Energy-Based Learning and EBT

August 28, 2026

Overview

- A Tutorial on Energy-Based Learning. NeurIPS Tutorial 2006.

- Energy-Based Transformers are Scalable Learners and Thinkers. ICLR 2026.

A Tutorial on Energy-Based Learning

Yann LeCun, Sumit Chopra, Raia Hadsell,
Marc'Aurelio Ranzato, and Fu Jie Huang
The Courant Institute of Mathematical Sciences,
New York University
{yann,sumit,raia,ranzato,jhuangfu}@cs.nyu.edu
http://yann.lecun.com

v1.0, August 19, 2006

To appear in "Predicting Structured Data",

G. Bakir, T. Hofman, B. Schölkopf, A. Smola, B. Taskar (eds)

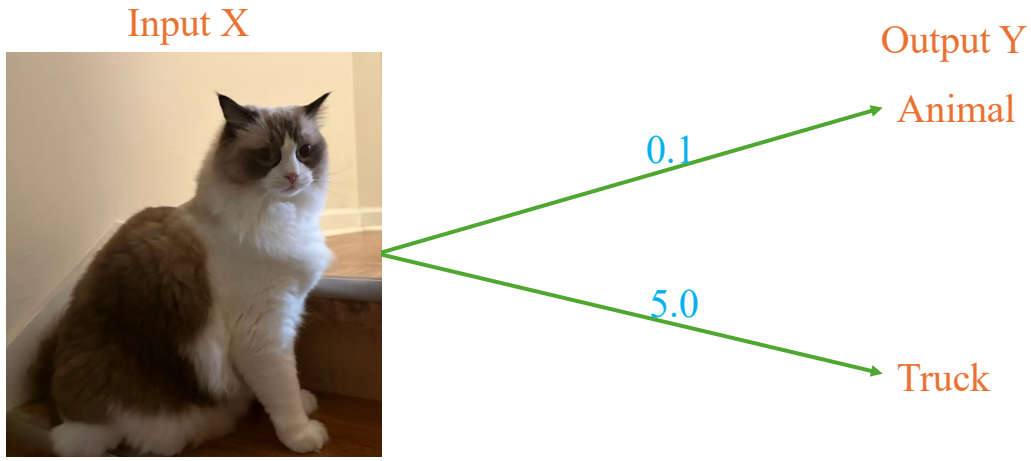
MIT Press, 2006

Energy-Based Models –Introduction

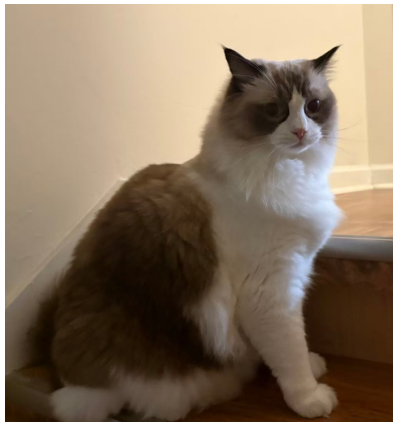
- Energy-Based Models (EBMs) capture dependencies (for example, label Y depends on input X) between variables by associating a scaler energy to each configuration of the variables.

$$Y^* = \text{argmin}_{Y \in \mathcal{Y}} E(Y, X)$$

- Energy-Based Inference:



Energy-Based Models – Training Objective for Discrete Case



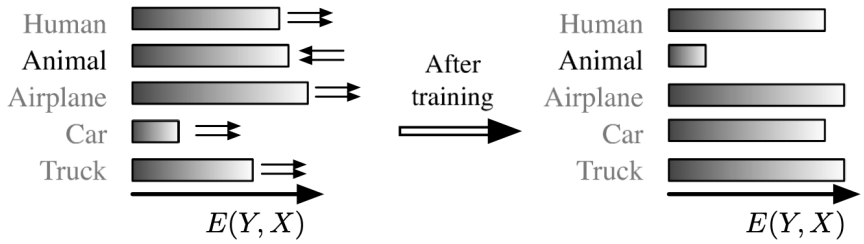
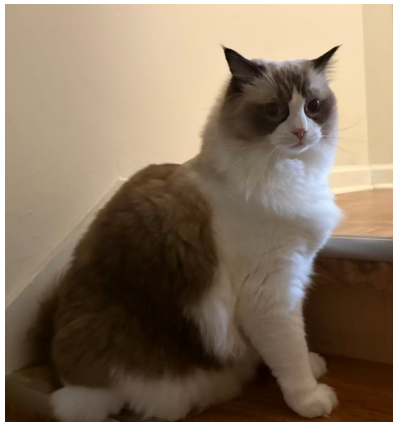


Figure 3: *How training affects the energies of the possible answers in the discrete case: the energy of the correct answer is decreased, and the energies of incorrect answers are increased, particularly if they are lower than that of the correct answer.*

Energy-Based Models – Training Objective for Continuous Case



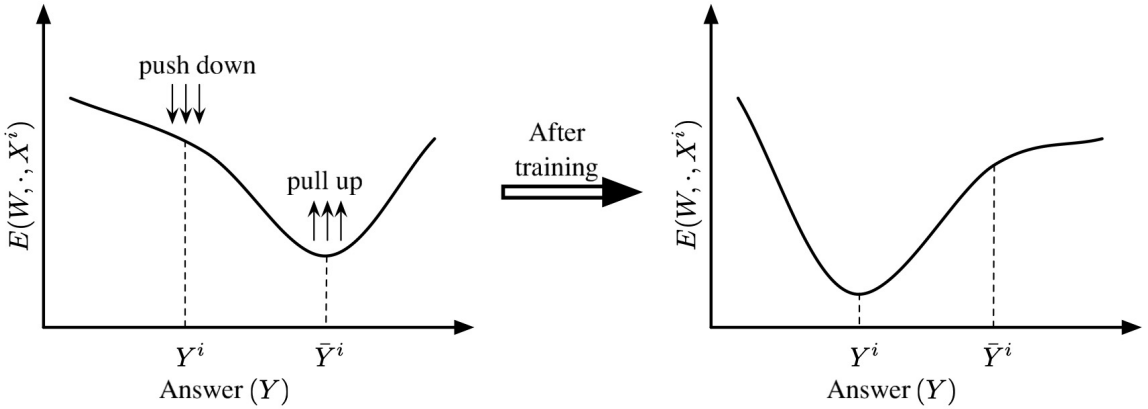


Figure 4: *The effect of training on the energy surface as a function of the answer Y in the continuous case. After training, the energy of the correct answer Y^i is lower than that of incorrect answers.*

Energy-Based Models – Important Terms

- The Model (Energy Function): $\mathcal{E} = \{E(W, Y, X) : \ W \in \mathcal{W}\}$
- Train Samples: $\mathcal{S} = \{(X^i, Y^i) : i = 1 \ldots P\}$
- The Learning problem is to find the W that minimizes the loss: $W^* = \min_{W \in \mathcal{W}} \mathcal{L}(W, \mathcal{S})$
- The loss functional is: $\mathcal{L}(E, \mathcal{S}) = \frac{1}{P} \sum_{i=1}^{P} L(Y^i, \underbrace{E(W, \mathcal{Y}, X^i)}) + R(W)$ regularizer

a slice of landscape with clamped X and varying Y

Energy-Based Models – As a Universal Framework

- As long as the model 1) outputs a scalar value; 2) lower value represents more compatible predictions; and 3) exist some methods to find the lower values over Y space, it can be viewed as energy-based model.

- Therefore, there are a variety of options for the combination of architecture and loss function.

- The tutorial lists:

Architecture	Loss
Regression	Energy Loss
Two-Class Classifier	Generalized Perceptron Loss
Multiclass Classifier	Generalized Margin Losses
Probablistic Latent Variables	Negative Log-Likelihood Loss
….	…

Energy-Based Models – As a Universal Framework

- The architectures are different, and how do we integrate energy-based learning into the varying architectures?

$$E(x,y)=R(F(x), H(y), \text{interaction})$$

Architecture	F(x)	H(y)	Interaction + Readout	Loss
Regressor	network G(x)	identity y	distance $\|\cdot\|^2$	Energy loss
Classifier	network → score vector	one-hot	inner product	Perceptron / Hinge / NLL / MCE
Siamese (implicit regression)	encoder G_1	encoder G_2 (often shared)	embedding distance	Square-square / Square-exp (contrastive)
Factor graph / GTN	local features	local labels	sum of local energies	NLL (→CRF) / margin / discriminative
EBT (not in the tutorial)	Joint encoding: x, y enter Transformer together		scalar head	Unrolled CE (not in tutorial)

Energy-Based Models – Challenge: Collapse

- Picking a proper combination of architecture and loss is important, otherwise the training may results in a collapsed landscape.

- Here is a toy example:

  1) Problem: consider of learning a function that computes Y = X^2.

  2) Training samples: 200 samples (X^i, Y^i) where $Y^i = X^{i2}$, randomly sampled with a uniform distribution between -1 and +1.

  3) The energy function: $E(W, Y, X) = ||G_W(X) - Y||_1$.

  4) The input X is passed through a parametric function G_W, a two layer neural network with 1 input unit, 20 hidden units with sigmoids, and 1output unit.

Energy-Based Models – Challenge: Collapse

- Picking a proper combination of architecture and loss is important, otherwise the training may results in a collapsed landscape.

- Here is a toy example:

  1) Problem: consider of learning a function that computes Y = X^2.

  2) Training samples: 200 samples (X^i, Y^i) where $Y^i = X^{i2}$, randomly sampled with a uniform distribution between -1 and +1.

  3) The energy function: $E(W, Y, X) = ||G_W(X) - Y||_1$.

  4) The input X is passed through a parametric function G_W, a two layer neural network with 1 input unit, 20 hidden units with sigmoids, and 1output unit.

Energy-Based Models – Challenge: Collapse

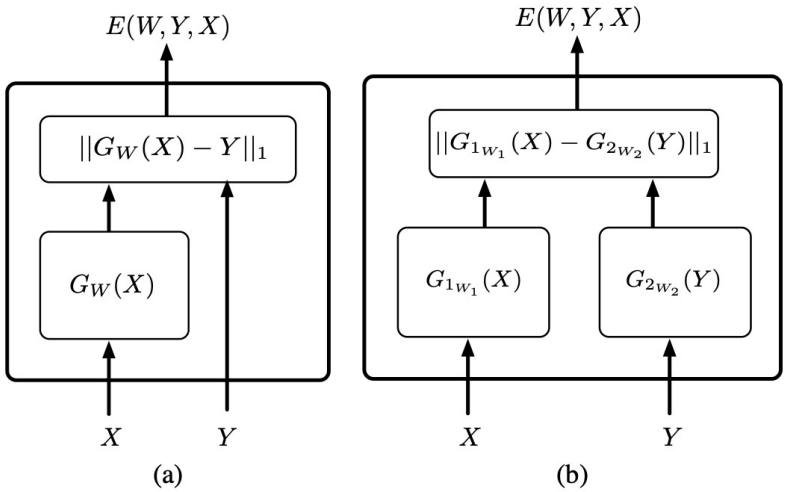


Figure 9: (a): *A simple architecture that can be trained with the* energy *loss*. (b): *An implicit regression architecture where X and Y are passed through functions $G_{1_{W_1}}$ and $G_{2_{W_2}}$ respectively. Training this architecture with the energy loss causes a collapse (a flat energy surface). A loss function with a contrastive term corrects the problem*.

Energy-Based Models – Challenge: Collapse

- If train with energy loss -- minimize E (W, X, Y_correct):

Why figure 10 not collapses?

The architecture (the V shape) makes pushing down on the expected prediction will automatically pulling up other Energies.

$$\mathcal{L}_{\text{energy}}(W, \mathcal{S}) = \frac{1}{P}\sum_{i=1}^{P} E(W, Y^i, X^i) = \frac{1}{P}\sum_{i=1}^{P} ||G_W(X) - Y||_1$$

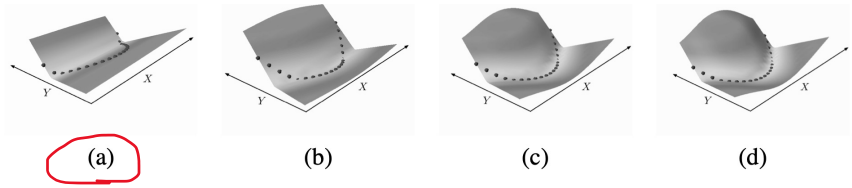


Figure 10: *The shape of the energy surface at four intervals while training the system in Figure 9(a) with stochastic gradient descent to minimize the* energy loss. *The X axis is the input, and the Y axis the output. The energy surface is shown (a) at the start of training, (b) after 10 epochs through the training set, (c) after 25 epochs, and (d) after 39 epochs. The energy surface has attained the desired shape where the energy around training samples (dark spheres) is low and energy at all other points is high.*

$$\mathcal{L}_{\text{energy}}(W, \mathcal{S}) = \frac{1}{P}\sum_{i=1}^{P} E(W, Y^i, X^i) = \frac{1}{P}\sum_{i=1}^{P} ||G_{1_{W_1}}(X) - G_{2_{W_2}}(Y)||_1$$

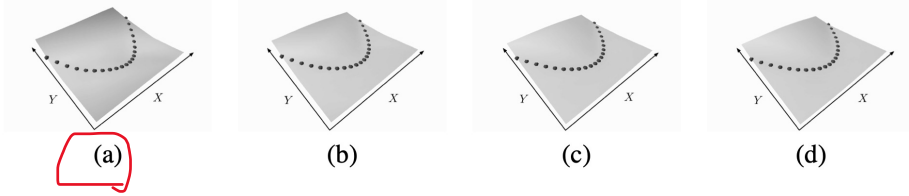


Figure 11: *The shape of the energy surface at four intervals while training the system in Figure 9(b) using the energy loss. Along the X axis is the input variable and along the Y axis is the answer. The shape of the surface (a) at the start of the training, (b) after 3 epochs through the training set, (c) after 6 epochs, and (d) after 9 epochs. Clearly the energy is collapsing to a flat surface.*

Energy-Based Models – Challenge: Collapse

m: a positive margin;
The most offending incorrect answer
(an undesired answer with min energy

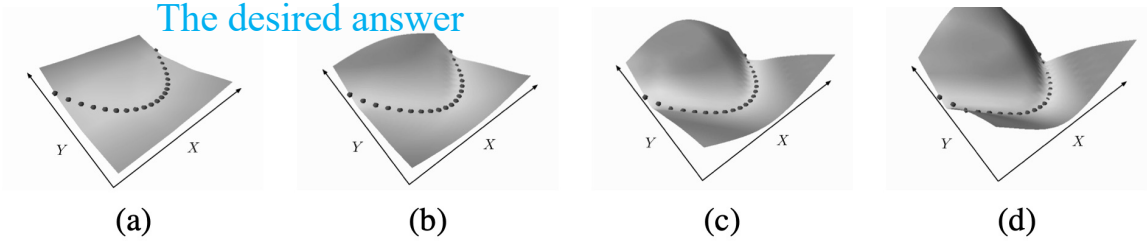
- But this can be fixed by adding a contrastive term (intentionally pull-up the wrong answers

Alternative loss function 1: Square-Square Loss.

Push down on the energy of the desired answers

$$L(W, Y^i, X^i) = E(W, Y^i, X^i)^2 + \left(max(0, m - E(W, \bar{Y}^i, X^i))\right)^2$$

The desired answer



(a) (b) (c) (d)

Case 1:
E is higher than m:
Good, Loss + 0.

Case 2:
E is lower than m:
Not good, Loss + (m-E)^2

Figure 12: *The shape of the energy surface at four intervals while training the system in Figure 9(b) using* square-square *loss. Along the x-axis is the variable X and along the y-axis is the variable Y. The shape of the surface at (a) the start of the training, (b) after 15 epochs over the training set, (c) after 25 epochs, and (d) after 34 epochs. The energy surface has attained the desired shape: the energies around the training samples are low and energies at all other points are high.*

Energy-Based Models – Challenge: Collapse

- But this can be fixed by adding a contrastive term (intentionally pull-up the wrong answers

Alternative loss function 2: Negative Loglikelihood Loss.

Push down on the energy of the desired answers

$$L(W, Y^i, X^i) = E(W, Y^i, X^i) + \frac{1}{\beta} \log \left(\int_{y \in \mathcal{Y}} e^{-\beta E(W, y, X^i)} \right)$$

The desired answer

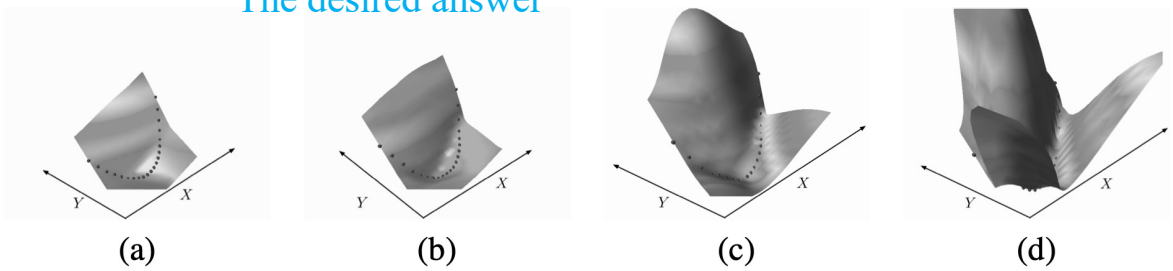


Figure 13: *The shape of the energy surface at four intervals while training the system in Figure 9(b) using the negative log-likelihood loss. Along the X axis is the input variable and along the Y axis is the answer. The shape of the surface at (a) the start of training, (b) after 3 epochs over the training set, (c) after 6 epochs, and (d) after 11 epochs. The energy surface has quickly attained the desired shape.*

- Pulls up on all answers.
- Y_i: still push down
- Contrary: to infinite

Expensive: adopt a simple sampling – the integral is Approximated by a sum of 20 points regularly spaced Between -1 and 1 in the Y Direction.

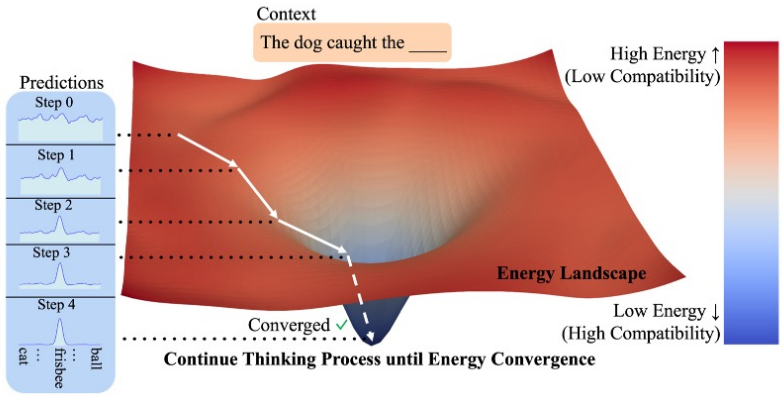
ENERGY-BASED TRANSFORMERS ARE SCALABLE LEARNERS AND THINKERS

Alexi Gladstone[1,2], **Ganesh Nanduru**[1], **Md Mofijul Islam**[1,3], **Peixuan Han**[2], **Hyeonjeong Ha**[2], **Aman Chadha**[3,4], **Yilun Du**[5], **Heng Ji**[2], **Jundong Li**[1], **Tariq Iqbal**[1]

[1]UVA [2]UIUC [3]Amazon GenAI† [4]Stanford University [5]Harvard University

 **energy-based-transformers.github.io**  **github.com/alexiglad/EBT**

The main topic: Energy-Based Transformer

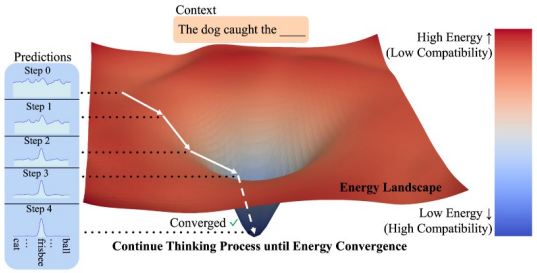


- Our focus:
  Guide the transformer produce compatible tokens via energy minimization through gradient descent over the energy landscape.

- This transformer can also be applied to vision tasks (such as next video frame prediction).

The main topic: Energy-Based Transformer

- Though the depiction is elegant, if you think from the perspective of 0 to 1, shaping a proper landscape is challenging.



Ideally,

1) We need to ensure that the lower location brings better results.
2) We need to ensure that the landscape is somewhat smooth so that the minimum is easy to be targeted.
3) We need to ensure the landscape is differentiable and 2nd differentiable so that we can apply gradient descent and other training paradigm.
4) We need to ensure the landscape has contrast – low only one/near the data manifold, high elsewhere; otherwise the landscape collapses to a flat surface.

Key Idea: Energy-Based Transformers

The energy model serves as both verifier (evaluate compatibility) and generator (optimize predictions according to energy minimization).

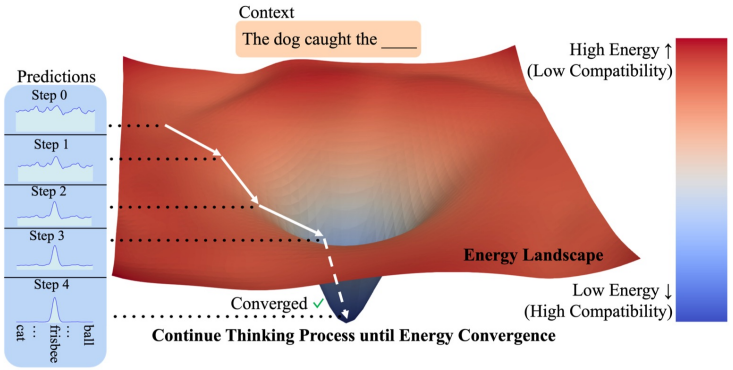
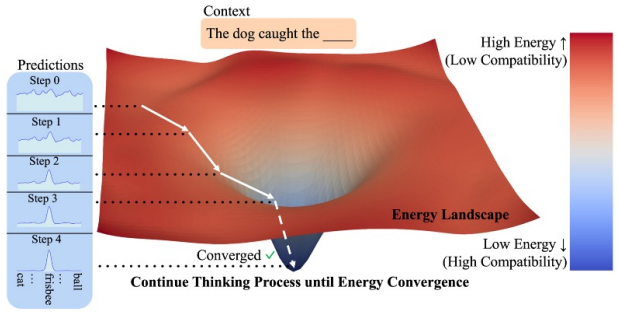


Figure 3: **Thinking Process Visualization.** A learned energy landscape and its optimization through gradient descent, interpreted as a thinking process. In this example, the model predicts a distribution over text tokens, progressively shifting from an initial random distribution toward the target distribution. At each step, the EBM assigns an energy scalar indicating how **compatible** the current prediction is with the context, visualized as the landscape's height (Facet 3). This scalar's convergence allows the model to determine whether the prediction is adequate or if further thinking is necessary. Uncertainty (Facet 2) can be represented by landscapes that are harder to optimize or by landscapes with many local minima, allowing the model to know when it requires more steps to think (Facet 1). Adapted from [57].

Energy-Based Transformers - Landscape

- Micro-Level: slice landscape $E_\theta(x_{\text{obs}}, \cdot)$, where the context x and transformer parameters θ are clamped.



- Mid-Level: Joint landscape $E_\theta(\cdot, \cdot)$, where the transformer parameters θ is clamped.

- Macro-Level: the family of landscapes $E_\theta : \theta \in \mathcal{W}$. Nothing is clamped.
  The landscape envolves as transformer parameters θ develope during training stage.

Energy-Based Transformers - Application

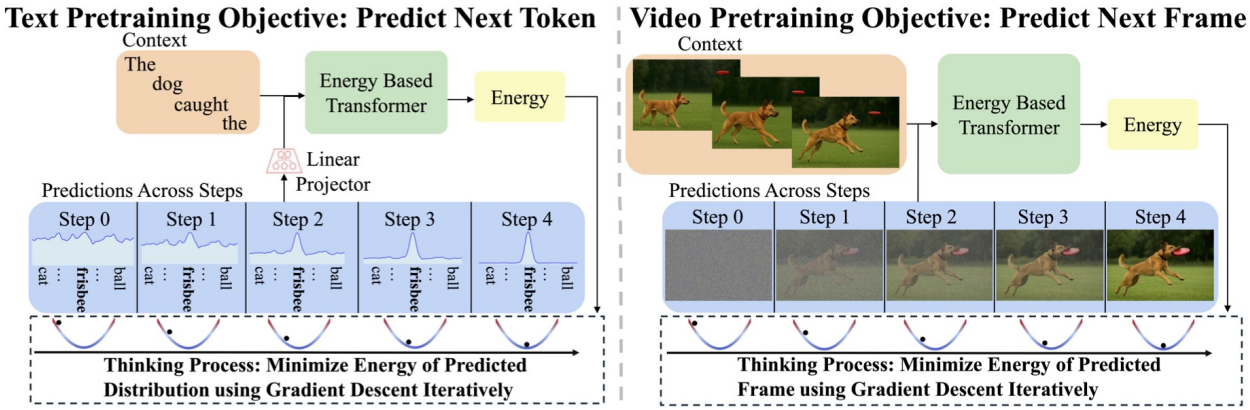


Figure 2: **EBT for Autoregressive Modeling.** Each blue box corresponds to a different prediction at each step of the thinking process, where the initial prediction starts as random. At each step, a new prediction is fed into the model, which gives an energy scalar for the prediction's current *compatibility* (unnormalized likelihood) with the context (Facet 2). Then, the gradient of this energy with respect to the prediction is calculated and used to update the prediction. This gradient descent update is done iteratively to refine the prediction until convergence of the predicted energy, which allows for dynamic use of computation (Facet 1).

Energy-Based Transformers – Model

- There's no foundation model for energy model + transformer.

  They had to rebuilt an existing transformer (Llama 2) and pretrain the model from scratch.

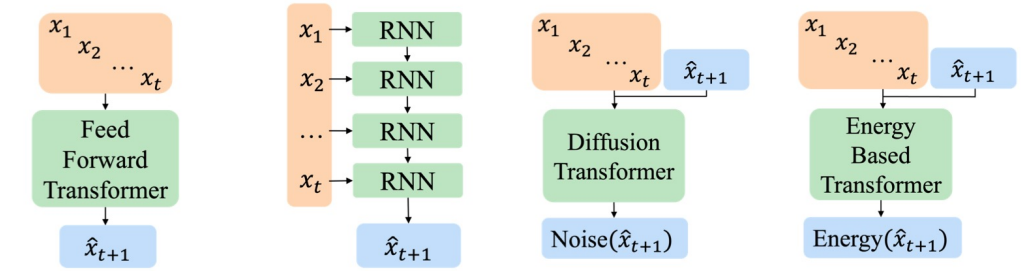


Figure 1: **Autoregressive Architecture Comparison.** (a) Autoregressive (AR) Transformer is the most common, with (b) RNNs becoming more popular recently [22, 23]. (c) Diffusion Transformers [26] (which are often bidirectional but can also be autoregressive) are the most similar to EBT, being able to dynamically allocate computation during inference, but predict the noise rather than the energy [27, 28]. Consequently, diffusion models cannot give unnormalized likelihood estimates at each step of the thinking process, and are not trained as explicit verifiers, unlike EBTs.

Energy-Based Transformers – Model Architecture

- There's no foundation model for energy model + transformer.

  They had to rebuilt an existing transformer (Llama 2) and pretrain the model from scratch.

Llama 2

```
LlamaForCausalLM(
  (model): LlamaModel(
    (embed_tokens): Embedding(32000, 4096)
    (layers): ModuleList(
      (0-31): 32 x LlamaDecoderLayer(
        (self_attn): LlamaAttention(
          (q_proj): Linear(in_features=4096, out_features=4096, bias=False)
          (k_proj): Linear(in_features=4096, out_features=4096, bias=False)
          (v_proj): Linear(in_features=4096, out_features=4096, bias=False)
          (o_proj): Linear(in_features=4096, out_features=4096, bias=False)
        )
        (mlp): LlamaMLP(
          (gate_proj): Linear(in_features=4096, out_features=11008, bias=False)
          (up_proj): Linear(in_features=4096, out_features=11008, bias=False)
          (down_proj): Linear(in_features=11008, out_features=4096, bias=False)
          (act_fn): SiLUActivation()
        )
        (input_layernorm): LlamaRMSNorm((4096,), eps=1e-06)
        (post_attention_layernorm): LlamaRMSNorm((4096,), eps=1e-06)
      )
    )
    (norm): LlamaRMSNorm((4096,), eps=1e-06)
    (rotary_emb): LlamaRotaryEmbedding()
  )
  (lm_head): Linear(in_features=4096, out_features=32000, bias=False)
)
```

EBT

```
EBT_NLP_Full(
  (embeddings): Embedding(50277, 384)
  (vocab_to_embed): Linear(in_features=50277, out_features=384, bias=False)
  (transformer): EBTTimeConcat(
    (layers): ModuleList(
      (0-5): 6 x TransformerBlock(
        (attention): Attention(
          (wq): Linear(in_features=384, out_features=384, bias=False)
          (wk): Linear(in_features=384, out_features=384, bias=False)
          (wv): Linear(in_features=384, out_features=384, bias=False)
          (wo): Linear(in_features=384, out_features=384, bias=False)
        )
        (feed_forward): FeedForward(
          (w1): Linear(in_features=384, out_features=384, bias=False)
          (w2): Linear(in_features=384, out_features=384, bias=False)
          (w3): Linear(in_features=384, out_features=384, bias=False)
        )
        (attention_norm): RMSNorm()
        (ffn_norm): RMSNorm()
      )
    )
    (norm): RMSNorm()
    (time_embeddings): Embedding(4, 384)
    (final_layer): Linear(in_features=384, out_features=1, bias=False)
  )
)
```

Energy-Based Transformers – Model Architecture

```
EBT_NLP_Full(
  (embeddings): Embedding(50277, 384)
  (vocab_to_embed): Linear(in_features=50277, out_features=384, bias=False)
  (transformer): EBTTimeConcat(
    (layers): ModuleList(
      (0-5): 6 x TransformerBlock(
        (attention): Attention(
          (wq): Linear(in_features=384, out_features=384, bias=False)
          (wk): Linear(in_features=384, out_features=384, bias=False)
          (wv): Linear(in_features=384, out_features=384, bias=False)
          (wo): Linear(in_features=384, out_features=384, bias=False)
        )
        (feed_forward): FeedForward(
          (w1): Linear(in_features=384, out_features=384, bias=False)
          (w2): Linear(in_features=384, out_features=384, bias=False)
          (w3): Linear(in_features=384, out_features=384, bias=False)
        )
        (attention_norm): RMSNorm()
        (ffn_norm): RMSNorm()
      )
    )
    (norm): RMSNorm()
    (time_embeddings): Embedding(4, 384)
    (final_layer): Linear(in_features=384, out_features=1, bias=False)
  )
)
```

Modifications:

- vocab_to_embed: encode the additional variable – candidate prediction ŷ;

- Time_embeddings: Embedding (M, D)
  -- M represents the step to find the minimum energy, and D represents the hidder layers dimension.

- Final_layer: (D, 1). Make sure the transformer outputs a single scalar $E\theta(x, \hat{y})$ for the input text x and the candidate prediction ŷ.

- Slight modifications for the layers and dimension size (varied by multiple sizes)

Energy-Based Transformers – Where is the Landscape?

```
EBT_NLP_Full(
  (embeddings): Embedding(50277, 384)
  (vocab_to_embed): Linear(in_features=50277, out_features=384, bias=False)
  (transformer): EBTTimeConcat(
    (layers): ModuleList(
      (0-5): 6 x TransformerBlock(
        (attention): Attention(
          (wq): Linear(in_features=384, out_features=384, bias=False)
          (wk): Linear(in_features=384, out_features=384, bias=False)
          (wv): Linear(in_features=384, out_features=384, bias=False)
          (wo): Linear(in_features=384, out_features=384, bias=False)
        )
        (feed_forward): FeedForward(
          (w1): Linear(in_features=384, out_features=384, bias=False)
          (w2): Linear(in_features=384, out_features=384, bias=False)
          (w3): Linear(in_features=384, out_features=384, bias=False)
        )
        (attention_norm): RMSNorm()
        (ffn_norm): RMSNorm()
      )
    )
    (norm): RMSNorm()
    (time_embeddings): Embedding(4, 384)
    (final_layer): Linear(in_features=384, out_features=1, bias=False)
  )
)
```

- The landscape is implicit.
- The forward pass gives a scalar value $E\theta(x, \hat{y})$, and this is the landscape height.
- So the energy scalar is shaped by the transformer parameters.
- In other words, we need to update the transformer parameters to shape the energy landscape.

Energy-Based Transformers – Training

Algorithm 1: Training

Inputs: Context x, Target y, EBM $E_\theta(x, \hat{y})$
Hparams: Steps N, Step Size α, Loss $J(\cdot)$

Sample $\hat{y}_0 \sim \mathcal{N}(0, I)$;
for $i = 0, \ldots, N-1$ **do**
  $\hat{y}_{i+1} \leftarrow \hat{y}_i - \alpha \nabla_{\hat{y}_i} E_\theta(x, \hat{y}_i)$;
$\mathcal{L} \leftarrow J(\hat{y}_N, y)$;
return $\mathcal{L}$, *update* E_θ;

Energy-Based Transformers – Inference

Algorithm 2: Inference with Verification

Inputs: Context x, EBM $E_\theta(x, \hat{y})$
Hparams: Steps N, Step Size α, Samples M
for $j = 1, \ldots, M$ **do**
  Sample $\hat{y}_{0,j} \sim \mathcal{N}(0, I)$;
  for $i = 0, \ldots, N-1$ **do**
    $\hat{y}_{i+1,j} \leftarrow \hat{y}_{i,j} - \alpha \nabla_{\hat{y}_{i,j}} E_\theta(x, \hat{y}_{i,j})$;
return $\hat{y}^* = \text{argmin}_j \, E_\theta\big(x, \hat{y}_{N,j}\big)$;

Energy-Based Transformers – Regularization

Motivation: achieve smooth energy landscape with a single local minimum is difficult, they found three key techniques for learning smoother.

- Replay buffer

- A variant of Langevin Dynamics:

$$\hat{y}_{i+1} = \hat{y}_i - \alpha \nabla_{\hat{y}_i} E_\theta\left(x, \hat{y}_i\right) + \eta_i, \quad \eta_i \sim \mathcal{N}(0, \sigma)$$

- Vary the paths length.

Energy-Based Transformers – Z is dropped

lower probability. To avoid the intractability of the partition function, it is common to work with **unnormalized EBMs**, which dispense of the partition function in favor of representing relative unnormalized probabilities. This formulation shifts the focus from addressing the partition function, to simply assigning low energy to the true data manifold and high energy elsewhere [42, 51], offering

$$p_\theta(x) = \frac{e^{-E_\theta(x)}}{Z(\theta)}$$

$$Z(\theta) = \int e^{-E_\theta(x)} dx$$

$$p_\theta(x, \hat{y}) \propto e^{-E_\theta(x, \hat{y})}$$

- **Is it okay that energies are unnormalized probabilities?**
  Yes, for most real-world applications, you only ever need sample **relative likelihood** comparison; it's significantly less common to need the exact likelihood of samples. An example of this is reward models, which can be seen as EBMs (just multiplying the reward by -1), where all that really matters is the relative reward for choosing which sample to use or which behavior to perform.

The dog caught the tissue (0.1)
English (5.0)

Z is the same as they share the same X (context) and θ
(The slice landscape)

Energy-Based Transformers – the cost of Z being dropped

- This is next-token generation, not reasoning.

The dog caught the tissue (0.1)
English (5.0)

Z is the same as they share the same X (context) and θ
(The slice landscape) $E_\theta(x_{\text{obs}}, \cdot)$

$$Z(\theta) = \int e^{-E_\theta(x)} dx$$

- If we consider evaluate the quality of different reasoning paths, Z cannot be simply dropped, as the context x could be varied.

Energy-Based Transformers - Experiments

The experienments results are reported by perplexity, but not accuracy.

The authors claimed:

- we focus on reporting perplexity as our relatively small models trained from scratch, when compared to current foundation models, do not achieve high accuracies on many of these benchmarks.

- perplexity often functions as a more linear metric than accuracy, enabling a more comparable analysis of downstream performance as we scale compute during inference.

Energy-Based Transformers - Experiments

Perplexity:

- It measures the model's surprise or uncertainty when guessing the next word or token. A lower PPL means the model is less surprised.

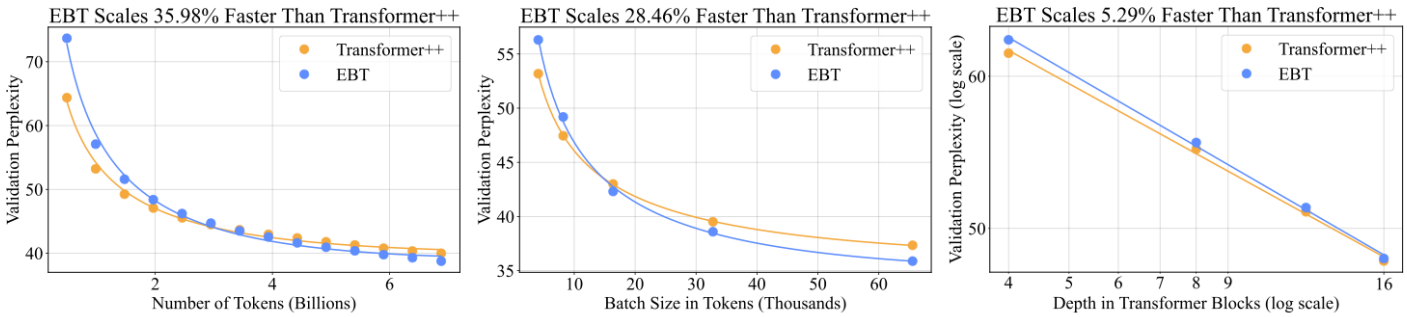
$$\mathrm{PPL}(X) = \exp\left\{-\frac{1}{t}\sum_{i}^{t}\log p_\theta(x_i|x_{<i})\right\}$$

Energy-Based Transformers - Experiments

Table 5: **Sudoku OOD Test Set Performance.** We compare a Feed-Forward (FF.) Transformer, an RNN based on recent work Jolicoeur-Martineau (2025), and an EBT on data-constrained algorithmic reasoning using a Sudoku dataset Palm et al. (2018).

Architecture	**Accuracy**
FF. Transformer	0.03%
RNN	17.7%
EBTs	**29.7%**

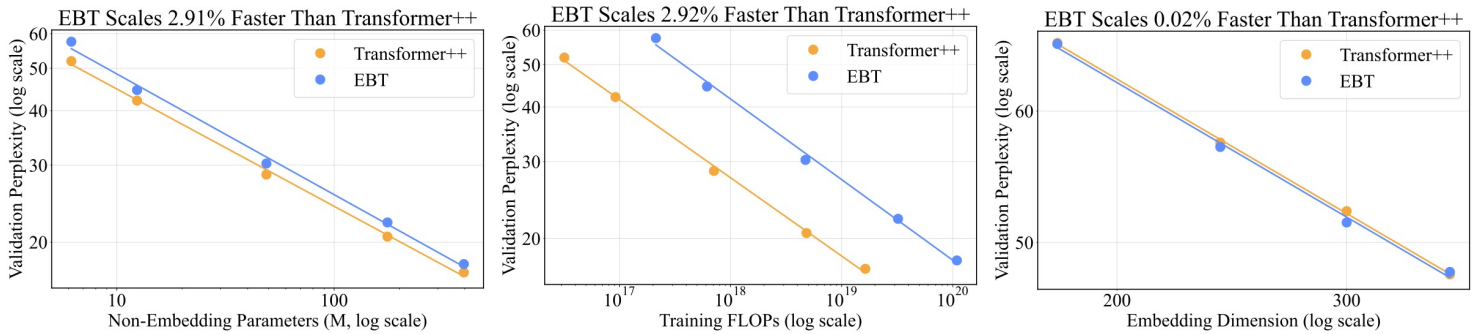
Energy-Based Transformers – Learning Scalability Experiment 1



(a) Scaling for data. (b) Scaling for batch size. (c) Scaling for depth.

Figure 4: **Language Learning Scalability—Data, Batch Size, and Depth.** A comparison between the scaling of the Transformer++ recipe [87] and EBTs across data, batch size, and depth during pretraining. On all of these axes, EBTs out-scale the Transformer++ recipe significantly, indicating improved data efficiency and suggesting potential benefits for generalization. Additionally, the improved depth scaling offers promise for reasoning, where depth is crucial [96]. These results suggest that if these scaling trends persist, EBTs would likely outperform Transformer++ models at foundation model data scale.

Energy-Based Transformers – Learning Scalability Experiment 2



(a) Scaling for number of Parameters. (b) Scaling for number of FLOPs. (c) Scaling for the embed. dimension.

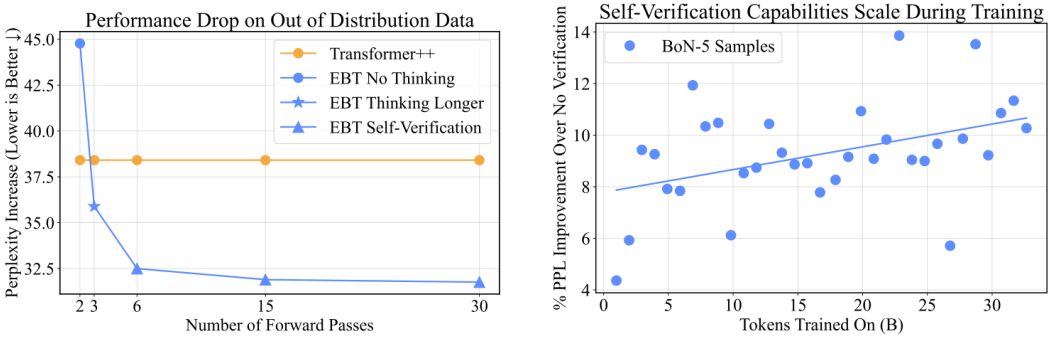
Figure 5: **Language Learning Scalability—Parameters, FLOPs, and Width.** Pretraining scaling comparisons between the Transformer++ recipe [87] and EBTs across model size (parameters), compute (FLOPs), and width (embedding dimension). EBTs slightly out-scale the Transformer++ in FLOP and parameter scaling, becoming the first approach to achieve a higher **scaling rate** without modifying the tokenizer [101] to our knowledge. These results suggest that EBTs offer high promise as a pretraining paradigm in both parameter and FLOP efficiency as scale increases.

Energy-Based Transformers – Think Scalability Experiment 1

Table 2: **System 2 Thinking Ablations.** All energy landscape regularization techniques described in Section 3.3 and their impact on System 2 Thinking performance, measured by percent perplexity improvement. Thinking Longer denotes more optimization steps and Self-Verification denotes optimizing many predictions and choosing the best. Bolded highlights the default System 2 Hyperparameters, leveraging all energy landscape regularization techniques described. This configuration results in the best performance when thinking longer and doing self-verification. Removing regularization, such as Langevin Dynamics, results in less energy landscape exploration, which improves single path performance (thinking longer) at the expense of self-verification performance.

Model	**Thinking Longer** ↑	**Thinking Longer and Self-Verification** ↑
No Random Step Size	-1.47	0.19
No Random Num. Steps	0.00	9.65
No Langevin Dynamics	**17.2**	17.0
No Replay Buffer	14.8	17.8
Full System 2 Configuration	7.19	**18.7**

Energy-Based Transformers – Think Scalability Experiment 2



(a) OOD Thinking Performance Comparison.

(b) Verification Capabilities as Scale Increases.

Figure 6: **EBT Thinking Analysis.** (a) Mean performance degradation of the standard Transformer++ recipe [87] and the Energy-Based Transformer (EBT) on four Out of Distribution (OOD) datasets. While the Transformer++ cannot reduce perplexity at a *per-token level*, EBTs can by performing more forward passes over a single token/sample (Thinking Longer) as well as generating many samples and choosing the minimum energy one (Self-Verifying/BoN in addition to longer thought). (b) The Self-Verification capabilities of EBTs using BoN-5 compared to not doing any self-verification. As data scale increases, the benefit from doing self-verification increases. These results suggest EBTs generalize OOD better than the Transformer++ because of their System 2 Thinking capabilities, and that the thinking capabilities of EBTs scale during training.

Energy-Based Transformers – Generalization 1

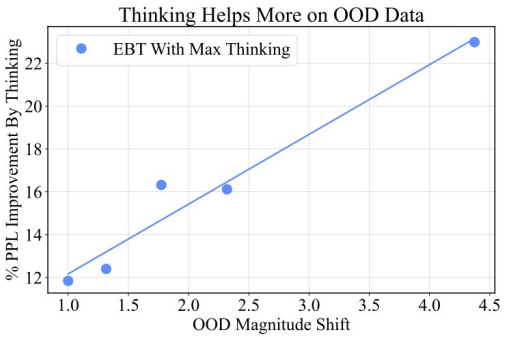


Figure 7: **OOD Thinking Performance.** As the data becomes more OOD, thinking leads to greater performance improvements, with a roughly linear trend. These findings highlight that EBTs thinking is especially critical for robust generalization to OOD data. Performance is measured on 5 different datasets varying in Out-of-Distribution (OOD) magnitude shift, which is measured as the ratio of downstream dataset perplexity to pretraining perplexity. Max Thinking denotes combining thinking longer and self-verification.

Energy-Based Transformers – Generalization 2

Table 3: **Language Model Task Generalization Comparison.** Despite exhibiting a slightly higher pretraining perplexity, EBTs usually achieve lower perplexity on downstream tasks than the Transformer++. This suggests that EBTs generalize better than the Transformer++. Additionally, because EBTs scale better than the Transformer++ during pretraining (Figure 4), these findings suggest that EBTs would outperform Transformer++ at foundation model scale. BB stands for BigBench.

Model	Pretrain	GSM8K ↓	SQuAD ↓	BB Math QA ↓	BB Dyck ↓
Transformer++	**31.36**	49.6	**52.3**	79.8	131.5
EBT	33.43	**43.3**	53.1	**72.6**	**125.3**

Energy-Based Transformers – Generalization 3

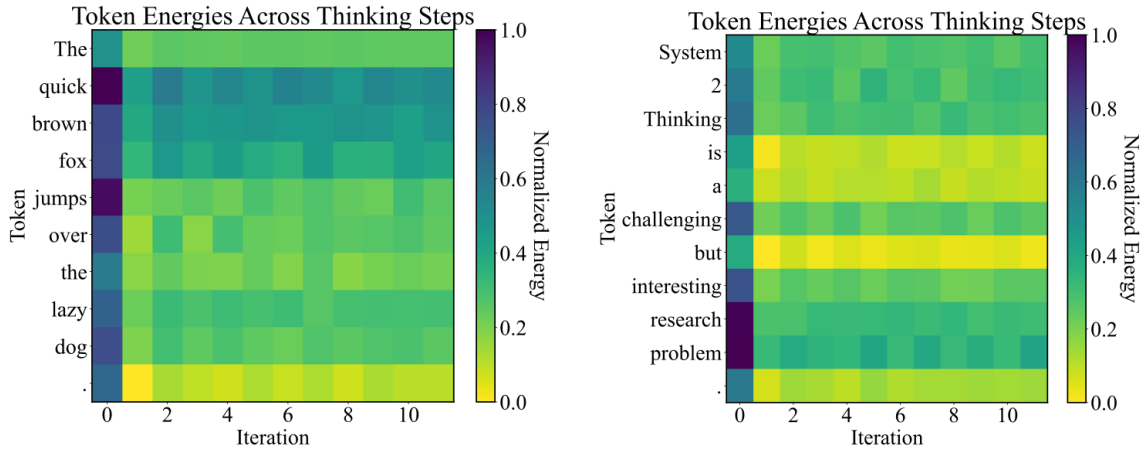


Figure 8: **Learning Uncertainty on Text Results.** EBTs learn to vary uncertainty across text tokens without any explicit supervision. As an example, in both (a) and (b), simple tokens such as ".", "is", "a", "but", or "the" have lower energies across inference-time optimization (thinking) steps, indicating lower uncertainty. On the other hand, harder to predict tokens such as "quick", "brown", "research", and "problem" have higher energies across optimization steps, and more difficulty in achieving energy convergence, meaning the model is more uncertain. Inspired by [25].