Masked Diffusion Training Induces Resampling-like Carry Representations in Addition

Anonymous Authors[1]

Abstract

Masked diffusion models (MDMs) have emerged as alternatives to autoregressive (AR) language models, with evidence of stronger generalization under data constraints. We study this gap mechanistically in a controlled addition task: one-layer Transformers add two six-digit numbers after training only on examples with limited carry complexity, then are evaluated on an out-of-distribution carry-generalization split requiring $N_{\text{carry}} > 2$. This tests whether models extrapolate the carry rule beyond the carry numbers observed during training. With C2, MDMs outperform AR models on high-carry examples, while C2-Resampled largely closes the gap. We trace this difference to the geometry of carry representations. Attention and MLP sublayers play similar roles in both model classes: attention aggregates base-addition information, while the MLP makes answer tokens more linearly decodable. However, MDM training yields stronger *linear alignment*, which we defined as the fraction of post-attention representation variance captured along the carry/non-carry direction. Then, theoretically, we show in a Gaussian model that higher *linear alignment* improves robustness to boundary perturbations. Retraining the MLP while freezing earlier representations preserves the same accuracy ordering, suggesting that MDMs generalize better in this setting because they learn better-aligned carry representations, not a qualitatively different layer-level algorithm.

1. Introduction

Masked diffusion models (MDMs) have recently emerged as a promising alternative to autoregressive (AR) language models for discrete sequence generation (Nie et al., 2025; Arriola et al., 2025; Swerdlow et al., 2025). Unlike AR models, which generate tokens in a fixed left-to-right order, MDMs learn to reconstruct randomly masked tokens and can generate sequences through iterative denoising (Ho et al., 2020; Gulrajani & Hashimoto, 2023; Austin et al., 2021). This difference in training and decoding raises a basic question: do MDMs and AR models learn qualitatively different algorithms, or do they learn similar algorithms with different internal representation geometry?

Prior work has compared AR and diffusion-style language models mainly through high-level performance metrics such as likelihood, accuracy, or generation quality (Fraij & Dauncey, 2025; Feng et al., 2025; Ni et al., 2025; Prabhudesai et al., 2026; Li et al., 2022). While such comparisons are important it is unclear whether this success comes from learning a more systematic rule, from better use of training data, or from a more robust internal representation of task-relevant features.

We study this question in a controlled addition task, a widely used benchmark for analyzing algorithmic generalization in language models(Quirke & Barez, 2023; Kantamneni & Tegmark, 2025; Li et al., 2025). We use the C2 and C2-Resampled data regimes, which restrict the number of carries to at most two. We then evaluate the models on examples requiring more than two carries. This setting directly probes whether a model has learned a carry-propagation rule that extrapolates beyond the carry complexity observed during training.

Our main empirical observation is that MDMs generalize better than AR models under a C2 training set. However at C2-Resampled both model classes improve substantially and the AR–MDM gap largely closes. This suggests that the key distinction is not simply between AR and MDM, but between fixed-data training and training procedures that induce more robust internal representations. We therefore ask what representation-level property is shared by MDM training and data resampling.

To answer this, we analyze the internal representations of one-layer Transformer models. We find that attention and MLP layers play broadly similar roles across AR and MDM models. Post-attention representations are organized pri-

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

marily by digit-pair information, while the MLP transforms these representations into a form from which answer digits are linearly decodable. Thus, the MDM advantage does not appear to come from a qualitatively different layer-level algorithm.

Instead, we identify a difference in the geometry of carry representations. Within each digit-pair group, the model must distinguish examples with and without carry. We introduce linear alignment, defined as the fraction of representation variance captured along the carry/non-carry direction, to measure how strongly carry information is organized along a single feature direction. MDM training under C2 regime induces stronger carry-direction alignment than AR training, and this alignment resembles the geometry induced by epoch-wise data resampling.

Our contributions are as follows:

1. We show that, in a carry-limited addition setting, MDMs generalize better than AR models under a fixed C2 regime, while epoch-wise C2-Resampled training substantially improves both model classes and largely closes the AR–MDM gap.
2. We provide representation-level evidence that this gap is unlikely to arise from a qualitatively different division of labor between attention and MLP layers.
3. We identify linear alignment of carry representations as a shared representation-level property associated with OOD carry generalization.

2. Preliminaries

2.1. Transformer Architecture

We use a one-layer Transformer with Pre-Layer Normalization (Pre-LN) and trainable positional encodings. Let d_{model} denote the Transformer hidden dimension, and let $h_i^{(l)} \in \mathbb{R}^{d_{\text{model}}}$ denote the residual-stream vector at position i after layer l. The residual stream is updated as

$$h_i^{(0)} = pos_i + token_i,$$
$$\tilde{h}_i^{(l)} = h_i^{(l-1)} + a_i^{(l)},$$
$$h_i^{(l)} = \tilde{h}_i^{(l)} + m_i^{(l)}.$$

Here $a_i^{(l)}$ is the attention update, $m_i^{(l)}$ is the MLP update, and $\tilde{h}_i^{(l)}$ is the post-attention representation, *i.e.*, the residual-stream vector after the attention update and before the MLP update. The sublayer outputs and final logits are

$$a_i^{(l)} = \text{MHA}^{(l)}\left(\{\text{LN}(h_j^{(l-1)}) : j \in \mathcal{A}_i\}\right),$$
$$m_i^{(l)} = W_{out}^{(l)}\,\text{ReLU}\left(W_{in}^{(l)}\,\text{LN}(\tilde{h}_i^{(l)})\right),$$
$$\text{logits}_i = W_{\text{unembed}}\,\text{LN}(h_i^{(L)}).$$

The set $\mathcal{A}_i$ contains the positions visible to position i under the attention mask. For AR, $\mathcal{A}_i = \{1, \ldots, i\}$. For MDM, we use bidirectional attention over the masked input sequence; for a length-T sequence, $\mathcal{A}_i = \{1, \ldots, T\}$. In later analyses, "post-attention representation" refers to $\tilde{h}_i^{(1)}$.

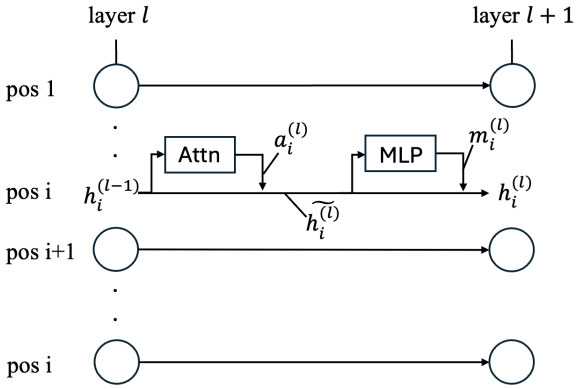

Figure 1. Abstract diagram of transformer architecture

2.2. Model Classes and Decoding

Autoregressive Model. AR models generate answer tokens in a fixed left-to-right order. They are trained with next-token cross-entropy under teacher forcing, and at inference time each generated token is appended to the context before predicting the next one.

Masked Diffusion Model. MDMs instead train a mask predictor on random partially masked answers: a random mask pattern is applied to the answer tokens, and cross-entropy is computed only on the masked positions. At inference, generation starts from a fully masked answer and the model iteratively fills masked positions. We distinguish the trained MDM model from its decoding schedule. In `MDM-random`, masked positions are filled in a random order, matching the random masking used by the training objective. In `MDM-AR-style`, the same trained MDM is decoded left-to-right over answer positions. Thus, "AR-style" refers only to the inference order, not to AR training. Other schedules, such as confidence-based greedy decoding, are also possible (Nie et al., 2025; Kim et al., 2025).

2.3. Addition Terms and Notation

For convenience, we write the operands and the seven non-padding result digits in decimal (base-10) notation:

$$a_6a_5a_4a_3a_2a_1 + b_6b_5b_4b_3b_2b_1 = c_7c_6c_5c_4c_3c_2c_1. \quad (1)$$

The implemented sequence prepends one additional leading-zero padding position, yielding an 8-position answer. Using this notation, we define the digit-pair index, the base-addition digit before incoming carry, and the carry-out variable at each position:

$$\begin{aligned}\mathrm{Pair}_i &= 10a_i + b_i,\\ \mathrm{Base}_i &= (a_i + b_i) \bmod 10,\\ \kappa_i &= \mathbf{1}\{a_i + b_i + \kappa_{i-1} \geq 10\},\ \kappa_0 = 0,\ i = 1, \ldots, 6.\end{aligned}$$

Here κ_i is the carry out of position i and κ_{i-1} is the incoming carry that affects digit c_i:

$$c_i = (a_i + b_i + \kappa_{i-1}) \bmod 10, \qquad i = 1, \ldots, 6.$$

For the six-digit addition task, $c_7 = \kappa_6$ and the additional eighth answer position is always zero.

We use the number of carries, denoted N_{carry} to describe the carry complexity of an example. The number of carries is

$$N_{\mathrm{carry}} = \sum_{i=1}^{6} \kappa_i.$$

3. Experimental Setup

3.1. Dataset Construction

We use two carry-limited training data regimes, denoted C2 and C2-Resampled. To construct one accepted example, we sample two 6-digit operands by drawing each digit independently and uniformly from $\{0, \ldots, 9\}$, compute the sum, and keep the example only if it has at most two carries, *i.e.*, $N_{\mathrm{carry}} \leq 2$.

In C2, for each seed, we construct one set of 1,600 accepted examples and reuse the same set throughout training. In C2-Resampled, at every epoch we resample a fresh set of 1,600 accepted examples using the same carry constraint. Thus, both regimes are trained only on examples with at most two carries; they differ only in whether the finite training set is fixed or resampled each epoch.

3.2. Model and Training Setup

All experiments use 1-layer pre-LN Transformer models with matched width, depth, and component structure across AR and MDM. The models differ in training objective: AR is trained with next-token prediction, while MDM is trained with random masking over answer positions, as described in Section 2.2. Each model is trained for 5,000 epochs under both C2 and C2-Resampled data regimes.

For MDM, random-order and AR-style decoding are evaluated using the same trained MDM checkpoints; thus, `MDM-random` and `MDM-AR-style` differ only in decoding schedule. Models are trained on fixed-length sequences containing two 6-digit operands and an 7-digit answer. See Equation 1 and Appendix A for the implementation details. Results are averaged across 30 seeds.

3.3. Answer-Digit Representations

Because AR and MDM attach logits to different sequence positions, analyses of representations must compare vectors associated with the same predicted answer digit. Here i indexes answer digits as in Equation 1, with c_1 the ones digit and c_7 the final carry digit; it is not the raw sequence position. Let p_i denote the sequence position of answer digit c_i. In AR, the logits at position $p_i - 1$ predict c_i. In MDM, when c_i is masked, the logits at position p_i predict c_i.

Following the notation in Section 2.1, we use $\mathrm{Rep}_i \in \mathbb{R}^{d_{\mathrm{model}}}$ as shorthand for the post-attention representation used to predict answer digit c_i:

$$\mathrm{Rep}_i = \begin{cases} \tilde{h}^{(1)}_{p_i - 1}, & \text{AR},\\ \tilde{h}^{(1)}_{p_i}, & \text{MDM}. \end{cases}$$

Thus, a post-attention representation refers to any vector $\tilde{h}^{(1)}_j$ at sequence position j, while Rep_i selects the one associated with predicting answer digit c_i.

3.4. OOD Carry-Generalization Evaluation

We evaluate carry generalization by grouping test examples according to the carry-complexity terms defined in Section 2.3: the number of carries N_{carry}. Because both training regimes include only examples with $N_{\mathrm{carry}} \leq 2$, examples with $N_{\mathrm{carry}} > 2$ are out of distribution with respect to the training support.

This evaluation tests whether the model learns a rule for carry propagation that extrapolates beyond the carry numbers observed during training. To produce the correct answer when carries occur, the model must represent both the local base-addition value and the relevant incoming carry. Larger N_{carry} make this dependency less local, since a multiple digit can depend on carry information propagated from lower-order positions.

Test samples are sampled separately from training samples.

4. `MDM-random` and AR Differ in Carry-Constrained OOD Generalization

Figure 2 reports accuracy as a function of N_{carry}. All models maintain high accuracy on examples with $N_{\mathrm{carry}} \leq 2$, but their performance separates sharply in the OOD region $N_{\mathrm{carry}} > 2$. Under the fixed C2 training set, `MDM-random` is consistently more accurate than AR, and the gap widens

as the number of carries increases. At $N_{\text{carry}} = 6$, MDM-random+C2 is roughly 0.16 more accurate than AR+C2. Resampling the C2 training distribution improves both model classes and largely closes the AR–MDM gap. At $N_{\text{carry}} = 6$, the C2-Resampled models are roughly 0.15 more accurate than MDM-random+C2. These results suggest that the main contrast is not only between AR and MDM, but also between fixed-data AR and representations induced either by MDM training or by data resampling.

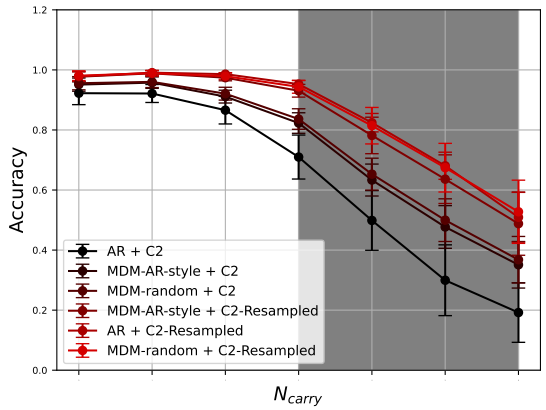

Figure 2. Accuracy grouped by N_{carry} for AR and random-order MDM decoding, averaged across 30 seeds. Legend is in increasing accuracy order at $N_{carry} = 6$. Training uses either a fixed C2 set or epoch-wise C2-Resampled samples; examples with $N_{\text{carry}} > 2$ are out of distribution with respect to the training support.

4.1. Failure mode analysis

We classify failures into two categories: "carry-state" and "non-carry". A carry-state failure occurs when the model makes exactly the error expected from getting the carry state wrong while preserving the underlying base-addition result. This includes two cases: when a carry is present in the ground truth but the model outputs the result without adding it($\kappa_2 = 1 \to \kappa_2 = 0$ failure), and when no carry is present but the model outputs the result as if a carry had been added($\kappa_2 = 0 \to \kappa_2 = 1$ failure). Equivalently, let $\hat{c}_2$ denote the predicted digit and let c_2 denote the true digit. If no carry is present in the ground truth, then a carry-state failure satisfies $\hat{c}_2 \equiv c_2 + 1 \pmod{10}$; if a carry is present, it satisfies $\hat{c}_2 \equiv c_2 - 1 \pmod{10}$. We detect carry-state failures using these conditions, and classify all remaining failures as non-carry.

We conduct this analysis on c_2 using 5,000 randomly generated samples. For MDM, we first apply the forward process and then analyze only those cases in which c_2 is masked.

Table 1 reports, for each model, the total number of failures over 5,000 sampled cases and the proportion of those failures that are carry-state failures. Across all models, most failures are carry-state failures. As accuracy improves, the total number of failures decreases substantially, while the dominant failure type remains the same. Under the fixed C2 regime, AR and MDM exhibit the same dominant failure type, but MDM makes substantially fewer such failures.

5. Shared Roles of Attention and MLP in MDM and AR

The results in Section 3.4 show that AR and MDM differ in OOD carry generalization, but do not by themselves identify where this difference arises within the Transformer. One possibility is that attention and the MLP play different representational roles in AR and MDM when predicting answer digits. Another is that these component roles are similar across model classes, but differ in how robustly the relevant information is represented. In this section, we test these possibilities by analyzing representations after attention and the MLP.

5.1. Post-Attention Representation Geometry

To examine the role of attention, we analyzed the geometry of the post-attention representation (Rep_i). Specifically, we passed the training samples used in C2 through the model and sampled Rep_i at c_2. For the analysis, we considered four grouping criteria: Pair_i, Base_i, κ_i, and the answer digit c_i.

Let group i contain vectors $\{v_{i,j}\}_{j=1}^{G_i}$, with center

$$\mu_i = \frac{1}{G_i}\sum_{j=1}^{G_i} v_{i,j}, \qquad \mu = \frac{1}{G}\sum_{i=1}^{G} \mu_i,$$

where μ is the mean of the group centers. We then measure group-center similarity and within-group compactness by

$$S_{\text{center}} = \frac{1}{G}\sum_{i=1}^{G} \cos(\mu_i, \mu),$$
$$S_{\text{within}} = \frac{1}{G}\sum_{i=1}^{G} \frac{1}{G_i}\sum_{j=1}^{G_i} \cos(v_{i,j}, \mu_i).$$

If the residual representations are well separated by a grouping criterion, then S_{center} should be low while S_{within} should be high.

Table 2 reports both S_{center} and S_{within} for AR+C2. Among the four grouping criteria, κ_i and Pair_i yield lowest group-center similarity, whereas Base_i and c_i do not. However, only Pair_i also yields highest within-group compactness. Taken together, these results suggest that the post-attention

Table 1. Failure statistics at c_2 over 5,000 uniformly sampled examples (averaged across 30 seeds). The third column reports the proportion of carry-state failures among all failures.

MODEL TYPE	FAILURES (/5000)	CARRY-STATE FAILURE(%)	$\kappa_2 = 1 \to \kappa_2 = 0$ FAILURE(%)	$\kappa_2 = 0 \to \kappa_2 = 1$ FAILURE(%)
AR + C2	342.9	97.85%	94.3%	3.55%
AR + C2-RESAMPLED	0.33	100%	100%	0.00%
MDM + C2	137.1	98.90%	97.7%	1.20%
MDM + C2-RESAMPLED	1.33	99.99%	98.6%	1.39%

Table 2. Group-center similarity and within-group compactness for post-attention representations in AR+C2 (averaged across 30 seeds). Lower S_{center} indicates better separation between group centers, while higher S_{within} indicates tighter clustering within each group.

GROUP TYPE	$S_{\text{center}} \downarrow$	$S_{\text{within}} \uparrow$
κ	0.5859	0.6569
Pair	0.5422	0.8601
Base	0.9604	0.4875
c_i	0.9679	0.4857

Table 3. Average accuracy of answer token predicted from Rep_i and $\text{Rep}_i + m^{(1)}_{\pi(i)}$ using one-vs-rest linear SVM(averaged across 30 seeds)

MODEL TYPE	ACCURACY Rep_i	ACCURACY $\text{Rep}_i + m^{(1)}_{\pi(i)}$
AR + C2	0.275	0.999
AR + C2-RESAMPLED	0.272	0.999
MDM + C2	0.218	0.999
MDM + C2-RESAMPLED	0.211	0.999

representations are organized primarily by Pair_i, rather than directly by carry state or answer digit. Pair_i dominance were observed across all other models, and these results are provided in the Appendix B. Repeated dominance of Pair_i across model classes suggests that attention plays a similar representational role: grouping by Pair_i in AR and MDM.

5.2. Linear Decodability of the Answer Digit Before and After the MLP

In 5.1, we observed that the post-attention representations Rep_i are organized primarily by Pair_i. This suggests that the attention block already structures the relevant base-addition information, but does not yet tell us whether the correct answer digit is directly readable from that representation. We next ask what the MLP contributes on top of Rep_i, and whether this contribution is similar across model classes.

To examine this, we test whether the correct answer digit is linearly decodable before and after the MLP. For each example in the training dataset, we extract the post-attention representation Rep_i and the corresponding post-MLP representation $\text{Rep}_i + m^{(1)}_{\pi(i)}$, where $m^{(1)}_{\pi(i)}$ denotes the MLP update at the same sequence position used to predict answer digit c_i. We then train a one-vs-rest linear SVM to predict the answer digit from each representation and evaluate it using 5-fold cross validation.

The results are shown in Table 3. Across all model and data settings, the answer digit is only weakly decodable from Rep_i, with accuracies around 0.2–0.3. In contrast, after the MLP the answer becomes almost perfectly linearly decodable in every case. This suggests that the MLP transforms the post-attention representation into a form directly aligned with answer prediction. Because the same before/after pattern appears across all models, the role of the MLP also appears similar in AR and MDM.

6. Linear Alignment and Generalization

The previous analyses suggest a more specific source of the OOD generalization gap. First, failures are dominated by carry-state errors, especially $\kappa_i = 1 \to \kappa_i = 0$ failures. Second, across model classes, attention organizes post-attention representations primarily by Pair_i. Because each Pair_i group contains both $\kappa_i = 0$ and $\kappa_i = 1$ examples, we hypothesize that carry-level failures arise from an insufficient separation of the $\kappa_i = 0$ and $\kappa_i = 1$ subgroups within each Pair_i group.

We therefore introduce *linear alignment*, a metric that measures how strongly the representation of a target feature is aligned with a single direction v. In this work, linear alignment is used to quantify how well the carry feature is encoded inside each Pair_i group. We define this metric formally in Definition 6.1.

Definition 6.1. Linear Alignment Let $x, v \in \mathbb{R}^d$, where $x \sim \mathcal{D}$, and let $\Sigma = \text{Cov}(\mathcal{D})$. We define linear alignment(P) as

$$P = \frac{\text{Var}(v^\top x)}{\text{tr}(\Sigma)}.$$

6.1. Mathematical Analysis of Linear Alignment on Gaussian Distribution

her linear alignment leads to robust boundary. Specifically with gaussian distribution condition we can derive accuracy monotonically increases with linear alignment given fixed boundary pertubation size.

Theorem 6.2. *(Higher linear alignment may indicate more robustness to orthogonal noise in decision boundary) Let $x \in \mathbb{R}^d$ be a random vector drawn from a multivariate normal distribution $\mathcal{N}(0, \Sigma)$. Consider a ground truth linear classifier defined by $e^T x \geq 0$, $||e||_2 = 1$ and a learned linear classifier defined by $a^T x \geq 0$. Suppose the covariance matrix Σ is a diagonal matrix of the form:*

$$\Sigma = \text{diag}\left(P, \frac{1-P}{d-1}, \ldots, \frac{1-P}{d-1}\right) \tag{2}$$

where $\text{tr}(\Sigma) = 1$ and $e^T \Sigma e = P$. If the learned weight vector a is decomposed as $a = e + e_\perp$, where $e^T e_\perp = 0$, then the probability of disagreement (error rate) between the learned classifier and the ground truth is given by:

$$\textit{Error Rate} = \frac{1}{\pi} \cos^{-1}\left(\frac{1}{\sqrt{1 + \frac{1-P}{P(d-1)}\|e_\perp\|^2}}\right) \tag{3}$$

Proof. First, we determine the general expression for the error rate. We can transform the data into an isotropic Gaussian space. Let $\Sigma = LL^T$ be the Cholesky decomposition of the covariance matrix. By applying the change of variables $z = L^{-1}x$ (or equivalently $x = Lz$), we have $z \sim \mathcal{N}(0, I)$ with $\mathbb{E}[zz^T] = I$.

In this transformed space, the decision boundaries are equivalently expressed as:

$$e^T x \geq 0 \iff e^T L z \geq 0$$
$$a^T x \geq 0 \iff a^T L z \geq 0$$

The error rate is the probability that the two classifiers disagree, which is proportional to the angle between their normal vectors, $L^T e$ and $L^T a$, in the isotropic space:

$$\text{Error Rate} = \frac{1}{\pi} \cos^{-1}\left(\frac{(L^T a)^T (L^T e)}{\|L^T a\| \|L^T e\|}\right) \tag{4}$$

Expanding the inner product and norms, and substituting $LL^T = \Sigma$, we obtain the general error rate formula:

$$\text{Error Rate} = \frac{1}{\pi} \cos^{-1}\left(\frac{a^T \Sigma e}{\sqrt{(a^T \Sigma a)(e^T \Sigma e)}}\right) \tag{5}$$

Next, we apply the specific assumptions of the theorem. We are given $e^T \Sigma e = P$ and $a = e + e_\perp$ with $e^T e_\perp = 0$. We evaluate the terms $a^T \Sigma e$ and $a^T \Sigma a$:

$$a^T \Sigma e = (e + e_\perp)^T \Sigma e = e^T \Sigma e + e_\perp^T \Sigma e = P + 0 = P \tag{6}$$

For the variance of the learned model, we have:

$$a^T \Sigma a = (e + e_\perp)^T \Sigma (e + e_\perp) = e^T \Sigma e + e_\perp^T \Sigma e_\perp \tag{7}$$

Since Σ is diagonal with its principal component variance as P (corresponding to the direction of e) and the remaining $d-1$ dimensions having variance $\frac{1-P}{d-1}$, and since $e_\perp$ lies entirely in the orthogonal subspace, it follows that $e_\perp^T \Sigma e_\perp = \frac{1-P}{d-1}\|e_\perp\|^2$. Thus:

$$a^T \Sigma a = P + \frac{1-P}{d-1}\|e_\perp\|^2 \tag{8}$$

Finally, substituting these components back into Equation 5 yields:

$$\begin{aligned}
\text{Error Rate} &= \frac{1}{\pi} \cos^{-1}\left(\frac{P}{\sqrt{\left(P + \frac{1-P}{d-1}\|e_\perp\|^2\right) P}}\right) \\
&= \frac{1}{\pi} \cos^{-1}\left(\frac{P}{P\sqrt{1 + \frac{1-P}{P(d-1)}\|e_\perp\|^2}}\right) \\
&= \frac{1}{\pi} \cos^{-1}\left(\frac{1}{\sqrt{1 + \frac{1-P}{P(d-1)}\|e_\perp\|^2}}\right)
\end{aligned}$$

□

6.2. Linear Alignment and Generalization Mode

For experiment we sampled Rep_2, the post-attention representation used to predict c_2, from across train samples and grouped by Pair_2. Also within the group we considered two subgroups each satisfying $\kappa_2 = 0$ and $\kappa_2 = 1$. We then calculated PC_1 with principal component analysis. For reliability of the result we skipped Pair_2 groups which has subgroup number of samples smaller than 3.

First we measured cosine of PC_1 and center difference between carry 1 and carry 0 subgroup. Table 4 shows cosine value is around $0.99 \sim 1.0$ which indicates carry information is encoded in PC_1.

We then measured explained variance ratio of PC_1 which is equivalent to linear alignment by definition. Table 5 shows C2-Resampled models have higher linear alignment. Which helps generalization as in Theorem 6.2. And MDM+C2's linear alignment is significantly higher than AR+C2's. This representation analysis is measured before decoding, so `MDM-random` and `MDM-AR-style` share the same value. This indicates MDM's training induces a similar internal mechanism to C2-Resampled, and it leads to higher accuracy.

Table 4. Cosine of PC1 and center difference between carry 1 subgroup and carry 0 subgroup(averaged across 30 seeds).

MODEL TYPE	COSINE
AR + C2	0.993 ± 0.005
AR + C2-RESAMPLED	0.999 ± 0.0008
MDM + C2	0.997 ± 0.003
MDM + C2-RESAMPLED	0.998 ± 0.004

Table 5. Linear alignment at each model(averaged across 30 seeds).

MODEL TYPE	LINEAR ALIGNMENT
AR + C2	0.674 ± 0.076
AR + C2-RESAMPLED	0.916 ± 0.03
MDM + C2	0.79 ± 0.06
MDM + C2-RESAMPLED	0.90 ± 0.1

6.3. MLP Retraining on Base Models

We freezed positional encoding and attention and retrained base model at Section 3.4 additional 5000 epochs. All of the models were trained with C2-Resampled during retraining. This isolates effect of linear alignment and ensures OOD accuracy gap measured at Section 3.4 isn't due to under trained MLP.

We note retrained model with + retrain. For example, MDM+C2+retrain refers to the MDM model trained additional 5000 epochs in retrain setting with freezing on a base model trained with C2.

Figure 3 is retrained accuracy of the model with random-order MDM decoding, grouped by N_{carry}. Accuracy difference exists and relationship is same to model without retraining. This matches the result of linear alignment at Section 6.2.

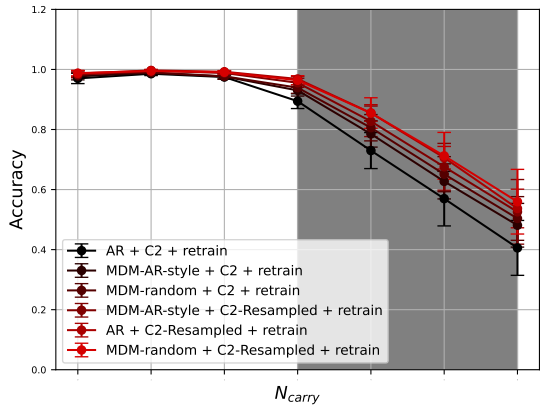

Figure 3. Accuracy by carry number N_{carry} with random-order MDM decoding after MLP retraining(averaged across 30 seeds). Legend is in increasing accuracy order. Accuracy order is `MDM-random` + C2-Resampled $>$ AR + C2-Resampled $>$ `MDM-AR-style` + C2-Resampled $>$ `MDM-random` + C2 $>$ `MDM-AR-style` + C2 $>$ AR + C2, and consistent across N_{carry}.

7. Conclusion

Masked diffusion models have recently emerged as an important alternative to autoregressive models, and prior work suggests that they can perform better in data-constrained regimes. However, it remains unclear whether this advantage comes from learning a qualitatively different mechanism.

In this work, we studied this question in a controlled addition setting. We trained AR and MDM models under two carry-constrained regimes: C2, where models are trained on a fixed set of examples with limited carry complexity, and C2-Resampled, where the dataset is resampled under the same carry constraint. Through representation-based analyses, we found that the roles of attention and the MLP are broadly similar across model types: post-attention representations primarily organize digit-pair information, while the MLP makes answer tokens more linearly decodable.

The main difference appears in the geometry of carry representations. C2-Resampled induces stronger linear alignment along the carry direction, which is associated with improved OOD carry generalization. Notably, MDMs trained under C2 also exhibit stronger linear alignment than AR models trained under the same fixed-data regime. These results suggest that, in this setting, the advantage of MDM training does not arise from a qualitatively different layer-level algorithm, but from inducing a resampling-like internal representation that better aligns carry information for downstream prediction.

Impact Statement

This paper presents a controlled study of how different training objectives shape internal representations in neural networks. The goal of this work is to advance scientific understanding of model behavior in data-constrained reasoning settings. Since our experiments are conducted on a synthetic addition task with small Transformer models, we do not anticipate direct societal harms from the specific methods or results presented here. More broadly, improved understanding of neural network representations may support the development of more reliable and interpretable AI systems, while also contributing to general advances in machine learning capability.

References

Arriola, M., Gokaslan, A., Chiu, J. T., Yang, Z., Qi, Z., Han, J., Sahoo, S. S., and Kuleshov, V. Block diffusion: Interpolating between autoregressive and diffusion language models. *arXiv preprint arXiv:2503.09573*, 2025.

Austin, J., Johnson, D. D., Ho, J., Tarlow, D., and Van Den Berg, R. Structured denoising diffusion models in discrete state-spaces. In *Advances in Neural Information Processing Systems*, volume 34, pp. 17981–17993, 2021.

Feng, G., Geng, Y., Guan, J., Wu, W., Wang, L., and He, D. Theoretical benefit and limitation of diffusion language model. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=fGBCRZQVse.

Fraij, A. and Dauncey, S. Double descent as a lens for sample efficiency in autoregressive vs. discrete diffusion models. *arXiv:2509.24974*, 2025.

Gulrajani, I. and Hashimoto, T. B. Likelihood-based diffusion language models. In *Advances in Neural Information Processing Systems*, volume 36, pp. 16693–16715, 2023.

Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, volume 33, pp. 6840–6851, 2020.

Kantamneni, S. and Tegmark, M. Language models use trigonometry to do addition. *arXiv:2502.00873*, 2025.

Kim, J., Shah, K., Kontonis, V., Kakade, S., and Chen, S. Train for the worst, plan for the best: Understanding token ordering in masked diffusions. *arXiv preprint arXiv:2502.06768*, 2025.

Li, H., Chen, X., XU, Z., Li, D., Hu, N., Teng, F., Li, Y., Qiu, L., Zhang, C. J., Li, Q., and Chen, L. Exposing numeracy gaps: A benchmark to evaluate fundamental numerical abilities in large language models. *arXiv:2502.11075*, 2025.

Li, X., Thickstun, J., Gulrajani, I., Liang, P. S., and Hashimoto, T. B. Diffusion-lm improves controllable text generation. In *Advances in Neural Information Processing Systems*, volume 35, pp. 4328–4343, 2022.

Ni, J., Liu, Q., Dou, L., Du, C., Wang, Z., Yan, H., Pang, T., and Shieh, M. Q. Diffusion language models are super data learners. *arXiv:2511.03276*, 2025.

Nie, S., Zhu, F., You, Z., Zhang, X., Ou, J., Hu, J., Zhou, J., Lin, Y., Wen, J.-R., and Li, C. Large language diffusion models. *arXiv preprint arXiv:2502.09992*, 2025.

Prabhudesai, M., Wu, M., Zadeh, A., Fragkiadaki, K., and Pathak, D. Diffusion beats autoregressive in data-constrained settings. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026. URL https://openreview.net/forum?id=W5Ht05jF4c.

Quirke, P. and Barez, F. Understanding addition in transformers. *arXiv:2310.13121*, 2023.

Swerdlow, A., Prabhudesai, M., Gandhi, S., Pathak, D., and Fragkiadaki, K. Unified multimodal discrete diffusion. *arXiv preprint arXiv:2503.20853*, 2025.

A. Implementation Details

Sequence format. Every example is represented as a fixed-length token sequence. The operands are six digits each, so the arithmetic result requires at most seven digits. The implementation stores the answer in eight positions by prepending one deterministic leading zero. We use <bos> at the beginning, <eoi> to separate the prompt from the answer, and <eos> after the answer. Digits, +, =, and special tokens are tokenized as individual tokens. For example,

<bos>123456+123456=<eoi>00246912<eos>.

Because the answer length is fixed, <eos> is not needed to infer the end of the answer; it is included only as a terminal token in the implemented sequence format.

AR training and decoding. AR models are trained with next-token cross-entropy under teacher forcing. At inference time, answer tokens are generated left-to-right from the prompt ending at <eoi>. The terminal <eos> token is generated after the fixed-length answer suffix.

MDM training. MDMs are trained with a random masking objective on the answer suffix. A random subset of answer positions is replaced by a mask token, and cross-entropy is computed only on the masked positions. This is the same training objective for all MDM checkpoints used in the paper.

MDM decoding. At inference time, MDM generation starts from a fully masked answer suffix and iteratively fills answer positions. In MDM-random, the remaining masked positions are filled in a random order. In MDM-AR-style, the same trained MDM checkpoint is decoded left-to-right over answer positions. Thus, MDM-random and MDM-AR-style differ only in decoding schedule, not in training or architecture. Confidence-based greedy decoding is another possible MDM decoding strategy, but it is not the schedule used for the main reported MDM-random results.

Model Configuration. Transformerlens library was used for experiment. Model runs on A100-SXM4-40GB GPU. The key parameters are $n\_layers = 1$ $n\_heads = 2$ $d\_model = 256$ $d\_mlp = 512$.

B. Additional Results of Section 5

Table 6. Similarity between group centers for post-attention representations. The metric is defined as $\frac{1}{G}\sum_{i=1}^{G}\cos(\mu_i, \mu)$. Lower values indicate more separated group centers.

Group type	AR + C2	AR + C2-Resampled	MDM + C2	MDM + C2-Resampled
κ	0.5859	0.8765	0.6110	0.7864
Pair	0.5422	0.5153	0.5413	0.5359
Base	0.9604	0.9609	0.9537	0.9576
c_i	0.9679	0.9667	0.9639	0.9658

Table 7. Within-group compactness for post-attention representations. The metric is defined as $\frac{1}{G}\sum_{i=1}^{G}\frac{1}{G_i}\sum_{j=1}^{G_i}\cos(v_{i,j}, \mu_i)$. Higher values indicate tighter within-group clustering.

Group type	AR + C2	AR + C2-Resampled	MDM + C2	MDM + C2-Resampled
κ	0.6569	0.5621	0.6591	0.6117
Pair	0.8601	0.9401	0.8820	0.9208
Base	0.4875	0.5230	0.5028	0.5302
c_i	0.4857	0.5202	0.4998	0.5263