Variable Skipping for Autoregressive Range Density Estimation

Eric Liang* [1] **Zongheng Yang*** [1] **Ion Stoica** [1] **Pieter Abbeel** [1 2] **Yan Duan** [2] **Xi Chen** [2]

Abstract

Deep autoregressive models compute *point* likelihood estimates of individual data points. However, many applications (i.e., database cardinality estimation) require estimating *range* densities, a capability that is under-explored by current neural density estimation literature. In these applications, fast and accurate range density estimates over high-dimensional data directly impact user-perceived performance. In this paper, we explore a technique, *variable skipping*, for accelerating range density estimation over deep autoregressive models. This technique exploits the sparse structure of range density queries to avoid sampling unnecessary variables during approximate inference. We show that variable skipping provides 10-100× efficiency improvements when targeting challenging high-quantile error metrics, enables complex applications such as text pattern matching, and can be realized via a simple data augmentation procedure without changing the usual maximum likelihood objective.

1. Introduction

Deep autoregressive (AR) models have achieved state-of-the-art density estimation results in image, video, and audio (Salimans et al., 2017; van den Oord et al., 2016; Van den Oord et al., 2016; Child et al., 2019; Radford et al., 2019; Weissenborn et al., 2019). Recent work has applied them to domains traditionally outside of machine learning, such as physics (Sharir et al., 2020), protein modeling (Rao et al., 2019), and database query optimization (Yang et al., 2019b; 2020). These use cases have surfaced the need for complex inference capabilities from deep AR models. For example, the database cardinality estimation task reduces to estimating the density mass occupied by sets of variables under sparse *range constraints*. In this problem, the database optimizer probes the fraction of records satisfying a query of high-dimensional constraints, e.g., $\Pr(\text{age} > 35 \;\&\&\; \text{salary} < 50K)$, and relies on accurate estimates to pick performant query execution strategies.

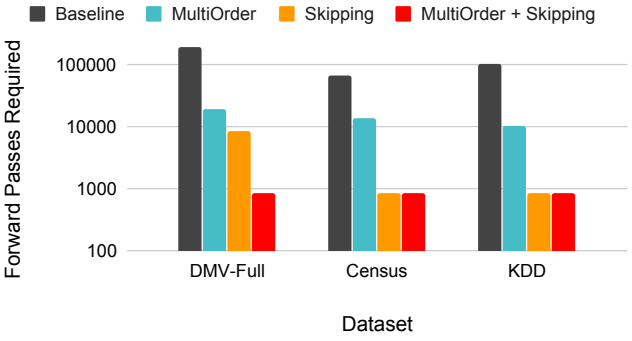

Figure 1. Approximate number of model forward passes required to achieve single-digit inference error at the *99th quantile*. Y-axis shown in *log scale*, lower is better. Variable skipping provides 10-100× compute savings for challenging high-quantile error targets. Refer to the Evaluation section for full results.

In this paper, we call for attention to such *range density estimation* problems in the context of deep AR models. Given rapid advances in model capabilities, fast and accurate range density estimation has broad potential applicability to a number of domains, including databases, text processing, and inpainting (Section 1.1).

Range density estimation involves two related challenges:

- **Marginalization**: the handling of unconstrained variables, and
- **Range Constraints**: variables that are constrained to a specific *range* or *subset* of values.

Exact inference or integration over the query region takes time exponential in the number of dimensions—a cost too high for all but the tiniest problems. Further, both marginalization and range constraints are difficult to implement on top of AR models since they are only trained to provide point density estimates. This motivates the use of approximate inference algorithms such as recently proposed by (Yang et al., 2019b), which show that AR models can significantly improve on the state-of-the-art in range estimation accuracy while remaining competitive in latency.

*Equal contribution [1]EECS, UC Berkeley, Berkeley, California, USA [2]covariant.ai, Berkeley, California, USA. Correspondence to: Eric Liang <ericliang@berkeley.edu>, Zongheng Yang <zongheng@berkeley.edu>.

Proceedings of the 37^{th} International Conference on Machine Learning, Online, PMLR 119, 2020.

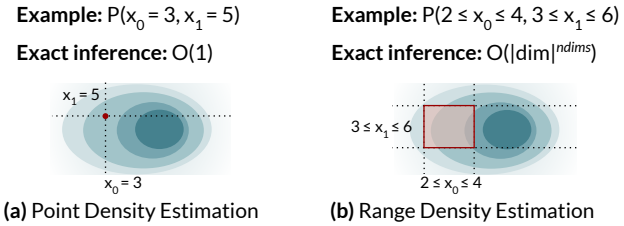

Figure 2. Comparison of point density and range density estimation. Naive marginalization to estimate range densities takes time proportional to the size of the query region (i.e., exponential in the number of dimensions of the joint distribution).

Building on prior work, we distill and evaluate a more general optimization for accelerating range density estimation termed *variable skipping*. The central idea is to exploit the sparsity of range queries, by avoiding sampling through the unconstrained dimensions (i.e., those to be marginalized over) during approximate inference. A training-time data augmentation procedure randomly replaces some dimensions in the input with learnable marginalization tokens, which are trained to represent the *absence* of those dimensions. During inference, the unconstrained dimensions take on these learned values instead of being sampled from their respective domains.

Variable skipping provides two key advantages. First, by not needing to sample a concrete value for certain variables, the number of forward passes is significantly reduced from $O(|\text{Vars}|)$ (e.g., hundreds) to $O(|\text{ConstrainedVars}|)$ (e.g., a few). Second, by avoiding sampling through the (potentially large) unconstrained region, it is possible to reduce the variance of the sampling-based estimator. We show that variable skipping realizes both advantages in practice (Figure 1).

Reducing the computation required for estimates can significantly impact the viability of model-based estimators for the aforementioned computer systems applications. For example, in database query optimization, cardinality estimation is typically run in the inner loop of a dynamic program (Selinger et al., 1979), and hence has to be executed many times in potentially unbatchable fashion. Further, this process must be re-run for each new query as it may have different variable constraints. In this setting, reducing estimation costs from tens or hundreds of forward passes (i.e., the number of columns in a typical production database) to just a handful (i.e, the number of constraints in a typical range query) is critical for adoption. Models that include rarely queried text columns (e.g., byte pair encoded, which exacerbates the problem) may benefit further still.

We start by discussing related work, then reviewing the previously proposed approximate inference algorithm (Yang et al., 2019b), termed Progressive Sampling (Section 3.1), which allows any trained autoregressive model to efficiently compute range densities. We then discuss an optimization, *variable skipping*, which allows dimensions irrelevant to a query to be skipped over at inference time, greatly reducing or eliminating sampling costs (Section 4). We show that, beyond accelerating range density estimation, variable skipping can enable related applications such as pattern matching and text completion. Finally, we study the performance of variable skipping (Section 5).

The contributions of this paper are as follows:

1. We distill the more general concept of *variable skipping*, a training and run-time optimization that greatly reduces the variance of range density estimates.
2. To show its generality, we apply variable skipping to text models, which can then support applications such as pattern matching.
3. We evaluate the effectiveness of variable skipping across a variety of datasets, architecture, and hyperparameter choices, and compare with related techniques such as multi-order training.
4. To invite research on this under-explored problem, we open source our code and a set of range density estimation benchmarks on high-dimensional discrete datasets at `https://var-skip.github.io`.

1.1. Applications of Range Density Estimation

Range density estimation is important for the following applications, among others:

Database Systems: A core primitive in database query optimizer is cardinality estimation (Selinger et al., 1979): given a query with user-defined predicates for a subset of columns, estimate the *fraction* of records that satisfy the predicates. Applying AR models to cardinality estimation was the topic of (Yang et al., 2019b).

Pattern Matching: A regular expression can be interpreted as a dynamically unrolled predicate (i.e., a nondeterministic finite automata) (Hopcroft & Ullman, 1979) over a series of character variables. Hence, its *match probability* can be estimated in the same way as a range query. Section 4.1 shows how this can be realized with variable skipping.

Completion and Inpainting: While an AR model can be straightforwardly used to extend a prefix in the variable ordering, completing a missing value from the *middle* of a sequence of variables requires sampling from the marginal distribution over missing values. We show that variable skipping allows this to be done efficiently (Section 4.2).

2. Related Work

Density Estimation with Deep Autoregressive Models have enjoyed vast interest due to their outstanding capability of modeling high-dimensional data (text, images, tabular).

Efficient architectures such as MADE (Germain et al., 2015) and ResMADE (Durkan & Nash, 2019) have been proposed, and self-attention models (e.g., Transformer (Vaswani et al., 2017)) have underpinned recent new advances in language tasks. Our work optimizes the approximate inference (of range density estimates) on top of such AR architectures.

Masked Language Models. Our variable skipping learns special MASK tokens (Section 4) by randomly masking inputs, which is similar to masked language models such as BERT (Devlin et al., 2019) and CMLMs (Ghazvininejad et al., 2019). These models differ from AR models in optimization goals: they typically predict *only* the masked tokens conditioned on present tokens, and may assume independence among the masked tokens. We study deep AR models for two reasons: (1) our problem settings are in density estimation, and deep AR models have generally shown superior density modeling quality than other generative models; (2) the approximate inference procedure we study (Section 3.1) assumes access to autoregressive factors.

Multi-Order Training handles marginalization by training over many orders and invoking a suitable order (or an ensemble over available orders) during inference. This technique has appeared in NADE (Uria et al., 2014), MADE (Germain et al., 2015), XLNet (Yang et al., 2019a), among others. Variable skipping shares the same goal of efficiently handling marginalization. These prior works have reported increased optimization difficulty as the number of orders to learn increases (some sample a fixed set of orders, while others keep sampling new orders). In the latter case, we posit that the difficulty is due to adding $n!$ input variations; in contrast, variable skipping only extends the vocabulary of each dimension by a MASK symbol, a relatively smaller increase in task difficulty. In Section 5, we compare variable skipping against multi-order training, and show that they can be combined to further reduce errors.

3. Range Density Estimation on Deep Autoregressive Models

We model a finite set of n-dimensional data points $D = \{\boldsymbol{x}^{(0)}, \boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(M)}\}$ as a discrete distribution using an autoregressive model $p_\theta(\boldsymbol{x})$, parameterized by θ. The model is trained on D using the maximum likelihood objective:

$$\min_\theta \mathcal{L}(\theta) = - \mathop{\mathbb{E}}_{\boldsymbol{x}\sim D} \log p_\theta(\boldsymbol{x}) \tag{1}$$

where $p_\theta(\boldsymbol{x}) = \prod_i p_\theta(x_i|\boldsymbol{x}_{<i})$ for each data point $\boldsymbol{x}$.

Range density. We consider range queries of the form

$$p_\theta(X_1 \in R_1, \ldots, X_n \in R_n), \tag{2}$$

where each region R_i is a subset of the domain $\mathrm{Dom}(X_i)$. This formulation encapsulates unconstrained dimensions, where we simply take $R_i = \mathrm{Dom}(X_i)$ (the whole domain).

3.1. Background: Progressive Sampling

Exact inference of Equation 2 is computationally efficient only for low dimensions or small domain sizes. Approximate inference is required to scale its computation.

To solve this problem, (Yang et al., 2019b) adapts classical forward sampling (Koller & Friedman, 2009) for range likelihoods, yielding an unbiased approximate inference algorithm. The algorithm works by drawing in-range samples and re-weighting each intermediate range likelihood. Each in-range sample is drawn from the first dimension to the last (in the AR ordering). As an example, consider estimating $p_\theta(X_1 \in R_1, X_2 \in R_2, X_3 \in R_3)$. Progressive sampling draws $x_1^{MC} \sim p_\theta(X_1|X_1 \in R_1)$ and stores $p_\theta(X_1 \in R_1)$—both tractable operations since a forward pass on the trained AR produces this single-dimensional distribution. It performs another forward pass to obtain $p_\theta(X_2|x_1^{MC})$, which then produces a sample x_2^{MC} and the range likelihood $p_\theta(X_2 \in R_2|x_1^{MC})$. Lastly it obtains $p_\theta(X_3 \in R_3|x_1^{MC}, x_2^{MC})$. It can be shown that the product of all n range likelihoods, e.g.,

$$p_\theta(X_1 \in R_1)p_\theta(X_2 \in R_2|x_1^{MC})p_\theta(X_3 \in R_3|x_1^{MC}, x_2^{MC})$$

is a valid Monte-Carlo estimate of the desired range density.

In the remainder of the paper, we invoke $\mathrm{ProgressiveSampling}(R_1, \ldots, R_n)$ as a black-box estimator, although our variable technique (described next) can work with other estimators for Equation 2.

4. Variable Skipping

Variable skipping works by (1) training special marginalization tokens, MASK_i, for each dimension i; (2) at approximate inference, rewriting each unconstrained variable, e.g., $X_i \in \mathrm{Dom}(X_i)$, into a constrained variable with the singleton range, $X_i \in \{\mathrm{MASK}_i\}$. The training process can be interpreted as dropout of the input, or as data augmentation.

Architecture. We assume a model architecture shown in Figure 3: the input layer, an autoregressive core, and the output layer. For the autoregressive core, we use ResMADE (Durkan & Nash, 2019) for tabular data and an autoregressive Transformer (Vaswani et al., 2017) (encoder only with correct masking) for text data. At the input layer, we embed each data point using a per-dimension trainable embedding table, denoted by $\mathrm{Embed}()$. (For text, we tie the embeddings across all dimensions since they share the same character vocabulary.) The output layer dots the hidden features with the input embeddings to produce logits.

Training-time input masking. First, we add a special token MASK_i to each dimension i's vocabulary. For each input $\boldsymbol{x}$ we uniformly draw the number of masked dimensions $n_{\text{mask}} \sim [0, |\boldsymbol{x}|)$, then sample the n_{mask} positions to

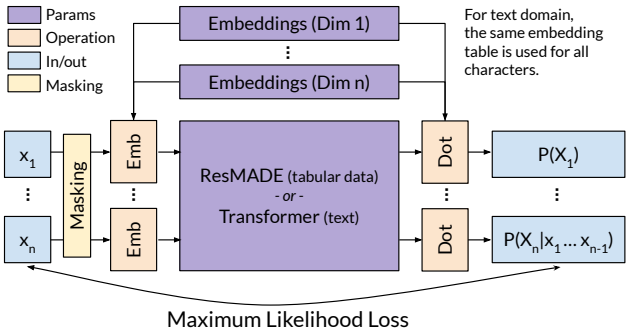

Figure 3. Model architecture.

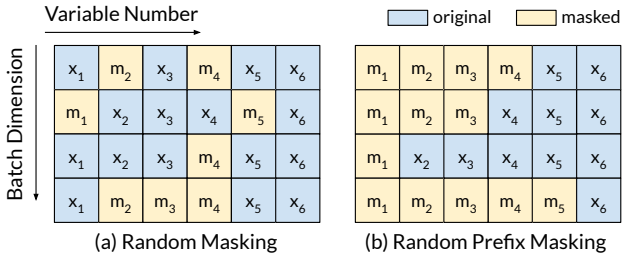

Figure 4. Masking strategies (Section 4). *(a)* For tabular data, we randomly sample the dimensions to mask out for each row. *(b)* For text, we mask a random prefix of each string, exploiting the natural left-to-right ordering.

mask, X_{mask}. For position $i \in X_{\text{mask}}$, we replace the original representation, $\text{Embed}(x_i)$, by the masked representation, $\text{Embed}(\text{MASK}_i)$:

$$\begin{aligned}\text{Encode}(x_i) = &\text{Embed}(x_i)\mathbb{1}(i \notin X_{\text{mask}}) \\ &+ \text{Embed}(\text{MASK}_i)\mathbb{1}(i \in X_{\text{mask}}).\end{aligned} \tag{3}$$

Importantly, the objective remains the MLE for *all* autoregressive factors: we train the parameters to predict the *original values* at each dimension, given *a mix of original and masked information* at previous dimensions. In other words, we minimize the negative log-likelihood

$$-\log p_\theta(x_i|\boldsymbol{x}_{<i}) = -\log p_\theta\left(x_i|\forall j < i, \text{Encode}(x_j)\right) \tag{4}$$

over all dimension i. Conditioning on the mask tokens ensures that those representations are trained. Since we do not alter the *output targets* and the mask positions are chosen independently of the data, no bias is introduced.

Infer-time skipping. Given a range query (Equation 2), we look for each unconstrained dimension and replace its domain with a singleton set of its marginalization token:

$$\begin{aligned}&p_\theta(\ldots, X_i \in R_i = \text{Dom}(X_i), \ldots) \\ \rightarrow\quad &p_\theta(\ldots, X_i \in R_i' = \{\text{MASK}_i\}, \ldots)\end{aligned} \tag{5}$$

We then invoke $\text{ProgressiveSampling}()$ which would thus *skip* the sampling for those dimensions.

Example. Suppose we have an AR model trained over the autoregressive ordering $\{\text{age}, \text{salary}, \text{city}\}$, and want to draw a sample from $p_\theta(\text{city}|\text{salary} > 50\text{k})$.

- Without skipping, first we draw $x_1 \sim p_\theta(\text{age})$, then $x_2 \sim p_\theta(\text{salary}|\text{age} = x_1, \text{salary} > 50\text{k})$, and finally $x_3 \sim p_\theta(\text{city}|\text{age} = x_1, \text{salary} = x_2)$.
- With skipping, we can directly sample $x_2 \sim p_\theta(\text{salary}|\text{age} = \text{MASK}_1, \text{salary} > 50\text{k})$, followed by $x_3 \sim p_\theta(\text{city}|\text{age} = \text{MASK}_1, \text{salary} = x_2)$.

4.1. Prefix Skipping for Text Pattern Matching

Any regex can be implemented as a nondeterministic finite automata (NFA) (Hopcroft & Ullman, 1979), which takes a stream of characters and determines acceptable next characters. We can use progressive sampling with any regex, treating its NFA like a dynamically unrolled predicate. For example, consider the regex $[\texttt{at}](\texttt{c} + |\texttt{i}+)\texttt{e}$. Possible matches include `ace`, `aiie`, and `tiiie`. Progressive sampling would work as follows: first we sample $x_1 \in \{a, t\}$, then $x_2 \in \{c, i\}$. Depending on whether $x_2 = c$ or $x_2 = i$, third we either sample $x_3 \in \{c, e\}$ or $\{i, e\}$ (this is the "dynamic" part), and so on. By retaining an NFA per sample, we obtain an estimate of the overall match probability.

However, this naive formulation is inefficient when there are long unconstrained sequences. Consider the regex $.*\texttt{icml}.*$, intended to match any string containing the token `icml`. The probability of sampling a random prefix from an AR model matching this is vanishingly small—perhaps millions of samples for a hit. To avoid this, we can try to *skip over* sequences of unconstrained characters and compute the probability of `icml` at specific offsets directly. All that would remain is sampling forward through the remainder of the variables to avoid double counting duplicate occurrences of the token. Using $m(x_i)$ to denote a string match at position i, and $m(x_{>i})$ the existence of a match at any position $> i$, the match probability is approximated as:

$$\begin{aligned}P_{\text{match}} &= \sum_{i=1}^{n} P(m(x_i)) \cdot (1 - P(m(x_{>i})|m(x_i)) \\ &\approx \sum_{i=1}^{n} p_\theta(m(x_i)|\{x_j = \text{MASK}_j : j < i\}) \\ &\quad (1 - p_\theta(m(x_{>i})|\{x_j = \text{MASK}_j : j < i\}, m(x_i))\end{aligned} \tag{6}$$

Due to the need for masking contiguous prefixes, the model is trained with *random prefix masking* (Figure 4) to allow such contiguous characters to be skipped. We show the effectiveness of this strategy in Section 5.7, which implements

Table 1. Datasets used in evaluation. "Domain" refers to the range of distinct values per table column (i.e., DRYAD-URLS contains 78 different character values).

Dataset	Rows	Cols	Domain	Type
DMV-FULL	11.6M	19	2–32K	Discrete
CENSUS	2.5M	67	2–18	Discrete
KDD	95K	100	2–896	Discrete
DRYAD-URLS	2.4M	100	78	Text

simple pattern queries over an AR Transformer model.

4.2. Other Mask Patterns

Finally, we note that more structured mask patterns can be used, such as sub-sequences in text or random patches in images (Dupont & Suresha, 2018). This allows for marginalization over complex subsets of dimensions with potential applicability to not only sample variables given prefixes of the AR ordering (i.e., from $P(x_i|x_1, \ldots, x_{i-1})$) but also variables *later on*, i.e., from $P(x_i|x_1, \ldots, x_{i-1}, x_{i+k}, \ldots, x_N)$ by marginalizing over $\{x_{i+1}, \ldots, x_{i+k-1}\}$. We leave investigation of these potential applications to future work.

5. Evaluation

Our evaluation investigates the following questions:

1. How much does variable skipping improve estimation accuracy compared to baselines, and how is this impacted by the sampling budget?
2. Can variable skipping be combined with multi-order training to further improve accuracy?
3. To what extent do hyperparameters such as the model capacity and mask token distribution impact the effectiveness of variable skipping?
4. Can variable skipping be applied to related domains such as text, or is it limited to tabular data?

Overall, we find that variable skipping robustly improves estimation accuracy across a variety of scenarios. Given a certain target accuracy, skipping reduces the required compute cost by one to two orders of magnitude.

5.1. Datasets

We use the following public datasets in our evaluation, also summarized in Table 1. When necessary, we drop columns representing continuous data. We consider supporting continuous variables an orthogonal issue, and limit our evaluation to discrete domains:

DMV-FULL (State of New York, 2019). Dataset consisting of vehicle registration information in New York (i.e., attributes like vehicle class, make, model, and color). We use all columns except for the unique vehicle ID (VIN). This dataset was also used in (Yang et al., 2019b), but there it was restricted to 11 of the smaller columns.

Table 2. Hyperparameters for all experiments. We used a ResMADE for tabular data, and a Transformer for text.

Hyperparameter	Value
Training Epochs	20 (200 for KDD)
Batch Size	2048
Architecture	ResMADE
Residual Blocks	3
Hidden Layers / Block	2
Hidden Layer Units	256
Embedding Size	32
Optimizer	Adam
Learning Rate	5e-4
Learning Rate Warmup	1 epoch
Mask Probability	$\sim$ Uniform$[0, 1)$
Transformer Num Blocks	8
Transformer MLP Dims (d_{ff})	256
Transformer Embed Size (d_{model})	32
Transformer Num Heads	4
Transformer Batch Size	512

KDD (Dua & Graff, 2017). KDD Cup 1998 Data. We used the first hundred columns, sans noexch, zip, and pop901-3, which were especially high-cardinality. This leaves 100 discrete integer domains with 2 to 896 distinct values each.

CENSUS (Dua & Graff, 2017). The US Census Data (1990) Data Set, which consists of a 1% sample made publicly available. We use all available columns, which range from 2 to 18 distinct values each.

DRYAD-URLS (Sen et al., 2016). For text domain experiments, we use this small dataset of 2.4M URLs, each truncated to 100 characters. This dataset was chosen to emulate a plausible `STRING` column in a relational database.

5.2. Evaluation Metric

We issue a large set of randomly generated range queries, and measure how accurately each estimator answers them. We report the multiplicative error, or *Q-error*, defined as the factor by which an estimate differs from the actual density (obtained by actually executing each query on the dataset):

$$\text{Error} := \max(\text{estimate}, \text{actual}) / \min(\text{estimate}, \text{actual})$$

Hence, a perfect estimate for a query has an error of 1.0. Moreover, we report the median, 99%-tile, and maximum Q-error across all queries. We note that the median error is typically within a fraction of 1.0 for all estimators. The reason is that most randomly generated queries are "easy" (i.e., hit few cross-dimension predicate correlations), and only a few are "hard". Because of this, even a naive estimator can achieve good performance in many cases. Hence, our focus is on high quantile errors for evaluation.

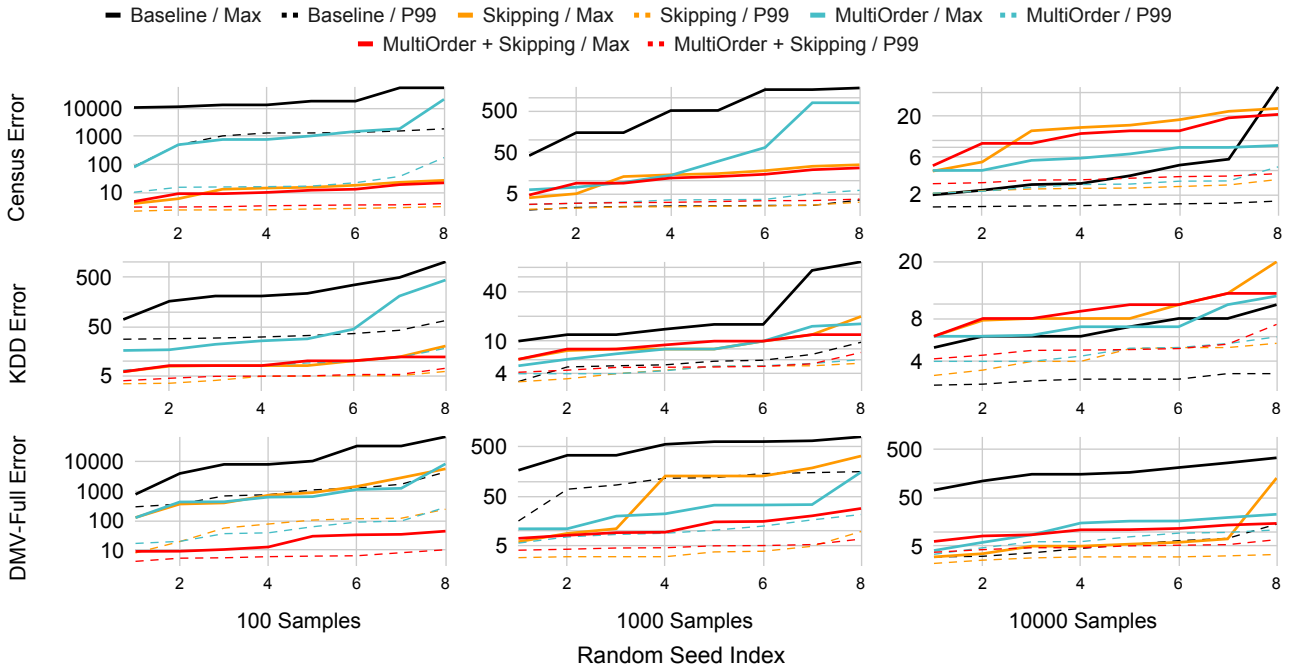

Figure 5. Variable skipping and skipping combined with multi-order training vs. baselines across different datasets, variable orderings, and sampling budgets. Error is plotted on the y-axis in *log scale* (lower is better). Each column reflects a $10\times$ increase in sampling budget as we move to the right. Results for 8 different variable orders within each plot are sorted by increasing error. Variable skipping provides $10-100\times$ max error reduction at low budgets, and still improves accuracy at high sampling budgets for large datasets such as DMV-FULL. This data is also shown in tabular form in Table 5, which additionally reports median errors.

5.3. Experiment Setup

For queries against tabular data, we used the experiment framework from (Yang et al., 2019b), randomly drawing between 5 and 12 conjunctive variable constraints per query[1]. It is important to not have too many or too few constraints, which would skew the distribution of true density estimates towards 0.0 (too many constraints lead to little density) or 1.0 (too few constraints lead to high density) respectively.

For text queries, we issued pattern glob queries of the form `value CONTAINS <str>`, where `<str>` is a character sequence between 3 and 5 characters in length drawn randomly from the full text corpus. This also provides a challenging spread of density from very common (e.g., `CONTAINS ".com"`), to quite rare (e.g., `CONTAINS "XVQ/i"`).

We compare between the following approaches, all of which use progressive sampling (Section 3.1) as the approximate inference procedure:

- **Baseline**: An autoregressive model queried using vanilla progressive sampling (Yang et al., 2019b).
- **Skipping**: An autoregressive model trained with random input masking and queried with the variable skipping optimization enabled (Section 4).
- **MultiOrder**: An autoregressive model trained under multiple variable orders to enable querying an ensemble of 10 orders at inference time (Uria et al., 2014; Germain et al., 2015).
- **MultiOrder + Skipping**: Combining the multi-order and variable skipping techniques.

The full list of training hyperparameters can be found in Table 2. Unless otherwise specified, we use a ResMADE (Durkan & Nash, 2019) with 3 residual blocks, two 256-unit hidden layers per block, and an 32-unit wide embedding for each input dimension. We choose hyperparameters known to optimize for progressive sampling performance (Yang et al., 2019b), but did not otherwise tune them for our experiments. In our ablations (Section 5.6) we found that the most sensitive hyperparameter to performance is the embedding size, which is closely related to model size.

The autoregressive variable ordering can significantly affect estimator variance. We thus evaluate each technique on 8 randomly chosen variable orderings and train a (fixed-order) model for each ordering[2]. For multi-order models,

[1] An example query for DMV-FULL may be: `record_type == 1 AND city == 17 AND zip > 10000 AND model_year < 1990 AND max_weight > 5000`.

[2] For ResMADE, this means we sample 8 sets of {input order-

Table 3. The model negative log-likelihoods at convergence in bits/datapoint (evaluated using non-masked data). We also report standard deviation across multiple random order seeds.

Dataset	Baseline	Random Input Masking	MultiOrder(5)	MultiOrder(10)	MultiOrder(15)
CENSUS	52.04 ± .009	52.34 ± .02	52.69 ± .02	52.79 ± .03	52.81 ± .03
DMV-FULL	43.12 ± .04	43.65 ± .06	44.15 ± .05	44.53 ± .06	44.65 ± .04
KDD	107.5 ± .3	116.58 ± .2	123 ± .9	127.6 ± .4	128.4 ± .5

we train 8 distinct sets of 10 randomly chosen orders (we saw diminishing returns past 10 orders), unless specified otherwise. To ensure fairness, when not using skipping, we use a model trained without masking.

For multi-order ResMADE, to condition on the current ordering statistics each masked linear layer is allocated an additional weight matrix that shares the existing mask and has an all-one vector as its input[3]. Due to the additional weights, we size down the hidden units appropriately to ensure that the multi-order models have about the same parameter count as other models.

5.4. Variable Skipping Performance

We evaluate the impact of the variable skipping optimization on the DMV-FULL, CENSUS, and KDD datasets. For each dataset, we generated 1000 random range queries.

In Figure 5 we show the results of variable skipping (orange and red lines) compared against baselines (black and turquoise lines). This data is also shown in tabular form in Table 5. We evaluate with sampling budgets of 100, 1000, and 10000 samples (left, center, and right columns respectively). Note that a sample refers to *all* the forward passes required to sample relevant variables (e.g., for CENSUS a single sample takes 67 forward passes without skipping). We limit to 10k samples for cost reasons (at 10k samples, each query takes multiple seconds to evaluate even with a GPU). There are several key takeaways:

High-quantile error differentiates estimators: Across all estimators, the median error is very close to 1.0 (not shown since it is indistinguishable in log scale). However, systems applications necessarily seek to minimize the worst-case error, which does vary significantly across samplers.

Skipping significantly improves sampling efficiency: Across all datasets, variable skipping provides between $10\times$ to $100\times$ max error reduction at low sampling budgets (i.e., 100 samples), compared to the baseline. It also provides up to $10\times$ improvement over the multi-order ensemble alone.

Concretely, at 100 samples the 99th-quantile error for CENSUS is reduced from ~1000 to 2.5, KDD from ~30 to 3, and DMV-FULL from ~1000 to 100. Moving up to 1000 samples, we continue to see a significant improvement at the max error, with CENSUS improved from ~500 to 10, KDD from ~15 to 8, and DMV-FULL from ~500 to 100.

Compared to multi-order, variable skipping provides $10\times$ better max error reduction for CENSUS and KDD at 100 samples. Interestingly, while multi-order and variable skipping provide comparable improvements for DMV-FULL at 100 samples, *combining multi-order and skipping* provides a further $10\times$ improvement in both max and 99th quantile error. This suggests that variable skipping and multi-order training are orthogonal mechanisms, and can be combined for larger datasets such as DMV-FULL to reduce both error *and* inference costs.

Variable skipping can help even at high sampling budgets: On the DMV-FULL dataset, variable skipping provides more than an order of magnitude reduction in max error (from ~150 to 5), even at 10000 samples. We hypothesize this is due to the large domain sizes of DMV-FULL (up to 32K distinct values), which in the worst case would require a much larger number of samples to achieve low estimation error. As evidence for this, a large number of samples are required to achieve good errors even with skipping enabled. This is in contrast to the smaller CENSUS and KDD datasets where skipping achieves close to single-digit errors even with as low as 100 samples. This suggests that for even larger datasets (common in industrial settings), skipping may have an even greater impact.

We have also provided a summary of the results for Figure 5 in Figure 1. For the concrete target of $10\times$ error at the 99th quantile, skipping significantly reduces the compute requirements over both the baseline and multi-order. This is due to both accuracy improvements that combine with those provided by multi-order ensembles, and also the reduced compute requirements of skipping.

5.5. Model Likelihoods vs. Training Scheme

Training with partially masked inputs makes the learning task more difficult: the number of examples increases by a factor of 2^N, and effectively the same model is learning multiple autoregressive distributions. Table 3 shows that in terms of negative log-likelihoods achieved, models trained with masking do have a slightly higher NLL than baseline, as expected. However, the NLLs achieved are lower than

ing, intermediate connectivity masks}.

[3]This treatment has appeared in MADE (Germain et al., 2015).

Table 4. Variable skipping vs. vanilla progressive sampling on the text domain. Naive sampling refers to generating samples (from the learned AR model) without constraints and then filtering the generated samples to estimate the probability of matches. We include naive sampling as a baseline for this experiment since it is competitive with progressive sampling in the text domain. We measure the estimation error over 100 random pattern queries against the DRYAD-URLS dataset, and show the bootstrap standard deviation.

Num Samples	Metric	Naive Sampling	Progressive Sampling (Baseline)	Variable Skipping
1000	Max Error	6412 ± 1280	4628 ± 1785	**115.2** ± 25.6
1000	P99 Error	4054 ± 1386	1741 ± 1824	**89.5** ± 30.6
1000	Median Error	**1.23** ± .06	1.66 ± .15	1.39 ± .08

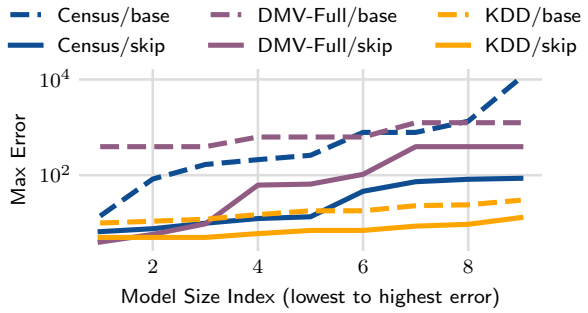

Figure 6. Model size vs. max estimation error on 1000 queries with 1000 samples each. The results for each dataset are sorted in increasing error. Errors are plotted on the y-axis in *log scale* (lower is better). Errors increase as the model embedding sizes are reduced from 32 to 2, and hidden layer sizes from 256 to 16. Variable skipping (solid lines) retains an advantage across this two orders of magnitude change in model capacity.

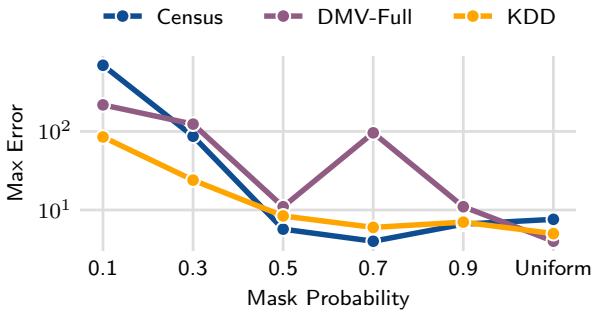

Figure 7. Varying the masking scheme. Here we measure the max estimator error with skipping enabled over 1000 queries with 1000 samples each, on the natural variable order. Errors are plotted on the y-axis in *log scale* (lower is better).

those of multi-order models, and the gap only widens with an increased number of orders.

Even though NLLs of masked models are higher than those of baseline, Figure 5 shows the benefit: estimation error is significantly improved when variable skipping is enabled. This highlights *the non-perfect alignment of optimizing for point likelihoods vs. downstream range query performance*, opening up interesting future directions.

5.6. Model Size and Masking Ablations

In Figure 6 we study the relationship between model size and estimation accuracy. For this experiment, we vary the model embedding size among $\{2, 8, 32\}$ dimensions, and the hidden layer size among $\{16, 64, 256\}$ units. We see that variable skipping retains a robust advantage across nearly two orders of magnitude variation in model size.

In Figure 7 we compare a few schemes for selecting the mask distribution, from fixed masking probabilities of $\{0.1, 0.3, 0.5, 0.7, 0.9\}$, vs. the random uniform scheme used in the main experiment. We see that drawing the mask probability uniformly at random obtains the lowest errors for KDD and DMV-FULL, and close to optimal for CENSUS as well, showing it to be a robust choice.

5.7. Application to Pattern Matching in Text Domain

Finally, we show that variable skipping can be applied to the text domain for estimating the probability of pattern matches. Pattern matching (or more generally, regex matching), can be thought of as unrolling a dynamic predicate as variables are sampled (Section 4.1). Here we evaluate a simple character-level Transformer model on the DRYAD-URLS dataset. We note that this is not a realistic application since scanning a dataset of this size is much faster than sampling from a model, however it demonstrates the applicability of variable skipping across domains. Table 4 shows that prefix skipping enables much lower variance estimates than naive sampling and vanilla progressive sampling.

6. Conclusion

To summarize, we identify the range density estimation task and important applications. We propose *variable skipping*, which greatly reduces sampling variance and inference latency. We validate the effectiveness of these techniques across a variety of datasets and model configurations.

Table 5. The full table of quantiles across all random orders evaluated in Figure 5. We show the mean and standard deviation of the quantiles across the random order seeds.

Samples	Metric	Dataset	Progressive (Baseline)	Multi-Order	Skipping	Multi-Order + Skipping
100	P50	Census	1.19 ± .01	1.53 ± .32	**1.09 ± .01**	1.21 ± .03
		KDD	1.24 ± .02	1.41 ± .16	**1.14 ± .02**	1.25 ± .05
		DMV-Full	1.22 ± .04	1.70 ± .22	**1.08 ± .01**	1.22 ± .04
	P99	Census	1150 ± 550	38.6 ± 52	**2.55 ± .29**	3.36 ± .32
		KDD	36.4 ± 11	9.82 ± 3.4	**4.58 ± .83**	5.07 ± .85
		DMV-Full	1130 ± 1300	80 ± 81	93 ± 69	**6.5 ± 1.7**
	Max	Census	24500 ± 18000	3490 ± 6700	15.1 ± 7.3	**12.3 ± 5.3**
		KDD	345 ± 280	99.9 ± 140	9.97 ± 4.1	**9.25 ± 1.9**
		DMV-Full	21200 ± 22000	1640 ± 2600	1560 ± 1700	**22.7 ± 13**
1000	P50	Census	**1.06 ± .01**	1.44 ± .31	1.09 ± .01	1.20 ± .02
		KDD	**1.08 ± .01**	1.34 ± .17	1.13 ± .02	1.25 ± .05
		DMV-Full	1.08 ± .01	1.53 ± .24	**1.05 ± .01**	1.18 ± .04
	P99	Census	2.58 ± .39	3.84 ± 1.1	**2.50 ± .28**	3.2 ± .28
		KDD	5.78 ± 1.7	4.69 ± .67	**4.41 ± .74**	5.05 ± .89
		DMV-Full	105 ± 44	11.3 ± 4.8	**4.24 ± 2.1**	4.92 ± .81
	Max	Census	798 ± 700	212 ± 330	14.7 ± 7.0	**12.8 ± 5.1**
		KDD	30.7 ± 30	9.42 ± 3.8	9.95 ± 4.1	**9.37 ± 1.9**
		DMV-Full	508 ± 200	39.4 ± 43	114 ± 100	**14 ± 6.7**
10000	P50	Census	**1.03 ± .01**	6.31 ± 1.6	1.09 ± .01	1.20 ± .24
		KDD	**1.05 ± .01**	1.33 ± .18	1.13 ± .02	1.25 ± .05
		DMV-Full	1.05 ± .01	1.48 ± .24	**1.05 ± .01**	1.18 ± .04
	P99	Census	**1.49 ± .08**	2.84 ± .70	2.48 ± .29	3.20 ± .29
		KDD	**2.99 ± .19**	4.43 ± .93	4.36 ± .75	5.08 ± .85
		DMV-Full	5.95 ± 3.4	7.23 ± 2.4	**2.89 ± .34**	4.96 ± .85
	Max	Census	8.86 ± 14	**6.31 ± 1.6**	14.7 ± 7.0	12.6 ± 5.0
		KDD	**7.0 ± 1.5**	7.57 ± 1.9	9.97 ± 4.1	9.37 ± 1.9
		DMV-Full	181 ± 78	13.5 ± 6.2	20.2 ± 40	**10.5 ± 2.6**

References

Child, R., Gray, S., Radford, A., and Sutskever, I. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423.

Dua, D. and Graff, C. UCI machine learning repository, 2017. URL http://archive.ics.uci.edu/ml.

Dupont, E. and Suresha, S. Probabilistic semantic inpainting with pixel constrained cnns. *arXiv preprint arXiv:1810.03728*, 2018.

Durkan, C. and Nash, C. Autoregressive energy machines. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97, pp. 1735–1744, Long Beach, California, USA, 2019.

Germain, M., Gregor, K., Murray, I., and Larochelle, H. MADE: Masked autoencoder for distribution estimation. In *International Conference on Machine Learning*, pp. 881–889, 2015.

Ghazvininejad, M., Levy, O., Liu, Y., and Zettlemoyer, L. Mask-predict: Parallel decoding of conditional masked language models. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 6114–6123, 2019.

Hopcroft, J. E. and Ullman, J. D. Introduction to automata theory, languages and computation. adison-wesley. *Reading, Mass*, 1979.

Koller, D. and Friedman, N. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

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

Rao, R., Bhattacharya, N., Thomas, N., Duan, Y., Chen, P., Canny, J., Abbeel, P., and Song, Y. Evaluating protein transfer learning with tape. In *Advances in Neural Information Processing Systems*, pp. 9689–9701, 2019.

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

Selinger, P. G., Astrahan, M. M., Chamberlin, D. D., Lorie, R. A., and Price, T. G. Access path selection in a relational database management system. In *Proceedings of the 1979 ACM SIGMOD international conference on Management of data*, pp. 23–34. ACM, 1979.

Sen, J., Lansdall-Welfare, T., Sudhahar, S., Carter, C., and Cristianini, N. Data from: Women are seen more than heard in online newspapers., 2016. URL https://datadryad.org/stash/dataset/doi:10.5061/dryad.p8s0j.

Sharir, O., Levine, Y., Wies, N., Carleo, G., and Shashua, A. Deep autoregressive models for the efficient variational simulation of many-body quantum systems. *Phys. Rev. Lett.*, 124:020503, Jan 2020. doi: 10.1103/PhysRevLett.124.020503.

State of New York. Vehicle, snowmobile, and boat registrations. catalog.data.gov/dataset/vehicle-snowmobile-and-boat-registrations, 2019. [Online; accessed March 1st, 2019].

Uria, B., Murray, I., and Larochelle, H. A deep and tractable density estimator. In *International Conference on Machine Learning*, pp. 467–475, 2014.

van den Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., and Kavukcuoglu, K. Wavenet: A generative model for raw audio. In *Arxiv*, 2016.

Van den Oord, A., Kalchbrenner, N., Espeholt, L., Vinyals, O., Graves, A., et al. Conditional image generation with pixelcnn decoders. In *Advances in neural information processing systems*, pp. 4790–4798, 2016.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. In *Advances in neural information processing systems*, pp. 5998–6008, 2017.

Weissenborn, D., Täckström, O., and Uszkoreit, J. Scaling autoregressive video models. *arXiv preprint arXiv:1906.02634*, 2019.

Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R. R., and Le, Q. V. XLNet: Generalized autoregressive pretraining for language understanding. In *Advances in neural information processing systems*, pp. 5754–5764, 2019a.

Yang, Z., Liang, E., Kamsetty, A., Wu, C., Duan, Y., Chen, X., Abbeel, P., Hellerstein, J. M., Krishnan, S., and Stoica, I. Deep unsupervised cardinality estimation. volume 13, pp. 279–292. VLDB, 2019b.

Yang, Z., Kamsetty, A., Luan, S., Liang, E., Duan, Y., Chen, X., and Stoica, I. NeuroCard: One cardinality estimator for all tables. *arXiv preprint arXiv:2006.08109*, 2020.