NRGPT: AN ENERGY-BASED ALTERNATIVE FOR GPT

Anonymous authors
Paper under double-blind review

ABSTRACT

Generative Pre-trained Transformer (GPT) architectures are the most popular design for language modeling. Energy-based modeling is a different paradigm that views inference as a dynamical process operating on an energy landscape. We propose a minimal modification of the GPT setting to unify it with the EBM framework. The inference step of our model, which we call eNeRgy-GPT (NRGPT), is conceptualized as an exploration of the tokens on the energy landscape. We prove, and verify empirically, that under certain circumstances this exploration becomes gradient descent, although they don't necessarily lead to the best performing models. We demonstrate that our model performs well for simple language (Shakespeare dataset), algebraic ListOPS tasks, and richer settings such as OpenWebText language modeling. We also observe that our models may be more resistant to overfitting, doing so only during very long training.

Transformers represent a dominant paradigm in autoregressive language modeling (Vaswani et al., 2017). In a typical setting, a sequence of tokens describing a text is passed through several transformer layers and mapped onto a new sequence, which is a copy of the original one shifted by one token and appended by the token that follows the initial sequence. At training time, this network is trained through self-supervised training, and at inference time the network is used for next token prediction. This is the standard Generative Pre-trained Transformer (GPT) setting, which is the first step in Large Language Model (LLM) design (Radford et al., 2018).

Energy-based modeling (LeCun et al., 2006) is another prominent paradigm in modern AI landscape that historically goes back to Hopfield Networks (Hopfield, 1982). In this framework the operation of the neural network is defined by a scalar energy function. Proper samples generated by the model (those that resemble training data) correspond to low energy states, while unrealistic samples (with large deviations from the training data distribution) correspond to high energy states.

Although at face value these two approaches look very different, in recent years a growing number of studies hint at deep connections. Von Oswald et al. (2023) showed evidence that in-context learning (ICL) may be gradient descent by constructed an explicit weights such that the forward pass was GD on MSE loss. Ahn et al. (2024) further showed that transformers learn a preconditioned GD for ICL. However, both of these works make significant simplifications, such as considering only *linear* transformers, omitting the softmax.

Other works have have attempted to reconcile transformers and EBM from several angles. For instance, the Energy Transformer (Hoover et al., 2023) is an architecture, which is simultaneously a transformer and an energy-based model. In the image domain, the typical setting would be to reconstruct a set of masked tokens (patches) given the set of open tokens. The network solves this task by performing a gradient descent of the energy on the space of tokens at inference time. This architecture is inspired by associative memory models (Krotov and Hopfield, 2016) and for this reason solves the following problem: given a partially incomplete pattern – complete it in a meaningful way. This aspect of the core design makes it difficult to apply Energy Transformers to GPT settings, in which the sequence needs to be transformed to a shifted sequence by means of going through the network. Intuitively, in Energy Transformers the masked tokens need to evolve rapidly to match the missing parts of the pattern (e.g., image or graph), while the open tokens need to stay almost constant to barely adjust for the smooth transitions between the masked and the open tokens within the pattern. This is in drastic contrast with the GPT setting, in which there are no masked tokens at all. Rather, every token needs to evolve into the following token in the sequence.

A different line of work is inspired by "System 2" thinking and attempts to design an energy-based network for processing language (Gladstone et al., 2025). In this study, transformers are used as an

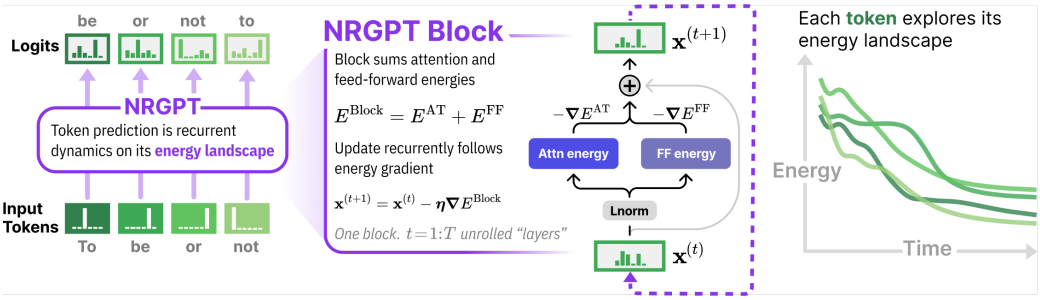

Figure 1: **NRGPT** casts the standard GPT setting into an energy-based framework. The network is defined as the sum of two energies: an **attention** energy and a **feedforward** energy. Each **token** is transformed into the next token by exploring the energy landscape. Recurrent application of the NRGPT block produces a dynamical system where each token can be thought of as a particle moving on the network's energy landscape.

architectural motif that casts text into a scalar energy function. While models of this nature have benefits for language processing, they belong exclusively to the class of energy-based models, and are unrelated to the GPT settings, commonly used in most LLMs.

Individual modules within the transformer block, such as attention, have also been studied from the perspective of inference time optimization (Geshkovski et al., 2023; 2024). In this line of work, peculiar clustering properties of tokens have been observed. Energy-based optimization has also been studied in (Yang et al., 2022) from the perspective of majorization-minimization algorithms.

Despite this growing list of studies dedicated to synergies between autoregressive transformers and energy-based models, at present it remains unknown how to cast the commonly used GPT setting into a well-defined energy-based framework. Our work tackles this gap. We refer to our model as e**N**e**R**gy **G**enerative **P**re-trained **T**ransformer or NRGPT. The input sequence of tokens is mapped onto a shifted sequence of tokens, which includes the next word, see Figure 1. The mapping is performed by a neural network, which recurrently applies the NRGPT block to the sequence of tokens. Each application of the block uses gradients of the network energy functions to update the state of the tokens. Each token has its own energy landscape, which is dependent on the states of other tokens. Specifically, our contributions are:

- We design an energy function and an update rule that describes the GPT setting with several possible variants including learnable inference rate and normalization operations: LayerNorm and RMSNorm.
- We obtain excellent results on nested ListOPS tasks, including arithmetic operations, min/max selection, etc.
- We show the feasibility of using NRGPT for language modeling on Shakespeare and OpenWebText datasets.
- We do a systematic comparison of performance scaling of recurrent transformers and NRGPT.
- We study empirically the properties of dynamical trajectories of tokens on the energy landscapes of our models.

1 Energy-based modeling

In generative modeling our goal is to generate samples with a distribution close to observed datapoints. If we manage to learn an approximate likelihood function for the dataset, we can generate data by sampling. This is also the premise Energy-Based Models (EBM). An example of EBM would be Dense Associative Memory (Krotov and Hopfield, 2016), where datapoints are stored in minima of an energy function. But more generally, the energy can represent a negative-log-likelihood, $E(\mathbf{x}) = -\log P(\mathbf{x})$. In this case, the global minima of the energy represent maximum likelihood solutions. The deeper the energy, the higher the likelihood of that datapoint. One strategy to train an

EBM is to first learn the energy function by fitting the distribution of the data. The sampling process would then be separate from learning the energy.

However, in high dimensional data, learning the distributions is notoriously difficult due to the curse of dimensionality. Diffusion models solve this problem by starting from high noise and cooling down. Diffusion models do not learn an explicit energy function, only its gradients, the score function. Yet, having an explicit energy function could enable us to explore the solution space in ways not easily afforded by the implicit score function of diffusion models. So can we build a model which learns the energy directly?

Similar to diffusion models, we use an end-to-end process, where learning the energy and generating datapoints are all done in one pass. The key idea is to have a differentiable sampling process which allows us to learn the parameters of the energy during sampling. Since real datapoints should have low energies, we choose a gradient-based sampling process. Note that we do not need to descend all the way to a minimum (i.e. maximum likelihood solution), since we want diverse samples. Instead, we could do a fixed number of GD steps and demand that the final point match real datapoints.

$$\text{Generated data: } \mathbf{x}^{(T)}, \qquad \mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} - \boldsymbol{\eta}^{(t)} \boldsymbol{\nabla} E(x^{(t)}) \tag{1}$$

for a fixed number of steps T, where $\mathbf{x}^{(0)}$ is some random initial point. Here $\boldsymbol{\eta}^{(t)}$ is a matrix that may depend on $\mathbf{x}$. This matrix has many different names, e.g., kinetic rates in physics, preconditioner in optimization, etc. We will call this matrix the *inference rate*, since it determines the size of the steps that the inference dynamics takes on the energy landscape. But how do we judge whether the output matches a real datapoint? One way would be to have a judge, like the discriminator in a GAN. Another setting where judging the output is more natural is autoregressive language modeling where the new datapoints are the next tokens and can be matched to the training text. In this case $\mathbf{x} \in \mathbb{R}^{N \times D}$ represents a real data sequence of length N embedded in D dimensions. In causal language modeling the energy should take $\mathbf{x}_{<N} = (\mathbf{x}_1 \dots \mathbf{x}_{N-1})$ as input and predict $\mathbf{x}_N$, as in

$$\mathbf{x}_N^{(t+1)} = \mathbf{x}_N^{(t)} - \boldsymbol{\eta} \boldsymbol{\nabla} E(\mathbf{x}_N^{(t)} | \mathbf{x}_{<N}^{(t)}) \tag{2}$$

Following the observations of the Energy Transformer (ET), we will show that one can choose a parametrization for E such that the process of T-step GD closely resembles the forward-pass through a T layer GPT transformer with a weight-sharing pattern.

2 NRGPT Module

$$E_A = -\frac{1}{\beta} \sum_{h=1}^{H} \log \left(\sum_{B<A} \exp \left(\beta \, \mathbf{g}_B^T \boldsymbol{J}^h \mathbf{g}_A \right) \right) - \sum_{B=1}^{N} \mathbf{g}_B^T \mathbf{W_2} \sigma \Big(\mathbf{W_1} \mathbf{g}_B \Big) \tag{3}$$

In this section we will start from the structure of the transformer model and derive the energy function whose gradients yield a layer which is very close in structure to a transformer layer. Let $\mathbf{x} \in \mathbb{R}^{D \times N}$ be an input sequence of length N embedded in D dimensions. We will denote its components by $\mathbf{x}_{Ai}$ with $A = 1 \dots N$ and $i = 1 \dots D$, or $\mathbf{x}_A$ suppressing the embedding index, but keeping the token index. Let $\mathbf{x}^{(t)}$ be the output sequence of layer t of the model with $\mathbf{x}^{(0)} = \mathbf{x}$. A conventional transformer layer has an Attention layer (AT) followed by a two-layer feedforward (FF) and LayerNorm (LN) in series

$$\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} + \text{FF}\Big[\text{LN}\Big(\mathbf{x}^{(t)} + \text{AT}\big(\text{LN}(\mathbf{x}^{(t)})\big)\Big)\Big] \tag{4}$$

But subsequent works such as GPT-J (Wang and Komatsuzaki, 2021), PaLM (Chowdhery et al., 2023), and Energy Transformer (Hoover et al., 2023) showed that the following parallel design has good performance too

$$\textbf{Parallel Transformer:} \quad \mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} + \text{AT}(\mathbf{g}^{(t)}) + \text{FF}(\mathbf{g}^{(t)}), \quad \mathbf{g}^{(t)} = \text{LN}(\mathbf{x}^{(t)}) \tag{5}$$

We choose this parallel transformer design as it is more suitable for our goal of replacing the transformer layer with the gradient of an energy.

If passing through a layer becomes one step of energy decent (ED), then all layers need to share weights. Therefore, our model will consist of a single module replacing the transformer block. Instead of different layers, we will be recurrently feeding the output of the layer back into itself, so that $\mathbf{x}^{(t)}$ will become step t of the ED instead of the layer number.

Update Rule An important point to note is that the update rule of NRGPT is slightly different from conventional gradient descent and is of the form

$$\dot{\mathbf{x}} = \mathbf{x}^{(t+1)} - \mathbf{x}^{(t)} = -\boldsymbol{\eta}^{(t)} \frac{\partial E}{\partial \mathbf{g}^{(t)}} \tag{6}$$

where $\boldsymbol{\eta}^{(t)} \in \mathbb{R}^{D\times D}$ is an inference rate matrix, which can be learnable. Nevertheless, this can be a valid descent on E as we can show that $E^{(t+1)} - E^{(t)} < 0$ under certain conditions, which depend on the normalization operation $\mathbf{g}$. We will derive these conditions for LayerNorm as well as RMSNorm, as well as when $\mathbf{g} = \mathbf{x}$, i.e., no normalization in Section 2.2. Next, we introduce the energy of NRGPT module.

2.1 Energy of NRGPT

Matching our update rule (6) to the parallel transformer (5), we define two terms in the energy, E^{AT} and E^{FF}

$$E = E^{\mathrm{AT}} + E^{\mathrm{FF}}, \qquad \eta\partial_g E^{\mathrm{AT}} = -\mathrm{AT}(\boldsymbol{g}), \qquad \eta\partial_g E^{\mathrm{FF}} = -\mathrm{FF}(\boldsymbol{g}), \tag{7}$$

We begin by introducing the attention layer and deriving the energy function for the self-attention mechanism. Then, we derive the energy function for the FF. Finally, we combine the two energy functions to obtain the total energy function for the transformer layer.

Attention. Consider a multi-head attention module with H heads, and hidden dimension $Y = D/H$, index h enumerates heads and runs $h = 1 \dots H$. Its query, key, value and projection weights are

$$\boldsymbol{W}^Q, \boldsymbol{W}^K, \boldsymbol{W}^V, \boldsymbol{W}^P \in \mathbb{R}^{H\times Y\times D}, \tag{8}$$

Using the standard $\boldsymbol{K} = \boldsymbol{W}^K\mathbf{g}, \boldsymbol{Q} = \boldsymbol{W}^Q\mathbf{g}, \boldsymbol{V} = \boldsymbol{W}^V\mathbf{g}$, The MHA output for token A is[1]

$$\mathrm{AT}(\mathbf{g})_A = \sum_{h=1}^{H} \left[\boldsymbol{W}_h^P\right]^T \boldsymbol{V}_h \mathrm{SM}\left(\boldsymbol{K}_h^T \boldsymbol{Q}_{Ah}\right) \tag{9}$$

denoting $\boldsymbol{J} = \left[\boldsymbol{W}^K\right]^T \boldsymbol{W}^Q$, the softmax is defined as (we omit the self-interaction term $C = A$)

$$\mathrm{SM}(\boldsymbol{K}^T\boldsymbol{Q})_{BA} = \frac{\exp\left(\beta \boldsymbol{g}_B^T \boldsymbol{J} \boldsymbol{g}_A\right)}{\sum_{C<A} \exp\left(\beta \boldsymbol{g}_C^T \boldsymbol{J} \boldsymbol{g}_A\right)}, \quad \beta = \frac{1}{\sqrt{Y}} \tag{10}$$

Following Hoover et al. (2023), define the attention energy

$$E_A^{\mathrm{AT}}(\boldsymbol{g}) = -\frac{1}{\beta} \sum_h \alpha_h \log\Big[\sum_{B<A} \exp\left(\beta \boldsymbol{g}_B^T \boldsymbol{J}_h \boldsymbol{g}_A\right)\Big] \tag{11}$$

where $\boldsymbol{\alpha} \in \mathbb{R}^H$ is a learnable weight. Taking the gradient of E^{AT} w.r.t. $\boldsymbol{g}_A$ and using (7), the resulting attention layer becomes

$$\mathrm{AT}(\boldsymbol{g})_A = -\boldsymbol{\eta}\frac{\partial E_A^{\mathrm{AT}}(\boldsymbol{g})}{\partial \boldsymbol{g}_A} = \sum_{h=1}^{H} \alpha_h \boldsymbol{\eta} \boldsymbol{J}_h^T \boldsymbol{g} \mathrm{SM}\left(\boldsymbol{g}^T \boldsymbol{J}_h \boldsymbol{g}_A\right) \tag{12}$$

Comparing to the original attention, we see that some weights are replaced

$$\text{Original: } \left[\boldsymbol{W}_h^P\right]^T \boldsymbol{W}_h^V \rightarrow \text{Energy: } \alpha_h \boldsymbol{\eta} \boldsymbol{J}_h^T \tag{13}$$

In principle $\boldsymbol{W}^V$ and $\boldsymbol{W}^P$ can be merged into one matrix. (He and Hofmann, 2024) also experimented with removing $\boldsymbol{W}^V$ and $\boldsymbol{W}^P$ and found that in the setting without skip connections, these two weights could be largely omitted.

[1] Usually the projection weights $\boldsymbol{W}^P$ are defined as $D \times D$ and the head outputs are concatenated, into an $N \times (YH) = N \times D$ matrix before multiplying by W^P. This is equivalent to our definition.

Feed-Forward network. The FF network generally has two layers $\mathrm{FF}(\boldsymbol{g}_A) = \boldsymbol{W}^{2^T}\sigma\left(\boldsymbol{W}^1\boldsymbol{g}_A\right)$ with weights $\boldsymbol{W}^1, \boldsymbol{W}^2 \in \mathbb{R}^{M\times D}$, with M being the size of the hidden layer. A possible choice for this network is a Dense Associative Memory (Krotov and Hopfield, 2016). In this case

$$E^{\mathrm{FF}} = -\sum_{A=1}^{N} \mathbf{1}^T F\left(\boldsymbol{W}^1\boldsymbol{g}_A\right), \quad \text{s.t. } F' = \sigma$$
$$\mathrm{FF}(\boldsymbol{g}_A) = -\eta\frac{\partial E^{\mathrm{FF}}}{\partial \boldsymbol{g}_A} = \boldsymbol{\eta}\boldsymbol{W}^{1^T}\sigma\left(\boldsymbol{W}^1\boldsymbol{g}_A\right) \tag{14}$$

where $\mathbf{1}$ is an M-dimensional vector of ones. So the energy gradient yields a structure similar to the FF in transformers but with different weights

$$\text{Original: } \boldsymbol{W}^2 \rightarrow \text{Energy: } \boldsymbol{W}^1\boldsymbol{\eta}^T \tag{15}$$

As an example, in order for E^{FF} to reproduce the FF of transformers with $\sigma(z) = \mathrm{ReLU}(z) = \max(z, 0)$, the function F should be

$$F(z) = \frac{1}{2}\sigma(z)^2 \tag{16}$$

Of course, the FF module can be replaced by other, more general, MLP networks. Essentially, any scalar function, which is additive in token index, can serve as a valid form of FF network. In the experiments (Section 3) we will detail our choices of E^{FF}.

2.2 Normalization of tokens

These normalizations have the form

$$\boldsymbol{g} = \boldsymbol{\gamma}\odot\frac{\boldsymbol{x}-\boldsymbol{\mu}}{\sqrt{\frac{1}{D}\|\boldsymbol{x}-\boldsymbol{\mu}\|^2+\epsilon}} + \boldsymbol{\delta} \tag{17}$$

with $\boldsymbol{\mu} = \mathbb{E}[\boldsymbol{x}]$ for LayerNorm, and $\boldsymbol{\mu} = 0, \boldsymbol{\delta} = 0$ for RMSNorm. Here $\boldsymbol{\gamma}, \boldsymbol{\delta} \in \mathbb{R}^D$ and $\odot$ is elementwise multiplication. Many recent models such as Qwen and Llama use RMSNorm.

Proposition 2.1 (Energy Descent)**.** *The update rule (6) results in decreasing energy, $\dot{E} = E^{(t+1)} - E^{(t)} < 0$, if the inference rate is $\eta = c\,\mathrm{diag}(\gamma)$ with $c \in \mathbb{R}_{>0}$.*

Sketch of proof. See Appendix B for full proof. The Jacobian of $\boldsymbol{g}_A$ can be written as $\partial\boldsymbol{g}_A/\partial\boldsymbol{x}_A = \frac{1}{r_A}\boldsymbol{\Gamma}\boldsymbol{P}_A$, where $\boldsymbol{\Gamma} = \mathrm{diag}(\gamma)$, $r_A > 0$ is some norm of $\boldsymbol{x}_A$ and $\boldsymbol{P}_A$ is an approximate $D\times D$ projection matrix and p.s.d. Using this and (6), we get $\dot{E} = -r_A^{-1}\sum_A \mathrm{Tr}\left[\partial_{\boldsymbol{g}_A}E\boldsymbol{\Gamma}\boldsymbol{P}_A\boldsymbol{\eta}^T\partial_{\boldsymbol{g}_A}E^T\right]$. When $\boldsymbol{\eta} = \boldsymbol{\Gamma}$ we get $\dot{E} < 0$. □

There also exist x-dependent solutions of the form $\boldsymbol{\eta} = \boldsymbol{M}(\boldsymbol{x})\boldsymbol{P}_A\boldsymbol{\Gamma}$ where $\boldsymbol{M}(\boldsymbol{x})$ is an arbitrary positive semi-definite (psd) matrix and $c > 0$, but they mean $\boldsymbol{\eta}$ is itself a neural network. To remain close to the structure of conventional transformers, we work with the x-independent $\boldsymbol{\eta} = c\boldsymbol{\Gamma}$ solution. In principle, $\boldsymbol{\eta}$ can also have a part contributing to the anti-symmetric part of $\boldsymbol{\Gamma}\boldsymbol{P}_A\boldsymbol{\eta}^T$, but we could not find an x-independent solution for it. While $\boldsymbol{\eta}^{(t)} = c^{(t)}\boldsymbol{\Gamma}$ puts severe restrictions on the preconditioning matrix, each layer is still allowed to have a different $c^{(t)}$ constant.

Preconditioner without layer normalization. Several works have shown careful initialization and scaling can replace normalization entirely. . Doing so would remove much of the restrictions on the preconditioner η. In an T-layer transformer, Instead of layer normalization, T-Fixup (Huang et al., 2020) and DeepNet (Wang et al., 2024) initialize some weights and x by a scale proportional to $T^{-1/4}$ and change most layer outputs to $f(g) \rightarrow \alpha f(x)$, with $\alpha \propto T^{-1/2}$. In this case, η becomes much less restricted.

Proposition 2.2 ($\dot{E}$ without normalization)**.** *When $\boldsymbol{g} = \boldsymbol{x}$, to get $\dot{E} < 0$, the symmetric part, $\boldsymbol{\eta}_+ = (\boldsymbol{\eta} + \boldsymbol{\eta}^T)/2$ needs to be psd.*

Proof. In this case $\dot{\boldsymbol{x}} = -\boldsymbol{\eta}\partial_{\boldsymbol{x}} E$ and

$$\dot{E} = -\sum_A \mathrm{Tr}\left[\partial_{\boldsymbol{x}_A} E \boldsymbol{\eta} \partial_{\boldsymbol{x}_A} E^T\right] = -\sum_A \mathrm{Tr}\left[\partial_{\boldsymbol{x}_A} E \boldsymbol{\eta}_+ \partial_{\boldsymbol{x}_A} E^T\right] \tag{18}$$

which is negative when $\boldsymbol{\eta}_+$ is psd. □

The antisymmetric part $\boldsymbol{\eta}_- = (\boldsymbol{\eta} - \boldsymbol{\eta}^T)/2$ is not constrained by this and can be arbitrary at this point. Hence, without layer normalization, we can have $\boldsymbol{\eta}^{(t)}$ as learnable parameters of the form

$$\boldsymbol{\eta} = \boldsymbol{U}^T\boldsymbol{U} + \boldsymbol{V} - \boldsymbol{V}^T, \qquad \boldsymbol{U} \in \mathbb{R}^{D\times D'}, \boldsymbol{V} \in \mathbb{R}^{D\times D} \tag{19}$$

where D' can be arbitrary, and each layer (depth) can have a different learnable $\boldsymbol{U}^{(t)}, \boldsymbol{V}^{(t)}$.

2.3 Asymptotic Stability of NRGPT

A peculiar aspect of NRGPT is the phenomenon of asymptotic stability. In order to illustrate it, consider a simplifying case when the inference rate matrix $\boldsymbol{\eta}$ is identity. In this case dynamical equations for tokens can be written as

$$\dot{x}_{iA} = -\frac{\partial E_A}{\partial g_{iA}} \tag{20}$$

Additionally, $\boldsymbol{\gamma}$ can always be absorbed into $\boldsymbol{J}$ and the weights of the FF model. This way, the Jacobian becomes $\partial \boldsymbol{g}_A/\partial \boldsymbol{x}_A = \boldsymbol{P}_A$, which is p.s.d.. The key observation is that due to causal attention mask the energy E_A of token A only depends on the states of tokens $B \le A$. Thus, for the first token

$$\dot{\boldsymbol{x}}_1 = -\frac{\partial E_1}{\partial \boldsymbol{g}_1} \tag{21}$$

Since the energy of that token decreases with time

$$\dot{E}_1 = \frac{\partial E_1}{\partial \boldsymbol{g}_1}\frac{\partial \boldsymbol{g}_1}{\partial \boldsymbol{x}_1}\frac{d\boldsymbol{x}_1}{dt} = -\dot{x}_1^T \boldsymbol{P}_1 \dot{x}_1 \le 0 \tag{22}$$

since $\boldsymbol{P}_1$ is psd. Additionally, since energy only depends on $\boldsymbol{g}$ – layernormalized tokens – it is bounded from below. Thus, the dynamics of $\boldsymbol{g}$ has to converge to a fixed point. This means that after a transitory period of time T_{tr} the derivative $\frac{d\boldsymbol{g}_1}{dt}$ vanishes.

Now, consider the network of two tokens

$$\dot{E}_2 = \frac{\partial E_2}{\partial \boldsymbol{g}_1}\frac{\partial \boldsymbol{g}_1}{dt} + \frac{\partial E_2}{\partial \boldsymbol{g}_2}\frac{\partial \boldsymbol{g}_2}{\partial \boldsymbol{x}_2}\frac{d\boldsymbol{x}_2}{dt} = -\dot{x}_2^T \boldsymbol{P}_2 \dot{x}_2 \le 0 \tag{23}$$

the second equality is written assuming that we are looking at this quantity at $t > T_{\text{tr}}$. Thus the first term is zero. Same argument applies, the energy decreases with time and is bounded from below. Thus, eventually $\boldsymbol{g}_2$ freezes.

One can apply this argument recursively to each token and conclude that after a transitory period of time, all tokens stabilize and all $\boldsymbol{g}_A$ will eventually become constants. From the perspective of energy profiles, this leads to the following behavior: during transitory regime energies of individual tokens will evolve in time (they can both increase and decrease). After that transitory period is over the energies must stabilize and reach their constant values that become unchanged in the future. One can run the inference dynamics as long as desired after that, but no changes in energies will occur. This behavior is apparent from the numerical profiles of energies, see Figure 2. It is a distinct aspect of our models - the phenomenon that we call asymptotic stability.

3 Experiments

We tested our model on three datasets: ListOps, Shakespeare and Open Web Text (OWT). The details of the experimental settings and hyperparameters can be found in Appendix C. TO evaluate the quality of text in the Shakespeare and OWT experiments, we use a number of metrics, including perplexity and diversity scores, explained in Appendix C.1. For

Model Variants. The choice of the energy function is equivalent to the choice of architectures in neural networks. As our goal is to be as close to the original transformer architecture as possible, we experimented with a few settings for the FF network and found the following variants to be the best performing:

1. `NRGPT_H_FF1`: $E^{\text{FF}} = -\|\sigma(\boldsymbol{W}\boldsymbol{g}_A)\|^2$, same as in equation 14, but with $\sigma = \text{GELU}$.
2. `NRGPT_H_FF2W`: $E^{\text{FF}} = -\sum_A \boldsymbol{g}_A^T \boldsymbol{W}^2 \sigma(\boldsymbol{W}^1 \boldsymbol{g}_A)$, which yields

$$\text{FF}(\boldsymbol{g}_A) = -\eta \left(\boldsymbol{W}^2 \sigma(\boldsymbol{W}^1 \boldsymbol{g}_A) + \sigma'(\boldsymbol{W}^1 \boldsymbol{g}_A)^T \odot \boldsymbol{W}^{1^T} \sigma(\boldsymbol{W}^2 \boldsymbol{g}_A) \right) \tag{24}$$

   Here, the first term is the conventional FF of transformers, but the second is a somewhat odd network.

In case with two weights, we choose the hidden dimension between $\boldsymbol{W}^1$ and $\boldsymbol{W}^2$ to be $4 \times D$. All of these performed well, with the residual version showing best performance on ListOps, learning at even smaller sizes than our baseline, while also easily training at larger sizes with embedding size over 256. However, since some of these models deviate significantly from the FF of a recurrent GPT (the gradient results in an FF module with four layers), we decided to focus more on `NRGPT_H_FF1` and `NRGPT_H_FF2W`, which are much closer to the GPT FF. As Baselines, we used `GPT_Rec_parallel`, which is a GPT-J model with a single transformer layer, feeding back recurrently into itself for a fixed number of times (mimicking number of layers). On Shakespeare and OWT, we also show results of a conventional GPT2-style deep transformer model.

3.1 Energy Dynamics

NRGPT without constraints on the inference rate η is not forced to strictly decrease energy during inference and it may learn other exploration strategies for inference. Nevertheless, we would like to examine whether models which are explicitly forced to perform GD and reduce energy during inference can learn the tasks well. To reach To better understand how our gradient-based update rule performs inference, we ran experiments on ListOps with large number of recurrent steps (30 steps). For these experiments, we set $\eta = 1$, which according to Proposition 2.1 forces the update rule to decreases energy. Figure 2 shows the evolution of total E, E^{AT} and E^{FF} along these trajectories. We observe that indeed in all trajectories the final energy is lower than initial. Each individual token trajectory is not required to be monotonically decreasing, as the dynamics of tokens is coupled. however, in accordance with our result in Section 2.3, after a transient stage, once all previous tokens start converging, the energy of the next token monotonically decreases.

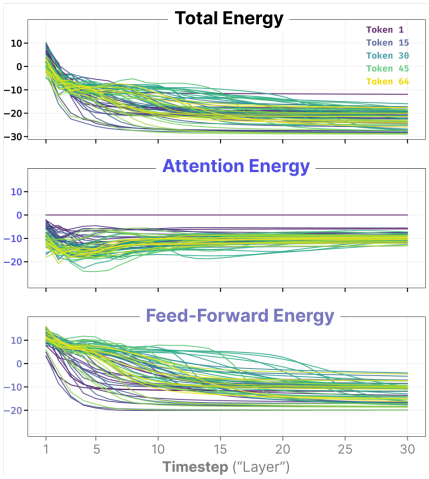

Figure 2: In NRGPT, tokens converge to stable states of low energy where the causal attention mask allows each token energy to fluctuate during inference. Shown are 64 tokens passed to an NRGPT model trained to predict ListOps equations.

3.2 ListOps

We perform experiment on nested math operations on lists of integers, which are a version of ListOps (Nangia and Bowman, 2018). Our ListOps setting consists of three functions: maximum, median and sum modulo 20. Our inputs range from 0 to 19. Each data sample begins with nested equations like `SUM(2,MAX(4,13,1),MEDIAN(5,3,16))`. As performance metrics, we looked at accuracy on the mixed task, as well as the training and validation loss. Figure 3 shows the results for two of our model variants, `NRGPT_H_FF1` and `NRGPT_H_FF2W` as compared to `GPT_Rec_parallel`.

3.3 Shakespeare

We compare the performance of our `NRGPT_H_FF2W` and `NRGPT_H_FF1` with `GPT_Rec_parallel` and deep GPT for embeddings sizes less than 1024 (Figure 4 In

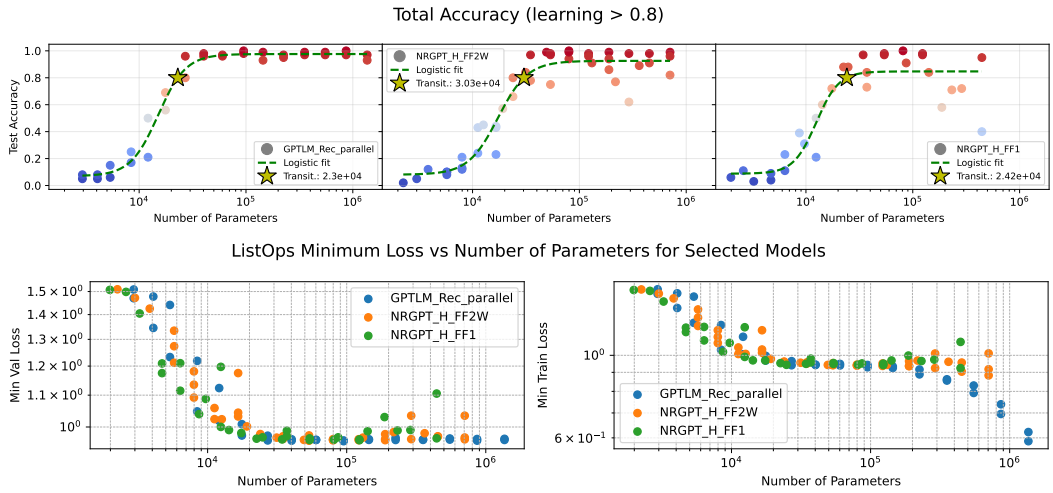

Figure 3: **Learning ListOps:** NRGPT variants match performance with a recurrent GPT model on ListOps accuracy parameter-transition points (top) and training/validation losses (bottom). The accuracy of models is tested on nested, mixed arithmetic tasks of maximum, median and sum modulo 20. For all plots, the x axis shows the *total parameter count* of the model. The yellow star indicates the transition to learning, which we define as where the logistic fit hit $> 80\%$ accuracy. The baseline model `GPT_Rec_parallel` shows the earliest learning transition at size 2.3×10^4, but our NRGPT variants are also similar, with `NRGPT_H_FF1` at 2.4×10^4 and `NRGPT_H_FF2W` at 2.98×10^4.

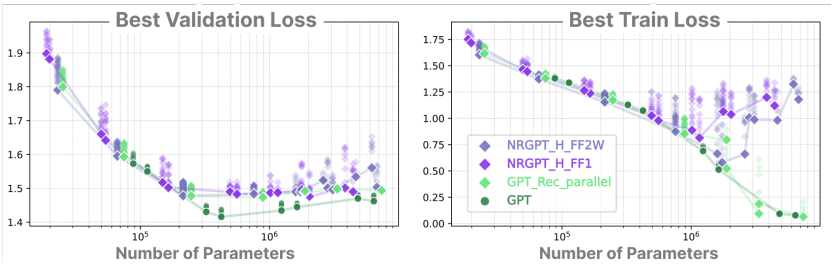

Figure 4: **Shakespeare scaling:** NRGPT achieves performance parity with recurrent GPT on Shakespeare across parameter sizes, as measured by *best validation loss* per number of parameters. For many embedding sizes, NRGPT also follows the same optimal training loss trajectory-per-parameter as both GPT and recurrent GPT baselines. However, NRGPT does not overfit Shakespeare at large parameter sizes. Connecting lines show the best performance at fixed parameter sizes. Transparent dots show different choices of hyperparameters — a larger spread indicates more sensitivity to hyperparameters. See Appendix C.3 for details.

larger sizes, we ran many sweeps to find suitable hyperparameters such as the range of learning rates, resulting in the wide spread. Interestingly, the well trained instances of our model at large sizes achieve low validation losses, close to baselines do so with much less overfitting.

3.4 GPT-2 LEVEL MODEL ON OPEN WEB TEXT

Table 2 shows the best model configuration for baseline GPT and RGPT-parallel and our model NRGPT with the respective generation quality metrics. We see that the generation quality of NRGPT is very competitive with GPT and RGPT-parallel while it contains around 34M less parameters than GPT. Figure 5 shows example of generated text by GPT, RGPT-parallel and NRGPT for which the generation quality metrics are provide in Table 2.

Table 1: OWT performance at $n_{embed} = 768$.

Model	Training loss mean ± std	min	max	# runs	# Param.
GPT	**2.905 ± 0.006**	**2.900**	**2.914**	5	124M
GPT_Rec_parallel	3.447 ± 0.046	3.395	3.494	5	85M
NRGPT_H_FF2W	3.456 ± 0.076	3.391	3.540	3	90M

Model	Validation loss mean ± std	min	max	# runs	# Param.
GPT	**2.921 ± 0.005**	**2.915**	**2.929**	5	124M
GPT_Rec_parallel	3.454 ± 0.037	3.411	3.491	5	85M
NRGPT_H_FF2W	3.467 ± 0.073	3.404	3.548	3	90M

Table 2: Best Model Configurations and Quality Metrics for OWT. Note abbreviations: RGPT-parallel → RGPT-P, no of parameters → n_param, grammar quality score → gqs, average pariwise cosine similarity → apcs, distinct-1 → d-1 and distinct-2 → d-2.

Model	Configuration						Metrics				
	lr	min_lr	n_layer	n_head	n_embed	n_params	perplexity	gqs	apcs	d-1	d-2
GPT	7e-4	7e-5	12	12	768	124M	**75**	**0.978**	0.306	0.619	0.965
GPT_Rec_Parallel	1e-4	7e-5	12	12	768	90M	104	0.966	0.306	**0.674**	**0.984**
NRGPT_H_FF2W	6e-4	4e-4	12	12	768	**85M**	99	0.976	**0.336**	0.615	0.975

4 Limitations

NRGPT is an appealing theoretical construct for the inference process of GPT. In our experiments, we observe that NRGPT can achieve similar performance to GPT and its recurrent variants on ListOps, Shakespeare, and OWT. However, NRGPT is computationally the gradient of an energy, which enforces weight sharing and limits how flexibly we can parameterize the architecture. We observe that this constraint also causes a larger amount of hyperparameter sensitivity than GPT variants. In contrast to standard transformers, increasing the number of attention heads in NRGPT actually increases the parameters. We additionally observe that NRGPT has a more difficult time overfitting the training set, which is beneficial in small data regimes but is undesirable in the massive datasets used to train modern LLMs.

5 Discussion and Conclusions

We have presented NRGPT, a minimal modification of the GPT architecture that unifies autoregressive language modeling with energy-based modeling. Our analysis show that under specific conditions on the inference rate matrix $\boldsymbol{\eta}$, this process provably decreases energy, providing a principled foundation for the dynamics. Moreover, relaxing this constraint allows the model to learn its own energy exploration strategy for inference. Thus, our work complements previous studies suggesting that transformers perform GD during inference. Unlike past work, in our model inference is explicitly a gradient-based dynamics, while still maintaining an architecture very similar to GPT. Our experiments show that this framework performs comparably to a fully recurrent GPT model across parameter sizes while generally requiring fewer parameters. NRGPT represents a meaningful step toward understanding the architecture of transformers using energies.

6 Reproducibility Statement

The code for our model, experiments, and analysis is available at https://anonymous.4open.science/status/nrgpt-iclr-E80F in a self-contained environment. The code began as a fork of the excellent nanoGPT repositoryand as such all experiments and models are implemented

using PyTorch (Paszke et al., 2019). Each Shakespeare and ListOps experiment was conducted on a single H100 GPU. OWT experiments were conducted over 4 nodes of 8xH100 GPUs.

REFERENCES

K. Ahn, X. Cheng, H. Daneshmand, and S. Sra. Transformers learn to implement preconditioned gradient descent for in-context learning. *Advances in Neural Information Processing Systems*, 36, 2024.

T. Barrus. pyspellchecker: Pure python spell checking, 2020. URL https://github.com/barrust/pyspellchecker.

A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton, S. Gehrmann, et al. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113, 2023.

R. Flesch. A new readability yardstick. *Journal of applied psychology*, 32(3):221, 1948.

B. Geshkovski, C. Letrouit, Y. Polyanskiy, and P. Rigollet. A mathematical perspective on transformers. *arXiv preprint arXiv:2312.10794*, 2023.

B. Geshkovski, C. Letrouit, Y. Polyanskiy, and P. Rigollet. The emergence of clusters in self-attention dynamics. *Advances in Neural Information Processing Systems*, 36, 2024.

A. Gladstone, G. Nanduru, M. M. Islam, P. Han, H. Ha, A. Chadha, Y. Du, H. Ji, J. Li, and T. Iqbal. Energy-based transformers are scalable learners and thinkers. *arXiv preprint arXiv:2507.02092*, 2025.

B. He and T. Hofmann. Simplifying transformer blocks. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=RtDok9eS3s.

B. Hoover, Y. Liang, B. Pham, R. Panda, H. Strobelt, D. H. Chau, M. Zaki, and D. Krotov. Energy transformer. *Advances in neural information processing systems*, 36:27532–27559, 2023.

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

X. S. Huang, F. Perez, J. Ba, and M. Volkovs. Improving transformer optimization through better initialization. In *International Conference on Machine Learning*, pages 4475–4483. PMLR, 2020.

D. Krotov and J. J. Hopfield. Dense associative memory for pattern recognition. *Advances in neural information processing systems*, 29, 2016.

Y. LeCun, S. Chopra, R. Hadsell, M. Ranzato, F. Huang, et al. A tutorial on energy-based learning. *Predicting structured data*, 1(0), 2006.

N. Nangia and S. Bowman. Listops: A diagnostic dataset for latent tree learning. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Student Research Workshop*, pages 92–99, 2018.

A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

A. Radford, K. Narasimhan, T. Salimans, I. Sutskever, et al. Improving language understanding by generative pre-training, 2018.

A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

N. Reimers and I. Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*, 2019.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

J. Von Oswald, E. Niklasson, E. Randazzo, J. Sacramento, A. Mordvintsev, A. Zhmoginov, and M. Vladymyrov. Transformers learn in-context by gradient descent. In *International Conference on Machine Learning*, pages 35151–35174. PMLR, 2023.

B. Wang and A. Komatsuzaki. Gpt-j-6b: A 6 billion parameter autoregressive language model, 2021.

H. Wang, S. Ma, L. Dong, S. Huang, D. Zhang, and F. Wei. Deepnet: Scaling transformers to 1,000 layers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(10):6761–6774, 2024.

Y. Yang, D. P. Wipf, et al. Transformers from an optimization perspective. *Advances in Neural Information Processing Systems*, 35:36958–36971, 2022.

A LLM USAGE STATEMENT

LLMs were not used for ideation, experiments, or model design. LLM generated code helped with experimental analysis (e.g., plot layouts) and grammar checking of the submission.

B LEARNING RATE AND PRECONDITIONER OF THE FORWARD PASS

$$\textbf{RMSNorm:} \quad g_{Ai} = \gamma_i \frac{x_{Ai}}{\sqrt{\frac{1}{D}\sum_i x_{Ai}^2}} = \sqrt{D}\gamma_i \hat{x}_{Ai} \tag{25}$$

$$\textbf{LayerNorm:} \quad g_{Ai} = \gamma_i \frac{x_{Ai} - \mathbb{E}[x_A]}{\sqrt{\mathrm{Var}[\mathrm{x_A}] + \epsilon}} + \beta_i \tag{26}$$

where x_{Ai} are the components with $A \in 1 \dots N$ and $i \in 1 \dots D$.

Proof of Proposition 2.1: Energy Descent. Using chain rule

$$\dot{E} = \partial_g E \cdot \partial_x g \cdot \dot{x} = \partial_g E \cdot \partial_x g \cdot \eta \partial_g E \tag{27}$$

Defining $\Gamma = \mathrm{diag}(\gamma)$, the Jacobian of g becomes

$$\frac{\partial g_{Ai}}{\partial x_{B\rho}} = \Gamma_{ij} \frac{\delta_{AB}}{r_A} \left(\delta_{j\rho} - y_{Aj} y_{A\rho}\right), \tag{28}$$

where

$$\textbf{RMSNorm:} \qquad r_A = \|x_A\| / \sqrt{D}, \qquad y_A = x_A / r_A \tag{29}$$

$$\textbf{LayerNorm:} \qquad r_A = \sqrt{\mathrm{Var}[x_A] + \epsilon}, \qquad y_A = (x_A - i_A)/r_A. \tag{30}$$

Since typically $\epsilon = 10^{-5}$ is a small constant, $\|y_A\| \approx 1$ for LayerNorm, and exactly 1 for RMSNorm. Therefore

$$\frac{\partial g_A}{\partial x_A} = \frac{1}{r_A} \Gamma P_A, \qquad P_A = P_A^T, \qquad P_A^2 = P_A + O(\epsilon) \tag{31}$$

where the approximate projection matrix P_A is positive semi-definite, with $\hat{y}_A / \|y_A\|$ being its sole null eigenvector. Plugging into equation 27

$$\delta E = -\sum_A \frac{1}{r_A} \mathrm{Tr}\left[\frac{\partial E}{\partial g_A} \Gamma P_A \eta^T \frac{\partial E}{\partial g_A}^T\right] \tag{32}$$

We want $\delta E < 0$, which can be achieved if the symmetric part of $\Gamma P_A \eta^T$ is p.s.d.. Two simple solutions to this are

$$\eta = c\Gamma, \qquad \text{or} \qquad \eta = M(x) P_A \Gamma \tag{33}$$

where $M(x)$ is an arbitrary p.s.d. matrix and $c > 0$. If we want η to be simple weights instead of an x dependent neural network, the solution is $\eta = c\Gamma$. □

Note that equation 32 does not restrict the anti-symmetric part of $\Gamma P_A \eta$. Using $\eta = c\Gamma + B$, the antisymmetric part satisfies $B^T P_A \Gamma = -\Gamma P_A B$. Since P_A is rank $D - 1$ for each A, the anti-symmetric part doesn't seem to have an x-independent solution.

C EXPERIMENTS

C.1 EVALUATION METRICS

To assess the generation quality of our language models, we utilize several complementary metrics. We use Perplexity as a measure of model uncertainty, computed using a pretrained GPT-2 model to evaluate how well the generated text aligns with expected language patterns. Lower perplexity

indicates more fluent and predictable text, with scores typically ranging from 10 (excellent) to 1000+ (poor quality). For lexical diversity, we utilize Distinct-1 and Distinct-2, which measure the ratio of unique unigrams and bigrams to total n-grams (or total words) in the generated text, respectively. These metrics range from 0 to 1, where higher values indicate greater vocabulary diversity and less repetitive generation. A Distinct-1 score near 0 suggests highly repetitive text, while scores above 0.8 indicate rich vocabulary usage. We utilize Average Pairwise Cosine Similarity using Sentence-BERT embeddings (Reimers and Gurevych, 2019) to measure semantic diversity within generated samples. This metric calculates the mean cosine similarity between all pairs of generated sentences, ranging from -1 to 1. Optimal values fall between 0.3 and 0.6, balancing semantic diversity with topical coherence. Values below 0.3 indicate excessive divergence with potentially incoherent topic-jumping between sentences, while values above 0.7 suggest repetitive or redundant content with insufficient variation. The target range of 0.3-0.6 represents healthy diversity where generated sentences explore different aspects of a topic while maintaining semantic relevance and coherent narrative flow.

Finally, We compute the Grammar Quality Score (GQS), a composite metric that combines rule-based grammar error detection, spelling accuracy via spellchecker (Barrus, 2020), and readability assessment using Flesch-Kincaid grade level (Flesch, 1948). GQS ranges from 0 (poor grammar) to 1 (perfect grammar), weighting grammatical correctness (50%), spelling accuracy (30%), and readability (20%). The metric identifies errors across ten categories including subject-verb agreement, tense consistency, and punctuation, with severity-weighted scoring. For complete context and to understand what the ideal ranges are for all of these metrics, see Table 3.

Table 3: Ideal ranges for generation quality metrics

Metric	**Good Range**	**Interpretation**
Perplexity	15-50	Lower is better (fluency)
Distinct-1	0.6–0.9	Higher is better (vocabulary diversity)
Distinct-2	0.8–0.95	Higher is better (bigram diversity)
GQS	0.8–1.0	Higher is better (grammatical quality)
Avg. Cosine Similarity	0.3–0.6	Moderate values best (semantic diversity)

C.2 LISTOPS

We perform experiment on nested math operations on lists of integers, which are a version of ListOps (Nangia and Bowman, 2018). Our ListOps setting consists of three functions: maximum, median and sum modulo 20. Our inputs range from 0 to 19.

C.3 SHAKESPEARE

As training data we used the full Shakespeare training set, tokenized such that each character constitutes a single token. Models were evaluated across the same held out validation subset. Across all models, we used a context window of 256 tokens, dropout of 10%, the AdamW(β_1=0.9, β_2=0.99), and 40k maximum update iterations using a minibatch size of 64. We varied model sizes by sweeping over embedding dimensions $(32, 64, 128, 256, 380, 512, 768)$ and the number of attention heads $(1, 2, 8)$. For the recurrent models, we additionally varied the number of layers across $(3, 6, 8)$, though this does not affect the parameter count.

We found the recurrent models to be quite sensitive to choices of learning rate and learning rate schedules. Hence, we explored several different maximum learning rates $(1e-3, 7.5e-4, 3e-4, 1e-4)$, schedules (cosine, exponential), and minimum learning rates ($10\times$, $20\times$, and $100\times$ smaller than the max learning rate). LR warm-up was 100 updates for all experiments.

In Figure 4 we emphasize the best losses we were able to achieve for each model size. In addition, we capture the model's sensitivity to hyperparams by showing the top 50% performing models across all hyperparameters.

C.3.1 Best Model Configurations and Metrics

Table 4 shows the best model configuration for baseline GPTs and our model NRGPT with the respective generation quality metrics. We see that NRGPT outperforms the baseline GPT, RGPT and RGPT-parallel in terms of generation quality while it has only half of the nparams of GPT.

Table 4: Best model configurations and quality metrics for Shakespeare. Note abbreviations: RGPT-parallel → RGPT-P, no of parameters → n_param, grammar quality score → gqs, average pariwise cosine similarity → apcs, distinct-1 → d-1 and distinct-2 → d-2.

Model	**Configuration**						**Metrics**				
	lr	min_lr	n_layer	n_head	n_embed	n_params	perplexity	gqs	apcs	d-1	d-2
GPT	5e-3	5e-4	8	4	64	0.4M	294	0.913	0.198	0.751	0.978
RGPT	1e-3	1e-4	8	1	256	0.8M	476	0.896	0.164	0.794	0.995
RGPT-P	2e-3	2e-4	8	1	128	**0.2M**	410	0.888	0.190	0.838	**1**
NRGPT_H_FF1	3e-4	3e-5	6	2	512	2M	318	0.901	**0.218**	**0.797**	0.995
NRGPT_H_FF2W	1e-3	1e-4	8	1	128	**0.2M**	**283**	**0.975**	0.176	0.765	0.995

C.4 OpenWebText

We perform experiments on natural language modeling using the OpenWebText corpus, which is an open-source recreation of GPT-2's WebText dataset (Radford et al., 2019). The dataset contains approximately 17GB of text with 9B tokens that came from 8 million documents. We tokenize using byte-pair encoding (BPE) with a vocabulary size of 50,257 tokens. Our training sequences are fixed-length contexts of 1024 tokens. Table 5 lists the hyperparameters and respective values/ranges used in our experiments.

Table 5: OWT Hyperparameters and range of values.

Hyperparameter	**Used Values**
batch_size	12
block_size	1024
n_layer	[3, 6, 9, 12, 24]
n_head	[1, 2, 4, 6, 12, 16]
n_embed	[768, 1020, 1536]
learning_rate, lr	[1e-3 - 1e-5]
min_lr	[lr/10 - lr]
beta1	0.9
beta2	0.99
weight_decay	[1e-1, 1e-2]
gradient_accumulation_steps	40
eval_interval	1,000
eval_iters	200
max_iters	100000
warmup_iters	[100, 2000]
dropout	0.0

C.4.1 Best Model Configurations and Metrics

Table 6 shows the best model configuration for baseline GPT and RGPT-parallel and our model NRGPT with the respective generation quality metrics. We see that the generation quality of NRGPT is very competitive with GPT and RGPT-parallel while it contains around 34M less parameters than GPT. Figure 5 shows example of generated text by GPT, RGPT-parallel and NRGPT for which the generation quality metrics are provide in Table 6.

Table 6: Best Model Configurations and Quality Metrics for OWT. Note abbreviations: RGPT-parallel → RGPT-P, no of parameters → n_param, grammar quality score → gqs, average pariwise cosine similarity → apcs, distinct-1 → d-1 and distinct-2 → d-2.

Model	**Configuration**						**Metrics**				
	lr	min_lr	n_layer	n_head	n_embed	n_params	perplexity	gqs	apcs	d-1	d-2
GPT	7e-4	7e-5	12	12	768	124M	**75**	**0.978**	0.306	0.619	0.965
RGPT-P	1e-4	7e-5	12	12	768	90M	104	0.966	0.306	**0.674**	**0.984**
NRGPT	6e-4	4e-4	12	12	768	**85M**	99	0.976	**0.336**	0.615	0.975

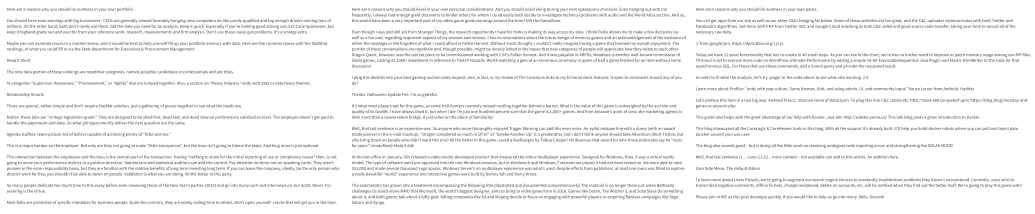

Figure 5: Best Generation Examples from GPT (left column), RGPT-parallel (middle column) and NRGPT (right column).