CAN GENERATIVE IMAGE MODELS REASON THROUGH EACH OTHER'S OUTPUTS?

Anonymous authors
Paper under double-blind review

ABSTRACT

Can separately trained image models use each other's drawings to solve a new problem? We train five models from scratch; each takes an image of two numbers and draws their sum modulo 23. Three meet a preset threshold on development problems and are tested on number pairs withheld from training. We then copy one model's drawing into a second addition problem for another model, without converting it to text. Both additions are correct in 76% of all planned trials. We also give the second model that problem with a clean image of the same number. When the drawing is correct and the second model solves the clean version, it solves the version with the drawing in 98% of trials. In a separate binary-addition experiment, seven models use one another's drawn digits correctly in 99% of tests between different models. Two models with different architectures also alternate through complete multiplication problems. In some cases, a correctly read drawing changes the second model's answer. These results show that image generation models can calculate from new combinations of pictured numbers and from other models' drawings. Their calculations can still depend on how a digit is drawn.

1 INTRODUCTION

Several visual reasoning systems let a model draw an intermediate result and use that picture in a later step (Chern et al., 2025; Gu et al., 2025; Wang et al., 2026). In these systems, the same model usually makes and uses the picture. We ask whether a separately trained model can calculate from another model's drawing.

Arithmetic lets us check both parts of the exchange. The first model sees a problem as an image and draws an answer. A program places that drawing in a new problem for the second model. We can check whether the drawing shows the right number and whether the second model calculates the right answer from it. We also give the second model the same problem with a clean image of that number. The operation and the digit's position stay fixed, so this comparison tests what changes when the generated drawing replaces the clean one.

The models usually succeed. They pass answers to unseen addition problems across separate training runs, and binary digits pass between models built in different ways. A pretrained image generator and a convolutional network also alternate through complete multiplication problems, passing drawn answers back and forth through several dependent calculations.

A drawing can show the right number and still change the next calculation. In some tests, two independent digit readers identify the same number before and after we change its appearance, but the second model gives a different answer. The models use information in pictured numbers, yet their calculations remain sensitive to how those numbers are drawn.

2 RELATED WORK

Visual arithmetic has long tested whether models can read a problem from an image and calculate its answer (Hoshen & Peleg, 2016). Other work has studied neural computation on longer arithmetic inputs (Kaiser & Sutskever, 2016). Image generators have also shown geometric problem-solving ability (Goren et al., 2026).

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Other systems generate pictures or visual representations during multistep work and then use them in later steps (Chern et al., 2025; Gu et al., 2025; Wang et al., 2026; Dong et al., 2026; Park et al., 2026). See2Think tests whether visual feedback affects later reasoning (Yan et al., 2026). In drawing-based communication games, agents learn to exchange sketches while training together (Mihai & Hare, 2021). Here the models are trained separately, and the image passed between them is a drawn answer. Giving the second model the same problem with a clean version of that answer shows whether the drawing itself changes its calculation.

3 Experimental setup

3.1 How a drawing enters the next problem

The first model sees an arithmetic problem and draws its answer. A program copies the drawing into a fixed position in a second problem, which another model solves. The second model also receives that problem with a clean image of the same number. The operation and position are identical in the two versions; only the digit image changes.

We report two results for the addition experiment. First, we count all planned two-problem trials in which both answers are correct. Second, among trials where the first drawing is read as the right number and the second model answers correctly with the clean digit, we count how often it also answers correctly with the drawing. The second comparison isolates the effect of replacing the clean digit with the generated one.

In the modulo-23 experiment, a drawing is read as the value of the most similar standard answer image, measured by pixel error. For binary digits, both a comparison with digit templates and Tesseract optical character recognition must read the correct value. Unreadable digits and disagreements count as errors.

3.2 Addition with held-out number pairs

Each input image shows two numbers from 0 to 22 as seven-segment digits. A small model trained from random weights draws their sum modulo 23. Both numbers pass through the same part of the model, so changing their order does not change the answer.

The task has 276 unordered number pairs. For every possible answer, we place eight pairs in training, two in development, and two in testing. This gives 184 training pairs and 46 each for development and testing. Every answer appears in each split, but the development and test pairs are new.

We train five runs independently. Before evaluating the test pairs, we set the rule that a run must answer at least 40 of 46 development pairs correctly to enter the exchange experiment. Three runs meet this rule. The appendix shows all five training curves.

For each test, the first model solves a held-out problem. From a fixed ordering, we take the first second problem that uses its answer and replace that number with the model's drawing. The second model solves this problem and a version with a clean digit in the same place. Both problems use number pairs absent from training. We test every ordered pair of the three selected runs, including cases where a run uses its own drawing.

3.3 One column of binary addition

The second task is one column of binary addition. The input contains the two bits being added, a and b, and an incoming carry, c. The model draws the answer bit s for that column and the outgoing carry c_{out}:

$$c_{\text{out}} = \left\lfloor \frac{a+b+c}{2} \right\rfloor, \qquad s = (a+b+c) \bmod 2.$$

Training includes all eight possible combinations of the three input bits. The three input and two output boxes have fixed positions in a 512×512 image. Every digit box measures 96×160 pixels, allowing an output to be copied directly into a later input.

We train seven models: a SANA image generator, three multilayer perceptrons (MLPs) that use SANA's fixed image compressor and decoder, and three convolutional neural networks (CNNs)

trained from scratch on pixels. Every model uses drawings from every model, including itself. We reuse the same drawings when testing different second models rather than generating a new image for every pairing.

Each source model supplies four drawings of zero and four of one. A fixed rule selects the drawing placed in each input box. We also repeat each test after converting the selected drawing to pure black and white. The arithmetic problem, second model, and random seed stay fixed.

3.4 Repeated addition and multiplication

To add longer binary numbers, a program starts at the rightmost column and copies the model's drawn carry into the next column. We test two independently trained SANA runs on numbers up to 64 bits long. We also change an incoming carry while holding the other inputs fixed to check whether the model uses it.

For multiplication, a program forms each bit product and adds the partial products using the same one-column operation. A SANA model and a CNN trained from scratch alternate after every model call. We test two-, four-, and eight-bit operands; the operand pairs and model order are fixed before generating any images. The four- and eight-bit tests each contain 16 fixed random operand pairs and four pairs chosen to exercise carries. Later problems use the models' drawings without correcting them. An independent audit rebuilds later inputs from saved output crops and checks the model order, image hashes, and every drawn digit. The appendix gives the full audit.

3.5 Models and training

The modulo-23 model is trained from random weights. It sees new visual renderings of the training pairs throughout training. We select checkpoints from development results before testing the model exchanges.

The binary experiments use a pretrained SANA image generator (Xie et al., 2024), three MLPs trained separately with SANA's fixed image compressor and decoder, and three CNNs trained from scratch directly on pixels. The CNNs let us test drawings passed between models that process images differently. Each CNN trains on the eight possible local additions in four image positions, with random pixel noise and one font. Model sizes, training schedules, and output-reading rules are specified in the appendix.

4 Results

4.1 Can another model use the drawing?

The three runs selected on development problems answer 89% of the held-out first additions correctly on average. Across all 276 planned exchanges between different training runs, both models answer correctly in 210 trials, or 76%. The denominator includes trials in which the first model's drawing is read as the wrong number or the second model fails when given a clean digit.

We then isolate what happens when the generated drawing replaces that clean digit. In 215 trials, the first drawing is read as the right number and the second model answers correctly with a clean image of the same number. It remains correct with the first model's drawing in 210 of those trials, or 98%. The five errors arise from changing the digit image while keeping the arithmetic problem fixed. A model using its own drawing succeeds in 107 of 108 comparable trials. Table 1 shows the result for each pair of runs.

Table 1: Correct second answers from the drawn digit / trials where the first drawing is read as the right number and the second model succeeds with a clean digit. Rows are first models; columns are second models. Different-model entries total 210/215.

First / second model	Run 1	Run 2	Run 3
Run 1	36/36	37/39	31/33
Run 2	37/38	41/42	36/36
Run 3	33/33	36/36	30/30

We next test drawings passed among seven binary-addition models. A different model gives the right next answer in 663 of 672 tests, or 99%. This includes 382 correct answers in 384 exchanges between scratch CNNs and models that use SANA's image compressor and decoder. The CNNs operate directly on pixels, so these exchanges do not depend on the models sharing that image-processing machinery. Drawings made by a CNN work in all 288 tests with another model; the nine different-model errors all involve drawings made by an MLP (Figure 2).

4.2 Can the drawings support repeated calculation?

A SANA image generator and a CNN trained from scratch take turns during multiplication. They produce the right final product for all tested two-, four-, and eight-bit operand pairs. Both independent digit readers also read every intermediate output correctly. In the four- and eight-bit tests alone, this covers 2,960 model outputs.

An eight-bit multiplication program makes 120 model calls, some of which compute independent partial products. The longest chain in which each step depends on an earlier answer contains 21 calls and 20 successive switches between the two models. The models therefore exchange generated digits repeatedly within a calculation. Table 2 separates total calls from the length of this chain.

Table 2: Products made by alternating SANA and a scratch CNN. The longest dependent sequence counts calls linked by generated outputs; other calls can compute separate partial products.

Operand width	Programs	Calls per program	Longest sequence	Correct products
2 bits	16	6	–	16/16
4 bits	20	28	9	20/20
8 bits	20	120	21	20/20

The models also reuse their own drawings in longer additions. Each of two SANA runs solves all tested problems at lengths up to 64 bits, for 320 correct full additions. In 160 paired checks, changing only the incoming carry changes the required answer; both runs answer every pair correctly. This shows that they use the drawn carry passed from one column to the next.

4.3 Is a recognizable drawing enough?

The five errors in the held-out addition test show that a drawing can be read as the right number yet change the next answer. Figure 1 shows one case: a drawn zero and a clean zero lead the same second model to different answers to the same problem.

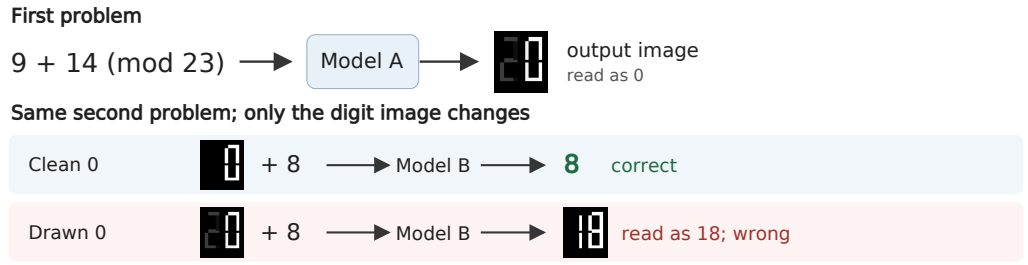

Figure 1: One incorrect exchange. The first model draws an answer read as zero. The second model answers correctly with a clean zero but incorrectly with that drawing; its problem is otherwise unchanged.

We change only the appearance of the digits drawn by the first model, turning them black and white. Both digit readers still identify all 56 drawings as the same digits. Yet the second model's answer changes in 20 of 784 tests: eight wrong answers become right, and twelve right answers become wrong. Each before-and-after comparison uses the same arithmetic problem, second model, and original drawing (Figure 2).

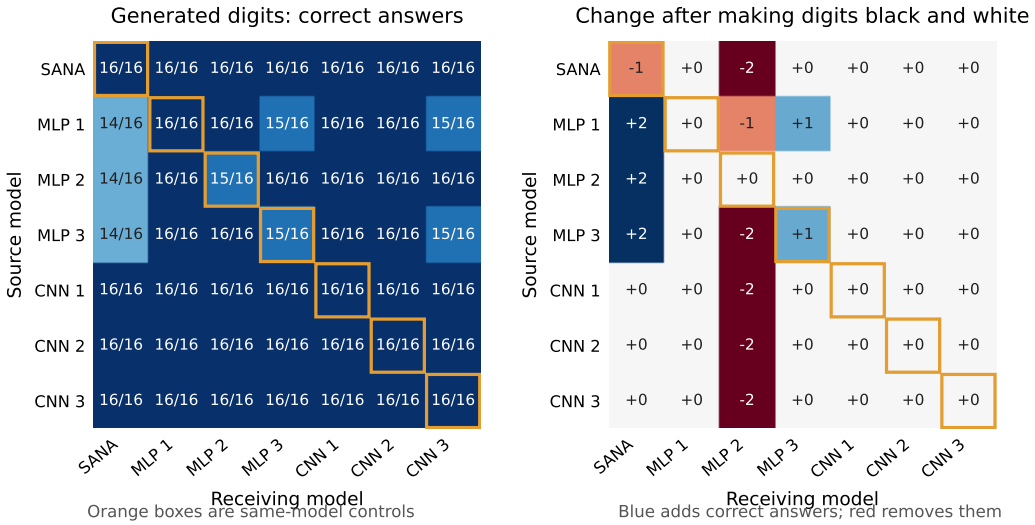

Figure 2: Correct next answers for every binary model pair (left) and changes after the drawn digits are made black and white (right). Each cell contains 16 tests. Orange boxes mark cases where a model uses its own drawing. Blue means more correct answers after conversion; red means fewer.

A separate font check shows another effect of appearance. With a font used in training, one SANA run answers every tested one-column addition correctly. With a font withheld from training, its accuracy falls to 88%; all 24 errors involve $0 + 0 + 1$. A readable value alone does not determine whether a model will use that value correctly in the next calculation.

5 Discussion

The models use arithmetic information in their input images. Three runs trained from scratch solve operand pairs excluded from training, so their success extends beyond the exact pairs they saw. In the carry test, changing only the incoming carry changes the required answer, and both SANA runs answer every paired case correctly. The models' outputs therefore depend on the pictured operands and on a specific input bit that the calculation requires.

This behavior also survives new visual inputs: models can continue from digits drawn by other models, including exchanges between SANA-based models and CNNs trained directly on pixels. The alternating multiplication programs repeat those exchanges through many dependent steps. At the same time, changing a digit's appearance sometimes changes the next answer while independent readers still identify the same digit. The calculation depends on visual details beyond the number those readers identify. Testing new operand combinations, changing one input at a time, and varying how the same digit is drawn each reveal a different part of what an image generation model has learned. In richer visual tasks, the same approach could test whether a model responds to the objects and relations depicted or to incidental details of how they are drawn.

6 LIMITATIONS

Every problem uses fixed boxes and digit layouts. A program chooses the operations, places the images, and selects the model for each step. Training includes every possible one-column binary addition case. The long additions pass carries within one SANA run; the repeated cross-model results come from multiplication with operands no longer than eight bits. The modulo-23 test uses new number pairs but the same family of digit renderings, and its reader always assigns one of the possible answers. Local accuracy falls from 100% to 88% with a font withheld from training. These input-output tests do not reveal whether the models represent numbers explicitly or what calculation they perform internally. We have not tested free-form drawings or models that choose their own sequence of steps.

7 CONCLUSION

Image generation models solved new addition pairs, used changed carry digits, and continued calculations from other models' drawings. Their answers also changed when a digit's appearance changed while two independent readers still identified the same number. The models use numerical information in images, but their calculations remain sensitive to visual details that leave the depicted number unchanged.

Reproducibility. Settings, code, fonts, model checkpoints, inputs, and generated outputs are saved with hashes for model calls. A separate program rebuilds later inputs from saved output boxes to check that each reported sequence uses the recorded drawings. Architectures, training schedules, image coordinates, and digit-reading rules appear in the appendix.

AI USE STATEMENT

OpenAI Codex assisted with designing experiments, implementing code, literature search, figure creation, and manuscript editing. The authors reviewed and verified all AI-assisted code, analyses, and references, and take responsibility for all final work.

REFERENCES

Ethan Chern, Zhulin Hu, Steffi Chern, Siqi Kou, Jiadi Su, Yan Ma, Zhijie Deng, and Pengfei Liu. Thinking with generated images. *arXiv preprint arXiv:2505.22525*, 2025. URL https://arxiv.org/abs/2505.22525.

Shuai Dong, Siyuan Wang, Xingyu Liu, Chenglin Li, Haowen Hou, and Zhongyu Wei. Interleaved latent visual reasoning with selective perceptual modeling. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 29316–29335, 2026. doi: 10.18653/v1/2026.acl-long.1351. URL https://aclanthology.org/2026.acl-long.1351/.

Nir Goren, Shai Yehezkel, Omer Dahary, Andrey Voynov, Or Patashnik, and Daniel Cohen-Or. Visual diffusion models are geometric solvers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 43187–43196, 2026. URL https:

//openaccess.thecvf.com/content/CVPR2026/html/Goren_Visual_Diffusion_Models_are_Geometric_Solvers_CVPR_2026_paper.html.

Jiawei Gu, Yunzhuo Hao, Huichen Will Wang, Linjie Li, Michael Qizhe Shieh, Yejin Choi, Ranjay Krishna, and Yu Cheng. ThinkMorph: Emergent properties in multimodal interleaved chain-of-thought reasoning. *arXiv preprint arXiv:2510.27492*, 2025. URL https://arxiv.org/abs/2510.27492.

Yedid Hoshen and Shmuel Peleg. Visual learning of arithmetic operation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 3733–3739, 2016. doi: 10.1609/aaai.v30i1.9882. URL https://ojs.aaai.org/index.php/AAAI/article/view/9882.

Łukasz Kaiser and Ilya Sutskever. Neural GPUs learn algorithms. In *International Conference on Learning Representations*, 2016. URL https://arxiv.org/abs/1511.08228.

Daniela Mihai and Jonathon Hare. Learning to draw: Emergent communication through sketching. In *Advances in Neural Information Processing Systems*, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/39d0a8908fbe6c18039ea8227f827023-Abstract.html.

Suhyeong Park, Junha Jung, and Jaewoo Kang. Reason through the latent! making latent visual reasoning necessary. *arXiv preprint arXiv:2609.06746*, 2026. URL https://arxiv.org/abs/2609.06746.

Pengjie Wang, Linger Deng, Zujia Zhang, Shaojie Zhang, Zhenbo Luo, Pei Fu, Jian Luan, Xiang Bai, and Yuliang Liu. DeltaV: Thinking with visual state updates in unified large multimodal models. *arXiv preprint arXiv:2607.08434*, 2026. URL https://arxiv.org/abs/2607.08434.

Enze Xie, Junsong Chen, Junyu Chen, Han Cai, Haotian Tang, Yujun Lin, Zhekai Zhang, Muyang Li, Ligeng Zhu, Yao Lu, and Song Han. SANA: Efficient high-resolution image synthesis with linear diffusion transformers. *arXiv preprint arXiv:2410.10629*, 2024. URL https://arxiv.org/abs/2410.10629.

Siyu Yan, Zhuoran Yan, Haiying Xu, Panhao Zhou, Jingyu Chen, Chenhao Ji, Shuo Cao, Yongheng Zhang, Haoze Liu, Siyu Zhang, Xiwen Gu, Yihao Liu, and Alex Jinpeng Wang. See2Think: Do multimodal models really use intermediate visual states? *arXiv preprint arXiv:2607.26769*, 2026. URL https://arxiv.org/abs/2607.26769.

A MODULO-23 TRAINING RUNS

Three of five runs meet the predeclared development criterion and enter the cross-model test. Run 2 has the highest development score and is used for the separate single-model result. Figure 4 shows all five training trajectories.

On the held-out first problems, the selected runs answer 41, 44, and 38 of 46 correctly. With clean digits in the second problem, they answer 40, 44, and 38 of 46 correctly, respectively.

Figure 3 shows the separate test in which one run uses its own drawings.

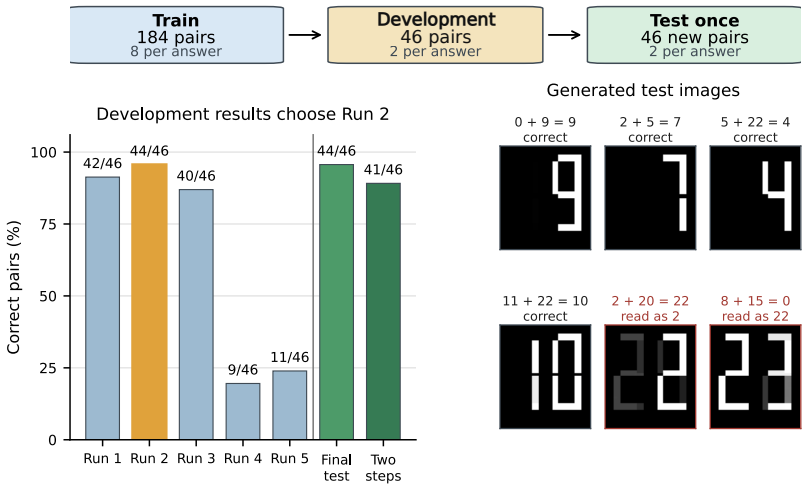

Figure 3: Run 2 answers 44 of 46 held-out addition problems correctly and completes both steps in 41 of 46 tests. The right side shows four correct answer images and the two errors from the first step.

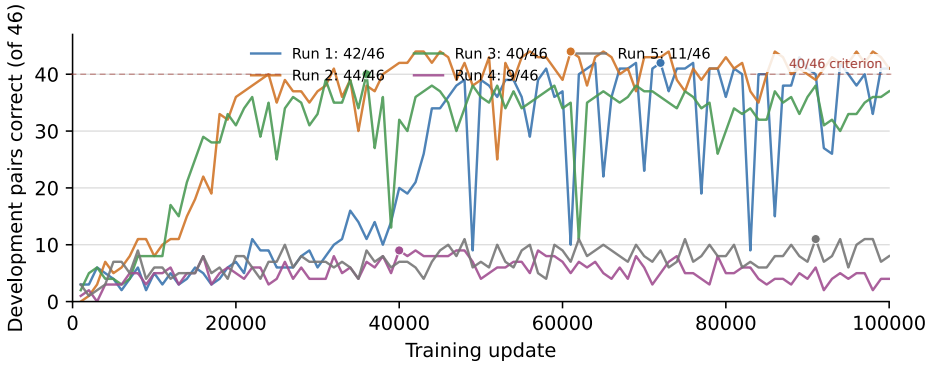

Figure 4: Development accuracy for all five modulo-23 runs. The selection rule is at least 40 of 46 correct pairs; three runs meet it. Dots mark selected checkpoints.

The same small network reads both operands, and both receive the same position marker. This makes the output invariant to operand order. Two transformer blocks combine their features with an output feature. A pixel writer converts that feature into 896 gray values, forming the 28×32 answer image. The model has width 64, four attention heads, and 285,760 parameters.

At each update, the model sees a new random rendering of every training pair. We train with AdamW at a learning rate of 10^{-3}, weight decay 1, and gradient clipping at one. We check the development images every 1,000 updates through update 100,000. Each run uses fixed model and training seeds. We leave the test images unused until the checkpoint and run-selection rules are fixed.

B BINARY MODEL DETAILS

The scratch CNN encoder uses channels $3 \to 16 \to 32 \to 32$. Each convolution has kernel size three, stride two, padding one, and a SiLU activation. Adaptive average pooling gives a 5×3 output. Each input box yields 480 features. Linear layers map the joined 1,440 features through 128 hidden values to 960 output values. A shared decoder uses five transposed convolutions with channels $32 \to 32 \to 16 \to 16 \to 8 \to 3$. All 353,803 parameters are trained.

Inputs and targets use values from -1 to 1. Training adds independent noise with standard deviation 0.04 and clips to this range. The loss weights each squared RGB error by $1 + 4(1 - y)/2$, where y is the target pixel value. AdamW uses a learning rate of 10^{-3}, weight decay of 10^{-4}, and gradient clipping at one. The batch size is 16. Each run uses 2,000 updates.

SANA starts from `Efficient-Large-Model/Sana_600M_512px_diffusers` at revision `83d7a190bfd1fd070570a793d2dab5c7a3231b9d` (Xie et al., 2024). We train rank-32 adapters at 224 attention projections. We also train the input and output projections, depthwise feedforward convolutions, scale-and-shift tables, the last four feedforward blocks, and a local image conditioner. This gives 58,313,120 trained parameters. The image decoder has 159,223,171 fixed parameters.

SANA training proceeds from decimal digits to binary digits in seven stages: decimal flow matching for 800 updates, decimal pixel training for 900, decimal contrast training for 1,200, decimal reader training for 2,000, binary training for 600, four-font binary training for 200, and eight-font binary training for 1,200. The batch size is four. In the final stage, AdamW uses learning rate 10^{-4} and weight decay 10^{-2}; the last feedforward blocks use learning rate 3×10^{-5}. The loss combines flow error, weighted error in decoded pixels, and 0.1 times the cross-entropy from a fixed reader used during training. At inference, SANA takes one Euler step from fresh Gaussian noise and produces the image with its fixed decoder.

Each MLP takes 8,192 noisy values in SANA's compressed image representation together with 4,608 input values. Its layers have sizes $12800 \to 512 \to 512 \to 8192$, with GELU after the first two. We train all 11,019,264 parameters for 1,200 updates per run. The MLP uses SANA's fixed image compressor and decoder and one sampling step. Its loss combines compressed-image error, weighted pixel error, and a digit-reader loss weighted by 0.1.

C IMAGE BOXES AND READERS

Within each 512×512 binary image, the three input boxes have pixel coordinates $(80, 48, 176, 208)$, $(208, 48, 304, 208)$, and $(336, 48, 432, 208)$, written as left, top, right, and bottom edges. The two output boxes are $(144, 320, 240, 480)$ and $(272, 320, 368, 480)$. A copied digit keeps the same box size. Training uses Liberation Sans Bold; the second set of reading templates uses Iosevka.

The modulo-23 experiment decodes each output by choosing the nearest of its 23 standard answer images in mean-squared pixel error. For the binary experiments, one digit reader compares an output image with two sets of templates, each containing the ten digits, and uses the set with the smaller pixel error. The other reader is Tesseract 5.5.3, configured for a single text line and restricted to `0123456789`. We remove spaces from its output before scoring. A binary output is correct only if both readers return the expected digit; an unreadable output or a disagreement is an error.

For each output box, each binary source model supplies four images showing 0 and four showing 1. When a test needs a particular digit, it chooses one of that digit's four images using $(i + j) \bmod 4$, where i identifies the arithmetic case and j the receiving input box. The original and black-and-white tests use the same chosen images. For conversion, we first make an image gray, then set values below 128 to 0 and all other values to 255.

D MIXED-MODEL MULTIPLICATION AUDIT

The four- and eight-bit programs use the same SANA checkpoint and first scratch-CNN run as the two-bit test. The reading rule was recorded before image generation.

The independent audit also checks input hashes and rereads every intermediate output with the frozen template reader and Tesseract 5.5.3. The readers agree on the expected digits at all 560 four-bit and 2,400 eight-bit calls. The four-bit programs have a maximum dependency depth of nine calls and eight consecutive cross-model transfers; the eight-bit programs have a maximum depth of 21 calls and 20 consecutive transfers. These are dependency-path lengths, not total call counts.