf{x}_{<k}) \sum_{x_k} e^{-\phi_\theta(x_k, \mathbf{x}_{<k})}} \\
&= \frac{\sum_{x_k} e^{-\phi(x_k, \mathbf{x}_{<k})}}{\mathbb{E}_{\mathbf{x}_{<k} \sim \tilde{q}_\theta(\mathbf{x}_{<k})}[\sum_{x_k} e^{-\phi_\theta(x_k, \mathbf{x}_{<k})}]}.
\end{aligned} \tag{17}$$

A.2 THE DERIVATION OF THE SECOND TERM

Then, we tackle the second term $\mathbb{E}_{x_k,\mathbf{x}_{<k}\sim p_\theta}[\frac{\partial}{\partial\theta}\phi_\theta(x_k,\mathbf{x}_{<k})]$ as follows

$$
\begin{aligned}
\mathbb{E}_{p_\theta}\big[\frac{\partial}{\partial\theta}\phi_\theta(x_k,\mathbf{x}_{<k})\big] &= \sum_{x_k,\mathbf{x}_{<k}} p_\theta(x_k,\mathbf{x}_{<k})\frac{\partial}{\partial\theta}\phi_\theta(x_k,\mathbf{x}_{<k}) \\
&= \sum_{x_k,\mathbf{x}_{<k}} p_\theta(x_k,\mathbf{x}_{<k})\frac{\tilde{q}_\theta(x_k,\mathbf{x}_{<k})}{\tilde{q}_\theta(x_k,\mathbf{x}_{<k})}\frac{\partial}{\partial\theta}\phi_\theta(x_k,\mathbf{x}_{<k}) \\
&= \sum_{x_k,\mathbf{x}_{<k}} \tilde{q}_\theta(x_k,\mathbf{x}_{<k})\frac{\tilde{q}_\theta(\mathbf{x}_{<k})\cdot e^{-\phi_\theta(x_k,\mathbf{x}_{<k})}}{\mathbf{Z}_\theta\cdot\tilde{q}_\theta(x_k,\mathbf{x}_{<k})}\frac{\partial}{\partial\theta}\phi_\theta(x_k,\mathbf{x}_{<k}) \\
&= \mathbb{E}_{x_k,\mathbf{x}_{<k}\sim\tilde{q}_\theta(x_k,\mathbf{x}_{<k})}[\frac{e^{-\phi_\theta(x_k,\mathbf{x}_{<k})}}{\tilde{q}_\theta(x_k|\mathbf{x}_{<k})}\cdot\frac{1}{\mathbf{Z}_\theta}\frac{\partial}{\partial\theta}\phi_\theta(x_k,\mathbf{x}_{<k})] \\
&= \sum_{\mathbf{x}_{<k}}\tilde{q}_\theta(\mathbf{x}_{<k})\sum_{x_k}\tilde{q}_\theta(x_k|\mathbf{x}_{<k})\frac{e^{-\phi_\theta(x_k,\mathbf{x}_{<k})}}{\tilde{q}_\theta(x_k|\mathbf{x}_{<k})}\cdot\frac{1}{\mathbf{Z}_\theta}\frac{\partial}{\partial\theta}\phi_\theta(x_k,\mathbf{x}_{<k}) \\
&= \sum_{\mathbf{x}_{<k}}\tilde{q}_\theta(\mathbf{x}_{<k})\sum_{x_k}e^{-\phi_\theta(x_k,\mathbf{x}_{<k})}\cdot\frac{1}{\mathbf{Z}_\theta}\frac{\partial}{\partial\theta}\phi_\theta(x_k,\mathbf{x}_{<k}) \\
&= \mathbb{E}_{\tilde{q}_\theta(\mathbf{x}_{<k})}[\sum_{x_k}\frac{e^{-\phi_\theta(x_k,\mathbf{x}_{<k})}}{\mathbf{Z}_\theta}\frac{\partial}{\partial\theta}\phi_\theta(x_k,\mathbf{x}_{<k})] \\
&= \mathbb{E}_{\tilde{q}_\theta(\mathbf{x}_{<k})}[\sum_{x_k}\frac{e^{-\phi_\theta(x_k,\mathbf{x}_{<k})}}{\sum_{x_k}e^{-\phi_\theta(x_k,\mathbf{x}_{<k})}}\cdot\frac{\sum_{x_k}e^{-\phi_\theta(x_k,\mathbf{x}_{<k})}}{\mathbf{Z}_\theta}\frac{\partial}{\partial\theta}\phi_\theta(x_k,\mathbf{x}_{<k})] \\
&= \mathbb{E}_{\tilde{q}_\theta(\mathbf{x}_{<k})}[\sum_{x_k}\tilde{q}_\theta(x_k|\mathbf{x}_{<k})\mathbf{w}(\mathbf{x}_{<k})\frac{\partial}{\partial\theta}\phi_\theta(x_k,\mathbf{x}_{<k})] \\
&= \mathbb{E}_{\tilde{q}_\theta(\mathbf{x}_{<k})}[\mathbb{E}_{a\sim\tilde{q}_\theta(x_k|\mathbf{x}_{<k})}[\mathbf{w}(\mathbf{x}_{<k})\frac{\partial}{\partial\theta}\phi_\theta(x_k,\mathbf{x}_{<k})]] \\
&= \mathbb{E}_{x_k,\mathbf{x}_{<k}\sim\tilde{q}_\theta(x_k,\mathbf{x}_{<k})}[\mathbf{w}(\mathbf{x}_{<k})\frac{\partial}{\partial\theta}\phi_\theta(x_k,\mathbf{x}_{<k})]
\end{aligned}
\tag{18}
$$

where $\mathbf{w}(\mathbf{x}_{<k})$ is also equal to $\frac{\sum_{x_k}e^{-\phi(x_k,\mathbf{x}_{<k})}}{\mathbf{Z}_\theta}$. Combining Eq. 16 and Eq. 18, we can obtain an equivalent form of the gradient of the negative phase without any expectation over p_θ as

$$
-\mathbb{E}_{\mathbf{x}_{<k}\sim\tilde{q}_\theta(\mathbf{x}_{<k})}[\mathbf{w}(\mathbf{x}_{<k})\frac{\partial}{\partial\theta}\log\tilde{q}_\theta(\mathbf{x}_{<k})] + \mathbb{E}_{x_k,\mathbf{x}_{<k}\sim\tilde{q}_\theta(x_k,\mathbf{x}_{<k})}[\mathbf{w}(\mathbf{x}_{<k})\frac{\partial}{\partial\theta}\phi_\theta(x_k,\mathbf{x}_{<k})], \tag{19}
$$

$$
\textbf{where}\quad \mathbf{w}(\mathbf{x}_{<k}) = \frac{\sum_{x_k}e^{-\phi(x_k,\mathbf{x}_{<k})}}{\mathbb{E}_{\mathbf{x}'_{<k}\sim\tilde{q}_\theta(\mathbf{x}_{<k})}[\sum_{x_k}e^{-\phi_\theta(x_k,\mathbf{x}'_{<k})}]}. \tag{20}
$$

B THE FURTHER REFINEMENT OF $\mathbf{w}$

The reweighing weight $\mathbf{w}$ can be further deduced as

$$
\begin{aligned}
\mathbf{w}(\mathbf{x}_{<k}) &= \frac{\sum_{x_k}e^{-\phi(x_k,\mathbf{x}_{<k})}}{\mathbb{E}_{\mathbf{x}'_{<k}\sim\tilde{q}_\theta(\mathbf{x}_{<k})}[\sum_{x_k}e^{-\phi_\theta(x_k,\mathbf{x}'_{<k})}]} = \frac{\sum_{x_k}\frac{p_\theta(x_k,\mathbf{x}_{<k})}{\tilde{q}_\theta(\mathbf{x}_{<k})}}{\mathbb{E}_{\mathbf{x}'_{<k}\sim\tilde{q}_\theta(\mathbf{x}_{<k})}[\sum_{x_k}\frac{p_\theta(x_k,\mathbf{x}_{<k})}{\tilde{q}_\theta(\mathbf{x}_{<k})}]} \\
&= \frac{\frac{p_\theta(\mathbf{x}_{<k})}{\tilde{q}_\theta(\mathbf{x}_{<k})}}{\mathbb{E}_{\mathbf{x}'_{<k}\sim\tilde{q}_\theta(\mathbf{x}_{<k})}[\frac{p_\theta(\mathbf{x}_{<k})}{\tilde{q}_\theta(\mathbf{x}_{<k})}]} = \frac{\mu(\mathbf{x}_{<k})}{\mathbb{E}_{\mathbf{x}'_{<k}}\mu(\mathbf{x}_{<k})},
\end{aligned}
\tag{21}
$$

where $\mu(\mathbf{x}_{<k})$ is defined as $\frac{p_\theta(\mathbf{x}_{<k})}{\tilde{q}_\theta(\mathbf{x}_{<k})}$.

C EXPERIMENTAL SETTINGS

In this section, we introduce the specific setup of different benchmarks in Table 6. We uniformly use Adam optimizer. The training will be stopped once the model has not obtained better performance for 20 epochs on the validation set. For translation tasks, the generation length is dependent on the source sequence whilst for language modeling tasks, we fix the generation length as 50. As for the model structures of the image generation task, we use the official structure reported by PixelCNN (van den Oord et al., 2016c) and Gated PixelCNN (van den Oord et al., 2016b) without modification. The source code will be released once upon acceptance.

Hyper-Parameters	**IWSLT14**	**WMT16**		**WiKiText103**	
	Tr-Base	**Tr-Base**	**Tr-Large**	**Tr-Base**	**Tr-XL**
Number of Layers	12	12	12	6	16
Hidden Embed Size	512	512	1024	512	410
FC-Layer Embed Size	1024	2048	4096	2048	2100
Attention Heads	4	8	16	8	10
Dropout	0.3	0.3	0.3	0.1	0.1
Learning Rate	5e-4	1e-3	1e-3	5e-4	2.5e-4
lr scheduler	inverse_sqrt	inverse_sqrt	inverse_sqrt	inverse_sqrt	cosine
Warm up Updates	4000	4000	4000	4000	10000
Weigth Decay	1e-4	0.0	0.0	1e-2	0.0
Coefficient λ	0.05	0.05	0.05	0.05	0.02
E-ARM Start Epoch	15	15	10	15	10

Table 6: Hyper-Parameters of different model structures and datasets. "Tr-Base", "Tr-Large", and "Tr-XL" indicate Transformer-Base, Transformer-Large, and Transformer-XL respectively