Learning Tractable Distributions of Language Model Continuations

Gwen Yidou-Weng[1], **Ian Li**[2], **Anji Liu**[3], **Oliver Broadrick**[1], **Guy Van den Broeck**[1], **Benjie Wang**[1]
[1]University of California, Los Angeles
[2]University of California, San Diego
[3]National University of Singapore

Abstract

Controlled language generation conditions text on sequence-level constraints (for example, syntax, style, or safety). These constraints may depend on future tokens, which makes directly conditioning an autoregressive language model (LM) generally intractable. Prior work uses tractable surrogates such as hidden Markov models (HMMs) to approximate the distribution over continuations and adjust the model's next-token logits at decoding time. However, we find that these surrogates are often weakly context aware, which reduces query quality. We propose *Learning to Look Ahead* (LTLA), a hybrid approach that pairs the same base language model for rich prefix encoding with a fixed tractable surrogate model that computes exact continuation probabilities. Two efficiency pitfalls arise when adding neural context: (i) naïvely rescoring the prefix with every candidate next token requires a sweep over the entire vocabulary at each step, and (ii) predicting fresh surrogate parameters for each prefix, although tractable at a single step, forces recomputation of future probabilities for every new prefix and eliminates reuse. LTLA avoids both by using a single batched HMM update to account for all next-token candidates at once, and by conditioning only the surrogate's latent state prior on the LM's hidden representations while keeping the surrogate decoder fixed, so computations can be reused across prefixes. Empirically, LTLA attains higher conditional likelihood than an unconditional HMM, approximates continuation distributions for vision–language models where a standalone HMM cannot encode visual context, and improves constraint satisfaction at comparable fluency on controlled-generation tasks, with minimal inference overhead.

1 Introduction

Autoregressive models are the dominant way to represent high-dimensional discrete distributions over language and sequence data (Grattafiori et al., 2024; Shin et al., 2021), factorizing the distribution into a sequence of next-token conditional distributions. Many useful queries, however, concern properties of the entire sequence: to generate autoregressively under a sequence-level constraint, we need the probability that the remaining suffix will satisfy that constraint given the current prefix (Boyd et al., 2022; Zhang et al., 2023). Computing this probability requires reasoning over exponentially many possible futures and is therefore intractable for standard autoregressive language models (LM). In practice, estimates come from sampling (Qin et al., 2022; Hu et al., 2023; Lew et al., 2023), which introduces significant computational overhead, or from learned heuristics (Krause et al., 2020; Yang & Klein, 2021; Meng et al., 2022) that must be tailored to each query or constraint.

Tractable probabilistic models (TPMs) are a broad class of generative models that admit efficient routines for exactly computing such conditional queries exactly and efficiently (Choi et al., 2020). As such, a natural approach employed by prior works (Zhang et al., 2023; Yidou-Weng et al., 2025) is to utilize TPMs as *tractable surrogates* to the LM, such that queries on the LM can be approximated by computing them on the surrogate. However, we identify that current TPMs used for language modeling, such as hidden Markov models (HMM), face two significant obstacles in this setting. Firstly, we find empirically that HMMs are comparatively insensitive to the prefix, which yields less accurate distributions over continuations, as illustrated in Fig. 1. Secondly, as models op-
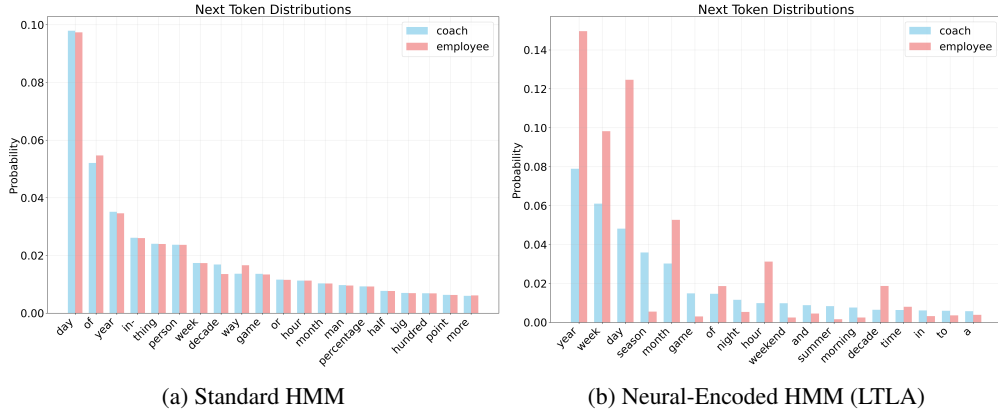
Figure 1: The encoder given by standard HMMs is often insensitive to information contained within the context. In this example, we show an example with the context `they fired the <x> after just one`, where `<x>` can be `coach` or `employee`. The distribution is almost identical for the standard HMM, while the neural HMM shows a significant shift in distribution (in particular, with `season` and `game` being more likely when `<x> = coach`).

erating only over discrete sequences, they cannot effectively encode some kinds of context, notably continuous multimodal features used by modern vision–language models.

To encode the past well and answer future queries quickly, we separate the two jobs. Understanding the past ("lookback") should preserve as much information as possible, whereas forecasting the future ("lookahead") must remain tractable. The same base LM already builds a rich representation of the prefix, which we reuse to encode the prefix into the tractable model (for example, a HMM) that models and computes exact probabilities over continuations. In short, we propose a *hybrid* tractable model where a transformer-based LM is used for lookback and a HMM for lookahead.

Making this hybrid model practical raises two concrete challenges. First, when autoregressively generating the next token, a naïve strategy would run the transformer on the prefix concatenated with every candidate token across the vocabulary to assess how each affects the downstream conditional query; this per-step sweep over the entire vocabulary is expensive *(see "Exhaustive LM Rescoring" in Table 1)*. Second, conditioning the surrogate itself on the prefix by predicting fresh parameters for every context remains tractable at a single step but prevents reuse: as the prefix grows, the surrogate must be rebuilt and its future probabilities recomputed at every step, which increases decode-time cost and memory *(see "Prefix-Parameterized TPM" in Table 1; cf. Sec. 3.1)*. Both issues undercut the efficiency that makes TPMs attractive.

Thus, we propose a solution that encodes rich context with the transformer while maintaining TPM efficiency by encoding the prefix only into the HMM surrogate's latent prior and keeping the rest of the HMM fixed. Specifically, the transformer's hidden states are fed into to a lightweight head that is trained to produce a prior over the surrogate's latent state, and the observed next token x_t is incorporated with a one-step HMM update as a single batched matrix–vector operation, avoiding separate computation for each token in the vocabulary. With fixed transition and emission parameters, future computations are reusable across steps, so conditional queries remain exact and fast even for long sequence lengths *(see "LTLA (Ours)" in Table 1)*. We call this approach *Learning to Look Ahead (LTLA)*: it learns to injects context through the LM representation, while preserving tractability of the lookahead model.

Our contributions can be summarized as follows:

1. We introduce LTLA, a novel approach for learning tractable surrogates to language models. LTLA reuses the base language model to predict the prior of a HMM modeling continuations given the current context, enabling accurate estimates of conditional queries at each time step of generation while only using a small number of matrix-vector multiplications. We further show how LTLA can be used to condition on multimodal context by leveraging vision-language models (VLM).

Table 1: Comparison of lookahead properties across models.

Model	Tractable Lookahead	Context Awareness (incl. multimodal)	No Extra LM Calls per Step?	Reuse Surrogate Precompute per Prefix?	Decoding Overhead
LLM	✗	✓	—	—	—
Standard HMM	✓	✗	✓	✓	**Low**
Exhaustive LM Rescoring	✗	✓	✗	✓	High
Prefix-Parameterized TPM	✓	✓	✓	✗	High
LTLA (Ours)	✓	✓	✓	✓	**Low**

2. We propose different architectural choices for both the neural encoder and tractable decoder, and investigate their effect on modeling performance, decoding overhead, and downstream performance.

3. Empirically, we show that LTLA achieves improved conditional log-likelihood compared with existing tractable models, especially for the next few tokens. We also demonstrate the application of LTLA to controlled-generation tasks for both language and vision–language models, where LTLA improves constraint satisfaction compared to existing approaches while adding only a small inference-time overhead.

2 TRACTABLE MODELING OF SEQUENCES

In this work, we are interested in autoregressive sequence models, and how to effectively answer *queries* about the distribution over sequences that they represent. An autoregressive model decomposes the distribution over a sequence of tokens $x_{1:T}$ as follows:

$$p(x_{1:T}) = \prod_{t=1}^{T} p(x_t|x_{<t}). \tag{1}$$

Queries Aside from generating or analyzing the next token distribution, we are often interested in more complex properties of the distribution. These can be represented generally as *conditional probability queries* (Boyd et al., 2022), which ask for the probability $p(\alpha|x_{1:t})$ of some event α, where $x_{1:t}$ is the prefix (or *context*) generated so far. Examples of events α might include the kth token x_{t+k} in the future taking a particular value, some token a appearing before token b in the sequence, the expected length of the sequence generated, or more complex properties involving grammatical or semantic constraints (Zhang et al., 2024; Yidou-Weng et al., 2025; Ahmed et al., 2025). What these queries have in common is that they require *looking into the future*: that is, aggregating over all possible continuations, weighted by their conditional probability given the context:

$$p(\alpha|x_{1:t}) = \sum_{x_{t+1:T}} p(x_{t+1:T}|x_{1:t})p(\alpha|x_{1:t}, x_{t+1:T}). \tag{2}$$

One of the key downstream applications of conditional probability queries is *controlled generation*: that is, generating from an autoregressive model *conditional on* some event α. In particular, observe that, by Bayes' rule, the distribution of the next token X_t conditional on previous tokens and α is given by:

$$p(X_t|x_{<t}, \alpha) \propto p(X_t|x_{<t}) \cdot p(\alpha|x_{<t}, X_t) \tag{3}$$

As such, if one has access to an oracle for conditional probability queries, then it is possible to sample autoregressively from the conditional distribution by explicitly computing the terms in Equation 3. In practice, however, such an oracle is not available, and one must resort to approximations.

Estimating Conditional Probability Queries Consider the problem of estimating the conditional probability query $p(\alpha|x_{1:t})$ given in Equation 2. The key tradeoff is between (i) the *accuracy* and (ii) the *computation cost* of the estimation. Explicitly enumerating all such continuations would result in an exact answer, but is clearly infeasible as the number of such continuations grows exponentially with sequence length. As such, for most models, one typically needs to approximate, for example by (i) using sampling-based techniques targeting the conditional continuation distribution (Qin et al., 2022; Zhao et al., 2024; Loula et al., 2025); or (ii) directly approximating the conditional probability

query using a neural classifier or generative model specialized to the constraint α (Krause et al., 2021; Yang & Klein, 2021; Meng et al., 2022).

In this work, we consider an alternative, computationally efficient approach based upon *tractable modeling* of continuations. This stems from the observation that, for certain distributions p and queries α, the computation of $p(\alpha|x_{1:t})$ can be done both (i) exactly and (ii) efficiently, breaking the tradeoff. In particular, tractable probabilistic models (TPMs) (Choi et al., 2020) are classes of probabilistic generative models that are known to enable computationally efficient *analytical* computation of many classes of queries, such as marginal probabilities.

Example 1 *Hidden Markov models (HMMs) are tractable sequence models, that represent a joint distribution over a sequence of T variables $X_1, X_2, \ldots, X_T$ each taking values in a size-V vocabulary $\mathcal{V} = \{0, \ldots, V - 1\}$ with latent variables $Z_1, Z_2, \ldots, Z_T$ each taking value in a discrete set of hidden states $\mathcal{H} = \{0, \ldots, H - 1\}$ of size H. The parameters of an HMM are given by an emission matrix $q(x_t|z_t) \in \mathbb{R}_{\geq 0}^{H \times V}$ and a transition matrix $q(z_t|z_{t-1}) \in \mathbb{R}_{\geq 0}^{H \times H}$. Then, the distribution of an HMM is defined by*

$$q(x_1, \ldots, x_T) = \sum_{z_1, \ldots, z_T} q(z_1)q(x_1|z_1) \prod_{t=2}^{T} q(z_t|z_{t-1})q(x_t|z_t).$$

HMMs enable efficient computation of various queries via (variants of) the forward *and* backward *algorithms. For example, if α is the event that the last token $X_T = \text{world}$, we can compute the conditional query $q(\alpha|x_{1:t})$ using the fact that (by conditional independence):*

$$q(\alpha|x_{1:t}) = \sum_{z_t} q(z_t|x_{1:t})q(\alpha|z_t). \tag{4}$$

where $q(z_t|x_{1:t})$ and $q(\alpha|z_t)$ can be computed using the forward and backward algorithms respectively, which each take linear time in the sequence length. For instance, $q(\alpha|z_t)$ can be computed using the following recurrence relation backward in time:

$$q(\alpha|z_{t-1}) = \sum_{z_t} q(z_t|z_{t-1})q(\alpha|z_t) \tag{5}$$

with base case $q(\alpha|z_T) = q(x_T = \text{world}|z_T)$. Prior work has also shown that HMMs support tractable querying of many other conditions α, including complex logical constraints[1] (Zhang et al., 2024; 2025b) or factorized classifiers for semantic constraints (Yidou-Weng et al., 2025).

The distribution p we are interested in will typically not be tractable in this way. However, we can aim to approximate the (prior) distribution $p(x_{t+1:T}|x_{1:t})$ using a simpler tractable surrogate model $q(x_{t+1:T}|x_{1:t})$ for which conditional queries are tractable, and estimating $p(\alpha|x_{1:t}) \approx q(\alpha|x_{1:t})$ using the tractable approximation q. This approach has been employed in prior works (Zhang et al., 2023) using HMMs as the TPM of choice, which has been shown to lead to state-of-the-art performance on controlled generation benchmarks.

The Promise and Challenges of Tractable Surrogate Modeling Besides state-of-the-art empirical performance on downstream applications, the tractable modeling approach offers a number of other benefits related to computational efficiency. Firstly, it amortizes the cost across different constraints α. That is, since q is trained to match the *prior*, and the computation of the query $q(\alpha|x_{1:t})$ is conducted using a symbolic algorithm, one does not need to commit in advance to any particular condition α. This is in contrast to approaches that train models to specifically target the *posterior* $p(x_{t+1:n}|x_{1:t}, \alpha)$. Secondly, tractable models can amortize the cost across different *contexts* $x_{1:t}$ by exploiting conditional independence. For example, in the HMM query computation in Equation 4, the backward quantity $q(\alpha|z_t)$ can be precomputed and cached independently of the context $x_{1:t}$.

This computational efficiency, however, also comes at a cost. The quality of the query estimate provided by the tractable model depends significantly on the quality of the approximation of q to p, which in turn has been shown to affect downstream performance (Zhang et al., 2023; Yidou-Weng et al., 2025). Unfortunately, the class of tractable models (e.g., HMMs) is fundamentally

---

[1]more specifically, deterministic finite automata (DFA) or unambiguous context-free grammars (uCFG).

less expressive than e.g., neural autoregressive models (Choi et al., 2019; Broadrick et al., 2025). This is reflected in our observation in Figure 1, where a HMM trained to approximate a GPT2-large language model is unable to effectively encode dependence on context $x_{1:t}$ in its distribution $q(x_{t+1:n}|x_{1:t})$. As such, a key challenge is to improve the expressivity and learning performance of the tractable model approximation, while maintaining the computational efficiency of the existing HMM-based approximation. In the next section, we will present our approach, Learning To Look Ahead (LTLA), which utilizes an amortized inference approach with a neural encoder to obtain the tractable approximation q.

3   Learning to Look Ahead

To formalize our problem from first principles, suppose that we have a set of contexts $\{x_{1:t}^{(j)}\}_{j=1}^{N}$, and let $\mathcal{Q}$ be a class of tractable distributions over continuations (e.g. HMMs). Then our goal is to infer, for each given context $x_{1:t}^{(j)}$, a distribution $q^{(j)} \in \mathcal{Q}$ over continuations $x_{t+1:n}$ such that $q^{(j)} \approx p(\cdot|x_{1:t}^{(j)})$, for example, by maximizing conditional log-likelihood

$$\text{LL} = \mathbb{E}_{x_{t+1:T} \sim p(\cdot|x_{1:t}^{(j)})} \left[\log q^{(j)}(x_{t+1:T}) \right]. \tag{6}$$

Unfortunately, in practice we cannot afford to optimize $q^{(j)}$ separately for every context $x_{1:t}^{(j)}$; instead we must take an amortized inference approach, in which we *learn* to predict $q(x_{t+1:T})$ given context $x_{1:t}$. In particular, an HMM trained on the joint distribution over contexts and continuations can be viewed as performing amortized inference by (i) applying a probabilistic *encoder* $q(z_t|x_{1:t}) \in \mathbb{R}_{\geq 0}^{H}$ to predict the latent state distribution and (ii) parameterizing the distribution over continuations via this latent state and the probabilistic *decoder* $q(x_{t+1:T}|z_t)$. Our key insight is that, for answering queries about the continuation, we only need to be able to (i) evaluate the encoder and (ii) answer queries about the decoder's distribution. As such, only the decoder needs to be tractable, and we can increase the expressivity of the continuation model by allowing the encoder to be an arbitrary neural network q_{enc}, giving rise to the following hybrid model:

$$q_{\text{hybrid}}(x_{t+1:T}|x_{1:t}) = \sum_{z_t} q_{\text{enc}}(z_t|x_{1:t}) q(x_{t+1:T}|z_t) \tag{7}$$

We can then jointly train the encoder and decoder to maximize the expected log-likelihood over a dataset of contexts and continuations $\{x_{1:t}^{(j)}, x_{t+1:T}^{(j)}\}_{j=1}^{N}$. By utilizing a more expressive encoder, our hypothesis is that the log-likelihood of the hybrid model will improve upon the pure HMM, in turn leading to better downstream performance.

We call our method Learning To Look Ahead (LTLA) as (i) we train a neural encoder to learn a language model's distribution over continuations, and (ii) we execute a symbolic algorithm to look ahead over exponentially many continuations with a probability distribution given by the output of the learned encoder. Although we only maintain the model's tractability over the continuation tokens $x_{t+1:T}$, this is sufficient for downstream applications where tokens are generated autoregressively. In the rest of this section, we will first describe the algorithms for and complexity of conditional probability queries of interest, before discussing architectural choices for the neural encoder.

3.1   Inference using Hybrid HMMs

After conditioning on a context $x_{\leq t}$, we obtain a tractable model of the distribution of continuations $q(x_{>t}|x_{\leq t})$, rendering many natural queries exactly and efficiently computable. For example, after observing a sequence $x_{\leq t}$, we might ask the following questions: (i) what is the probability that the sequence ends within the next k tokens; (ii) what is the probability that the sequence contains a particular keyword; (iii) what is the probability that the generated sequence will be labeled "toxic"/"biased"/etc by a given tractable classifier? In general, queries $q(\alpha|x_{\leq t})$, where α is some subset of the possible continuations $x_{>t}$, can be written $q(\alpha|x_{\leq t}) = \sum_{x_{>t}} q(x_{>t}|x_{\leq t})\mathbb{1}\{x_{>t} \in \alpha\}$ where $\mathbb{1}\{x_{>t} \in \alpha\}$ is an indicator function of the membership of $x_{>t}$ in α. So, to efficiently compute $q(\alpha|x_{\leq t})$, it suffices to be able to *multiply* $q(x_{>t}|x_{\leq t})$ by the indicator function for α in such a way that the summation remains tractable; general sufficient conditions are known for such tractable

multiplications (Darwiche & Marquis, 2002; Zhang et al., 2025b). More generally, tractable models also support other types of queries, including for example information-theoretic quantities like entropies and divergences (Vergari et al., 2021; Wang et al., 2024).

Controlled Generation As a concrete application of the tractability of a neural-encoded HMM, we consider the task of generated text conditioned on a (logical or semantic) constraint α. A natural decoding procedure is to sample autoregressively from the conditional distribution factored by Bayes' rule as Equation 3, where the first term $p(x_t|x_{<t})$ is computed with the autoregressive model, and the second term $p(\alpha|x_{\leq t})$ is computed with the tractable neural-encoded HMM serving as a proxy for the autoregressive model. As an example, we describe the case where α is the event that the text is accepted by a given deterministic finite automaton (DFA); see Section B for the definition of a DFA. We follow closely the algorithm Ctrl-G designed for unconditional HMMs (Zhang et al., 2024). Given a DFA M, let S_t be the random variable representing the state of M after reading $x_{\leq t}$ sampled from $p(x_{\leq t}|\alpha)$. Then,

$$p(\alpha|x_{\leq t}) = \sum_{z_t} p(z_t|x_{\leq t})p(\alpha|z_t, x_{\leq t}) = \sum_{z_t} p(z_t|x_{\leq t})p(\alpha|z_t, s_t) \qquad (8)$$

where the first equality follows from the law of total probability, and the second from the Markov properties of HMMs and DFAs and the fact that s_t is fully determined by $x_{\leq t}$. The term $p(z_t|x_{\leq t})$ may be estimated by the neural encoder. In practice, to avoid evaluating the neural encoder V times, once for each candidate next token x_t, we instead evaluate the encoder once to compute $p(z_{t-1}|x_{<t})$ and then perform a single HMM forward step to obtain $p(z_t|x_{\leq t})$ for each x_t.

Then, observe that the term $p(\alpha|z_t, s_t)$ is entirely independent of the context $x_{\leq t}$, and so all TVH such probabilities $p(\alpha|z_t, s_t)$ can be precomputed and stored in a lookup table (before decoding); there is an efficient backward-style algorithm for this computation (Zhang et al., 2024). Finally $p(\alpha|x_{\leq t})$ is computed by the aggregation in Equation (8). Similar approaches work for other constraints. TRACE is a similar algorithm derived by Yidou-Weng et al. (2025) for the case where the constraint is an attribute s predicted by a fully factorized probabilistic classifier $p(s|x_{1:n})$.

Complexity We emphasize an advantage to changing only the *encoder* of the HMM. Specifically, we contrast neural-encoded HMMs with an alternative wherein a neural network predicts, given a context $x_{\leq t}$, fresh parameters of a full HMM. With such full conditioning, the backward computation of $p(\alpha|z_t, s_t)$ can no longer be precomputed but needs to be carried out at each step of decoding on the freshly predicted HMM, resulting in a decoding time that scales quadratically in sequence length. Moreover, fresh HMM parameters are needed for each context, resulting in an additional blowup in a memory with batch size. The time and space complexities for training and inference of standard HMMs, neural-encoded HMMs, and full conditional HMMs are summarized in Section C. We use the parameter τ for the complexity of a single HMM forward (or backward) step (always assuming $\tau = \Omega(hV)$); for standard (dense) transition and emission matrices, $\tau = O(h^2 + hV)$.

3.2 ARCHITECTURE

We now specify the architectural designs for our hybrid model, which includes two components: the HMM decoder and the neural network encoder.

3.2.1 STRUCTURED SPARSITY

The main parameter controlling the expressivity of an HMM is its *hidden size*, the number of states of each latent variable. Specifically, for any index t an HMM forms a Markov chain $X_{<t} \to Z_t \to X_{\geq t}$, and so any dependence of the continuation on the context must 'flow through' the latent state. Indeed, we can show that the *logarithm* of the hidden size is an upper bound on the mutual information between the context and continuation. Denoting the mutual information between A and B by $I(A; B)$, the entropy of A by $H(A)$, and the support of A by supp(A), we have the following.

Proposition 1 *For any Markov chain $X_{<t} \to Z_t \to X_{\geq t}$, we have*

$$I(X_{<t}; X_{\geq t}) \leq H(Z_t) \leq \log |supp\,(Z_t)|.$$
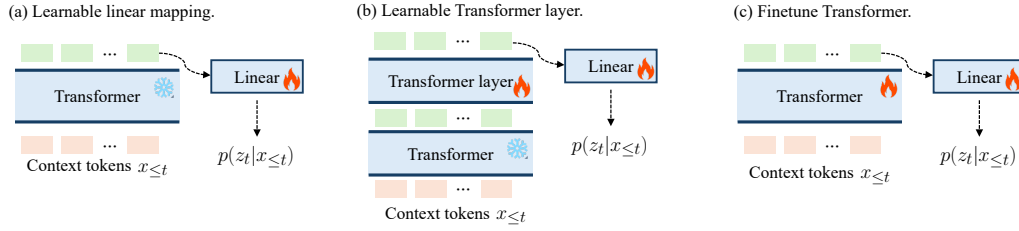
Figure 2: Neural architectures for neural-encoded HMMs: (a) frozen Transformer with linear mapping, (b) frozen Transformer with additional learnable layer, (c) fully finetuned Transformer.

This bound (proven in Section A) holds regardless of the encoder $p(z_t|x_{<t})$ or decoder $p(x_{\geq t}|z_t)$, i.e., it holds for standard HMMs as well as neural-encoded HMMs. Although Proposition 2 provides a strong motivation for increasing the hidden size H to improve the model's capacity, the number of parameters in an HMM with dense transition and emission matrices grows quadratically in hidden size. Therefore, we also consider using Monarch matrices (Dao et al., 2022), structured matrices that require polynomially fewer parameters in the hidden size yet can still express complex, high-rank matrices, which have been used for scaling tractable models (Zhang et al., 2025a). Monarch matrices for the transition and emission matrices in an HMM use a combined $O(h^{3/2} + h^{1/2}V)$ set of parameters, giving a single-step complexity of $\tau = O(h^{3/2} + h^{1/2}V)$.

3.2.2 NEURAL NETWORK ARCHITECTURES

Another key factor that determines the performance of neural-encoded HMMs is the choice of the neural network. Aside from achieving better conditional log-likelihoods, we want the neural encoder to be as lightweight as possible to minimize the computational overhead in downstream applications. One way to achieve this is by reusing the same neural network backbone of existing autoregressive sequence models (e.g., LLMs) and adding minimal modules on top of them.

Figure 2 illustrates the three neural architectures we adopted. The first one (in sub-figure (a)) keeps the (causal) Transformer backbone frozen and only adds a learnable linear mapping on top of its last hidden embedding to predict the latent variable distribution $p(z_t|x_{\leq t})$. In downstream applications, the Transformer embeddings are already computed, so we only add a negligible overhead of running the linear layer. The second variant (in sub-figure (b)) extends the frozen Transformer with an additional learnable Transformer layer before the output linear mapping, which strikes a better balance between the expressiveness of the encoder and the computational overhead. The third design (in sub-figure (c)) fully finetunes the Transformer backbone together with the linear prediction layer. While this variant has the largest number of trainable parameters, it also provides the most flexibility to tailor the representation for the HMM.

4 EXPERIMENTS

Datasets In our experiments, we utilize datasets consisting of data sampled from LLMs. Our main evaluation is on datasets from **GPT2-large**, **GPT2-large finetuned for CommonGen**, and **Qwen2.5-VL-2B**. For each of these datasets, we unconditionally sample sequences of length $T = 32$, and uniformly at random choose a split position in the sequence to construct context and continuation pairs of variable length.

Methods and Baselines In the distillation performance experiments, we primarily compare our neural HMMs to corresponding HMMs in terms of their performance in modeling continuations. For reference, we also compare to HMMs trained using the method of Lee & Berg-Kirkpatrick (2025). In the controlled generation benchmarks, we compare LTLA with the standard HMMs used in Ctrl-G, in order to show the benefits of the neural encoder, and with prompting and sampling baselines for the novel VLM detoxification setting.

Research Questions The key research questions we seek to investigate are as follows. **(RQ1)** Do neural HMMs achieve better modeling performance compared to standard HMMs, and if so, in what way? **(RQ2)** How does the choice of neural encoder architecture affect these results? **(RQ3)** Do any
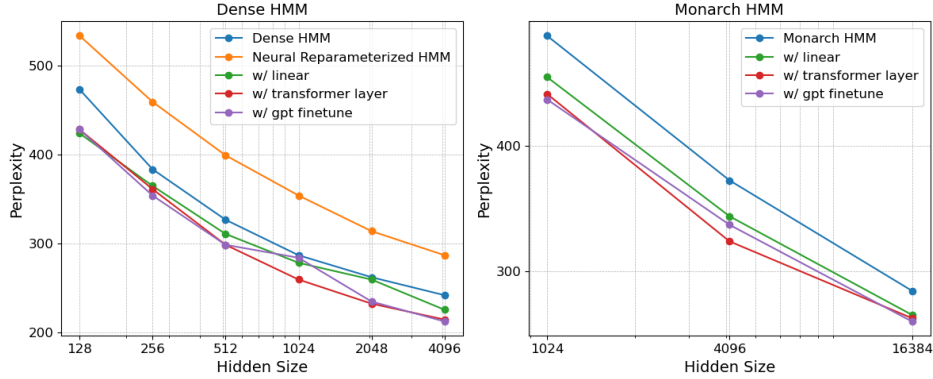
Figure 3: Perplexity of neural-encoded HMMs and baseline HMM for varying hidden sizes with dense transition and emission matrices on the left, Monarch matrices on the right.
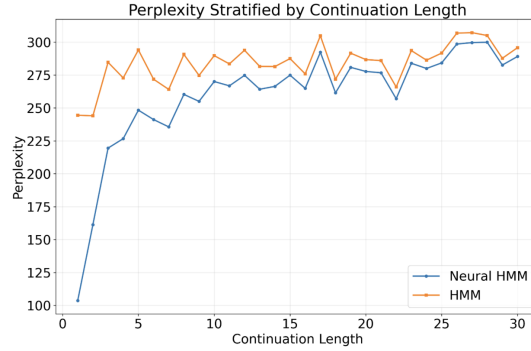

Figure 4: Perplexity of (Monarch) HMMs vs neural HMMs on the GPT2-large dataset for different continuation lengths.

gains in terms of modeling performance translate to improvements in downstream benchmarks via LTLA? **(RQ4)** Can neural HMMs effectively condition on multimodal context data?

4.1 DISTILLATION PERFORMANCE & ANALYSIS

Figure 3 reports the scaling behavior of neural-encoded HMMs compared to standard HMM baselines. The a-axis refers to the hidden size of the HMM. Across both dense and Monarch parameterizations (as introduced in Section 3.2.1), augmenting HMMs with neural components consistently reduces perplexity relative to the baseline. Across the three neural architectures, adding a trainable Transformer layer (i.e., Figure 2(b)) or finetuning GPT (i.e., Figure 2(c)) achieves stronger performance compared to the learnable linear layer. These results demonstrate that neural HMMs scale favorably with hidden dimension, leading to better conditional log-likelihood per token/perplexity. In Figure 4, we further see that the neural encoder achieves better perplexity particularly for shorter continuation lengths, which is to be expected as the first few tokens have the strongest dependence on the context and are where the neural encoder has the most direct effect.

In terms of the architectural choices, both adding a transformer layer and finetuning the full GPT model lead to greater improvements in perplexity compared to using a linear head, at the cost of additional compute. In terms of the HMM architecture, the results show that, perhaps surpisingly, the larger hidden sizes of Monarch HMMs did not perform significantly better when normalized for compute. For instance, the Monarch HMM with size 16384 uses theoretical compute comparable to the Dense HMM with hidden size 1024, but does not show significantly better perplexity.

For reference, we also compared to the method of Lee & Berg-Kirkpatrick (2025), who conduct an empirical analysis of techniques designed to improve the learning dynamics of HMM language models, and suggest utilizing a neural reparameterization (Chiu & Rush, 2020) combined with latent

Table 2: Generation quality of HMM variants on the CommonGen benchmark using Ctrl-G.

Model	BLEU-4	Cider	RougeL	Avg. Perplexity($\downarrow$)	Max Perplexity ($\downarrow$)
Max Sequence Length 20					
Standard HMM	0.301	1.552	**0.448**	39.59	1569.51
HMM with Linear NN	**0.303**	**1.566**	**0.448**	41.99	1458.38
HMM with Transformer Block	0.297	1.536	0.446	**36.24**	**616.73**
Max Sequence Length 32					
Standard HMM	0.303	1.566	**0.448**	36.00	1569.51
HMM with Linear NN	**0.311**	1.566	**0.448**	34.16	**671.88**
HMM with Transformer Block	0.305	**1.573**	**0.448**	**33.98**	1065.47

variable distillation (Liu et al., 2023). Contrary to their findings, we found that this worsens performance over standard training in our distillation setting. In particular, we found that the performance of the HMM is quite sensitive to the hyperparameters of the AdamW optimizer, but by choosing appropriate hyperparameters it outperforms the neural parameterization and LVD.

4.2 CONTROLLED GENERATION

LTLA delivers (i) improved distillation quality and (ii) compatibility with autoregressive models beyond the reach of a standalone TPM. The stronger predictive fit of our neural-encoded HMMs translates into better performance on standard LLM controlled-generation benchmarks. In addition, LTLA's flexible design allows us to extend this tractable lookahead approach to VLMs. For VLM image captioning, LTLA uses its tractable lookahead to compute the probability of a future semantic property (i.e. toxicity) that is then used to reweight the VLM's original next-token distribution. In our experiments, this approach outperforms sampling-based and heuristic controllers with negligible inference overhead. We demonstrate versatility across two comprehensive constraint families—*hard logical constraints* and *soft semantic attributes*—using the established Ctrl-G (Zhang et al., 2024) and TRACE (Yidou-Weng et al., 2025) frameworks. By conditioning the TPM on transformer prefixes and learning to look ahead better, LTLA improves constraint satisfaction while maintaining the base model's fluency and diversity.

Logical Constraints We enforce hard constraints (keywords, propositions) by combining the tractable lookahead model with deterministic finite automata (DFAs) in the Ctrl-G framework (Zhang et al., 2024). We distill both standard HMMs and LTLA variants from a GPT-2 model fine-tuned on CommonGen. For each keyword set, we construct a DFA that accepts only sequences satisfying the constraints, then run Ctrl-G beam search (128 beams) and select the top hypothesis by base-LM log-likelihood. We evaluate against references using BLEU-4, CIDEr, and ROUGE-L, and report average and maximum perplexity for fluency.

As shown in Table 2, LTLA leads to significant improvements in generation quality under hard constraints. Neural HMMs not only demonstrate consistent syntactical gains over the standard HMM across metrics including BLEU-4, Cider, and RougeL, but also achieve significant fluency gains by dramatically reducing perplexity, slashing the maximum perplexity by more than half. This suggests that the neural encoders are highly effective at providing more context-aware guidance for the Ctrl-G framework to satisfy constraints without forcing the model to generate unnatural sequences.

Semantic constraints in VLMs. We introduce a plug-and-play tractable lookahead for VLMs, compatible with autoregressive decoders where a standalone TPM is ill-suited. LTLA enforces soft constraints such as style (emotion, sentiment) and topicality with only the addition of a log-linear classifier, making it extremely lightweight and adaptable for new constraints. We adopt TRACE (Yidou-Weng et al., 2025) to detoxify image captioning on the Hateful Memes dataset (Kiela et al., 2020).

Concretely, we distill LTLA from Qwen2-VL-Instruct-2B and pair it with a log-linear toxicity model trained on RealToxicityPrompts (Gehman et al., 2020), scored with the Perspective API (Lees et al., 2022). The predicted future toxicity reweights the next-token distribution during decoding, providing probabilistic guidance without architectural changes.

Table 3: Toxicity vs. fluency across methods. Lower is better.

Method	Avg. toxicity ($\downarrow$)	Avg. max toxicity ($\downarrow$)	Perplexity ($\downarrow$)
Vanilla	0.087	0.249	**2.359**
Neural Monarch HMM	**0.064**	**0.188**	3.564
Prompting	0.078	0.224	2.463
Sampling	0.097	0.309	43.142

Baselines. We compare against (i) default Qwen generation, (ii) prompt engineering with "Avoid any toxic, offensive, or hateful language.", and (iii) Monte Carlo sampling that reweights tokens using the mean sampled toxicity of N continuations per step.

Results. As shown in Table 3, LTLA achieves the strongest detoxification while preserving fluency. Under a comparable compute budget, the Monte Carlo (MC) baseline was capped at $k=1$ reweighting pass per step, which still ran at approximately $10\times$ the LTLA decoding time. Both methods reweight continuations to approximate the same future-toxicity query, MC by sampling and LTLA via a tractable predictive surrogate, but at equal budget the MC estimator is high variance and costly, yielding higher toxicity and lower fluency..

5 RELATED WORK AND CONCLUSION

We study the problem of computing conditional queries on autoregressive sequence models (Boyd et al., 2022), which has many existing approaches, especially for constrained generation (Qin et al., 2022; Hu et al., 2023; Lew et al., 2023; Krause et al., 2020; Yang & Klein, 2021; Meng et al., 2022). We follow a recent line of work, proposing a method based around tractable modeling (Zhang et al., 2023; 2024; Yidou-Weng et al., 2025), adopting hidden Markov models as our primary structure (Rabiner & Juang, 2003; Chiu & Rush, 2020; Lee & Berg-Kirkpatrick, 2025). Our amortized inference technique of neurally conditioning on context is similar to hybrid neural and tractable models in other settings (Shao et al., 2020; Dos Martires, 2024). Our work complements existing research in tractable modeling, that aims to scale tractable models in practice (Loconte et al., 2024; Liu et al., 2024; Maene et al., 2025; Wang & Van den Broeck, 2025) while characterizing theoretical limits (Zhang et al., 2021; Bläser, 2023; Harviainen et al., 2023; Broadrick et al., 2024).

In conclusion, we proposed Learning to Look Ahead (LTLA), a novel approach to modeling and probabilistically reasoning about language model continuations using tractable models. LTLA uses a neural encoder to predict the latent state of a tractable hidden Markov model, enabling more accurate dependence on the context while maintaining the tractability and computational efficiency of the approach. Empirically, we show that LTLA significantly boosts the perplexity of tractable models over the prior state-of-the-art, particularly for shorter continuation sequences, and improved performance on controllable generation benchmarks.

REFERENCES

Kareem Ahmed, Catarina G Belem, Padhraic Smyth, and Sameer Singh. Semantic probabilistic control of language models. *arXiv preprint arXiv:2505.01954*, 2025.

Markus Bläser. Not all strongly rayleigh distributions have small probabilistic generating circuits. In *International Conference on Machine Learning*, pp. 2592–2602. PMLR, 2023.

Alex Boyd, Samuel Showalter, Stephan Mandt, and Padhraic Smyth. Predictive querying for autoregressive neural sequence models. *Advances in Neural Information Processing Systems*, 35: 23751–23764, 2022.

Oliver Broadrick, Honghua Zhang, and Guy Van den Broeck. Polynomial semantics of tractable probabilistic circuits. In *Proceedings of the 40th Conference on Uncertainty in Artificial Intelligence (UAI)*, 2024. URL http://starai.cs.ucla.edu/papers/BroadrickUAI24.pdf.

Oliver Broadrick, Sanyam Agarwal, Guy Van den Broeck, and Markus Bläser. The limits of tractable marginalization. In *Proceedings of the 42th International Conference on Ma-*

chine Learning (ICML), jul 2025. URL `https://starai.cs.ucla.edu/papers/BroadrickICML25.pdf`.

Justin Chiu and Alexander M Rush. Scaling hidden markov language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1341–1349, 2020.

Arthur Choi, Ruocheng Wang, and Adnan Darwiche. On the relative expressiveness of bayesian and neural networks. *International Journal of Approximate Reasoning*, 113:303–323, 2019.

YooJung Choi, Antonio Vergari, and Guy Van den Broeck. Probabilistic circuits: A unifying framework for tractable probabilistic models. oct 2020. URL `http://starai.cs.ucla.edu/papers/ProbCirc20.pdf`.

Tri Dao, Beidi Chen, Nimit S Sohoni, Arjun Desai, Michael Poli, Jessica Grogan, Alexander Liu, Aniruddh Rao, Atri Rudra, and Christopher Ré. Monarch: Expressive structured matrices for efficient and accurate training. In *International Conference on Machine Learning*, pp. 4690–4721. PMLR, 2022.

Adnan Darwiche and Pierre Marquis. A knowledge compilation map. *Journal of Artificial Intelligence Research*, 17:229–264, 2002.

Pedro Zuidberg Dos Martires. Probabilistic neural circuits. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 17280–17289, 2024.

Samuel Gehman, Suchin Gururangan, Maarten Sap, Yejin Choi, and Noah A. Smith. Realtoxicityprompts: Evaluating neural toxic degeneration in language models, 2020.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Juha Harviainen, Vaidyanathan Peruvemba Ramaswamy, and Mikko Koivisto. On inference and learning with probabilistic generating circuits. In *Uncertainty in artificial intelligence*, pp. 829–838. PMLR, 2023.

Edward J Hu, Moksh Jain, Eric Elmoznino, Younesse Kaddar, Guillaume Lajoie, Yoshua Bengio, and Nikolay Malkin. Amortizing intractable inference in large language models. *arXiv preprint arXiv:2310.04363*, 2023.

Douwe Kiela, Hamed Firooz, Aravind Mohan, Vedanuj Goswami, Amanpreet Singh, Pratik Ringshia, and Davide Testuggine. The hateful memes challenge: Detecting hate speech in multimodal memes. *Advances in neural information processing systems*, 33:2611–2624, 2020.

Ben Krause, Akhilesh Deepak Gotmare, Bryan McCann, Nitish Shirish Keskar, Shafiq Joty, Richard Socher, and Nazneen Fatema Rajani. Gedi: Generative discriminator guided sequence generation. *arXiv preprint arXiv:2009.06367*, 2020.

Ben Krause, Akhilesh Deepak Gotmare, Bryan McCann, Nitish Shirish Keskar, Shafiq Joty, Richard Socher, and Nazneen Fatema Rajani. Gedi: Generative discriminator guided sequence generation. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pp. 4929–4952, 2021.

Ivan Lee and Taylor Berg-Kirkpatrick. Optimizing hidden markov language models: An empirical study of reparameterization and initialization techniques. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 7712–7723, 2025.

Alyssa Lees, Vinh Q. Tran, Yi Tay, Jeffrey Sorensen, Jai Gupta, Donald Metzler, and Lucy Vasserman. A new generation of perspective api: Efficient multilingual character-level transformers, 2022.

Alexander K Lew, Tan Zhi-Xuan, Gabriel Grand, and Vikash K Mansinghka. Sequential monte carlo steering of large language models using probabilistic programs. *arXiv preprint arXiv:2306.03081*, 2023.

Anji Liu, Honghua Zhang, and Guy Van den Broeck. Scaling up probabilistic circuits by latent variable distillation. In *Proceedings of the International Conference on Learning Representations (ICLR)*, may 2023. URL http://starai.cs.ucla.edu/papers/LiuICLR23.pdf.

Anji Liu, Kareem Ahmed, and Guy Van den Broeck. Scaling tractable probabilistic circuits: A systems perspective. In *Proceedings of the 41th International Conference on Machine Learning (ICML)*, jul 2024. URL http://starai.cs.ucla.edu/papers/LiuICML24.pdf.

Lorenzo Loconte, Aleksanteri Sladek, Stefan Mengel, Martin Trapp, Arno Solin, Nicolas Gillis, and Antonio Vergari. Subtractive mixture models via squaring: Representation and learning. In *International Conference on Learning Representations (ICLR)*, 2024.

João Loula, Benjamin LeBrun, Li Du, Ben Lipkin, Clemente Pasti, Gabriel Grand, Tianyu Liu, Yahya Emara, Marjorie Freedman, Jason Eisner, et al. Syntactic and semantic control of large language models via sequential monte carlo. In *The Thirteenth International Conference on Learning Representations*, 2025.

Jaron Maene, Vincent Derkinderen, and Pedro Zuidberg Dos Martires. Klay: Accelerating arithmetic circuits for neurosymbolic ai. In *The Thirteenth International Conference on Learning Representations*, 2025.

Tao Meng, Sidi Lu, Nanyun Peng, and Kai-Wei Chang. Controllable text generation with neurally-decomposed oracle. *Advances in Neural Information Processing Systems*, 35:28125–28139, 2022.

Lianhui Qin, Sean Welleck, Daniel Khashabi, and Yejin Choi. Cold decoding: Energy-based constrained text generation with langevin dynamics. *Advances in Neural Information Processing Systems*, 35:9538–9551, 2022.

Lawrence Rabiner and Biinghwang Juang. An introduction to hidden markov models. *ieee assp magazine*, 3(1):4–16, 2003.

Xiaoting Shao, Alejandro Molina, Antonio Vergari, Karl Stelzner, Robert Peharz, Thomas Liebig, and Kristian Kersting. Conditional sum-product networks: Imposing structure on deep probabilistic architectures. In *International Conference on Probabilistic Graphical Models*, pp. 401–412. PMLR, 2020.

Jung-Eun Shin, Adam J Riesselman, Aaron W Kollasch, Conor McMahon, Elana Simon, Chris Sander, Aashish Manglik, Andrew C Kruse, and Debora S Marks. Protein design and variant prediction using autoregressive generative models. *Nature communications*, 12(1):2403, 2021.

Antonio Vergari, YooJung Choi, Anji Liu, Stefano Teso, and Guy Van den Broeck. A compositional atlas of tractable circuit operations for probabilistic inference. *Advances in Neural Information Processing Systems*, 34:13189–13201, 2021.

Benjie Wang and Guy Van den Broeck. On the relationship between monotone and squared probabilistic circuits. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 21026–21034, 2025.

Benjie Wang, Denis Mauá, Guy Van den Broeck, and YooJung Choi. A compositional atlas for algebraic circuits. *Advances in Neural Information Processing Systems*, 37:141318–141355, 2024.

Kevin Yang and Dan Klein. Fudge: Controlled text generation with future discriminators. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 3511–3535, 2021.

Gwen Yidou-Weng, Benjie Wang, and Guy Van den Broeck. Trace back from the future: A probabilistic reasoning approach to controllable language generation. In *Proceedings of the 42th International Conference on Machine Learning (ICML)*, jul 2025. URL https://starai.cs.ucla.edu/papers/YidouWengICML25.pdf.

Honghua Zhang, Brendan Juba, and Guy Van den Broeck. Probabilistic generating circuits. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, jul 2021. URL http://starai.cs.ucla.edu/papers/ZhangICML21.pdf.

Honghua Zhang, Meihua Dang, Nanyun Peng, and Guy Van den Broeck. Tractable control for autoregressive language generation. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, jul 2023. URL `http://starai.cs.ucla.edu/papers/ZhangICML23.pdf`.

Honghua Zhang, Po-Nien Kung, Masahiro Yoshida, Guy Van den Broeck, and Nanyun Peng. Adaptable logical control for large language models. In *Advances in Neural Information Processing Systems 37 (NeurIPS)*, dec 2024. URL `https://starai.cs.ucla.edu/papers/ZhangNeurIPS24.pdf`.

Honghua Zhang, Meihua Dang, Benjie Wang, Stefano Ermon, Nanyun Peng, and Guy Van den Broeck. Scaling probabilistic circuits via monarch matrices. In *Proceedings of the 42th International Conference on Machine Learning (ICML)*, jul 2025a. URL `https://starai.cs.ucla.edu/papers/ZhangICML25.pdf`.

Honghua Zhang, Benjie Wang, Marcelo Arenas, and Guy Van den Broeck. Restructuring tractable probabilistic circuits. In *Proceedings of the 28th International Conference on Artificial Intelligence and Statistics (AISTATS)*, may 2025b. URL `https://arxiv.org/pdf/2411.12256`.

Stephen Zhao, Rob Brekelmans, Alireza Makhzani, and Roger Grosse. probabilistic inference in language models via twisted sequential monte carlo. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 60704–60748, 2024.

A PROOFS

We provide a proof of Proposition 2, restated here for convenience.

Proposition 2 *For any Markov chain $X_{<t} \to Z_t \to X_{\geq t}$, we have*

$$I(X_{<t}; X_{\geq t}) \leq H(Z_t) \leq \log |supp\,(Z_t)|.$$

Proof. We have $I(X_{<t}; X_{\geq t}) \leq I(X_{<t}; Z_t) = H(Z_t) - H(Z_t \mid X_{<t}) \leq H(Z_t)$. The first inequality is the 'data processing inequality' (e.g., proved via the chain rule of mutual information), the equality is a standard identity that follows from definitions, and the final inequality holds because entropies are nonnegative. $\square$

B DFAS

We give a formal definition of a deterministic finite automaton.

Definition 1 *A deterministic finite automaton (DFA) is a tuple $M = (Q, \Sigma, \delta, q_0, F)$, where Q is a finite set of states, Σ a finite set of symbols, $\delta : Q \times \Sigma \to Q$ a transition function, q_0 an initial state, and $F \subseteq Q$ a set of accept states. A string of tokens $w_1 w_2 \ldots w_n$ is accepted by M if there exists a sequence of states $q_0, q_1, \ldots, q_n$ such that $q_i = \delta(q_{i-1}, w_i)$ for $1 \leq i \leq n$ and $q_n \in F$.*

C COMPLEXITIES

		Standard HMM	Neural-Encoded HMM	Full Conditioning HMM
Training	Time	$O(Bn(\tau + s))$	$O(Bn(\tau + s))$	$O(Bn(n\tau + s))$
	Space	$O(n(v + B))$	$O(n(v + B))$	$O(Bn\tau)$
Ctrl-G	Time	$O(Bn(\tau m + s))$	$O(Bn(\tau m + s))$	$O(Bn(n\tau m + s))$
	Space	$O(n(vm + B))$	$O(n(vm + B))$	$O(n(vm + B))$
TRACE	Time	$O(Bn(\tau + s))$	$O(Bn(\tau + s))$	$O(Bn(n\tau + s))$
	Space	$O(n(v + B))$	$O(n(v + B))$	$O(Bn\tau)$

Table 4: Time and space complexities of training and constrained generation algorithms with hybrid HMM variants. The parameters are generation length n; vocabulary size V; batch size B; HMM single-step complexity τ; number of edges m in the DFA (for Ctrl-G); and time s for a single evaluation of the neural model.