Diverse capability and scaling of diffusion and auto-regressive models when learning abstract rules

Binxu Wang
Kempner Institute, Harvard University
binxu_wang@hms.harvard.edu

Jiaqi Shang
Program in Neuroscience, Harvard Medical School
jiaqishang@g.harvard.edu

Haim Sompolinsky
Center for Brain Science, Harvard University
Edmond and Lily Safra Center for Brain Sciences, Hebrew University
hsompolinsky@mcb.harvard.edu

Abstract

Humans excel at discovering regular structures from limited samples and applying inferred rules to novel settings. We investigate whether modern generative AI systems can similarly learn underlying rules from finite samples and perform reasoning through conditional sampling. Inspired by Raven's Progressive Matrices task, we designed GenRAVEN dataset, where each sample consists of three rows, and one of the 40 relational rules governing the object position, number, or attributes applies to all three rows. We trained generative models to learn the data distribution, where samples are encoded as 3×9×9 integer arrays to focus on rule learning. We compared two major families of generative models: diffusion models (EDM, DiT, SiT) and autoregressive models (GPT2, Mamba). We evaluated their ability to generate structurally consistent samples and perform panel completion via unconditional and conditional sampling. We found that diffusion models excel at unconditional generation, producing novel and more consistent samples from scratch and memorize less, but perform less well in panel completion, even with advanced conditional sampling methods like Twisted Diffusion Sampler. Conversely, autoregressive models excel at completing missing panels in a rule-consistent manner but generate less consistent samples unconditionally. We observe diverse data scaling behaviors: for both model families, rule learning emerges at a certain dataset size – around thousands examples per rule. With more training data, diffusion models improve both their unconditional and conditional generation capabilities. However, for autoregressive models, while panel completion improves with more training data, unconditional generation consistency declines. Our findings highlight complementary capabilities and limitations of diffusion and autoregressive models in rule learning and reasoning tasks, suggesting avenues for further research into their mechanisms and potential for human-like reasoning.

1 Background

Human excels at discovering regular structure from a small number of samples, and they can further apply such rule to novel settings to generate new samples or complete missing parts based on the same rule. The Raven's progressive matrix (RPM) [25] is a famous task in human reasoning literature. In the generative version of this task (GenRAVEN), the subject observes two complete rows of panels and is tasked to complete the third row in a manner that is consistent with the first two rows (Fig.1A).

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

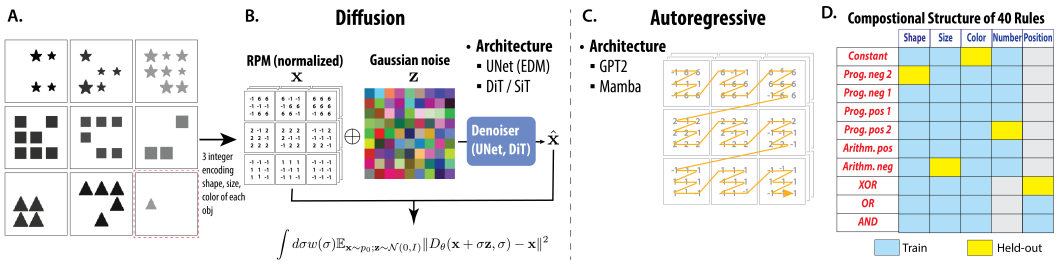

Figure 1: **Design of the study A.** Example Raven's progression matrix, and its encoding as 3x9x9 integer array. The underlying rule is constant shape. **B.C.** Two families of generative models: Diffusion and autoregressive model, and their training method: denoising and predicting the next token. **D.** The 40 relational rules, with 5 rules held out during training.

Ideally, the subjects need to infer the underlying rule consistent with the first two rows and apply it correctly to the third row. How can we train a general learning system to solve such reasoning task?

If we conceptualize all rule-conforming samples as a joint distribution, then rule learning can be framed as a generative modeling problem or learning the correct joint. Further, reasoning about the missing panel can be framed as sampling from the conditional probability [23]. One conceptual problem is, that given finite training samples, the rule governing them, or the 'true' joint distribution is under-specified. The rules or the distribution learned by the system should be affected by its inductive bias, be it human or AI. Given this ambiguity, we asked whether modern generative AI systems could learn the correct "joint distribution" given finite samples. If so, can they reason and fill in the missing parts in a sample through conditional sampling?

In the current age of Generative AI, there are two prominent families of generative models: autoregressive models and diffusion models. The autoregressive model generally dominates discrete sequence data, such as language, music, genomics [4, 14, 13], while the diffusion model excels at continuous data, such as image, video, audio, molecule structure, robot trajectories [26, 32, 27, 5, 3]. Both of them are capable of unconditional generation and conditional sampling based on partial observation: e.g. prompting for autoregressive model or inpainting for diffusion models. Given their similar capability, it's interesting to compare them back-to-back on the same reasoning task and see how their different modeling method might lead to different learning and scaling behaviors.

In this work, we trained a diverse set of generative models from both the diffusion family (EDM, DiT, SiT) and autoregressive family (GPT, Mamba) to learn the data distribution; and then use conditional sampling techniques to perform inference i.e. reasoning about missing panel. We studied their performance as a function of data scale and model scale. We found that generally, diffusion models excel at unconditional sampling from the joint, creating structurally consistent samples from scratch, but perform less well for sampling conditioned on given panels. Conversely, autoregressive models excel at completing missing panels in a rule-consistent manner, but they perform less well in generating consistent samples from scratch with unconditional sampling. They also exhibit diverse scaling behavior when the size of datasets is varied. Our results call for further investigation into the seemingly complementary capabilities and limitations of diffusion and autoregressive models.

2 Method

GenRAVEN Dataset We introduce the GenRAVEN dataset, comprising RPMs associated with 40 relational rules. Each RPM features a 3×3 matrix of panels, where the three panels in each row follow a unique relational rule. To focus on rule learning and reasoning, we abstracted away the visual aspect of the sample, encoding each object with three integers representing its shape, size, and color, with $(-1, -1, -1)$ representing empty positions. Each attribute has a discrete set of allowed values, 0-6 for shape and 0-9 for size and color. Taking together, each sample RPM is represented by an integer array of shape $3 \times 9 \times 9$, which is the target of generative modeling (Fig. 1B).

Each rule is composed of an abstract relation (constant; progression ±2, ±1; arithmetic ±; XOR; OR; AND) applied to an attribute (shape, size, color, number, position). The dataset is designed so that the rule governing each row remains ambiguous when examining only the first two panels and only

becomes evident when all three panels are considered. This design ensures that the rule governing the row cannot be directly deduced from the first two panels alone and the model must reason the entire matrix. We generate 4000 random samples per rule for training. To study the generalization of abstract relations such as constant to new attributes, we held out 5 rules (Fig 1D) during training.

Diffusion Model The diffusion models have been a prominent approach for generative modeling [9] in continuous domains. It takes a holistic approach of modeling: it learns a vector field in the data space $s_\theta(x, \sigma)$, which approximates the gradient of smoothed data distribution. When states flow along the vector field, they will be transported from a base distribution e.g. Gaussian into the target distribution $p(X)$. During sampling, samples initialized from Gaussian will be transformed via this dynamic process into generated samples.

Given the spatial structure of the task, we treated each RPM as a 9×9 image with 3 attribute channels and adapted existing diffusion models for image generation. Specifically, we experimented with two network architectures, UNet [15] and Diffusion Transformer (DiT) [22]. UNet is a Convolutional Neural Network (CNN) based backbone, designed to extract information at multiple resolutions, with self-attention modules at each resolution. For a long time, it has been the default backbone for diffusion models for images [12, 8, 26]. For our task, we adapted its architecture to match the 3x3 panel structure of the RPM samples: we changed the filter size to 3 and the downsampling ratio to 3. DiT is a recent transformer-based backbone adapted for diffusion modelling [22]. For our task, we treated each object in RPM as a token with three features (patch size = 1), totaling 81 tokens. We also tested SiT model [21], which had the same network architecture as DiT, but based their training and inference on the theoretical framework of Stochastic Interpolant [1].

We treated the integer attributes in RPM samples as continuous values and normalized them by mean and std before training. During sampling, since diffusion models generate samples with continuous values, we rounded the generated attribute value to the closest integer. We used deterministic samplers to generate samples: Heun's 2nd order sampler with 18 steps for UNet model [15]; DDIM sampler with 100 steps for DiT [28]; off-the-shelf Runge-Kutta sampler of order 5 (`dpori5`) for SiT [21]. Leveraging the in-painting capability of diffusion models [20, 30], we also challenged them with the RPM inference tasks: given the first 8 panels, let the model sample the missing panel.

Autoregressive models Another major family of generative models is the autoregressive models, which have demonstrated impressive capabilities to model complex sequence data like natural language and music [4, 13]. These models have also been applied to model natural images, as sequences of patches and quantized tokens [10, 6]. These models decompose the joint distribution into the conditional probability of each variable over all previous variables $p(X) = p(x_1)p(x_2|x_1)p(x_3|x_{1:2})...p(x_d|x_{1:d-1})$. During inference, it samples variables autoregressively as a sequence. Here we used auto-regressive models **GPT2** and **Mamba** [24, 11] to solve RAVEN's task, learning the sequence of objects in an RPM. Similar to DiT, we treated each object as a token. The three integer attributes of each token were embedded through separated embedding layers with 1/3 hidden dimensions and then concatenated as the token embedding. To reflect such latent space structure, the hidden state outputs from the transformer or Mamba were chunked into 3 parts and decoded separately into attributes of the next token. We used the default positional encoding for GPT2 and no position encoding for Mamba. We train these models with the next token prediction objective. A key design decision is an order to sequentialize RPM data into 1d. Here we first scan the objects within each panel and then follow the raster order of panels (panel1 in row 1, P2 row1, ...P3 row 3), forming a sequence of 81 tokens (see Fig.1C). We prepend the same starting token `[SOS]` for each sample. For unconditional generation, we started from `[SOS]` and autoregressively sampled 81 tokens with a temperature of 1.0. For panel completion, we started from `[SOS]` and 72 existing tokens (corresponding to the first 8 panels), and sampled 9 tokens with temperature 1.0.

Evaluation One benefit of having a distribution defined by explicit rule is that, we can evaluate the generated samples more precisely. For unconditional generation, there is no sense of accuracy, so we evaluated the internal consistency of generated samples. For each row in a generated sample, we inferred the set of applicable rules. If any rule from the 40 rule set applies, the row is called *valid*. We calculated the fraction of valid rows throughout training. For each sample, we examined whether the same rule applies to two or three rows in the sample, which we call the two or three-row consistent fraction (*C2, C3*).

For conditional generation, we selected 50 new samples per rule, unseen during training, as our test cases. We removed their final panel, and let the model generate the missing panel given the other 8 panels. We evaluated whether the completed panel allowed the same rule to apply to all three rows (C3). The accuracy of panel completion was quantified via the frequency of C3 completion.

3 Results

3.1 Diffusion models learn to generate structurally consistent samples better

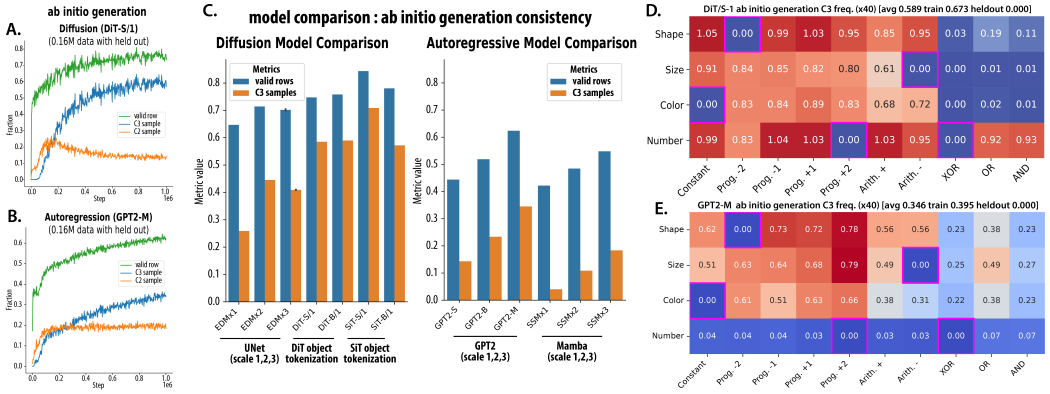

Figure 2: **Diffusion models lead in generating structurally consistent samples *ab initio* A.B.** Dynamics of sample consistency per valid row fraction and C2,C3 fraction during training, for diffusion model (**A.** DiT-S/1) and autoregressive model (**B.** GPT2) **C.** Comparison of *ab initio* generation consistency across model families. **D.E.** Frequency of generating C3 samples of each rule (showing the value $\times 40$ to normalize) for DiT-S and GPT2-M. Magenta frames showing the 5 rules held out from generative model training.

Diffusion models excel in *ab initio* generation We found that, through generative modeling, both diffusion and autoregressive models learn to generate structurally consistent samples measured via valid row fraction and C3 sample fraction (Fig. 2A,B). Comparing across model families (Fig. 2C), SiT models perform better than DiT, and EDM models, which generally perform better than GPT and Mamba models. Notably, the best-performing diffusion model SiT-S/1 had 70.9% of generated samples that were C3 rule consistent; in contrast, the best-performing auto-regressive model (GPT2-M) only had 34.6% C3 samples while doubling the layer number and hidden space dimension of SiT-S. Within each model family, we found scaling up model size helps only for lower performing families e.g. EDM, GPT, and Mamba, but for DiT and SiT models, it didn't help or even hurt performance. The limited effect of model scaling may be related to a fixed training set size, which we will manipulate below (Fig.5). Notably, across model families, SiT-S/B, DiT-S/B, and GPT2-S/B form nice pairs of comparison, since they match in their number of attention heads, layers, hidden dimension, tokenization, and training steps. Given this fair comparison, DiT models have a higher rate of generating structurally consistent samples than GPT2.

Different rule types were challenging for each family of models When separated by the rule types, the two classes of models fell short at different rule types (Fig.2**D,E**). Diffusion models rarely generate consistent samples with logical operation applying to attribute sets (e.g. Shape-XOR, which means when the first panel contains circles and triangles, the second panel contains triangles and squares, then the third panel could only contain circles and squares.) For the held-out rules, the models did not generate C3 samples consistent with those rules after training, though single rows consistent with held-out rules were still generated with some frequency. For Autoregressive models, they failed spectacularly for rules related to number and position, (e.g. Prog.+1-Number, which means 3 objects in the first panel, 4 objects in the 2nd panel, 5 in the 3rd panel; Position-AND, which means 3rd panel can only place objects at locations where objects exist in both 1st and 2nd panel). They had a higher frequency of generating according to logic-attribute rules, compared to diffusion models. Comparatively, their success rate in generating progression and arithmetic rules on attribute values is also lower.

3.2 Diffusion models and autoregressive model display diverse memorization behavior

When models create structurally consistent samples or valid rows, are they creating something completely novel or simply recalling their training set? To answer this, we evaluated the memorization properties of generated samples and their parts: rows, panels, and rows and panels when considering single attributes. We evaluated what fraction of samples and parts have exact copies in the training set for the model. As a control, we also generated another control dataset with 4000 samples per rule unseen during training and computed the memorization fraction towards the control set.

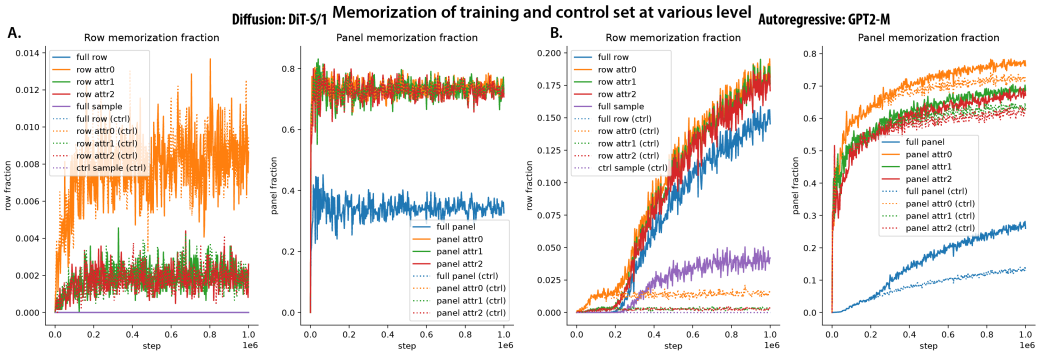

Figure 3: **Diffusion and autoregressive models show diverse data memorization property A. B.** Memorization of training and control set at multiple levels for samples generated through training, for **A.** DiT-S, and **B.** GPT2-M. Solid lines show what fraction of samples, rows, and panels have copies from the training set, and dashed lines show the control, i.e. the fraction of those with copies from the control set of samples unseen during training.

Diffusion models show little memorization on row and sample level Remarkably, within the 800,000 samples generated by the DiT-S model through training, not a single sample or row has exact copy from the training set! Even if we only consider one attribute of a row, only 0.9%, 0.2%, and 0.2% of rows generated by DiT-S model were memorized after training. On the local level, 34.0% of the generated panels were memorized, and their single attributes components have over 70% memorization rate. Intriguingly, the "memorization" on the panel level happened very rapidly during initial training and then stabilized at a fixed level (Fig.3A).

We further found the memorization rates of row and panel attributes are comparable with the control memorization rate, i.e. generated samples overlapping with the control set not used for training. Thus, we can conclude this is "benign" memorization, or approaching the true data distribution. The negligible deviation between the solid and dashed lines (Fig.3A.) highlights diffusion models are learning the true data distribution without over-focusing on the training set. Similar results were observed in other diffusion models. This shows that the diffusion models are capable of generating novel rule consistent rows and samples, without memorizing training rows, demonstrating its "understanding" of the rules is on the abstract level. Further given its memorization of panels, it seems they are capable of recombining memorized local parts (panels, attribute panels) to create novel rows and samples consistent with rules.

Autoregressive models show significantly more memorization with lower consistency In contrast, after training, generation from GPT2-M model had 4.1% of samples, 14.9% of rows and 26.8% of panels that were memorized from the training set. This shows that a significant portion of valid generated rows from GPT2 models are simply regurgitating their training sequences. This compounds the low generation consistency and highlights the "lack of understanding" of abstract rules in autoregressive models (Fig.3B.). Intriguingly, panels are memorized less in GPT2-M than DiT-S. Further, the panel memorization rate of GPT2-M converged to a high level much slower than DiT-S. Comparing to the control memorization rate, we can see the memorization rate with training and control set are initially aligned and both increasing, before they diverge around 0.25M steps, showing the model starts to over-memorize the specific training samples.

This analyses highlights a dichotomy between the memorization behavior in diffusion and autoregressive model: while diffusion models tend to rapidly memorize local panels and attribute combination; autoregressive models default to memorize more global rows and samples. Diffusion models mostly

showed "benign" memorization that is consistent with the statistics of the true distribution; while autoregressive models show significant over-memorization of training set.

3.3 Autoregressive models excel at rule consistent panels completion

Next, we examined their capability of completing missing panels. Similar to unconditional generation, their panel completion accuracy also grows with training in both diffusion and autoregressive models (Fig. 4 **A.B.**). Notably, the learning curve of conditional generation saturates faster than unconditional generation, suggesting making a fully consistent sample from scratch (unconditional sample from joint) may be a harder task than panel completion (sample conditionally).

Inpainting accuracy significantly depends on the conditional sampling method for diffusion model One salient observation is that, for diffusion models, their panel completion accuracy depended critically on the conditional sampling method. Repaint and DDNM [30, 20] are popular methods for general image inpainting or linear inverse problem. They are heuristic methods that basically fix the values of observed pixels and only let the masked pixels diffuse with diffusion. However, these methods do not perform exact conditional sampling [2]. Recently, twisted diffusion sampler (TDS) [33] was shown to be asymptotically exact. TDS maintains a population of particles and re-weights them during sampling, leveraging the twisting technique from Sequential Monte Carlo (SMC) literature. We tested both methods on our task and found TDS performed much better than the Repaint method: on DiT-S/1 model, TDS yielded an overall C3 completion accuracy (across all 40 rules) of 45.2% while Repaint yielded accuracy 26.1% (Fig. 4 C).

This is interesting, since for natural image completion, evaluation is usually done through visual inspection or classification, which is a relatively loose criterion. Then heuristic samplers e.g. Repaint and DDNM seem to suffice. But for the RPM task, where we have access to the precise form of underlying rules, the evaluation of sample completion can be done much more precisely. For such a task, advanced sampling algorithms seem to be required to perform well.

Autoregressive models lead in panel completion, while generalizing less well. Intriguingly, even with the advanced TDS sampler, diffusion models fall short in their panel completion accuracy, falling behind GPT2 models: GPT2-M model had panel completion accuracy of 62.4% (Fig.4C). This is the opposite of unconditional generation. Notably, for held-out rules, diffusion models with or without twisted sampler still show comparable or higher completion accuracy than chance and GPT2 models, showing that diffusion models still exhibit better rule-level generalization.

Panel completion ability correlates with unconditional generation capability Notebaly, within each model, when analyzed per rules, the panel completion accuracy of each rule correlates with the frequency of C3 samples in unconditional generation (Fig.4F.G.): Pearson correlation 0.843 ($p = 8.4 \times 10^{-12}$) for DiT-S/1 and 0.587 ($p = 6.7 \times 10^{-5}$) for GPT2-M. This suggests that a similar mechanism within the models may contribute to both panel completion and unconditional generation capabilities. Comparing between DiT and GPT2, note that the dots generally fall on different sides of the diagonal, where the completion accuracy per rule is generally lower than the unconditional C3 frequency for DiT, but the opposite is true for GPT2.

3.4 Diverse scaling property of diffusion model and autoregressive model

Next, we studied the effect of dataset scale and model scale on the diffusion and autoregressive model.

Rule learning collapses at low data scale for diffusion model For diffusion models, across the three classes, we found that at a small dataset scale (400 samples per rule, or 0.016M total sample), the training "collapsed" and both the unconditional consistency and the panel completion accuracy dropped to a low level (Fig.5A,B). Beyond that scale, scaling up the dataset results in a higher C3 ratio and completion accuracy, with larger models benefiting more from the larger data (Fig.5A). We didn't see much modulation in the learning of heldout rule by dataset scale (Fig.5B). Generally across models and across scales, the unconditional and conditional generation capabilities of diffusion models are correlated, suggesting similar mechanisms behind them.

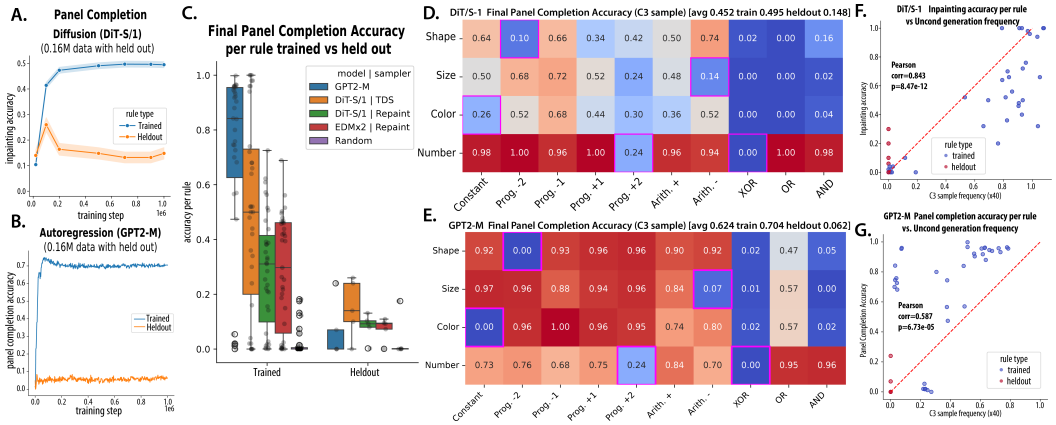

Figure 4: **Autoregressive models lead in rule consistent panel completion A. B.** Learning dynamics of panel completion accuracy for trained and heldout rules, for diffusion model (**A.** DiT-S/1) and autoregressive model (**B.** GPT2) **C.** Comparison of panel completion accuracy across model class and sampler. **D. E.** Panel completion accuracy per rule for DiT-S and GPT2-M after training. **F. G.** Correlation between panel completion accuracy and C3 sample generation frequency (Fig.2D.E.) per rule for DiT-S and GPT2-M after training.

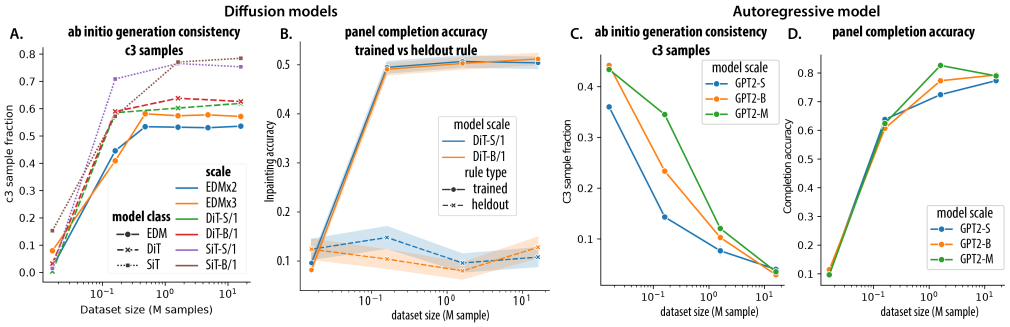

Figure 5: **Diverse scaling behavior of Diffusion and Autoregressive models A.B.** Data scaling curve of Diffusion models (EDM, DiT, SiT) **A.** *ab initio* generation consistency (C3 fraction) and **B.** panel completion accuracy. **C.D.** Analogous data scaling curve of autoregressive model (GPT2)

Divergent data scaling for autoregressive model For autoregressive model, however, as the dataset size grows, we observed contrasting trends between the two tasks. Similar to diffusion models, their panel completion accuracy also collapsed at the same scale (400 samples per rule), and the accuracy grows with the data scale up to a ceiling (Fig. 5**D**). Across model scales, larger models (GPT2-M) benefit more from larger data than smaller ones, similar to diffusion models. Conversely, as for unconditional generation (Fig. 5**C**), *ab initio* sample consistency decreases sharply as a function of dataset size, where smaller models (GPT2-S) decays faster. Thus, although the panel completion accuracy benefits from bigger training datasets; scaling up datasets has a detrimental effect on generating structurally consistent samples esp. for smaller models. Such diverging data scaling properties of conditional and unconditional sampling capability for autoregressive models are worth further investigation.

4 Discussion and Future direction

Weakness of autoregressive models in structurally consistent samples generation from scratch Given the high panel completion accuracy of GPT models, we are surprised by the low structural consistency (valid row, C3) of their samples when we let them generate unconditionally. Why did they struggle to generate samples from scratch? We are reminded of a comment Yann LeCun made on his twitter: even if the conditional probability of each token is slightly off, autoregressive sampling will lead to exponentially compounding errors; further when one error is made during sampling, the model has no way to come back and revise it. In contrast, diffusion models have a holistic view of the sample from the start, and can always change any token on the RPM, which may lead to higher

structural consistency of the samples. It is interesting to further validate this compounding error hypothesis, for the autoregressive model and see how we can rescue it with a better sampling method (e.g. beam search).

Weakness of diffusion models in rule consistent panel completion On the flip side, we are also surprised that the panel completion accuracy of diffusion models is generally lower than the C3 frequency of their unconditional generation. As we may naively think, modeling conditional density should be easier than modeling joint distribution. One potential reason is that, even though unconditional sampling algorithms for diffusion have been studied extensively [16, 34, 17, 19], we still do not have the perfect method for sampling the conditional distribution from diffusion models. Just as TDS performs better than simple heuristic methods e.g. Repaint, a better conditional sampler may boost the reasoning capability of diffusion models further. Our results also provide evidence that for tasks that are evaluated precisely (e.g. abstract rules), a better sampler is critical.

Comparative study of rule learning with humans and animals We showed that modern generative models can learn abstract rules from finite rule-conforming samples. However, their sample complexity and generalization still fall short behind humans. As we showed, when only 400 examples were provided per rule, both diffusion and autoregressive models didn't learn the rule (fail in panel completion and generation). Whereas human subjects can learn these rules with fewer samples, and without observing all the 40 rules. It is interesting to see what inductive biases or pre-training could be done to close the sample complexity and generalization gap between human and current generative models. Given the diverse scaling behavior and learning behavior per rule of two families of models, comparative studies with human subjects could potentially adjudicate between the models and answer which algorithm is behaviorally more similar to humans.

Advanced reasoning techniques in autoregressive model In our study, we focus on autoregressive models in their most basic form. Clearly, we haven't explored more advanced and intricate designs for autoregressive models. Many recent ideas may be interesting to try to improve their performance, for example more informative spatial positional encoding [18, 7]; autoregression across spatial scales or Fourier modes instead of across patches [29]. Further, we haven't explored sampling techniques designed for enhancing reasoning for autoregressive models esp. LLMs, e.g. chain-of-thoughts [31]. We hope our work motivates further investigations into these models and their limitations in reasoning capability.

Acknowledgments and Disclosure of Funding

Many thanks to Hidenori Tanaka, Martin Wattenberg, Michael Albergo, Andy Keller, Ekdeep Singh Lubana, Talia Konkle, George Alvarez, and Sham Kakade for helpful discussions and insightful feedback. We appreciate the Kempner Institute for the Study of Natural and Artificial Intelligence at Harvard University for providing funding, computing resources, and space for this research; we also thank the Swartz Foundation, ONR grant No. N0014-23-1-2051, and a generous gift from Amazon.

References

[1] Michael S Albergo, Nicholas M Boffi, and Eric Vanden-Eijnden. Stochastic interpolants: A unifying framework for flows and diffusions. *arXiv preprint arXiv:2303.08797*, 2023.

[2] Michael S Albergo, Mark Goldstein, Nicholas M Boffi, Rajesh Ranganath, and Eric Vanden-Eijnden. Stochastic interpolants with data-dependent couplings. *arXiv preprint arXiv:2310.03725*, 2023.

[3] Andreas Blattmann, Robin Rombach, Huan Ling, Tim Dockhorn, Seung Wook Kim, Sanja Fidler, and Karsten Kreis. Align your latents: High-resolution video synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22563–22575, 2023.

[4] Tom B Brown. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

[5] Joao Carvalho, An T Le, Mark Baierl, Dorothea Koert, and Jan Peters. Motion planning diffusion: Learning and planning of robot motions with diffusion models. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1916–1923. IEEE, 2023.

[6] Huiwen Chang, Han Zhang, Lu Jiang, Ce Liu, and William T Freeman. Maskgit: Masked generative image transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11315–11325, 2022.

[7] Xiangxiang Chu, Zhi Tian, Yuqing Wang, Bo Zhang, Haibing Ren, Xiaolin Wei, Huaxia Xia, and Chunhua Shen. Twins: Revisiting the design of spatial attention in vision transformers. *Advances in neural information processing systems*, 34:9355–9366, 2021.

[8] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021.

[9] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021.

[10] Patrick Esser, Robin Rombach, and Bjorn Ommer. Taming transformers for high-resolution image synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12873–12883, 2021.

[11] Albert Gu, Tri Dao, et al. Mamba: Linear-time sequence modeling with selective state spaces. In *Proceedings of the COLM Conference*, 2024. Available at OpenReview.

[12] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[13] Cheng-Zhi Anna Huang, Ashish Vaswani, Jakob Uszkoreit, Noam Shazeer, Ian Simon, Curtis Hawthorne, Andrew M Dai, Matthew D Hoffman, Monica Dinculescu, and Douglas Eck. Music transformer. *arXiv preprint arXiv:1809.04281*, 2018.

[14] HyenaDNA et al. Genomic language models: Opportunities and challenges. *arXiv preprint arXiv:2407.11435*, 2023.

[15] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. *Advances in Neural Information Processing Systems*, 35: 26565–26577, 2022.

[16] Tero Karras et al. Elucidating the design space of diffusion-based generative models. In *Advances in Neural Information Processing Systems*, volume 35, pages 26565–26577. Curran Associates, Inc., 2022.

[17] Luping Liu, Yi Ren, Zhijie Lin, and Zhou Zhao. Pseudo numerical methods for diffusion models on manifolds. *arXiv preprint arXiv:2202.09778*, 2022.

[18] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10012–10022, 2021.

[19] Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps. *Advances in Neural Information Processing Systems*, 35:5775–5787, 2022.

[20] Andreas Lugmayr, Martin Danelljan, Andres Romero, Fisher Yu, Radu Timofte, and Luc Van Gool. Repaint: Inpainting using denoising diffusion probabilistic models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11461–11471, 2022.

[21] Nanye Ma, Mark Goldstein, Michael S Albergo, Nicholas M Boffi, Eric Vanden-Eijnden, and Saining Xie. Sit: Exploring flow and diffusion-based generative models with scalable interpolant transformers. *arXiv preprint arXiv:2401.08740*, 2024.

[22] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12345–12355. IEEE, 2023.

[23] Ben Prystawski, Michael Li, and Noah Goodman. Why think step by step? reasoning emerges from the locality of experience. *Advances in Neural Information Processing Systems*, 36, 2024.

[24] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[25] James C Raven. Mental tests used in genetic, the performance of related indiviuals on tests mainly educative and mainly reproductive. *MSC thesisUniv London*, 1936.

[26] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.

[27] Ludan Ruan, Yiyang Ma, Huan Yang, Huiguo He, Bei Liu, Jianlong Fu, Nicholas Jing Yuan, Qin Jin, and Baining Guo. Mm-diffusion: Learning multi-modal diffusion models for joint audio and video generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10219–10228, 2023.

[28] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020.

[29] Keyu Tian, Yi Jiang, Zehuan Yuan, Bingyue Peng, and Liwei Wang. Visual autoregressive modeling: Scalable image generation via next-scale prediction. *arXiv preprint arXiv:2404.02905*, 2024.

[30] Yinhuai Wang, Jiwen Yu, and Jian Zhang. Zero-shot image restoration using denoising diffusion null-space model. *arXiv preprint arXiv:2212.00490*, 2022.

[31] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

[32] Lemeng Wu, Chengyue Gong, Xingchao Liu, Mao Ye, and Qiang Liu. Diffusion-based molecule generation with informative prior bridges. *Advances in Neural Information Processing Systems*, 35:36533–36545, 2022.

[33] Luhuan Wu, Brian Trippe, Christian Naesseth, David Blei, and John P Cunningham. Practical and asymptotically exact conditional sampling in diffusion models. *Advances in Neural Information Processing Systems*, 36, 2024.

[34] Wenliang Zhao, Lujia Bai, Yongming Rao, Jie Zhou, and Jiwen Lu. Unipc: A unified predictor-corrector framework for fast sampling of diffusion models. *Advances in Neural Information Processing Systems*, 36, 2024.

A Detailed Methods

A.1 Dataset construction

We have a procedural algorithm to generate rows following the 40 rules. The basic workflow is the following:

1. Choosing the rule-following dimension (Shape, Size, Color, Number, Position);
2. Choosing the rule type (Constant, Progression, Arithmetic, XOR, OR, AND);
3. Switching based on the type of rules
   - If it's Constant, Progression, Arithmetic of object attributes (shape, size, color), then each panel will only have single value for the attribute i.e. object all have same size in one panel. We will choose three values (X_1, X_2, X_3) that follow the rule (e.g. Progression +2, will be $X_1 = 1, X_2 = 3, X_3 = 5$).
   - If it's XOR, OR, AND of object attributes, then each panel will have a set of attributes, which suffice the logic operation. Decide the number of objects of each attribute value.
4. Choose the other attribute dimensions of the object.
5. Check if the other attribute follow other rules, if so, regenerate the other attributes

We synthesized 1,200,000 rows per rule. Three rows following the same rule will be randomly combined into a Raven's Progression Matrix sample (400,000 samples per rule). From this base dataset, we will subsample 400, 4000, 40000, and 400000, samples to construct training sets for the data scaling experiments. By default, we will use the 4000 samples to train most of the models. These RPMs are encoded as a $3 \times 9 \times 9$ integer matrix. We used $[1.5, 2.5, 2.5]$ and $[2.5, 3.5, 3.5]$ as the mean and std of the three channels, to normalize the RPM encoding tensors.

A.2 Diffusion model and training details

A.2.1 EDM/UNet

Our implementation was based on the original EDM code base [16] and its simplified version `https://github.com/yuanzhi-zhu/mini_edm`.

For model scaling, we tested EDM models with three scales, EDMx1, EDMx2, EDMx3, which double and triple the width and depth of the UNet model. All UNet models have 3 resolution blocks $(9, 3, 1)$, at each resolution level the channel numbers are increased by a multiplier $(1, 2, 4)$. At each resolution, there are self-attention modules. Specifically, `EDMx1`,`EDMx2`,`EDMx3`, used base channel number 64, 128 and 192; while they used 1, 2, and 3 Residual Layer per resolution.

A.2.2 DiT

Our implementation was based on the original DiT code base [22], `https://github.com/facebookresearch/DiT`

For the major part of the paper, we used patch size 1, which treats each object as a token, with 81 tokens. In the ablation experiments, we also tested the model with patch size 3, which treats each panel as a token, with 9 tokens. The panel tokenization trains much faster but didn't perform as well.

For model scaling, we adapted the configuration from DiT and SiT, and used their Small (-S) (12 layers, 384 hidden dimension, 6 heads) and Big (-B) scale (12 layers, 768 hidden dimension, 12 heads). Other hyper parameters:

A.2.3 SiT

Our implementation was based on the original SiT code base [21], `https://github.com/willisma/SiT`.

The network architecture of SiT is almost identical to DiT, with the same object tokenization and configuration for Small and Big scale. For the training configuration, we used Velocity prediction, and Linear Path and none loss weighting.

A.2.4 Default samplers

By default, during model training, we used deterministic samplers to generate samples for efficiency: Heun's 2nd order sampler with 18 steps for UNet model [15]; DDIM sampler with 100 steps for DiT [28]; off-the-shelf Runge-Kutta sampler of order 5 (`dpori5`) for SiT [21].

A.3 Autoregressive models baseline

For all auto-regressive models, we treated each object as a token. The three integer attributes of each token were embedded through separated embedding layers with 1/3 hidden dimensions and then concatenated as the token embedding. During decoding, we chunked the latent states output from the final layer into 3 parts and decoded them separately into the three integer attributes of the next token. All the training samples were pre-pended with a `[SOS]` token, with a learned embedding vector. These models are trained on the next token prediction objective.

A.3.1 GPT2

Our implementation was based on the `GPT2model` from huggingface `transformers` library `https://github.com/huggingface/transformers/blob/v4.46.0/src/transformers/models/gpt2/modeling_gpt2.py`. We trained GPT2 models at Small, Base, Medium scale. GPT2-S has 384d hidden states, 6 attention heads and 12 layers; GPT2-B has 768d hidden states, 12 attention heads and 12 layers; GPT2-M has 768d hidden states, 12 attention heads and 24 layers.

A.3.2 Mamba

Our implementation was based on the original `https://github.com/state-spaces/mamba` code base. We used Mamba1 blocks in the backbone. We trained Mamba models at Base, Medium, Big, and Huge scale. Mamba-Base has 768d hidden states and 12 layers; Mamba-Medium has 768d hidden states and 24 layers; Mamba-Big has 1152d hidden states and 36 layers; Mamba-Huge has 1536d hidden states and 48 layers.

A.3.3 Training details

We used AdamW optimizer, with a linear learning rate schedule with 100 warmup steps and a base learning rate 1E-4. Both GPT2 and Mamba models were trained with batch size 64. GPT2s were trained with 1000000 gradient update steps and Mamba by 250000 gradient steps.

A.3.4 Sampling details

For unconditional generation, we sampled the next token with temperature 1.0.

A.4 Image conditional sampling method (inpainting)

For diffusion models, we used the Twisted Diffusion Sampler based on their official implementation `https://github.com/blt2114/twisted_diffusion_sampler`. We compared that to a custom implementation of the Repaint method combined with Heun (for EDM) and DDIM sampler (for DiT).

For autoregressive models (GPT2, Mamba), we used the greedy sampling with the first 8 panels (and [SOS] token) as their "prompt".