DEEP EQUILIBRIUM NON-AUTOREGRESSIVE SEQUENCE LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

In this work, we argue that non-autoregressive (NAR) sequence generative models can equivalently be regarded as iterative refinement process towards the target sequence, implying an underlying dynamical system of NAR model: $\boldsymbol{z} = \{(\boldsymbol{z}, \boldsymbol{x}) \to \boldsymbol{y}$. In such a way, the optimal prediction of a NAR model should be the equilibrium state of its dynamics if given infinitely many iterations. However, this is infeasible in practice due to limited computational and memory budgets. To this end, we propose DEQNAR to directly solve for the equilibrium state of NAR models based on deep equilibrium networks (Bai et al., 2019) with black-box root-finding solvers and back-propagate through the equilibrium point via implicit differentiation with constant memory. We conduct extensive experiments on four WMT machine translation benchmarks. Our main findings show that DEQNAR can indeed converge to a more accurate prediction and is a general-purpose framework that consistently yields substantial improvement for several strong NAR backbones.

1 INTRODUCTION

Transformer (Vaswani et al., 2017) has recently become the most prevailing neural architecture for sequence-to-sequence learning (Bahdanau et al., 2015). Transformer is originally an autoregressive (AR) sequence generative models, which adopts a sequential factorization to estimate the conditional probability of a target sequence $\boldsymbol{y} = \{y^{[1]}, \cdots, y^{[N]}\}$ conditioned on a source sequence $\boldsymbol{x}$: $p(\boldsymbol{y}|\boldsymbol{x}) = \prod_n^N p(y^{[n]}|\boldsymbol{y}^{[1:n-1]}, \boldsymbol{x})$. Albeit simple and effective, such a fixed left-to-right restriction is not necessarily the unique and the best formulation for sequence modeling, limiting the design space of neural networks and applicable tasks for AR models. Hence researchers are motivated to study non-autoregressive (NAR) sequence generative models (Gu et al., 2018) as an alternative to AR models, which instead use a per-token factorization $p(\boldsymbol{y}|\boldsymbol{x}) = \prod_n^N p(y^{[n]}|\boldsymbol{x})$. Despite their favorable decoding speed and flexible formulation to introduce constraints, NAR models still lag behind their AR counterparts and require data distillation.

NAR models can be viewed as generating sequences by iteratively denoising from an initial guess (Figure 1(a)). Several studies based on this idea of iterative refinement show promising and competitive results compared AR models. For instance, Lee et al. (2018) and Savinov et al. (2021) propose to regard NAR models as denoising autoencoders, while Ghazvininejad et al. (2019) task NAR models with conditional masked language modeling. More recently, discrete denoising diffusion models have started to attract the community's attention. Besides iteratively manipulating sequences of discrete tokens, research also finds that for fully NAR models Gu et al. (2018), layer recurrence also calibrates intermediate continuous representations towards the target discrete sequence (Huang et al., 2021; Elbayad et al., 2020; Li et al., 2022). In other words, fully NAR and iterative-based NAR models are tasked with approaching their equilibrium states, in terms of either discrete or continuous representation, which is also found in our empirical observation in Figure 1(c).

In this paper, we argue that NAR models, including fully NAR and iterative-based NAR models, can be regarded as a dynamical system in the form of $\boldsymbol{z}_{t+1} = f_\theta(\boldsymbol{z}_t, \boldsymbol{x})$, implying a dynamics of parallel denoising or iterative refinement process over the whole sequence (Figure 1(b)). More concretely, NAR models apply a Markov chain factorization to a series of intermediate predictions from the bottom up, where a neural parametric transition kernel f_θ learns denoising sequences in a coarse-to-fine manner, while $\boldsymbol{z}_t$ is the t-th running discrete or continuous state.

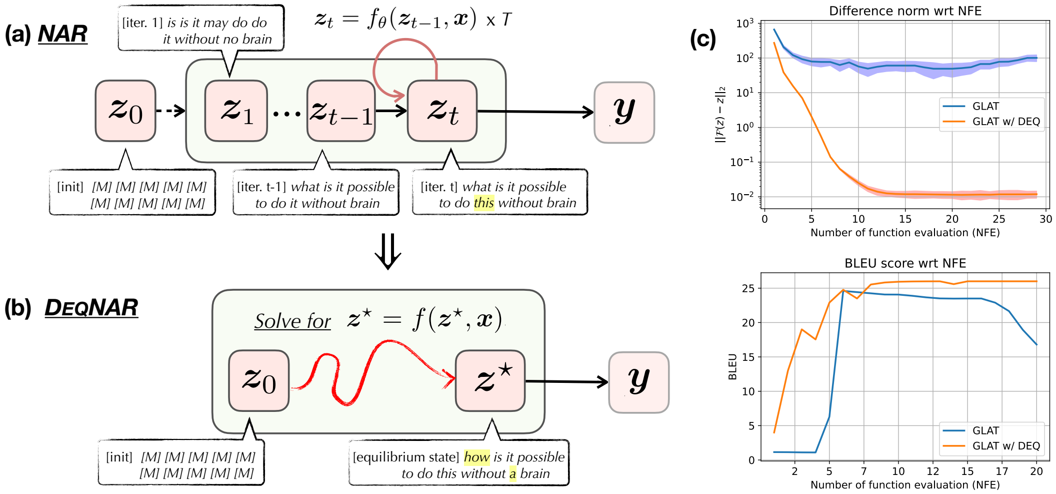

(a) NAR
[iter. 1] is is it may do do it without no brain
$z_t = f_\theta(z_{t-1}, x)$ x T
z_0
z_1
z_{t-1}
z_t
y
[init] [M] [M] [M] [M] [M] [M] [M] [M] [M] [M]
[iter. t-1] what is it possible to do it without brain
[iter. t] what is it possible to do this without brain
(b) DEQNAR
Solve for $z^\star = f(z^\star, x)$
z_0
$z^\star$
y
[init] [M] [M] [M] [M] [M] [M] [M] [M] [M] [M]
[equilibrium state] how is it possible to do this without a brain
(c)
Difference norm wrt NFE
$||\mathcal{F}(z) - z||_2$
10^3
10^2
10^1
10^0
10^{-1}
10^{-2}
0
5
10
15
20
25
30
Number of function evaluation (NFE)
GLAT
GLAT w/ DEQ
BLEU score wrt NFE
BLEU
25
20
15
10
5
0
2
5
7
10
12
15
17
20
Number of function evaluation (NFE)
GLAT
GLAT w/ DEQ

Figure 1: Comparative illustration of (a) non-autoregressive model (NAR), (b) the proposed DEQNAR model in the view of dynamical systems, and (c) the evolution of representation and performance of the NAR systems (GLAT (Qian et al., 2021) with and without DEQNAR).

From such a unified dynamical system perspective, intuitively, the state of an NAR system is supposed to evolve towards the target sequence $\lim_{t\to\infty} f_\theta(z_t, x) = z^\star \to y$, where we may obtain a solution $z^\star$ of this system that can best estimate the target y while no further improvement could be made. However, the current NAR systems, which naively evaluate the transition function $\mathcal{F}$ up to a manually-defined maximum iteration N, cannot guarantee to reach such a stationary equilibrium state, making the final output z_N a sub-optimal representation with regard to the target sequence. This motivates us to solve for such an equilibrium state of the NAR dynamical system for better understanding and modeling.

To this end, in this paper, we reformulate the sequence generation problem as solving the equilibrium state of NAR models. We propose our framework, the DEQNAR, and apply it to the cases where the iterative refinement can be conducted either in continuous feature state space, discrete data state space, or a combination of both. This enables multiple preferable properties for our model over previous studies. **(1)** Instead of naive iterative layer stacking, DEQNAR models define the output as fixed point of $\mathcal{F}_\theta$ given the input x, *i.e.*, $z^\star = f(z^\star, x)$, modeling an equilibrium representation. **(2)** Compared with typical NAR systems, the proposed DEQNAR permits better convergence to the equilibrium point. We can leverage any advanced black-box solvers, *e.g.*, quasi-Newton methods, to directly solve for the equilibrium solution, leading to better results. **(3)** The DEQNAR is also orthogonal to existing advanced techniques for NAR models, for which we studied its effectiveness when combined with the current best practices, including better modeling approach (VAE, Gu & Kong, 2021), training objective (CTC, Graves et al., 2006) and training strategy (GLAT, Qian et al., 2021).

We conduct extensive experiments on WMT14 English-German and WMT16 English-Romanian machine translation benchmarks. Based on the empirical results, our main findings are as follows: **(1)** DEQNAR is a general-purpose framework that can supplement several existing NAR techniques, including vanilla NAR, VAE, CTC loss, and GLAT training, giving rise to considerable performance gains. **(2)** We verify that convergence to an equilibrium state in DEQNAR is almost indeed via quantitative and qualitative evaluation. The closer to the equilibrium state, the more likely DEQNAR achieves more accurate performance.

2 Revisiting NAR Models as Dynamical Systems

NAR Models as Markov Process of Iterative Refinement in General. We formulate NAR models based on Transformer (Vaswani et al., 2017) as a Markov chain. There are mainly two categories of NAR models: fully NAR and iterative-based NAR models. Both of them can be unified under a general perspective of *dynamical systems conducting iterative refinement process over some intermediate state*, where the parametric transition function is in a form of $z_t = f_\theta(z_{t-1}, u_t)$ with z_t as the running state of the systems.

Formally, let $\boldsymbol{y} = [y^{[1]}, \dots, y^{[N]}] \in \{0,1\}^{N \times |\mathcal{V}|}$ within the vocabulary space $\mathcal{V}$ be a target sequence of interest, and $\boldsymbol{x} = [x^{[1]}, \dots, x^{[|\boldsymbol{x}|]}]$ be the conditional source sequence. Non-autoregressive sequence-to-sequence learning aims to learn a probabilistic model $p(\boldsymbol{y}|\boldsymbol{x})$ measuring the likelihood of target sequence given its source sequence:

$$p_\theta(\boldsymbol{y}|\boldsymbol{x}) = \sum_{\boldsymbol{z}_0,\cdots,\boldsymbol{z}_T} p_\theta(\boldsymbol{y}, \boldsymbol{z}_0, ..., \boldsymbol{z}_T|\boldsymbol{x}) = \sum_{\boldsymbol{z}_0,\cdots,\boldsymbol{z}_T} p_\theta(\boldsymbol{y}|\boldsymbol{z}_T, \boldsymbol{x}) \prod_{t=1}^{T} f_\theta(\boldsymbol{z}_t|\boldsymbol{z}_{t-1}, \boldsymbol{x}),$$

where $\boldsymbol{z}_t$ is the t-th intermediate state which is varied across different NAR models, $f(\boldsymbol{z}_t|\boldsymbol{z}_{t-1}, \boldsymbol{x})$ is the transition function from the $(t-1)$-th step to the $(t$-th step parameterized by f_θ, and $p(\boldsymbol{y}|\boldsymbol{z}_t, \boldsymbol{x}) = \prod p(y^{[n]}|\boldsymbol{z}_t, \boldsymbol{x})$ is the predicted probability made in parallel under the conditional independence assumption among the elements of $\boldsymbol{y}$.

Such parameterization shares a similar form with a first-order Markov chain, where the probability of $\boldsymbol{y}$ is a marginalization over all possible intermediate paths $\boldsymbol{z}_{0\cdots T}$. The state $\boldsymbol{z}_t$ evolves through layers in a bottom-up fashion, and the input $\boldsymbol{u}_t = \boldsymbol{x}$ is time-invariant or constant in sequence-to-sequence learning scenarios.

We provide an illustration in Figure 1(a). For clarity of notations, we can classify NAR models by the explicitness of tokens in the intermediate states:

- ***Explicit iterative refinement***: Iterative-based NAR models (Lee et al., 2018; Ghazvininejad et al., 2019) perform iterative refinement within discrete space, producing discrete tokens explicitly for each iteration. The t-th system state is the discrete representation $\boldsymbol{z}_t \in \{0,1\}^{N \times |\mathcal{V}|}$ (i.e., the index of tokens). The transition function f learns to refine the tokens in the previously generated sentence until meeting a certain condition (e.g., no further improvement or reaching a maximum number of iterations).
- ***Implicit iterative refinement***: Fully NAR models (Gu & Kong, 2021; Qian et al., 2021) can also be viewed as implicitly conducting iterative refinement within continuous feature space, given the nature of multi-layer neural networks. The t-th system state for such *implicit* iterative refinement is contextualized continuous representation $\boldsymbol{z}_t \in \mathbb{R}^{N \times d}$ (i.e., dense vectors). The transition function f is supposed to learn to refine representations layer by layer such that the discrete data can be best described.

Motivation. Based on such a dynamical system view of the NAR sequence-to-sequence learning, one can use dynamical system-inspired methods for better understanding and improved modeling. For an NAR system that conducts iterative refinement over the whole sequence towards the target sequence $\lim_{t \to \infty} f(\boldsymbol{z}_t, \boldsymbol{x}) = \boldsymbol{z}^\star \to \boldsymbol{y}$, we may want to find the solution $\boldsymbol{z}^\star$ of such a system that best estimate the target data, which is a local optima, or an equilibrium state of this system. However, as seen in Figure 1, the current NAR systems can be considered as resorting to a naive solver that recurrently applies the transition function f up to a manually-defined maximum iteration N, which cannot guarantee reaching the equilibrium solution, leading to a sub-optimal representation in terms of the target sequence. This motivates us to seek the answer to an arisen question: *Can we find such an equilibrium state of the NAR dynamical system, which can give rise to a better solution?*

3 DEQNAR: A DEEP EQUILIBRIUM NAR SEQUENCE LEARNING FRAMEWORK

To answer this question, we propose to directly solve for such an equilibrium state of NAR systems based on the use of DEQ networks (Bai et al., 2019) as a critical tool. Formally, given the input $\boldsymbol{x}$, a transition kernel f_θ parameterized by deep neural networks θ (*e.g.*, Transformer), we define a NAR sequence generative model by the following dynamical system and solve its equilibrium state $\boldsymbol{z}^\star$ as a root-finding problem:

$$\boldsymbol{z}_t = f_\theta(\boldsymbol{z}_{t-1}, \boldsymbol{x}) \qquad \Longrightarrow \quad \boldsymbol{z}_\star = \mathsf{RootFind}(g_{\boldsymbol{z}} = 0; \boldsymbol{z}_0, \theta), \text{ where } g_{\boldsymbol{z}} := f_\theta(\boldsymbol{z}, \boldsymbol{x}) - \boldsymbol{z}, \qquad (1)$$

where $\boldsymbol{z}_0$ is the initial condition.

As aforementioned, NAR models can be categorized by performing either explicit or implicit iterative refinement. We will introduce how to model implicit, explicit NAR models and the combination of both under the proposed DEQNAR framework, in accordance with different choices of the definition of the state $\boldsymbol{z}$ and the transition function f_θ, which we summarize in Table 1.

Table 1: Comparison between different type of NAR system under our framework. $\mathcal{F}$ denotes a Transformer layer, $\circ$ denotes function composition, and $\oplus$ denotes concatenation.

Category	**State**: $\boldsymbol{z}$	**Transition:** f_θ
DEQNAR-IMPLICIT	continuous feature	$f_\theta = f$
DEQNAR-EXPLICIT	discrete tokens	$f_\theta = \mathrm{softmax} \circ \mathcal{F} \circ \cdots \circ \mathcal{F}$
DEQNAR-MIXED	mixed	$f_\theta = \mathcal{F} \oplus \mathrm{softmax} \circ \mathcal{F}$

3.1 CASE I: IMPLICIT ITERATIVE REFINEMENT IN CONTINUOUS (FEATURE) STATE SPACE

Here we explain the intuition behind how solving for the equilibrium state is connected with implicit iterative refinement through an extreme case. Assuming an infinite-depth Transformer that is powerful enough, and each layer is capable of refining the representation. Intuitively, the quality of a series of intermediate states $\{\boldsymbol{z}_0, \cdots, \boldsymbol{z}_{t-1}, \boldsymbol{z}_t, \boldsymbol{z}_{t+1}, \cdots, \boldsymbol{z}_\infty\}$ would be *approximately* sorted in an ascending order. Since the goodness is bounded, it is reasonable to assume that $\boldsymbol{z}_t$ may converge to some fixed point, denoted by $\boldsymbol{z}^\star$, which is an equilibrium state that satisfies $\boldsymbol{z}^\star = f(\boldsymbol{z}^\star)$. Therefore, the inference problem of our interest becomes how to compute the equilibrium state $\boldsymbol{z}^\star$.

Formally, for NAR models that conduct implicit iterative refinement (Gu et al., 2018; Gu & Kong, 2021), the state $\boldsymbol{z}_t$ is defined as the continuous hidden representation of intermediate layer, while the transition function f_θ is parameterized by a Dirac delta distribution $\delta(\boldsymbol{z}_t - \boldsymbol{\mu})$ where $\boldsymbol{\mu}$ is the output of a single Transformer layer $\mathcal{F}$, which sequentially computes a self-attention, cross-attention and feed-forward blocks, each of which module is followed by layer normalization (Ba et al., 2016). The initial condition $\boldsymbol{z}_0 = \mathrm{emb}(\langle \texttt{mask} \rangle)$ is set to be an embedding sequence full of $\langle \texttt{mask} \rangle$ tokens.

Our solution to find the continuous variable $\boldsymbol{z}^*$ is to use advanced black-box root solving algorithms, *e.g.*, Newton or quasi-Newton methods like Broyden's methods (Broyden, 1965), or Anderson acceleration (Anderson, 1965). These methods guarantee a much faster and better-quality convergence than the case where we perform infinitely many naïve unrolling steps, which is not even realistic due to computational and memory budgets.

3.2 CASE II: EXPLICIT ITERATIVE REFINEMENT IN DISCRETE (DATA) STATE SPACE

As for NAR models that conduct explicit iterative refinement on discrete tokens (Lee et al., 2018; Ghazvininejad et al., 2019), the state $\boldsymbol{z}_t$ is defined as a sequence of one-hot vectors corresponding to each of the intermediate predicted tokens. The transition function $f_\theta : \{0, 1\}^{N \times |\mathcal{V}|} \to \mathbb{R}^{N \times |\mathcal{V}|}$ is parameterized by a multi-layer Transformer decoder followed by a softmax normalization, and a final discretization operator such as argmax or sampling. The initial condition $\boldsymbol{z}_0 = \langle \texttt{mask} \rangle$ is set to be a sequence full of $\langle \texttt{mask} \rangle$ tokens.

Two challenges exist while aiming to solve for the equilibrium point of $f_\theta(\boldsymbol{z}, \boldsymbol{x}) = \boldsymbol{z}$. (1) *Intractable*: $\boldsymbol{z}$ lies in a very high-dimensional space, and the cardinality of the feasible set is very small (the vocabulary size). Finding the solution is almost intractable, especially for highly non-linear neural networks. (2) *Non-differentiable*: Root-finding algorithms such as Newton or quasi-Newton methods require computing or estimating Jacobian inverse, which is numerically unstable or even infeasible to obtain, for transition functions f that contain non-differentiable sampling operators.

Our solution is to leverage the expected embedding weighted by the softmax probabilities as a continuous relaxation of $\boldsymbol{z}$:

$$\boldsymbol{z}_t \approx \tilde{\boldsymbol{z}}_t = \mathbb{E}\left[\mathrm{emb}(\tilde{z}_t)\right], \quad \text{where } \tilde{z}_t \sim f_\theta(\cdot|\boldsymbol{z}_{t-1}) \tag{2}$$

Such approximation helps ease the two problems: (1) a point is projected to the simplex formed by feasible points, greatly restricting the search space. (2) the "soft" embedding makes the neural network differentiable. Another possible solution is to use score function gradient estimators (e.g., REINFORCE (Williams, 1992)) for these non-differentiable operators, which, however, are known to be computationally expensive and of high variance nature.

The major difference with the implicit case is that the continuous relaxation $\tilde{\boldsymbol{z}}_t$ represents *context-less* token identity instead of *contextualized* deep representation. Issues may arise if the token identity is non-informative and not context-aware, preventing the model from evolving the state efficiently.

3.3 Case III: Mixed Explicit and Implicit Iterative Refinement

In practice, explicit iterative refinement methods are aware of the strong condition signal by the immediate prediction of the last iteration, usually leading to better results than the implicit (or the fully NAR) methods. As aforementioned, however, it is non-trivial if we want to use DEQ to solve for explicit iterative refinement, in which continuous surrogates are required.

To take the best of both implicit and explicit iterative refinement, we propose an indirect way to extend DEQ by introducing *layer-wise prediction-awareness* (Huang et al., 2021), and refer such hybrid variant to DEQNAR-MIXED. Concretely, we make an intermediate prediction $\tilde{p}(\boldsymbol{y}_t|\cdot)$ in every layer evaluation, and feed $\mathrm{emb}[\boldsymbol{y}_t]$ the embedding of the most probable predicted token into $\boldsymbol{z}_t$:

$$\boldsymbol{z}_t \leftarrow \sigma(\boldsymbol{z}_t, \mathrm{emb}[\boldsymbol{y}_t]), \quad \text{where } \tilde{\boldsymbol{y}}_t = \arg\max \tilde{p}(\cdot|\boldsymbol{z}_t)$$

where a fusion operator $\sigma : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}^d$ parameterized by a position-wise MLP. Our goal is that in such as way DEQNAR-MIXED can endow f with the awareness of running prediction made so far, which helps the model for better calibration.

3.4 Learning via Implicit Differentiation

Typically for explicit neural networks, we can directly back-propagate through the stacking layers using automatic differentiation tools (Baydin et al., 2018). However, for implicit models like DEQ, it is computationally expensive if we unroll the iteration path of the internal optimization problem. In this section, we introduce how to train the proposed model with only knowing its equilibrium state. Moreover, we also introduce regularizations to stabilize its convergence dynamics (see Appendix A).

Based on the implicit function theorem (IFT) (Krantz & Parks, 2002), the DEQ model can differentiate through its fixed point without unfolding and storing intermediate states in the forward trajectory. Specifically, given fixed-point state $\boldsymbol{z}^\star$, the task-specific loss function $\mathcal{L}$ (*e.g.*, cross-entropy), the gradients of DEQ with regard to the parameter θ and input $\boldsymbol{x}$ are given by:

$$\frac{\partial \mathcal{L}}{\partial \theta} = \frac{\partial \mathcal{L}}{\partial \boldsymbol{z}^\star}\Big(I - \frac{\partial f_\theta}{\partial \boldsymbol{z}^\star}\Big)^{-1}\frac{\partial f_\theta(\boldsymbol{z}^\star, \boldsymbol{x})}{\partial \theta} \qquad \frac{\partial \mathcal{L}}{\partial \boldsymbol{x}} = \frac{\partial \mathcal{L}}{\partial \boldsymbol{z}^\star}\Big(I - \frac{\partial f_\theta}{\partial \boldsymbol{z}^\star}\Big)^{-1}\frac{\partial f_\theta(\boldsymbol{z}^\star, \boldsymbol{x})}{\partial \boldsymbol{x}}. \tag{3}$$

This theorem enables us to decouple the forward and backward passes of DEQ-based models, *i.e.* for parameter update, we only need the final output $\boldsymbol{z}^\star$ and do not need to perform back-propagation through the unrolled chain of forwarding passes. This allows one to train implicit networks in a modern end-to-end manner while consuming only $O(1)$ memory cost.

For the explicit case we discussed in §3.2, we introduce a two-stage training scheme to prevent being hindered by the inaccurate and over-smooth predicted probability distribution in Equation 2. We first pretrain the NAR model as denoising autoencoders similar to Savinov et al. (2021) except that we use full-masking as the corruption function instead of uniform sampling from the vocabulary. We then integrate the pretrained model into our DEQNAR framework and use implicit differentiation for further finetuning, leading to the final model that learns to make predictions at its equilibrium states.

On the Differences with DEQ-Transformer (Bai et al., 2019). Note that the original work of Bai et al. (2019) has proposed DEQ-Transformer demonstrating its success for autoregressive language modeling. Compared with evolving a single word state in the DEQ-Transformer, evolving all words' states simultaneously is a highly structured and complicated problem. Using DEQ to solve NAR problems is non-trivial since the inter-dependencies among all words are prone to cause inconsistency and instability, which remains challenging and yet unexplored.

4 Experiments

We conduct extensive experiments on standard machine translation benchmarks to inspect DEQNAR's performance on sequence-to-sequence tasks. We demonstrate that DEQNAR produces better results over its NAR backbones. We also show that the proposed approach can achieve competitive performance compared with state-of-the-art NAR models.

Datasets. We evaluate our proposal on three standard translation benchmarks, *i.e.*, WMT14 English (EN) ↔ German (DE) (4.5M training pairs), and WMT16 English (EN) ↔ Romanian (RO) (610K training pairs), and use IWSLT14 De-En for preliminary study. We apply the same prepossessing steps as mentioned in prior work (En↔De: Zhou et al., 2020, En↔Ro: Lee et al., 2018). BLEU Papineni et al. (2002) is used to evaluate the translation performance for all models.

Knowledge Distillation (KD). Sequence-level knowledge distillation (Kim & Rush, 2016) is found to be crucial for training NAR models. Following previous NAR studies Gu et al. (2018); Zhou et al. (2020), all of our implemented models are trained on distilled data generated from pre-trained autoregressive Transformer models. Noticeably, DEQNAR is designed to be a general-purpose method. In this work, we resort to KD is follow the convention of previous work, which helps alleviate the general challenge of the multi-modality problem of NAR translation. No theoretical constraint prevents DEQNAR from leveraging the latest technique (*e.g.*, DAT (Huang et al., 2022)) that can directly build up NAR models on raw data. We leave it as future work.

Implementation Details. We design DEQNAR based on Transformer-*base* Vaswani et al. (2017) hyperparameters, where the number of head is 8, the model dimension is 512, the inner dimension of FFN is 2048, and 6-layer encoder/decoder are used. For variants of DEQNAR, decoders differ regarding how to parameterize their transition functions. For DEQNAR-EXPLICIT, the transition function consists of full 6-layer Transformer decoder, while for DEQNAR-IMPLICIT and DEQNAR-MIXED the transition function is one Tranformer layer only. We investigate the generality of DEQNAR by applying it to different NAR backbone models, including vanilla NAR model Gu & Kong (2021), GLAT training Qian et al. (2021), CTC loss Graves et al. (2006). For the CTC-based variant, we upsample the source input by 2. We use Anderson acceleration (Anderson, 1965) as the root-finding solver. All models are trained for 200K updates using NVIDIA V100 GPUs with a batch size of approximately 128K tokens. For both AR and NAR models, we set the dropout rate 0.1 for WMT14 En↔De and WMT16 En↔Ro. We adopt weight decay with a decay rate 0.01 and label smoothing with $\epsilon = 0.1$. Following prior studies Vaswani et al. (2017), we compute tokenized case-sensitive BLEU. We measure the validation BLEU for every 2,000 updates, and average the best 5 checkpoints to obtain the final model. As in previous NAR studies, we measure the GPU latency by running the model with a single sentence per batch on a single Nvidia V100 GPU. Partial implementation was inspired by `https://github.com/locuslab/deq` and all models were implemented on `fairseq` Ott et al. (2019).

4.1 Main Results

We first compare DEQNAR on the three cases we discussed in §3 to study the performance regarding the fundamental choices of state and transition functions. We then summarize the results of applying DEQNAR to different NAR models in Figure 2. We also compare DEQNAR and DEQNAR-LP with existing iterative and non-iterative NAR approaches in Table 3. We are now discussing our main findings in detail as follows:

Both implicit and explicit iterative refinement can be modeled under DEQNAR framework, as well as the combination of the both. To investigate the effectiveness of DEQNAR on different cases in §3, we conducted a comparative study on the top of the SoTA NAR model based on VAE and CTC from Gu & Kong (2021). As shown in Table 2, DEQNAR-IMPLICIT can improve over the CTC+VAE backbone with a large margin, verifying our motivation that the equilibrium solution of the NAR system can better represent the target data. However, DEQNAR-EXPLICIT can show a good result but is still behind the other setting. The major drawback is the challenges of learning a system with discrete states, in which our continuous relaxation by expected embedding plays an essential role in making it realizable while other attempts failed clearly (see Appendix C. Despite the interior performance, DEQNAR-EXPLICIT opens new opportunities to cast explicit iterative refinement to solve the dynamical system. As explicit iterative refinement is among the currently strongest NAR systems, we expect further studies on optimization and relaxed surrogate for the discrete state would permit further improvement under our DEQNAR framework. Finally, we find that DEQNAR-MIXED takes advantages of both implicit and explicit refinement, which helps further improve results with less NFE. In the rest of

Table 2: Preliminary comparison DEQNAR applying to different cases on IWSLT14 De-En. "NFE" refers to number of function evaluation with one Transformer decoder layer as the function herein.

Model	NFE	BLEU
AR Transformer	$6 \times N$	35.1
base NAR model: CTC+VAE	6	**33.0**
case I : DEQNAR-IMPLICIT	20	34.2
case II : DEQNAR-EXPLICIT	18 (6×3)	32.2
case III: DEQNAR-MIXED	14	34.5

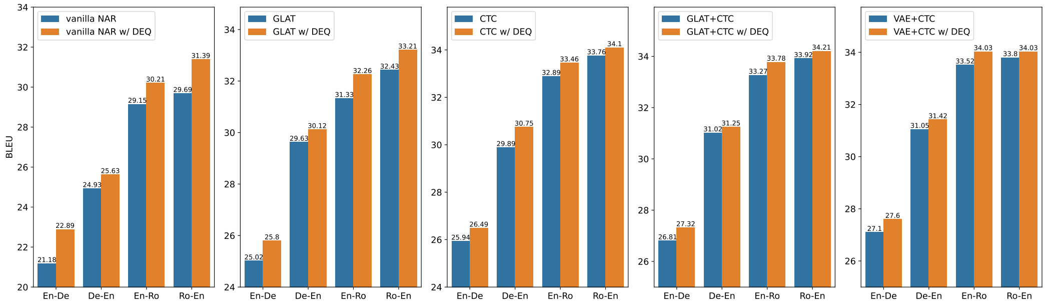


Figure 2: Applying DEQNAR to different NAR backbone models, *i.e.*, vanilla NAT (Gu et al., 2018), GLAT (Qian et al., 2021), CTC (Graves et al., 2006), GLAT+CTC (using greedy decoding, and VAE+CTC (Gu & Kong, 2021)). Note that here all the backbone models are our own implementations.

Table 3: Comparisons between our models with state-of-the-art NAR models, whose results are directly quoted from the cited publications. All our NAR models are trained with KD, where "*" and "underline" means models using KD from Transformer-big and their apparently-higher results, respectively. "NFE": numbers of evaluation of a single decoder layer, where N is the number of decoder layers (typically $N = 6$ for Transformer base configuration), T is the length of target sequence. The speed-up is measured on the WMT14 EN-DE test set, with batch size of 1 as previous NAT papers usually did. Notice that speedups from previous papers are generally not fully comparable due to inconsistent hardware and baselines and hence only for reference.

	Systems	NFE	**Speed**	**WMT14**		**WMT16**	
				EN-DE	DE-EN	EN-RO	RO-EN
AR	Transformer-*base* (KD teacher, 65m params)	$N \times 6$	1.0×	27.60	31.50	33.85	33.70
	Transformer-*big*	-	-	29.20	-	-	
Implicit	vanilla NAT (Gu et al., 2018)	6	15.6×	17.69	21.47	27.29	29.06
	CTC w/o KD (Libovický & Helcl, 2018)	6	-	16.56	18.64	19.54	24.67
	Flowseq (Ma et al., 2019)	6	1.1 ×	23.72	28.39	29.73	30.72
	*AXE (Ghazvininejad et al., 2020a)	6	15.3×	23.53	27.90	30.75	31.54
	CTC (Saharia et al., 2020)	6	18.6×	25.70	28.10	32.20	31.60
	GLAT (Qian et al., 2021)	6	15.3×	25.21	29.84	31.19	32.04
	GLAT+CTC (Gu & Kong, 2021)	6	16.8×	27.20	31.39	33.71	34.16
	DEQNAR-IMPLICIT [CTC+GLAT] (43.6m params)	20	4.2×	26.90	31.25	33.78	34.21
	beam search & reranking	20	2.9×	**27.50**	**31.65**	34.01	34.40
	comparable model size (64.4m params)	8	1.6 ×	**27.40**	**31.90**	-	-
	Transformer-*big* KD	18	4.4 ×	**27.51**	-	-	-
	DEQNAR-IMPLICIT [CTC+VAE]	20	4.2×	**27.60**	31.42	34.03	34.03
Explicit	iter-NAT (Lee et al., 2018)	6×10	1.5×	21.61	25.48	29.32	30.19
	*CMLM$_{10}$ (Ghazvininejad et al., 2019)	6×10	1.7×	27.03	30.53	33.08	33.31
	LevT (Gu et al., 2019)	$< 6 \times T$	4.0×	27.27	-	-	33.26
	*SMART$_{10}$ (Ghazvininejad et al., 2020b)	6×10	1.7×	27.65	31.27	-	-
	*DisCO$_4$ (Kasai et al., 2020)	6×4	3.5×	27.34	31.31	33.22	33.25
	*Imputer$_8$ (Saharia et al., 2020)	6×8	3.9×	28.20	**31.80**	34.40	34.10
	CTC+DSLP (Huang et al., 2021)	6	14.8×	27.02	31.61	34.17	**34.60**
	DEQNAR-MIXED [CTC+VAE]	16	1.8×	**27.80**	-	-	-

the paper, we will discuss and compare DEQNAR-IMPLICIT and DEQNAR-MIXED and other strong models.

DEQNAR is a general-purpose framework. DEQ is supposed to be a model-agnostic framework that helps converge to better representation for all NAR models. It is also orthogonal to existing advanced strategies for building up NAR systems. As shown in Figure 2, the DEQ framework can consistently improve four backbone approaches with substantial margins, including vanilla NAR model (Gu & Kong, 2021), GLAT (Qian et al., 2021), CTC (Graves et al., 2006), and the combination of GLAT and CTC, on every translation task.

Compared to the state-of-the-art approaches. As seen in Table 3, we compare our best model (CTC+GLAT w/ DEQ) with state-of-the-art approaches, both iterative and non-iterative. We found that our method can outperform all non-iterative approaches, except for Gu & Kong (2021)'s implementation of CTC+GLAT. As for the comparison with explicit iterative-based methods, DEQNAR can also match the strong results among them. Moreover, these models necessitate explicitly re-iterating the whole 6-layer decoder, usually ten times. Our approach enjoys a faster inference speed with at least half fewer layer evaluations, outperforming Huang et al. (2021) that incorporates layer-wise prediction within conventional NAR systems.

Model variants. (1) *Advanced decoding* If we further equip our CTC-based model with beam search and reranking by AR models, which is a commonly-used tactic as in previous studies, we can further boost the performance by 0.3∼0.6 BLEU score. (2) *Scaling up model capacity.* By matching the model scale to roughly the same parameters counts as the 6-layer-decoder baseline, where we held the encoder the same[1], DEQNAR can get further improved by 0.4 BLEU score (from 26.90 to 27.30). This indicates that DEQNAR can scale effectively and are more parameter-efficient per se. (3) *Learning from Transformer-big distillation.* For a more fair comparison with previous systems like Imputer (Saharia et al., 2020) that uses KD data produced from Transformer-*big*, we conduct experiments in a similar setting. We find that DEQNAR can also benefit from improving the KD performance bound by using larger teacher models, achieving 27.51 BLEU score.

4.2 Analysis of Convergence Stability and Accuracy-Efficiency Trade-off

We are really interested in whether we can converge to the equilibrium state $z^\star$, and the stability of the convergence. We first compare DEQNAR with a weight-tied GLAT model given the same maximum number of function evaluations (NFE). For DEQNAR we solve for its equilibrium up to the maximum NFE as the threshold, whereas for the GLAT model, we iteratively apply its layer for maximum NFE times. We present our observation in the upper-right part of the Figure 1. We find that without DEQ, the feature representation of a vanilla GLAT model could not converge to a stationary point, in which the differences between two consecutive iteration exhibits large in the magnitude of $\epsilon = 10^2$. As for DEQNAR-IMPLICIT GLAT, it quickly converges to a stable solution in which the residual errors are fairly small. And we suggest that such manner of convergence of DEQNAR is the reason behind its superior performance over the corresponding backbone model. Moreover, as shown in the bottom-right of the Figure 1, we also find that the DEQ-based model becomes more accurate when it gets closer to its equilibrium state during the convergence path. Finally, as shown in the right of the Figure 3, The more stable the state is, the more accurate the prediction becomes.

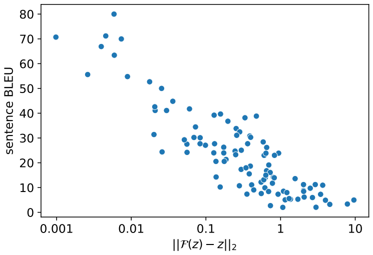


Figure 3: Scatter plot of quality (BLEU) against convergence stability (changed difference norm) on WMT14 EN-DE

As shown in Table 3 and Figure 5, DEQNAR gives rise to additional overhead in both training and decoding since it takes longer for DEQNAR to converge to more precise equilibrium states. Fortunately, we find DEQNAR would perform at least as effectively as the baseline models given the same decoding/training budget. (1) Given the same decoding time budget, we can infer from Figure 3 that DEQNAR achieves comparable performance when evaluating 6 layers as the baseline. This makes the use of DEQNAR more flexible, where you can safely ask DEQNAR as fast as its backbone model given a limited decoding budget, while you can also maximize the accuracy when the decoding budget is not a problem. (2) Given the same training time budget, we can also infer from Figure 5 that when restricting training time to 19 hours (as the time for the baseline model to finish training), DEQNAR yields a comparable some 25 BLEU as well. It can get further improved if allowing more budget as such a magnitude of training budget is often not a problem in practice.

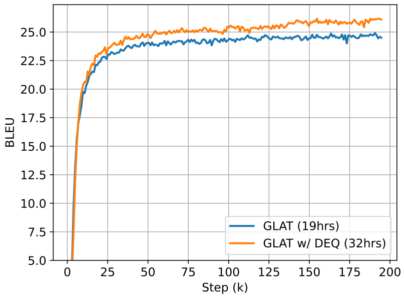


Figure 4: Comparison of the training process WMT14 En-De validation set.

[1] We set d_{model} from 512 to 1024 and d_{FFN} from 2048 to 8192, so model size expands from 46.4m to 64.6m.

5 Related Work

5.1 Non-autoregressive Sequence Generative Models in General

Non-autoregressive (NAR) models (Gu et al., 2018) are initially motivated to alleviate the decoding inefficiency of typical autoregressive seq2seq models. NAR models can be divided into two categories. Fully NAR models or non-iterative NAR models aim to generate sequence in parallel within only one shot but often sacrifice performance (Ma et al., 2019; Shu et al., 2020; Bao et al., 2019; Wei et al., 2019; Qian et al., 2021; Gu & Kong, 2021). Besides, iterative-based models significantly improve the performance of NAR models, which perform iterative refinement of translations based on previous predictions (Lee et al., 2018; Ghazvininejad et al., 2019; Gu et al., 2019; Kasai et al., 2020; Ghazvininejad et al., 2020b; Savinov et al., 2021).

5.2 Dynamical System View of Deep Neural Networks

A promising study subject is viewing a neural network as the discretization of a dynamical system. The resemblance between a residual block and an ODE's forward Euler scheme, in particular, has pushed this field forward significantly (E, 2017). One direction is to advance the widely-used residual neural architecture (He et al., 2016) by the inspiration of dynamical systems (Lu et al., 2018) The second class is to parameterize a dynamical system with trainable neural network modules (Chen et al., 2018; Dupont et al., 2019).

Deep Implicit Models. Unlike conventional, explicit neural networks, implicit models generalize the hierarchical layer stacking of neural networks to be the solution of an underlying dynamical system (Kolter et al., 2020; Amos & Kolter, 2017; Chen et al., 2018; Bai et al., 2019; El Ghaoui et al., 2019). For example, ODE-based methods (Chen et al., 2018) treat the residual block as Euler discretization of an ODE, which could be solved by any black-box ODE solver. Other studies define the output of the networks to be the solution to convex optimization problems (Amos & Kolter, 2017; Agrawal et al., 2019).

Deep Equilibrium Network. DEQs (Bai et al., 2019; 2020; 2021) is another class of implicit models that directly solves for fixed-point representation of a neural layer f_θ: $z^\star = f_\theta(z^\star, x)$ via root-finding. Intuitively, this could represent a neural network of infinite depth. One can perform nonlinear fixed point iterations of the discrete dynamical system using Broyden's method (Broyden, 1965) or Anderson acceleration (Anderson, 1965) to reach this stationary solution. Back-propagation can be done by directly differentiating through the fixed point based on the implicit function theorem. Work based on DEQs has manifested competitive performance on challenging tasks, *e.g.*, language modeling (Bai et al., 2019), flow-based generative modeling (Lu et al., 2021), semantic segmentation (Bai et al., 2020) and optical flow estimation (Bai et al., 2022).

6 Conclusions and Limitations

In this work, we revisit non-autoregressive (NAR) sequence generative models from the perspective of dynamical systems. We then propose DEQNAR that can directly solve for its equilibrium state to better estimate the desired target sequence. We conduct extensive empirical experiments demonstrating that the proposed DEQNAR models can indeed converge to the equilibrium state, which consistently improves several NAR backbones.

While these findings are promising, there remain several limitations, *e.g.* the accuracy-efficiency trade-off we have discussed above. Another one is the existence of knowledge distillation performance bound. Typical NAR models need sequence-level knowledge distillation (KD) by an AR teacher model, which imposes an upper bound of performance for the NAR student models. This is an obvious limitation for NAR models in general, since the current strong NAR baselines have been closely approaching this KD upper bound (e.g., AR Transformer's 27.60 BLEU on WMT14 DE-EN). As such, the NAR community has reached a point to call for progress in eliminating the need of KD, where we notice the recent advances of KD-free NAR sequence models such as DA-Transformer (Huang et al., 2022) which uses directed acyclic graph (DAG) for probabilistic modeling on raw data. In principle, DEQNAR is architecture-agnostic, and taking advantage of these new KD-free approaches is a promising future work, and we leave for further study.

REFERENCES

Akshay Agrawal, Brandon Amos, Shane Barratt, Stephen Boyd, Steven Diamond, and J Zico Kolter. Differentiable convex optimization layers. *Advances in neural information processing systems*, 32, 2019.

Brandon Amos and J. Zico Kolter. OptNet: Differentiable optimization as a layer in neural networks. In *International Conference on Machine Learning (ICML)*, 2017.

Donald G. Anderson. Iterative procedures for nonlinear integral equations. *Journal of the ACM (JACM)*, 12(4):547–560, October 1965.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In Yoshua Bengio and Yann LeCun (eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL `http://arxiv.org/abs/1409.0473`.

Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. Deep equilibrium models. In *Neural Information Processing Systems (NeurIPS)*, 2019.

Shaojie Bai, Vladlen Koltun, and J. Zico Kolter. Multiscale Deep Equilibrium Models. In *Neural Information Processing Systems (NeurIPS)*, pp. 5238–5250, 2020.

Shaojie Bai, Vladlen Koltun, and J. Zico Kolter. Stabilizing Equilibrium Models by Jacobian Regularization. In *International Conference on Machine Learning (ICML)*, 2021.

Shaojie Bai, Zhengyang Geng, Yash Savani, and J Zico Kolter. Deep equilibrium optical flow estimation. *arXiv preprint arXiv:2204.08442*, 2022.

Yu Bao, Hao Zhou, Jiangtao Feng, Mingxuan Wang, Shujian Huang, Jiajun Chen, and Lei Li. Non-autoregressive transformer by position learning. *arXiv preprint arXiv:1911.10677*, 2019.

Atilim Gunes Baydin, Barak A Pearlmutter, Alexey Andreyevich Radul, and Jeffrey Mark Siskind. Automatic differentiation in machine learning: a survey. *Journal of Marchine Learning Research*, 18:1–43, 2018.

Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

Charles G Broyden. A Class of Methods for Solving Nonlinear Simultaneous Equations. *Mathematics of computation*, 19(92):577–593, 1965.

Tian Qi Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In *Neural Information Processing Systems (NeurIPS)*, 2018.

Emilien Dupont, Arnaud Doucet, and Yee Whye Teh. Augmented neural odes. *Advances in Neural Information Processing Systems*, 32, 2019.

Weinan E. A proposal on machine learning via dynamical systems. *Communications in Mathematics and Statistics*, 1(5):1–11, 2017.

Laurent El Ghaoui, Fangda Gu, Bertrand Travacca, and Armin Askari. Implicit deep learning. *arXiv:1908.06315*, 2019.

Maha Elbayad, Jiatao Gu, Edouard Grave, and Michael Auli. Depth-adaptive transformer. In *International Conference on Learning Representations*, 2020.

Zhengyang Geng, Xin-Yu Zhang, Shaojie Bai, Yisen Wang, and Zhouchen Lin. On training implicit models. *Advances in Neural Information Processing Systems*, 34, 2021.

Marjan Ghazvininejad, Omer Levy, Yinhan Liu, and Luke Zettlemoyer. Mask-predict: Parallel decoding of conditional masked language models. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 6112–6121, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1633. URL https://www.aclweb.org/anthology/D19-1633.

Marjan Ghazvininejad, Vladimir Karpukhin, Luke Zettlemoyer, and Omer Levy. Aligned cross entropy for non-autoregressive machine translation. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pp. 3515–3523. PMLR, 2020a. URL http://proceedings.mlr.press/v119/ghazvininejad20a.html.

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

Alex Graves, Santiago Fernández, Faustino J. Gomez, and Jürgen Schmidhuber. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In William W. Cohen and Andrew W. Moore (eds.), *Machine Learning, Proceedings of the Twenty-Third International Conference (ICML 2006), Pittsburgh, Pennsylvania, USA, June 25-29, 2006*, volume 148 of *ACM International Conference Proceeding Series*, pp. 369–376. ACM, 2006. doi: 10.1145/1143844.1143891. URL https://doi.org/10.1145/1143844.1143891.

Jiatao Gu and Xiang Kong. Fully non-autoregressive neural machine translation: Tricks of the trade. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pp. 120–133, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.findings-acl.11. URL https://aclanthology.org/2021.findings-acl.11.

Jiatao Gu, James Bradbury, Caiming Xiong, Victor O. K. Li, and Richard Socher. Non-autoregressive neural machine translation. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL https://openreview.net/forum?id=B1l8BtlCb.

Jiatao Gu, Changhan Wang, and Junbo Zhao. Levenshtein transformer. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (eds.), *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. 11179–11189, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/675f9820626f5bc0afb47b57890b466e-Abstract.html.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pp. 770–778. IEEE Computer Society, 2016. doi: 10.1109/CVPR.2016.90. URL https://doi.org/10.1109/CVPR.2016.90.

Chenyang Huang, Hao Zhou, Osmar R Zaïane, Lili Mou, and Lei Li. Non-autoregressive translation with layer-wise prediction and deep supervision. *arXiv preprint arXiv:2110.07515*, 2021.

Fei Huang, Hao Zhou, Yang Liu, Hang Li, and Minlie Huang. Directed acyclic transformer for non-autoregressive machine translation. In *ICML*, 2022.

Jungo Kasai, James Cross, Marjan Ghazvininejad, and Jiatao Gu. Non-autoregressive machine translation with disentangled context transformer. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pp. 5144–5155. PMLR, 2020. URL http://proceedings.mlr.press/v119/kasai20a.html.

Kenji Kawaguchi. On the Theory of Implicit Deep Learning: Global Convergence with Implicit Layers. In *International Conference on Learning Representations (ICLR)*, 2020.

Yoon Kim and Alexander M. Rush. Sequence-level knowledge distillation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 1317–1327, Austin, Texas, November 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1139. URL https://www.aclweb.org/anthology/D16-1139.

J. Zico Kolter, David Duvenaud, and Matthew Johnson. Deep implicit layers tutorial - neural ODEs, deep equilibirum models, and beyond. *Neural Information Processing Systems Tutorial*, 2020.

Steven George Krantz and Harold R Parks. *The implicit function theorem: history, theory, and applications*. Springer Science & Business Media, 2002.

Jason Lee, Elman Mansimov, and Kyunghyun Cho. Deterministic non-autoregressive neural sequence modeling by iterative refinement. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 1173–1182, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1149. URL `https://www.aclweb.org/anthology/D18-1149`.

Xiang Lisa Li, John Thickstun, Ishaan Gulrajani, Percy Liang, and Tatsunori Hashimoto. Diffusion-lm improves controllable text generation. In *Advances in Neural Information Processing Systems*, 2022.

Jindřich Libovický and Jindřich Helcl. End-to-end non-autoregressive neural machine translation with connectionist temporal classification. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 3016–3021, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1336. URL `https://www.aclweb.org/anthology/D18-1336`.

Cheng Lu, Jianfei Chen, Chongxuan Li, Qiuhao Wang, and Jun Zhu. Implicit normalizing flows. In *International Conference on Learning Representations (ICLR)*, 2021.

Yiping Lu, Aoxiao Zhong, Quanzheng Li, and Bin Dong. Beyond finite layer neural networks: Bridging deep architectures and numerical differential equations. In *International Conference on Machine Learning*, pp. 3276–3285. PMLR, 2018.

Xuezhe Ma, Chunting Zhou, Xian Li, Graham Neubig, and Eduard Hovy. FlowSeq: Non-autoregressive conditional sequence generation with generative flow. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 4282–4292, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1437. URL `https://www.aclweb.org/anthology/D19-1437`.

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 the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pp. 48–53, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-4009. URL `https://www.aclweb.org/anthology/N19-4009`.

Avik Pal, Alan Edelman, and Christopher Rackauckas. Mixing implicit and explicit deep learning with skip deqs and infinite time neural odes (continuous deqs). *arXiv preprint arXiv:2201.12240*, 2022.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pp. 311–318, Philadelphia, Pennsylvania, USA, July 2002. Association for Computational Linguistics. doi: 10.3115/1073083.1073135. URL `https://www.aclweb.org/anthology/P02-1040`.

Lihua Qian, Hao Zhou, Yu Bao, Mingxuan Wang, Lin Qiu, Weinan Zhang, Yong Yu, and Lei Li. Glancing transformer for non-autoregressive neural machine translation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 1993–2003, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.155. URL `https://aclanthology.org/2021.acl-long.155`.

Chitwan Saharia, William Chan, Saurabh Saxena, and Mohammad Norouzi. Non-autoregressive machine translation with latent alignments. In *Proceedings of the 2020 Conference on Empirical*

Methods in Natural Language Processing (EMNLP), pp. 1098–1108, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.83. URL https://www.aclweb.org/anthology/2020.emnlp-main.83.

Nikolay Savinov, Junyoung Chung, Mikolaj Binkowski, Erich Elsen, and Aaron van den Oord. Step-unrolled denoising autoencoders for text generation. In *International Conference on Learning Representations*, 2021.

Raphael Shu, Jason Lee, Hideki Nakayama, and Kyunghyun Cho. Latent-variable non-autoregressive neural machine translation with deterministic inference using a delta posterior. In *EMNLP*, 2020.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (eds.), *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. URL https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

Bingzhen Wei, Mingxuan Wang, Hao Zhou, Junyang Lin, and Xu Sun. Imitation learning for non-autoregressive neural machine translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 1304–1312, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1125. URL https://www.aclweb.org/anthology/P19-1125.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

Ezra Winston and J. Zico Kolter. Monotone operator equilibrium networks. In *Neural Information Processing Systems (NeurIPS)*, pp. 10718–10728, 2020.

Chunting Zhou, Jiatao Gu, and Graham Neubig. Understanding knowledge distillation in non-autoregressive machine translation. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=BygFVAEKDH.

A More Details about Learning

A.1 Inexact Gradient Estimation.

The Jacobian-inverse term, i.e., $(I - \frac{\partial f}{\partial \boldsymbol{z}})^{-1}$, is the most important component when estimating the gradient as in Equation 3. Due to the cubic complexity, computing the inverse term by brute force is unattainable. Previous implicit models (Bai et al., 2019) tackle this by solving a linear system involving a Jacobian-vector product iteratively via a root-finding solver, resulting in expensive computational overhead in the backward pass. Furthermore, if the ill-conditioning problem occurs, estimating the gradient via this linear system can become numerically unstable. Inspired by recent advances in training implicit models (Bai et al., 2022; Geng et al., 2021), we attempt approximate gradient estimation for the backward pass to accelerate training. Take the gradient of θ as an example, we instead approximate equation 3 by:

$$\frac{\partial \mathcal{L}}{\partial \theta} \approx \frac{\hat{\partial \mathcal{L}}}{\partial \theta} = \frac{\partial \mathcal{L}}{\partial \boldsymbol{z}^\star} A \frac{\partial f_\theta(\boldsymbol{z}^\star, \boldsymbol{x})}{\partial \theta}, \tag{4}$$

where A is a approximation term for Jacobian inverse. We follow Bai et al. (2022) to let $A = I$, which simplifies the backward pass of DEQ to $\frac{\partial \mathcal{L}}{\partial \boldsymbol{z}^\star} \frac{\partial f_\theta(\boldsymbol{z}^\star, x)}{\partial \theta}$ requiring no additional iterations of gradient solvers. We will show the empirical comparison between exact and inexact gradient estimation later.

A.2 Equilibrium Dynamic Control

Albeit the existence of the equilibrium points and convergence (Kawaguchi, 2020; Winston & Kolter, 2020), the growing instability problem is a longstanding challenge in training implicit networks. As a result, the equilibrium point is often computationally expensive to reach during training (especially when stochastic regularization such as dropout is applied), slowing down the training process. Plus, equilibrium points cannot be obtained within an acceptable threshold, leading to degenerate performance when testing. We hereby introduce two constraints to stabilize the dynamics of convergence.

Stochastic dynamic correction. Inspired by Huang et al. (2021) and Bai et al. (2022), we propose to impose directly supervised signals upon some intermediate states to help stabilize DEQ dynamics and accelerate convergence. Suppose our root-finding solver yields a convergence path of $\{\boldsymbol{z}^{[1]}, \cdots, \boldsymbol{z}^\star\}$, we then randomly select some $\boldsymbol{z}_t$ (we use one in our case) and minimize the cross-entropy between its corresponding predictions against the groundtruth $\boldsymbol{y}$:

$$\ell_{\text{corr}} = \log \tilde{p}(\boldsymbol{y}|\boldsymbol{x}), \;\; \text{where } \tilde{p}(\boldsymbol{y}|\boldsymbol{x}) = \text{softmax}(\langle \boldsymbol{z}_t, \text{emb}[\boldsymbol{y}])\rangle) \tag{5}$$

Improved initial condition. In the original DEQ literature (Bai et al., 2019) and many of its followups (Bai et al., 2021; 2020), the initial condition $\boldsymbol{z}^{[0]}$ are typically set up to non-informative values (*e.g.*, all zeros) for all instances $\boldsymbol{y}$. Even if we assume that the equilibrium state of the system exists and could be reached by our solvers given enough budget of iterations, a poor, non-informative initial condition leads to a more lengthy convergence path. To mitigate this, we would like to improve the initial condition to help the model simplify its dynamics. Inspired by Pal et al. (2022), we propose to treat the first evaluation of the f as a predictive model and minimize its $L1$ distance toward the final DEQ equilibrium state $\boldsymbol{z}^\star$, given by

$$\ell_{\text{init}} = ||f(\boldsymbol{z}^\star, \boldsymbol{x}) - f(\boldsymbol{z}^{[0]}, \boldsymbol{x})||_1, \tag{6}$$

Final objective. Taken together, given parallel dataset $\mathcal{D} = \{\boldsymbol{x}, \boldsymbol{y}\}_{m=1}^M$, the final objective becomes

$$\mathcal{L}_{\text{final}}(\theta, \mathcal{D}) = \mathbb{E}_{\boldsymbol{x},\boldsymbol{y}\sim\mathcal{D}} \left[\ell_{\text{ce}} + \lambda_{\text{corr}}\ell_{\text{corr}} + \lambda_{\text{init}}\ell_{\text{init}}\right] \tag{7}$$

where ℓ_{ce} denotes cross-entropy loss, $\lambda_{\text{corr}} < 1$ and $\lambda_{\text{init}} < 1$ are weight hyperparameter for two auxiliary regularization terms, respectively.

B More Experiments and Analyses

B.1 Effect of Gradient Estimation

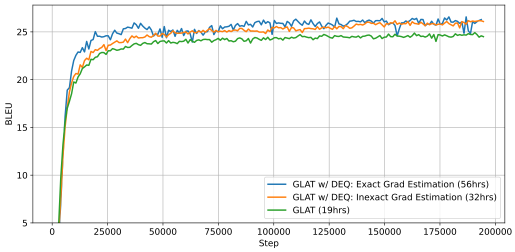


Figure 5: Comparison of training process of exact and inexact gradient estimation on WMT14 En-De validation set.

Neural networks are learned via back-propagation. DEQ uses the implicit differentiation theorem (IFT) to compute its gradient. However, the IFT requires solving another linear system to estimate the exact gradient (equation 3), which results in extra dozens of iterations, thus increasing the computational overhead for the backward pass. We thus attempt to inexact gradient estimator (equation 4). However, a natural question arises: will such approximation hurt performance? As shown in Figure 5, we plot the training curves of BLEU scores of both gradient estimators. We can find that the BLEU score of the exact gradient estimator grows more quickly than the inexact estimator in the early training stage, but tend to converge to a similar level. Furthermore, the exact estimator tends to oscillate in a larger magnitude, whereas the inexact estimator works more stable. Most importantly, the inexact estimator is fairly cheaper than the exact one, reducing more than 40% of training time (from 56 hrs to 32 hrs). Hence we choose to use the inexact gradient estimator for all our experiments.

B.2 Ablation Study on Equilibrium Dynamic Control

We present our ablation study on the proposed auxiliary regularizations for equilibrium dynamic control in Table 4. Notably, we find that these dynamic control strategies can help improve the model's performance. More importantly, we also observe that both auxiliary signals can greatly shorten the convergence path in terms of the number of iterations, which helps stabilize the equilibrium dynamics to reach the stationary state within the threshold.

Table 4: Ablation study of dynamic control on WMT14 En-De

Model	depth	BLEU
GLAT w/ DEQ	$\sim$25	**25.8**
- ℓ_{corr}	$\sim$40	25.5
- ℓ_{init}	$\sim$33	25.6
- ℓ_{corr} - ℓ_{init}	$\sim$58	25.3

B.3 Memory consumption

. We inspect the memory consumption on V100-32GB GPUs, where each device is allocated with a mini-batch of 16k tokens. The 6-layer GLAT baseline requires 14.8GB GPU memory, whilst its DeqNAR variant needs 11.4GB. This is due to the use of implicit differentiation for optimization, not needing to store the intermediate layer activations for back-propagation (see details in Bai et al. (2019)). In addition, DeqNAR-mixed only adds negligible memory overhead.

C On the Connections between Implicit (Continuous) and Explicit (Discrete) Iterative Refinement under DeqNAR Framework

One potential concern about DeqNAR could be that rooting finding and optimization algorithms like Newton methods primarily operates on continuous variable instead of actual discrete word tokens. Therefore, we would attempt to answer this interesting question as follows.

C.1 Other Preliminary Attempts on Modeling Explicit Refinement via DEQ

DeqNAR formulation for modeling explicit refinement. When applying the DeqNAR framework to discrete tokens, the state $\mathbf{z}^{[i]} = [z_1^{[i]}, ..., z_L^{[i]}]$ denotes a sequence of intermediate predicted tokens, wherein each $z_t^{[i]} \in \{0, 1\}^{|\mathcal{V}|}$ is a one-hot vector obtained by a final argmax or sampling operator on probability simplex $\Delta^{|\mathcal{V}|-1}$, which is given by the softmax output of the new corresponding layer $\hat{f}$.

Recall that with DEQ-NAR, we want to find the root of $\hat{f}(\mathbf{z}, \mathbf{x}) - \mathbf{z}$. The major obstacles are (1) that z is very high-dimensional and sparse; (2) argmax/sampling operator provides no gradients for training with back-propagation. As a result, we need continuous relaxation of z . We tried the following choices of relaxations, either deterministic or stochastic:

1. (deterministic) Let $\mathbf{z}$ be the probability of the categorical distribution over the vocabulary, i.e., the softmax result.
2. (deterministic) Let $\mathbf{z}$ be the logits/potentials, i.e., the pre-softmax scores.
3. (stochastic) Let $\mathbf{z}$ be sampled from the categorical distribution reparameterized by the Gumbel-softmax.

Note that due to the computationally expensive and the high variance nature of score function gradient estimators (e.g., REINFORCE or policy gradient), we only tried the aforementioned continuous surrogates or reparameterization in our experiments.

Experimental Results. We conducted experiments based on GLAT-based approaches on the IWSLT'14 DE-EN dataset to quickly test the ideas, which contains 160k sentence pairs.

The decoder layer $\tilde{f}$ is parameterized based on the original f_{cont} with (1) an additional up-projection linear layer ($\mathbb{R}^d \to \mathbb{R}^{|\mathcal{V}|}$, tied with embedding matrix) followed by a softmax at the end of the layer, and (2) a down-projection linear layer ($\mathbb{R}^{|\mathcal{V}|} \to \mathbb{R}^d$, also tied with embedding matrix) at the beginning of the layer.

The results are shown in the following Table 5. Unfortunately, we can find that all our attempts to directly apply DEQNAR to discrete tokens or their relaxations failed.

Analysis. We suggest that such poor performance of all these parameterizations of discrete DEQNAR could be attributed to *the lack of contextual information* as in the continuous version of DEQNAR, while contextualized representation learning is the key factor of the success of deep learning in NLP.

To expose contextual information, one solution is to additionally provide the contextualized representation, say the $\mathbf{z}_{\text{cont}}$ of the layer f of the continuous/implicit version, along with the (relaxed) discrete state $\mathbf{z}$. It is easy to find that this equivalently and essentially results in our DEQNAR-MIXED** variant, which has been shown to perform well in our paper. This is why we turn to propose DEQNAR-MIXED as a more robust solution that takes the best of both implicit and explicit refinement.

Modeling implicit refinement and decoding with the aid of explicit refinement. We want to show that despite the challenges of directly modeling explicit refinement, DEQNAR can also benefit from explicit refinement when decoding.

We study the Mask-Predict approach (Ghazvininejad et al., 2019), which is a popular explicit iterative decoding strategy. As shown in the Table above (last two rows), GLAT and its DEQNAR-powered variant can obtain subtle gains (0.2 0.3) when decoding with mask-predict. These results indicate that we can regard explicit refinement as a ready-to-use decoding strategy, which can supplement solving implicit refinement for optimal representation.

To conclude, our findings are:

1. Modeling pure explicit refinement as DEQNAR layer could be theoretically challenging and empirically not feasible (so far).
2. DEQNAR-MIXED is a good approach that combines both implicit and explicit refinement.
3. DEQNAR can also work with explicit iterative refinement techniques (i.e., mask-predict) for additional moderate gains with fewer refinement passes.

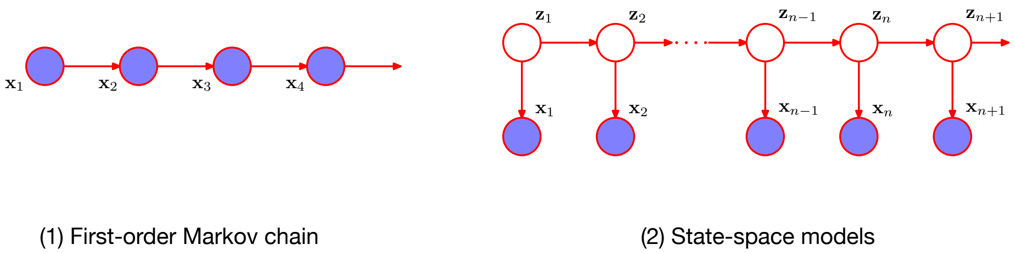
C.2 CONNECTIONS BETWEEN EXPLICIT AND IMPLICIT REFINEMENT UNDER DEQNAR

Interestingly, from the preliminary results of DEQNAR with Mask-Predict, we find that decoupling explicit refinement from DEQ training not only yields empirical gains but doing explicit refinement only during decoding perfectly also avoids the challenging back-propagating through discreteness. It could be intriguing to investigate how training on continuous embeddings and decoding on discrete tokens relate to one another and how the DEQ framework explains both.

Here we first let $\boldsymbol{x}$ denote the sequence of our general interest and ignore the conditional variable for simplicity.

Table 5: Experimental results of modeling explicit refinement via DEQ on IWSLT14 DE-EN.

Model	Result (BLEU)
Transformer	34.8
GLAT	32.2 (+0.0)
GLAT-DEQNAR	33.4 (+1.2)
softmax	∼5
logtis	∼16
gumbel-softmax	<3
GLAT + Mask-Predict (iter=4)	32.5 (+0.3)
GLAT-DEQNAR + Mask-Predict (iter=2)	33.6 (+1.4)



(1) First-order Markov chain

(2) State-space models

Figure 6: Illustration of explicit iterative refinement. Figure credit: Bishop (2006).)

As stated before, an explicit iterative refinement over sequence data in general is a process that incrementally improves the intermediate predicted discrete tokens towards the true target token sequence: $\boldsymbol{x}^{[0]} \to \boldsymbol{x}^{[1]} \to ... \to \boldsymbol{x}^{[i]}$, where $\boldsymbol{x}^{[i]}$ is expected to be close to the ground truth $\boldsymbol{x}^{\text{gt}}$. In other words, explicit refinement generates data in a coarse-to-fine, denoising manner, from an initial uninformative sequence $\boldsymbol{x}^{[0]}$ to $\boldsymbol{x}^{[i]}$.

As in iterative NAR models, if each refinement step only takes as input $\boldsymbol{x}^{[i-1]}$ the prediction of the immediate previous step, and produces an improved output $\boldsymbol{x}^{[i]}$, it can be described by a first-order Markov chain, where each $\boldsymbol{x}^{[i]}$ could be treated as one of sequential observations of sequence data, illustrated as in Figure 6(1).

Now, if we introduce an additional corresponding latent variable $\boldsymbol{z}_t$ for each $\boldsymbol{x}^{[i]}$, and assume that it is the latent variables that form a Markov chain, which is known as state-space models (Figure 6(2), e.g., HMM is a special kind of state-space models). We can readily find that the graphical structure of state-space models in Figure 6(2) gives rise to a layer-stacked NAR decoder (Transformer decoder for example), where $\boldsymbol{z}_t$ is the continuous hidden/embedding states of the i-th layer that corresponds to its discrete $\boldsymbol{x}^{[i]}$ through layer-wise and parameter-shared multinomial conditional $p(\boldsymbol{x}^{[i]}|\boldsymbol{z}_t) = \text{softmax}(W_E^\top \boldsymbol{z}_t)$, namely layer-wise prediction where W_E is the token embedding matrix shared across layers. The state transition function $\boldsymbol{z}_t = f(\boldsymbol{z}_{t-1})$ of latent variable $\boldsymbol{z}$ are now parameterized by a Transformer layer, where the initial condition/state $\boldsymbol{z}^{[0]}$ is set to be all-zeros.

With the state-space models' resemblance, we now try to study our questions in two aspects (1) how to find the fixed point of discrete sequence and why we need DEQ; (2) what is the role of ad-hoc iterative refinement decoding strategies like Mask-Predict in DEQ.

(1) How to find the fixed point of discrete sequence and why we need DEQ?

In this case, our primary goal is to find the fixed point $\boldsymbol{x}^*$ for a NAR models such that $\boldsymbol{x}^*\hat{f}(\boldsymbol{x}^*)$, regardless $\boldsymbol{x}^*$ is optimal or not.

Intuitively, it is easy the see that being a fixed point of discrete $\boldsymbol{x}^*$ is a necessary but not sufficient condition for being a fixed point of its corresponding continuous $\boldsymbol{z}^*$:

1. When the discrete fixed point $\boldsymbol{x}^*$ exists, its corresponding continuous states $\boldsymbol{z}^*$ might not be a fixed point of f. $\boldsymbol{z}^*$'s could lie in a certain region of the continuous embedding space if only the inner-product of $\boldsymbol{z}^*$ and the embedding of $\boldsymbol{x}^*$ is less than the embedding of all the other sequence $\boldsymbol{x}^{'}$.
2. In contrast, when the continuous states $\boldsymbol{z}^*$ is a fixed point of f, it is apparent that its corresponding $\boldsymbol{x}^*$ is a fixed point of the discrete sequence.

As a result, if we want to find a fixed point of discrete tokens $\boldsymbol{x}^*$ of the NAR system and the fixed point of the continuous states always exists (just under some mild conditions), we can always equivalently do this by alternatively finding the fixed points $\boldsymbol{z}^*$ of its underlying implicit NAR system over continuous embeddings states.

So this is why we need a tool to solve fixed points of such non-linear systems, and this can be effectively achieved by introducing DEQNAR based on the deep equilibrium networks (Bai et al., 2019).

(2) What is the role of ad-hoc iterative refinement decoding strategies like Mask-Predict Ghazvininejad et al. (2019) in DEQ.

Arguably, we know that, like any other optimization problem, there could exist many different solutions, and the solution of fixed point equations could likewise be affected by the initial condition $\boldsymbol{z}^{[0]}$. As a result, the fixed point of $\boldsymbol{z}^*$ we find in (1) is not necessarily the best or optimal one.

Because the primary run of DEQ solving process starts from a non-informative all-zeros state, the resulting fixed point could be too much ”contextualized” and lie in a position in the embedding vector space which is not that close to token embeddings of the discrete tokens.

As such, we suggest that ad-hoc iterative refinement decoding strategies like Mask-Predict serves to project the continuous states $\boldsymbol{z}_t$ onto the closest points that belongs to the embedding of the discrete tokens, and thus **providing a better initial condition for the next run of DEQ** solving process and hence leading to a better fixed point solution.

This could also explain for DEQNAR-MIXED from another angle that DEQNAR constantly pushes the continuous states of $\boldsymbol{z}_t$ to approach the points associated with the embeddings of the discrete tokens by providing the token embeddings themselves of the intermediate layer-wise predictions.

To conclude, we suggest that (1) DEQ is a nice tool for finding the fixed/equilibrium point of continuous embedding state $\boldsymbol{z}^*$ as a proxy so as to find the fixed point of discrete $\boldsymbol{x}^*$; (2) ad-hoc iterative refinement decoding strategies like Mask-Predict serves to provide a better initial condition for the second pass of DEQ process for a better solution.