Neural Networks Learn Statistics of Increasing Complexity

Nora Belrose [1] **Quintin Pope** [2] **Lucia Quirke** [1] **Alex Mallen** [1] **Xiaoli Fern** [2]

Abstract

The *distributional simplicity bias* (DSB) posits that neural networks learn low-order moments of the data distribution first, before moving on to higher-order correlations. In this work, we present compelling new evidence for the DSB by showing that networks automatically learn to perform well on maximum-entropy distributions whose low-order statistics match those of the training set early in training, then lose this ability later. We also extend the DSB to discrete domains by proving an equivalence between token n-gram frequencies and the moments of embedding vectors, and by finding empirical evidence for the bias in LLMs. Finally we use optimal transport methods to surgically edit the low-order statistics of one class of images to match those of another, and show early-training networks treat the edited images as if they were drawn from the target class. Code is available at https://github.com/EleutherAI/features-across-time.

1. Introduction

Neural networks exhibit a remarkable ability to fit complex datasets while generalizing to unseen data points and distributions. This is especially surprising given that deep networks can perfectly fit random labels (Zhang et al., 2021), and it is possible to intentionally "poison" networks so that they achieve zero training loss while behaving randomly on a held out test set (Huang et al., 2020).

A recently proposed explanation for this phenomenon is the **distributional simplicity bias (DSB)**: neural networks learn to exploit the lower-order statistics of the input data first– e.g. mean and (co)variance– before learning to use its higher-order statistics, such as (co)skewness or (co)kurtosis. Refinetti et al. (2023) provide evidence for the DSB by training networks on a sequence of synthetic datasets that act as increasingly precise approximations to the real data, showing that early checkpoints perform about as well on real data as checkpoints trained directly on the real data.

We build on Refinetti et al. (2023) by inverting their experimental setup. We train our models on real datasets, then test them throughout training on synthetic data that probe the model's reliance on statistics of different orders. We believe this experimental design provides more direct evidence about the generalization behavior of commonly used models and training practices.

Our primary theoretical contributions are to **(1)** motivate the DSB through a Taylor expansion of the expected loss, **(2)** propose criteria quantifying whether a model "uses" statistics up to order k by checking that the model is sensitive to interventions on the first k statistics, while being robust to interventions on higher-order statistics, **(3)** describe efficient methods of producing synthetic data that let us investigate whether models satisfy the above criteria, and **(4)** extend the DSB to discrete domains by proving an equivalence between token n-gram frequencies and the moments of sequences of embedding vectors.

We use a Taylor series expansion to express a model's expected loss as a sum over the central moments of an evaluation dataset. This connection provides some motivation for the DSB. Specifically, if during training, a network's loss is well approximated by the first k terms of its Taylor expansion, then the model should only be sensitive to statistics up to order k, and we argue that earlier terms of the expansion will generally become relevant before later terms.

We describe two intuitive criteria that a model sensitive to statistics up to order k should satisfy: **(1)** changing the first k statistics of data from class A to match class B should cause the model to classify the modified data as class B, and **(2)** models should be unaffected by "deleting" higher-order data statistics. We evaluate whether image classification networks satisfy the above criteria during training through extensive empirical experiments across a variety of network architectures and image datasets.

We evaluate whether the network satisfies criterion **(1)** by generating synthetic datasets where we "graft" the means and covariances of one class onto images of another class, and evaluating whether the network's classifies the result-

*Equal contribution [1]EleutherAI [2]Oregon State University. Correspondence to: Nora Belrose <nora@eleuther.ai>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

Figure 1. **(left)** Pekinese dog image from the ImageNet training set. **(center)** Image after quantile normalizing its pixels to match the marginal distribution of the goldfish class on ImageNet. The grass is now a slightly darker shade of green and the dog's fur has a reddish hue. **(right)** Synthetic "goldfish" generated by sampling each pixel independently from its marginal distribution.

Figure 2. Non-cherrypicked "fake" images produced by maximum entropy sampling using only the first two moments of the class-conditional distributions, and a hypercube constraint. Fake MNIST digits are clearly recognizable, SVHN digits less so, whereas fake CIFAR-10 images look nothing like their respective classes: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck.

ing data as belonging to the target class. We formalize this notion of "grafting" statistics with optimal transport (OT) theory, using an analytic formula to map samples from one class-conditional distribution to another, while minimizing the expected squared Euclidean distance the samples are moved. We also describe coordinatewise quantile normalization, an OT method that changes the marginal distribution of each coordinate of the input to match a target class.

We evaluate the degree to which networks satisfy criterion **(2)** by generating synthetic data that match the class-conditional means and covariances, but are otherwise maximum entropy.[1] We generate two datasets for this purpose. One dataset comes from sampling from a Gaussian distribution with matching mean and covariances. The other dataset comes from incorporating the constraints on image pixel values. We propose a novel gradient-based optimization method to produce samples from a hypercube-constrained maximum entropy distribution. We additionally describe independent coordinate sampling, a first order method of generating hypercube-constrained maximum entropy samples using only means.

Across models and datasets, we find a common pattern where criteria **(1)** and **(2)** hold early in training, with networks largely classifying images according to the means and covariances of the distributions from which they're drawn. But as training progresses, networks become sensitive to higher-order statistics, resulting in a U-shaped loss curve.

We also evaluate EleutherAI's Pythia autoregressive language models (Biderman et al., 2023) on synthetic data sampled from unigram and bigram models trained on the Pile (Gao et al., 2020). We find a fascinating "double descent" (Vallet et al., 1989; Belkin et al., 2019) phenomenon where models initially mirror the same U-shaped scaling observed in image classifiers, then use in-context learning to achieve even lower loss later in training.

[1]Appealing to the principle of maximum entropy to operationalize the notion of "deletion" in criterion **(2)**.

For a thorough review of related work on simplicity biases in machine learning, see Appendix A.

2. Theory and Methods

Let $\mathcal{L}(\mathbf{x})$ denote the loss of a neural network evaluated on input $\mathbf{x}$. If $\mathcal{L}_\theta$ is analytic[2] with an adequate radius of convergence, we can Taylor expand the loss for any given $\mathbf{x}$ around the mean input $\boldsymbol{\mu}$ as:

$$\mathcal{L}(\mathbf{x}) = \sum_{\alpha \in \mathbb{N}^d} \frac{(\mathbf{x}-\boldsymbol{\mu})^\alpha}{\alpha!} (\partial^\alpha \mathcal{L})(\boldsymbol{\mu}), \tag{1}$$

where α is a multi-index, or a d-tuple assigning an integer to each coordinate of $\mathbf{x}$. Recall that taking a vector to the power of a multi-index denotes a product of the components of the vector, where each component of the index indicates the multiplicity: e.g. if $\alpha = (1, 4, 6)$, the expression $(\mathbf{x}-\boldsymbol{\mu})^\alpha$ denotes the product $(x_1-\mu_1)(x_2-\mu_2)^4(x_3-\mu_3)^6$. Similarly, $(\partial^\alpha \mathcal{L})$ is shorthand for the mixed partial derivative $\frac{\partial^{11}}{\partial x_1 \partial x_2^4 \partial x_3^6}\mathcal{L}$. The factorial $\alpha!$ denotes the product of the factorials of the components: $1! \times 4! \times 6! = 17280$.

If $\mathbf{x}$ is drawn from a distribution with compact support,[3] which is true for images and text, we can take the expectation of both sides of Eq. 1. This leads to an expression summing over all the *central moments* of $\mathbf{x}$ multiplied by the corresponding partial derivatives of $\mathcal{L}$ evaluated at $\boldsymbol{\mu}$:

$$\mathbb{E}[\mathcal{L}(\mathbf{x})] = \sum_{\alpha \in \mathbb{N}^d} \frac{(\partial^\alpha \mathcal{L})(\boldsymbol{\mu})}{\alpha!} \underbrace{\mathbb{E}[(\mathbf{x}-\boldsymbol{\mu})^\alpha]}_{\text{central moment}} \tag{2}$$

Equation 2 suggests a close connection between the moments of the data distribution and the expected loss of a neural network evaluated on that distribution.[4]

[2]Famously, the ReLU activation function is not analytic, but it is possible to construct arbitrarily close approximations to ReLU that are analytic (Hendrycks & Gimpel, 2016, Sec. 4).

[3]The requirement is slightly weaker than this: we require that the distribution has finite moments of all orders, which is true

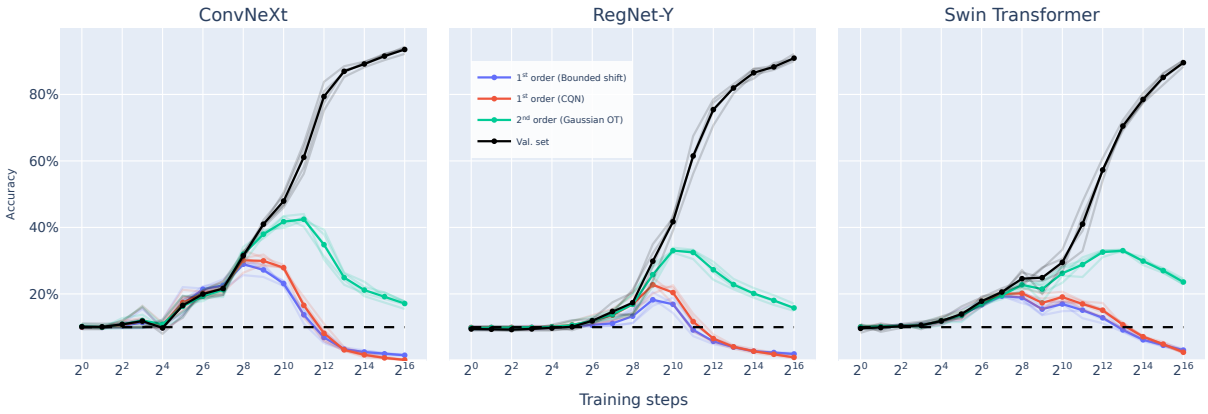


Figure 3. Accuracy of computer vision models when evaluated on images edited with optimal transport maps as described in Sec. 2.3, using the *target class*, not the source class, as the label. Between roughly 2^4 and 2^{12} training steps, all models classify the CQN-edited images coming from target class, with a peak in accuracy at 2^9.

2.1. Intuition

Without loss of generality, assume the input is constrained to the unit hypercube.[5] Since every coordinate of $\mathbf{x}$ is no greater than 1, the moments will have magnitudes that monotonically decrease with increasing order; for example, $\mathbb{E}[x_i x_j] \leq \mathbb{E}[x_i]$ for any $i, j \in 1 \ldots d$.

Indeed, we would expect the moment magnitude to decay *exponentially* with order when the coordinates are independent, roughly counterbalancing the exponential increase in the number of distinct moments at higher orders. Assuming the higher derivatives of $\mathcal{L}$ are reasonably well-behaved at initialization, the $\frac{1}{n!}$ Taylor coefficients will then cause the contribution of higher-order moments to Eq. 2 to decay monotonically and factorially fast with order.

As training progresses however, the derivatives of $\mathcal{L}$ become correlated with the corresponding moments, potentially inflating the magnitude of higher-order terms in Eq. 2. It then seems natural to suppose that the magnitude of higher-order terms will grow in roughly monotonic order– that is, the second order term will become important first, followed by the third order term, and so on[6].

when the support is compact.

[4]Expanding around an an arbitrary point $\mathbf{a}$ would yield an expression containing moments about $\mathbf{a}$, and our analysis would otherwise be unchanged.

[5]This is true of images with standard PyTorch preprocessing and one-hot encoded token sequences; other inputs can be rescaled to match this criterion, given our assumption of compact support.

[6]Another argument for monotonicity is that earlier terms account for factorially more of the loss at initialization, and are thus plausibly higher-priority targets for gradient descent, until the optimizer is no longer able to easily reduce the loss further by better matching the associated statistics and moves on to higher-order terms.

2.2. Criteria

Intuitively, if a model only "uses" low-order statistics of the input distribution, this means its behavior should be strongly affected by interventions on the lower-order statistics of the input, but largely unaffected by interventions on the higher-order statistics. More specifically:

1. "Grafting" the low-order statistics of class B onto class A should cause the model to treat examples from A as if they were from B.

2. "Deleting" the information contributed by higher-order statistics should not harm the model's performance.

We operationalize both criteria more precisely below and explain how we produce synthetic data that lets us evaluate the degree to which a given model satisfies each criterion.

2.3. Optimal Transport

We operationalize Criterion 1 using optimal transport (OT) theory, which provides tools for transforming samples from one probability distribution into samples from another while minimizing the average distance that samples are moved. We use three OT methods in our experiments: bounded shift and coordinatewise quantile normalization, which primarily affect the first order moments of the distribution, and Gaussian OT, which affects all first and second-order moments.

Bounded Shift In Appendix E, we derive an algorithm for shifting the mean of an empirical distribution to a desired value, while keeping it constrained to the interval $[0, 1]$, and minimizing the transport cost. We use this algorithm to graft the mean of one class onto another, while ensuring the pixel intensities of the edited images are valid.

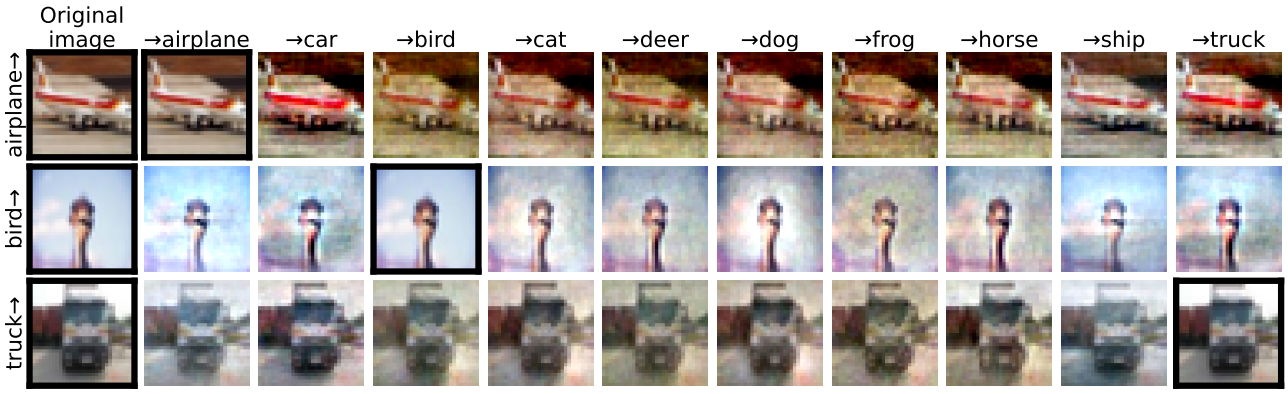


Figure 4. Rows 1-3 show how Gaussian optimal transport affects the example CIFAR-10 airplane, bird and truck images. Each row starts with the original unedited image on the left, with each subsequent column showing the effects of editing that image's first two moments to match the class-conditional distributions of a particular target class **(top)**. The bottom left image is the average of all test set images. Black borders indicate either an original image, or a "self-to-self" edited image.

Coordinatewise Quantile Normalization (CQN) Quantile normalization is a technique for making two scalar random variables identical in their statistical properties. When applied *coordinatewise* to the input of a neural network, such as an image, it ensures that the coordinatewise marginals match those of a target distribution, while keeping the correlations between coordinates largely intact, as illustrated by how the edited Pekinese dog image in Fig 1 (**center**) remains a recognizable dog image.

CQN works as follows. If a random variable X has cumulative distribution function $F_X(x)$, the transformed variable $F_X(X)$ will have the standard uniform distribution $\mathrm{Unif}(0,1)$. Conversely, given variables $U \sim \mathrm{Unif}(0,1)$ and Y, the transformed variable $F_Y^{-1}(U)$ will be equal in distribution to Y. Composing these transformations together yields quantile normalization. It can be shown that $F_Y^{-1} \circ F_X$ is the optimal transport map from X to Y for a large class of cost functions (Santambrogio, 2015, Ch. 2.2), and is thus ideal for editing the first order statistics of a distribution while minimally perturbing higher-order statistics.

Gaussian Optimal Transport Given two Gaussians $P = \mathcal{N}(\boldsymbol{\mu}_P, \boldsymbol{\Sigma}_P)$ and $Q = \mathcal{N}(\boldsymbol{\mu}_Q, \boldsymbol{\Sigma}_Q)$ supported on $\mathbb{R}^d$, the map $T(\mathbf{x}) = \mathbf{A}(\mathbf{x} - \mathbf{m}_P) + \mathbf{m}_Q$ is the optimal transport map from P to Q under the squared Euclidean cost function, where

$$\mathbf{A} = \boldsymbol{\Sigma}_P^{-1/2}\left(\boldsymbol{\Sigma}_P^{1/2}\boldsymbol{\Sigma}_Q\boldsymbol{\Sigma}_P^{1/2}\right)^{1/2}\boldsymbol{\Sigma}_P^{-1/2}. \tag{3}$$

More generally, if P is an arbitrary distribution with finite second moments, $T(\mathbf{x})$ will transport it to a distribution with mean $\boldsymbol{\mu}_Q$ and covariance $\boldsymbol{\Sigma}_Q$, and this map will minimize the cost $\mathbb{E}_P[\|\mathbf{x} - T(\mathbf{x})\|_2^2]$ (Dowson & Landau, 1982).

Given k image classes, each containing tensors of shape $C \times H \times W$, we unroll the tensors into vectors of size CHW, then compute their sample means and covariance matrices,[7] and plug these statistics into Eq. 3 to get the $k(k-1)$ optimal transport maps from each class to every other class.

2.4. Maximum Entropy Sampling

We can operationalize Criterion 2 using the principle of maximum entropy, which provides a principled method for constructing probability distributions based on "partial knowledge" (Jaynes, 1957). Here the partial knowledge consists of low-order statistics derived from a training dataset, but we otherwise want to minimize the information content of the higher-order statistics. We therefore want to construct the maximum entropy distribution P consistent with these low-order statistics,[8] then evaluate a neural network on samples drawn from P.

Famously, the maximum entropy distribution supported on $\mathbb{R}^d$ with known mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$ is the Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$. We therefore use Gaussians in many of our experiments in Sec 3. In addition to Gaussians, we use hypercube constrained sampling to generate a set of synthetic samples using 2^{nd} order statistics and another set using both 2^{nd} and 3^{rd} order statistics, though that last set suffered from significant convergence issues when trying to match the 3^{rd} order statistics (see Appendix C for 3^{rd} order results and Appendix D for discussion of convergence issues). We also use two first-order methods

[7] Because this is a high-dimensional covariance matrix with dimension only 1-3 times smaller than the sample size, we apply the asymptotically optimal linear shrinkage method proposed by Bodnar et al. (2014) to improve our estimate of the population covariance and increase numerical stability.

[8] P can be thought of as the "least informative" distribution that satisfies the constraints that its mean and covariance should match those of our original data distribution.

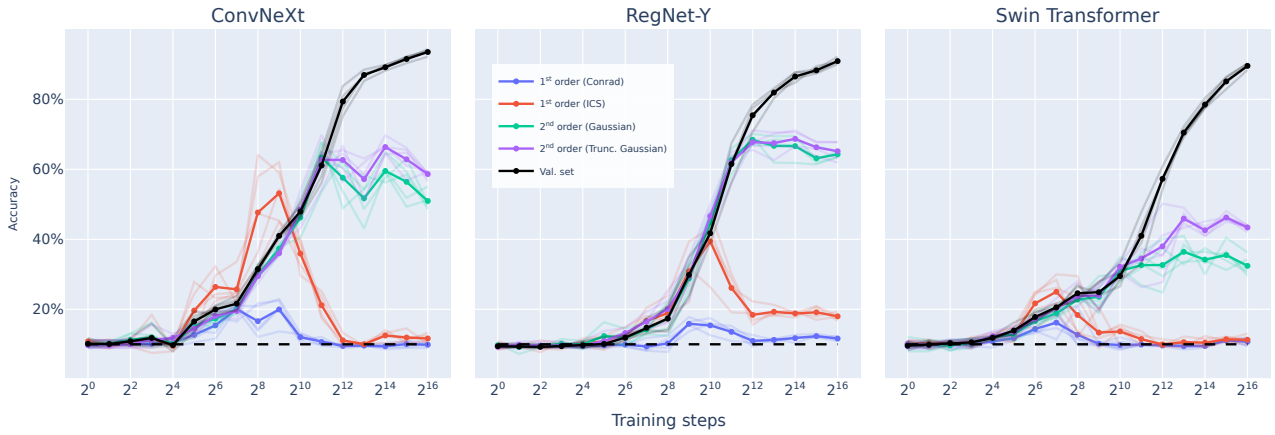


Figure 5. Accuracy of computer vision models being trained on the *standard* CIFAR-10 training set, and being evaluated on maximum-entropy synthetic data with matching statistics of 1st or 2nd order. 3rd order results are in Appendix C, with Appendix D discussing issues with those results.

(Conrad sampling and independent coordinate sampling). We describe these methods below.

Hypercube Constraints One problem with using Gaussians to generate synthetic images is that natural images are constrained to a hypercube: RGB pixel intensities are in the range $[0, 255]$, but nonsingular Gaussian distributions assign positive probability density to all of $\mathbb{R}^d$, so that a typical sample will often lie outside the hypercube of natural images.[9] We might expect neural networks to quickly adapt to such a simple box constraint on the support, so we would ideally like to subject our synthetic images to this constraint.

Conrad Distribution We prove in Theorem F.1 that the maximum entropy distribution supported on $[0, 1]$ with a fixed mean $\mu \neq \frac{1}{2}$ and unconstrained variance takes the form $p(x) = \frac{b \exp(-bx+b)}{\exp(b)-1}$, where the parameter b can be found using Newton's method. This formula is not well-known, although an alternative derivation can be found in Conrad (2004). To isolate the effect of first-order statistics, we first fit a Conrad distribution to the mean of each coordinate of the images. We then generate synthetic images using inverse transform sampling to produce a value for each coordinate independently.

Approximate Sampling For many sets of constraints, there is no known closed-form solution for the density, precluding standard sampling techniques like Markov chain Monte Carlo. For example, in the 1D case the maximum entropy probability density with known mean and variance supported on a finite interval $[a, b]$ has the form $p(x) = \exp(-\lambda_0 - \lambda_1 x - \lambda_2 x^2)$ (Dowson & Wragg, 1973),[10] but we are unaware of any such analytical formula for the multidimensional case.[11]

For these cases, we propose a novel technique for approximate sampling: use gradient-based optimization to directly produce a finite set of samples whose statistics match the desired ones, while maximizing the Kozachenko-Leonenko estimate for the entropy of the implicit population distribution (Kozachenko & Leonenko, 1987; Sablayrolles et al., 2018). See Appendix H for implementation details and Appendix I for a discussion of computational requirements.

Independent Coordinate Sampling (ICS) In the preceding sections, we decomposed the input distribution into its moments. Another possible decomposition is given by Sklar's theorem, which states that the distribution of any random vector $(X_1, \ldots, X_d)$ is uniquely determined by its coordinatewise marginal CDFs $F_{X_i}(x) = \mathbb{P}(X_i \leq x)$ and a *copula* function $C : [0, 1]^d \to [0, 1]$ that combines the marginal CDFs into a multivariate CDF $F_X(\mathbf{x}) = \mathbb{P}(X_1 \leq x_1, \ldots, X_d \leq x_d)$ (Sklar, 1959). The maximum entropy copula simply takes the product of the marginal CDFs, and corresponds to a random vector with independent coordinates. We can efficiently sample from this distribution by estimating an empirical CDF for each coordinate, then sampling from each CDF independently.

[9] Strictly speaking, Gaussians also violate the assumption of compact support that we made earlier. In high dimension, though, almost all the probability mass of a Gaussian is contained in the typical set, a compact region near the boundary of an ellipsoid surrounding the mean (Carpenter, 2017).

[10] For some values of the Lagrange multipliers, the formula corresponds to a truncated normal distribution. For sufficiently large variances, the density takes on a U-shape.

[11] The log-density of the multidimensional max entropy distribution must be a quadratic form, just like the multivariate normal, but the pseudo-precision matrix may not be p.s.d., and solving for the parameters may be challenging.

By constraining the coordinatewise marginals, we ensure that all of the *homogeneous* moments, or moments of the form $\mathbb{E}[(x_i)^n]$, match those of the true data distribution, while the *mixed* moments, e.g. $\mathbb{E}[x_i x_j]$ for $i \neq j$, will generally not match. In high dimension, almost all moments of order greater than one are mixed rather than homogeneous, so ICS matches the first order moments and almost none of the higher order ones. For this reason we classify it as a "first order" method.

2.5. Discrete Domains

Neural networks use embeddings to convert discrete inputs into vectors of real numbers. The embedding operation can be viewed as a matrix multiplication, wherein the discrete inputs are converted into one-hot vectors we then multiply by the embedding matrix. If the input is a sequence, the result is a sequence of one-hot vectors, or a one-hot matrix.

Just as we unroll images into vectors to compute their moments, we can similarly unroll one-hot matrices to compute their moments. Strikingly, we find that these moments correspond to token n-gram frequencies:[12]

Theorem 2.1. *[n-gram statistics are moments] Let $\mathcal{V}^N$ be the set of token sequences of length N drawn from a finite vocabulary $\mathcal{V}$, let P be a distribution on $\mathcal{V}^N$, and let $f : \mathcal{V}^N \to \{0,1\}^{N\cdot|\mathcal{V}|}$ be the function that encodes a length-N sequence of tokens as a flattened concatenation of N one-hot vectors of dimension $|\mathcal{V}|$. Let $f_\sharp P$ be the pushforward of P through this one-hot encoding, i.e. its analogue in $\{0,1\}^{N\cdot|\mathcal{V}|}$.*

Then every moment of $f_\sharp P$ is equal to an n-gram statistic of P and vice versa.

Furthermore, for a fixed embedding matrix $\mathbf{E}$, two distributions over token sequences that have the same n-gram frequencies up to order k will induce distributions over embedding space with the same moments up to k:

Theorem 2.2. *[Equal embedding moments] Let $\mathbf{E} \in \mathbb{R}^{|\mathcal{V}|\times d}$ be an embedding matrix, and let P and Q be two distributions over $\mathcal{V}^N$. Then if P and Q have the same n-gram statistics up to order $k \geq 1$, their embeddings under $\mathbf{E}$ have the same moments up to order k.*

For proofs, see Appendix G.

Given this equivalence, we can test the DSB in language models with maximum entropy sampling, just like the computer vision case. Given known n-gram frequencies up to order k, we produce maximum entropy samples using a k-gram autoregressive language model. For example, if only bigram frequencies are known, this corresponds to a Markov chain where the distribution of each token depends only on the token immediately preceding it.

3. Image Classification

3.1. Datasets.

Because Gaussian optimal transport (Sec. 2.3) requires $O(d^3)$ compute and $O(d^2)$ memory,[13] we focus on datasets with 32×32 or 64×64 resolution images for our primary experiments. Specifically, we examine the popular image classification datasets CIFAR-10 (Krizhevsky et al., 2009), Fashion MNIST (Xiao et al., 2017), MNIST (LeCun et al., 1998), and SVHN (Netzer et al., 2011).

We also build a new image classification dataset, **CIFARNet**, consisting of 200K images at 64×64 resolution sampled from ImageNet-21K, using ten coarse-grained classes that roughly match those of CIFAR-10. The larger number of images per class (20K) allows us to get a good estimate of the class-conditional covariance matrices needed for Gaussian optimal transport, which at this resolution contain $(3 \times 64 \times 64)^2 \approx 1.5 \times 10^8$ entries each. See Appendix B for more details on CIFARNet.

3.2. Architectures.

We focus on state-of-the-art computer vision architectures in our experiments. Specifically, we use ConvNeXt V2 (Woo et al., 2023) and Swin Transformer V2 (Liu et al., 2022), which Goldblum et al. (2023) recently found to have the best performance on a variety of tasks. We train for 2^{16} steps with batch size 128, using the AdamW optimizer (Loshchilov & Hutter, 2018) with $\beta_1 = 0.9, \beta_2 = 0.95$, and a linear learning rate decay schedule starting at 10^{-3} with a warmup of 2000 steps (Ma & Yarats, 2021).[14] For data augmentation, we employ RandAugment (Cubuk et al., 2020) followed by random horizontal flips and random crops.

To examine the effect of model scale on our results, we sweep over the Atto, Femto, Pico, Nano, and Tiny sizes for ConvNeXt V2, and we also construct Swin Transformers of

[12] Our formal definition of the term "n-gram statistic" is nonstandard in two respects: first, we include skip-grams (e.g. *the ___ dog*, where the underscore is a wildcard token), and second, it is tied to an absolute position in the sequence. However, the Pythia language models we will consider in this paper were trained on chunks of text of uniform length sampled from larger documents (Biderman et al., 2023), so the absolute position should not significantly affect the n-gram probabilities. We therefore assume in what follows that n-gram statistics exhibit translation invariance.

[13] The covariance matrix has d^2 elements, where d is the number of pixels, so it is actually $O(n^4)$ in the height or width dimension of the image. We also ran into a software limitation in early experiments where NumPy and PyTorch eigensolvers would crash when fed the very large covariance matrices produced by high-resolution image datasets; see PyTorch issue #92141 for discussion.

[14] We found in early experiments that many models require a lower learning rate to converge on SVHN. We therefore use a learning rate of 10^{-4} for ConvNeXt and Swin on this dataset.

roughly analogous sizes.[15]

To ensure our results are insensitive to the choice of optimizer and learning rate schedule, we also perform experiments with RegNet-Y (Radosavovic et al., 2020) using SGD with momentum and no LR warmup.

3.3. Results

We display our results on CIFAR10 in Figures 3 and 5, see Appendix C for other datasets.

Optimal transport We measure the effect of optimal transport interventions by computing the accuracy or loss of the model with respect to the *target class*, rather than the source class. If the intervention is ineffective, we would expect the accuracy to be much lower than the random baseline of 10%, because the model should confidently classify the images as belonging to the source class. Strikingly, all models we tested get substantially higher than 10% accuracy w.r.t. the target labels, with ConvNeXt peaking at over 40% accuracy on 2^{nd} order-edited images after 2^{10} training steps.

Maximum entropy sampling We include five different conditions in our maximum entropy sampling experiments: 1^{st} order (Conrad and ICS), 2^{nd} order (Gaussian sampling), and both 2^{nd} and 3^{rd} order approximate sampling plus a hypercube constraint. However, we place the 3^{rd} order approximate sampling results in Appendix C because we were unable to match the coskewness and covariance statistics simultaneously when generating the synthetic datasets, making their interpretation difficult. See Appendix D for discussion.

Overall, we find that accuracy on first order samples peaks earlier in training and has a lower maximum than accuracy on second order samples, followed by the 2^{nd} order hypercube-constrained samples. Remarkably, for some datasets early in training, we find some models achieve *higher* accuracy on the independent pixel samples than they do on images sampled from the real validation set!

Non-monotonicity Across all datasets, we observe some degree of *non-monotonicity* in the accuracy curves: while models are quite sensitive to low-order moments early in training, they become less sensitive by the end, with accuracy often dipping below the random baseline. The degree of non-monotonicity varies by dataset, however. Very simple datasets like MNIST and Fashion MNIST show very little non-monotonicity, likely because the first and second moments of the data distribution are sufficient to produce very realistic-looking samples (Fig. 2).

Overall, we found that model scale has a remarkably small effect on the learning curves, so we display curves averaged over scales in bold, with individual model scales shown as translucent lines. Each curve is itself averaged over three random seeds.

4. Language Modeling

To test the distributional simplicity bias in a discrete domain, we study EleutherAI's Pythia language model suite (Biderman et al., 2023), for which checkpoints are publicly available at log-spaced intervals throughout training. Model parameter counts range from 14 million to 12 billion.

There are ten independent training runs for Pythia 14M, 70M, and 160M publicly available on the HuggingFace Hub, each using a different random seed. There are also five available seeds for Pythia 410M. We take advantage of these additional runs to examine the effect of random seed on our results. We also trained custom variants of Pythia 14M and 70M with an extended learning rate warmup period of 14,300 steps to isolate the effect of LR warmup.

While we include skip-grams (e.g. *the* ___ *dog*) in our formal definition of n-gram frequency (Def. G.1), we do not include them in these experiments for tractability reasons: they would greatly increase the memory and storage requirements of maximum entropy sampling. We hope to explore the effect of skip-gram statistics in future work.

n-gram language models We compute token unigram and bigram frequencies across Pythia's training corpus, the Pile (Gao et al., 2020), and use these statistics to construct maximum entropy n-gram language models. We autoregressively sample sequences of length 2049 from the n-gram LMs, and evaluate Pythia's cross entropy loss on these maximum entropy samples at each checkpoint.

Additionally, we evaluate Pythia 12B's cross entropy loss over each token position in the maximum entropy n-gram sequences for four different checkpoints in order to detect the development of in-context learning. If in-context learning is involved in making predictions for these sequences at a training step, cross entropy loss should decrease over successive token positions in the sequence for that step (Olsson et al., 2022).

4.1. Results

We display our results on the Pythia suite in Fig. 6. See Appendix C for alternate model seeds and learning rate warmup. Overall we find that the random seed has very little effect on the learning curves, and lengthening the LR warmup period did not consistently affect their overall shape.

[15] The smallest model described in Liu et al. (2022) is Swin V2 Tiny, which weighs in at 49M parameters. We construct smaller Swin V2 sizes by copying the embedding dimension from the corresponding ConvNeXt V2 size.

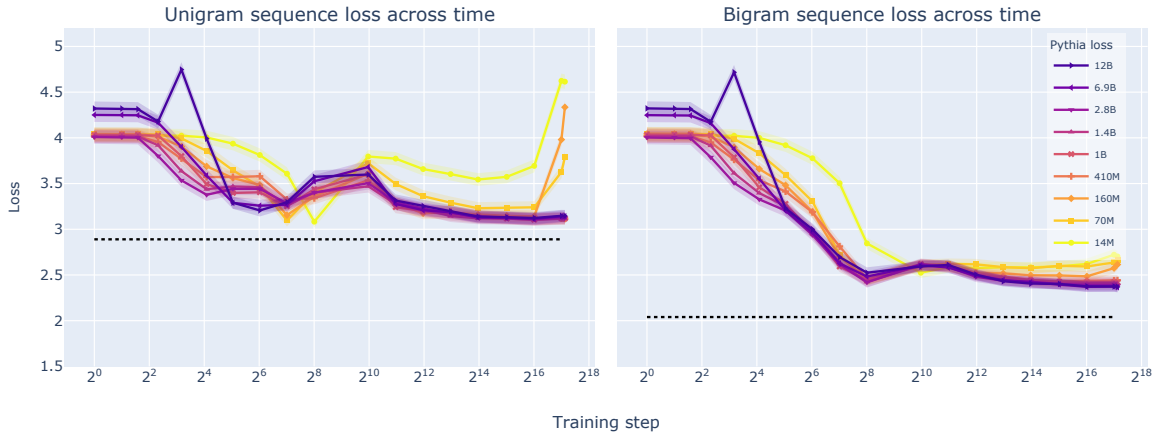


Figure 6. Cross entropy loss of Pythia suite evaluated on unigram and bigram sequences (N=1024). For comparison, the Shannon entropy of the unigram distribution is 2.89 bits per byte (bpb), and is 2.04 bpb for the bigram distribution. Both plots exhibit "double descent" scaling: loss reaches a trough between 2^6 and 2^8 steps, increases until 2^{10} steps, then decreases again as the model learns to match the data generating process in-context (Fig. 7).

n-gram sequence loss Consistent with the image classification tasks, unigram sequence loss consistently reaches its lowest point before bigram sequence loss and has a higher minimum value.

Across all models, we observe non-monotonicity in the n-gram sequence loss curves, where loss steeply reduces and then increases to a lesser extent. However, unlike in the image classification tasks, the loss reverts to a monotonic regime later in training. We hypothesize that this is caused by the development of in-context learning sufficient to improve n-gram sequence predictions. We observe correlational evidence in the n-gram sequence loss over increasing token indices and training steps in Pythia 12B (Fig. 7), where in-context learning seems to emerge in the same training step where the non-monotonic regime ends. Fascinatingly, smaller models seem to resume the standard 'U'-shaped loss pattern in the later portions of training.[16]

We speculate that this behavior may arise from a form of "catastrophic forgetting", in which all models initially learn low-order n-gram statistics, which are eventually eclipsed by more sophisticated features. Larger models have greater representational capacity, and so are better able to retain these early n-gram features.

In-context learning We follow Kaplan et al. (2020) in defining in-context learning as decreasing loss at increasing token indices. We find that loss is uniform across token positions in early training steps, but slowly decreases at increasing token indices in later steps, consistent with the presence of in-context learning (Fig. 7).

We observe an initial increase in loss early in each sequence. This is likely due to the fact unigram sequences are indistinguishable from real sequences at the first position, and bigram model predictions are indistinguishable from real sequences at the first and second positions.

5. Conclusion

We propose two criteria that operationalize what it means for models to exploit moments of a given order, then describe methods of generating synthetic data that test whether a network satisfies both criteria, using theoretically grounded approaches relying on optimal transport theory and the principle of maximum entropy. We extend our analysis to discrete sequences by proving a novel equivalence between n-gram statistics and statistical moments.

We find new compelling empirical evidence that neural networks learn to exploit the moments of their input distributions in increasing order, and further find "double descent" in the degree to which language model predictions match low-order data statistics, driven by in-context learning on longer sequences. Our contributions strengthen the case for the distributional simplicity bias (DSB), refine our understanding of how DSB influences early learning dynamics, and provide a foundation for further investigations into DSB.

Acknowledgements

We are thankful to Open Philanthropy for funding this work. We also thank New Science and Stability AI for providing computing resources. Finally, we thank Brennan Dury for

[16] Arguably, this pattern applies to models of all sizes on unigram sequences, but the tiny increases in loss for the larger models are within the margin of error for these experiments.

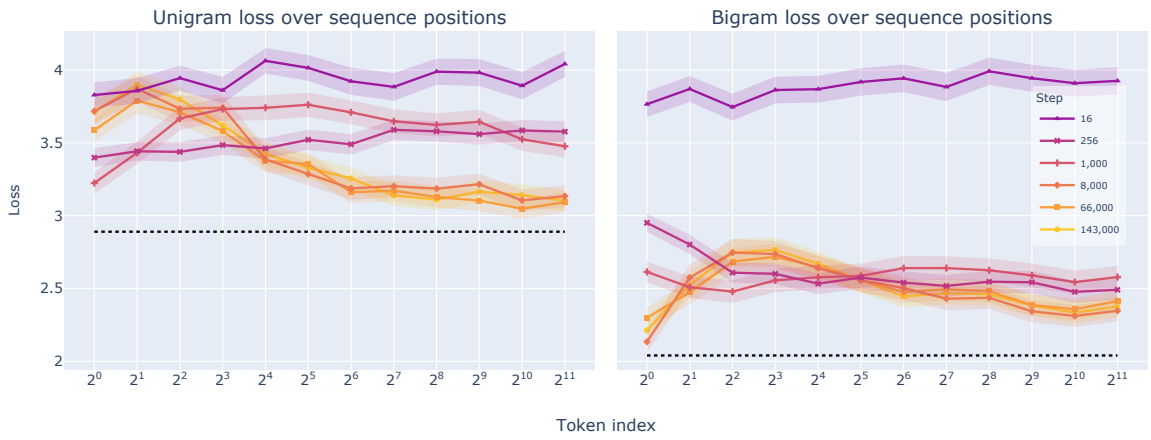


Figure 7. In-context learning of the maximum entropy bigram sequence occurs after step 1,000 in Pythia 12B. Some in-context learning of the maximum entropy unigram sequence occurs by step 1000, with more at step 143,000.

independently deriving the Conrad distribution.

Impact statement

The goal of this work was to advance our understanding of the generalization behavior of neural networks throughout training, in the hope that this will enable the development of more robust and predictable machine learning models.

References

Baratin, A., George, T., Laurent, C., Hjelm, R. D., Lajoie, G., Vincent, P., and Lacoste-Julien, S. Implicit regularization via neural feature alignment. In *International Conference on Artificial Intelligence and Statistics*, pp. 2269–2277. PMLR, 2021.

Basri, R., Galun, M., Geifman, A., Jacobs, D., Kasten, Y., and Kritchman, S. Frequency bias in neural networks for input of non-uniform density. In III, H. D. and Singh, A. (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 685–694. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/basri20a.html.

Belkin, M., Hsu, D., Ma, S., and Mandal, S. Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy of Sciences*, 116(32):15849–15854, 2019. doi: 10.1073/pnas.1903070116. URL https://www.pnas.org/doi/abs/10.1073/pnas.1903070116.

Biderman, S., Schoelkopf, H., Anthony, Q. G., Bradley, H., O'Brien, K., Hallahan, E., Khan, M. A., Purohit, S., Prashanth, U. S., Raff, E., et al. Pythia: A suite for analyzing large language models across training and scaling. In *International Conference on Machine Learning*, pp. 2397–2430. PMLR, 2023.

Bietti, A. and Mairal, J. On the inductive bias of neural tangent kernels. *Advances in Neural Information Processing Systems*, 32, 2019.

Bodnar, T., Gupta, A. K., and Parolya, N. On the strong convergence of the optimal linear shrinkage estimator for large dimensional covariance matrix. *Journal of Multivariate Analysis*, 132:215–228, 2014.

Canatar, A., Bordelon, B., and Pehlevan, C. Spectral bias and task-model alignment explain generalization in kernel regression and infinitely wide neural networks. *Nature communications*, 12(1):2914, 2021.

Carpenter, B. Typical sets and the curse of dimensionality. *Stan Software*, 2017.

Chiang, P., Ni, R., Miller, D. Y., Bansal, A., Geiping, J., Goldblum, M., and Goldstein, T. Loss landscapes are all you need: Neural network generalization can be explained without the implicit bias of gradient descent. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=QC10RmRbZy9.

Choshen, L., Hacohen, G., Weinshall, D., and Abend, O. The grammar-learning trajectories of neural language models. In Muresan, S., Nakov, P., and Villavicencio, A. (eds.), *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8281–8297, Dublin, Ireland, May 2022. Association for Computational Linguis-

tics. doi: 10.18653/v1/2022.acl-long.568. URL https://aclanthology.org/2022.acl-long.568.

Conrad, K. Probability distributions and maximum entropy. *Entropy*, 6(452):10, 2004.

Cubuk, E. D., Zoph, B., Shlens, J., and Le, Q. V. Randaugment: Practical automated data augmentation with a reduced search space. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pp. 702–703, 2020.

Dowson, D. and Landau, B. The fréchet distance between multivariate normal distributions. *Journal of multivariate analysis*, 12(3):450–455, 1982.

Dowson, D. and Wragg, A. Maximum-entropy distributions having prescribed first and second moments (corresp.). *IEEE Transactions on Information Theory*, 19(5):689–693, 1973.

Gao, L., Biderman, S., Black, S., Golding, L., Hoppe, T., Foster, C., Phang, J., He, H., Thite, A., Nabeshima, N., et al. The Pile: An 800GB dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020.

Goldblum, M., Souri, H., Ni, R., Shu, M., Prabhu, V. U., Somepalli, G., Chattopadhyay, P., Ibrahim, M., Bardes, A., Hoffman, J., et al. Battle of the backbones: A large-scale comparison of pretrained models across computer vision tasks. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023.

Hendrycks, D. and Gimpel, K. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

Huang, W. R., Emam, Z., Goldblum, M., Fowl, L., Terry, J. K., Huang, F., and Goldstein, T. Understanding generalization through visualizations. In Zosa Forde, J., Ruiz, F., Pradier, M. F., and Schein, A. (eds.), *Proceedings on "I Can't Believe It's Not Better!" at NeurIPS Workshops*, volume 137 of *Proceedings of Machine Learning Research*, pp. 87–97. PMLR, 12 Dec 2020. URL https://proceedings.mlr.press/v137/huang20a.html.

Jacot, A., Gabriel, F., and Hongler, C. Neural tangent kernel: Convergence and generalization in neural networks. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/file/5a4be1fa34e62bb8a6ec6b91d2462f5a-Paper.pdf.

Jaynes, E. T. Information theory and statistical mechanics. *Physical review*, 106(4):620, 1957.

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., and Amodei, D. Scaling laws for neural language models, 2020.

Kozachenko, L. F. and Leonenko, N. N. Sample estimate of the entropy of a random vector. *Problemy Peredachi Informatsii*, 23(2):9–16, 1987.

Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. 2009.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Lee, J., Sohl-dickstein, J., Pennington, J., Novak, R., Schoenholz, S., and Bahri, Y. Deep neural networks as gaussian processes. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=B1EA-M-0Z.

Liu, Z., Hu, H., Lin, Y., Yao, Z., Xie, Z., Wei, Y., Ning, J., Cao, Y., Zhang, Z., Dong, L., et al. Swin transformer v2: Scaling up capacity and resolution. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 12009–12019, 2022.

Loshchilov, I. and Hutter, F. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2018.

Ma, J. and Yarats, D. On the adequacy of untuned warmup for adaptive optimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 8828–8836, 2021.

Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., and Ng, A. Y. Reading digits in natural images with unsupervised feature learning. 2011.

Olsson, C., Elhage, N., Nanda, N., Joseph, N., DasSarma, N., Henighan, T., Mann, B., Askell, A., Bai, Y., Chen, A., Conerly, T., Drain, D., Ganguli, D., Hatfield-Dodds, Z., Hernandez, D., Johnston, S., Jones, A., Kernion, J., Lovitt, L., Ndousse, K., Amodei, D., Brown, T., Clark, J., Kaplan, J., McCandlish, S., and Olah, C. In-context learning and induction heads. *Transformer Circuits Thread*, 2022. https://transformer-circuits.pub/2022/in-context-learning-and-induction-heads/index.html.

Radosavovic, I., Kosaraju, R. P., Girshick, R., He, K., and Dollár, P. Designing network design spaces. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10428–10436, 2020.

Rahaman, N., Baratin, A., Arpit, D., Draxler, F., Lin, M., Hamprecht, F., Bengio, Y., and Courville, A. On the spectral bias of neural networks. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 5301–5310. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/rahaman19a.html.

Refinetti, M., Ingrosso, A., and Goldt, S. Neural networks trained with sgd learn distributions of increasing complexity. In *International Conference on Machine Learning*, pp. 28843–28863. PMLR, 2023.

Sablayrolles, A., Douze, M., Schmid, C., and Jégou, H. Spreading vectors for similarity search. In *International Conference on Learning Representations*, 2018.

Santambrogio, F. Optimal transport for applied mathematicians. *Birkäuser, NY*, 55(58-63):94, 2015.

Sklar, M. Fonctions de répartition à n dimensions et leurs marges. In *Annales de l'ISUP*, volume 8, pp. 229–231, 1959.

Valle-Perez, G., Camargo, C. Q., and Louis, A. A. Deep learning generalizes because the parameter-function map is biased towards simple functions. In *International Conference on Learning Representations*, 2018.

Vallet, F., Cailton, J.-G., and Refregier, P. Linear and nonlinear extension of the pseudo-inverse solution for learning boolean functions. *Europhysics Letters*, 9(4):315, 1989.

Woo, S., Debnath, S., Hu, R., Chen, X., Liu, Z., Kweon, I. S., and Xie, S. Convnext v2: Co-designing and scaling convnets with masked autoencoders. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16133–16142, 2023.

Xiao, H., Rasul, K., and Vollgraf, R. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

Xu, Z. J. and Zhou, H. Deep frequency principle towards understanding why deeper learning is faster. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(12):10541–10550, May 2021. doi: 10.1609/aaai.v35i12.17261. URL https://ojs.aaai.org/index.php/AAAI/article/view/17261.

Xu, Z.-Q. J., Zhang, Y., Luo, T., Xiao, Y., and Ma, Z. Frequency principle: Fourier analysis sheds light on deep neural networks. *arXiv preprint arXiv:1901.06523*, 2019a.

Xu, Z.-Q. J., Zhang, Y., and Xiao, Y. Training behavior of deep neural network in frequency domain. In *International Conference on Neural Information Processing*, pp. 264–274, 2019b.

Zhang, C., Bengio, S., Hardt, M., Recht, B., and Vinyals, O. Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM*, 64(3):107–115, 2021.

A. Additional Related Work

Extensive prior work has investigated neural network simplicity bias and learning dynamics. We highlight several prior research directions that usefully contrast our own approach.

A.1. Simplicity bias

One common approach studies simplicity biases in the parameter-function maps of neural network architectures. Such explanations posit that neural networks implement favorable priors, meaning that most network parameterizations, under commonly used initialization distributions, that reach good performance on the training data will also generalize to the test data, regardless of specific details about the optimization process used to find such parameterizations.

Valle-Perez et al. (2018) investigated such architectural simplicity biases by using Gaussian process-based approximations to neural networks (Lee et al., 2018) to estimate the Bayesian posterior produced by randomly sampling neural network parameterizations, conditional on those networks achieving perfect training loss, and showed the resulting posterior correlated well with the odds of SGD-based training finding a given function. Chiang et al. (2023) validate this perspective by showing that a variety of non-gradient based optimizers, including unbiased sampling of random initializations, are still able to generalize from training to testing data.

Another approach is to construct a simplified, theoretically tractable model of neural network learning dynamics, then analyzing the resulting model to find which types of functions it predicts networks will be most inclined to learn. The neural tangent kernel (Jacot et al., 2018), scales network widths to infinity, whereupon networks are limited to performing kernel regression with their initialization kernel. Thus, model inductive biases are determined by the spectrum of the initialization kernel's eigenfunctions, which have strong simplicity biases for commonly used architectures (Canatar et al., 2021; Baratin et al., 2021; Bietti & Mairal, 2019).

A.2. Learning order

Xu et al. (2019b) proposed the Frequency Principle, the tendency of neural networks to first fit low-frequency Fourier components of a given target function, before moving on to fit higher frequency components, and empirically demonstrated this tendency on real image classification problems and synthetic datasets. Subsequent works further explored how neural network learning dynamics relate to the representation of training data in the frequency domain (Rahaman et al., 2019; Xu et al., 2019a; Basri et al., 2020; Xu & Zhou, 2021). Our work is similar in that we also aim to connect neural network learning order to simple mathematical properties of the training data, though we use distributional statistics, rather than frequency.

Choshen et al. (2022) empirically studied learning dynamics of neural language models by tracking which grammatical patterns different networks learn to model across their training trajectories, and comparing network behavior across training to alternative language modeling approaches, such as n-gram models. They found that neural language models initially match the behaviors of unigram and bigram models early in training, then diverge as training progresses. These results are inline with our own findings on learning order in neural language models, and are consistent with a DSB-driven perspective on neural network learning dynamics.

B. CIFARNet dataset

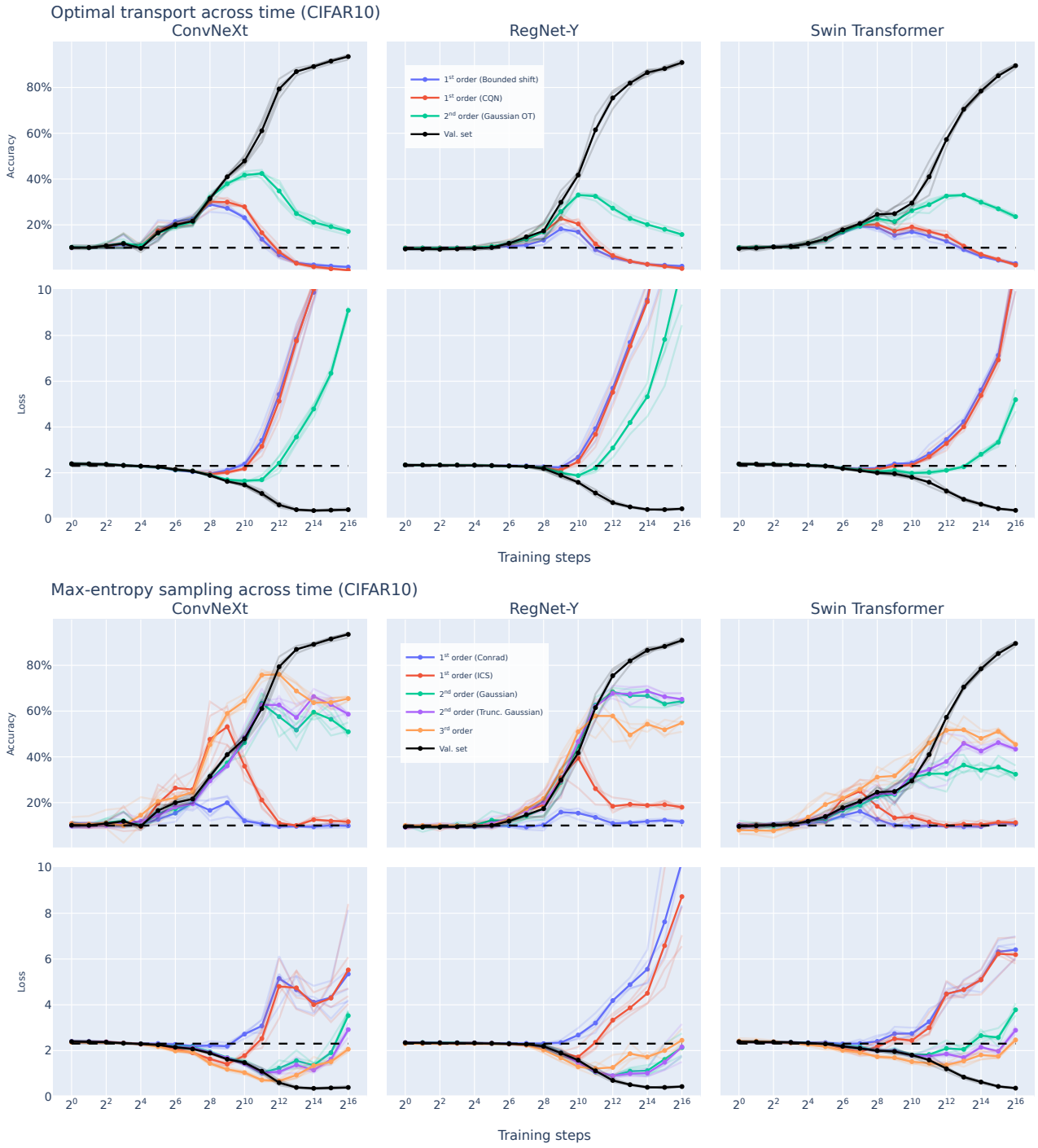
CIFARNet is based on the Winter 2019 version of ImageNet-21K. We selected the ten synsets from the ImageNet hierarchy which most closely matched the ten CIFAR-10 classes, with a bias toward broader synsets to maximize the dataset size:

- Airplane: `n02691156`
- Automobile: `n02958343`
- Bird: `n01503061`
- Cat: `n02121620`
- Deer: `n02430045`
- Dog: `n02083346`
- Frog: `n01639765`
- Horse: `n02374451`
- Ship: `n04194289`
- Truck: `n04490091`

We ensured class balance by randomly sampling 20K images from each synset. Images were directly resized to 64×64 resolution without center cropping.

C. Detailed experimental results

C.1. CIFAR-10



C.2. CIFARNet

Figure 8. These results qualitatively mirror those of the lower resolution CIFAR-10 dataset (see above), except that the maximum accuracies attained on 2^{nd} order samples are somewhat lower. This may suggest that networks more quickly learn to use higher-order statistics when the input has higher dimensionality.

C.3. Fashion MNIST

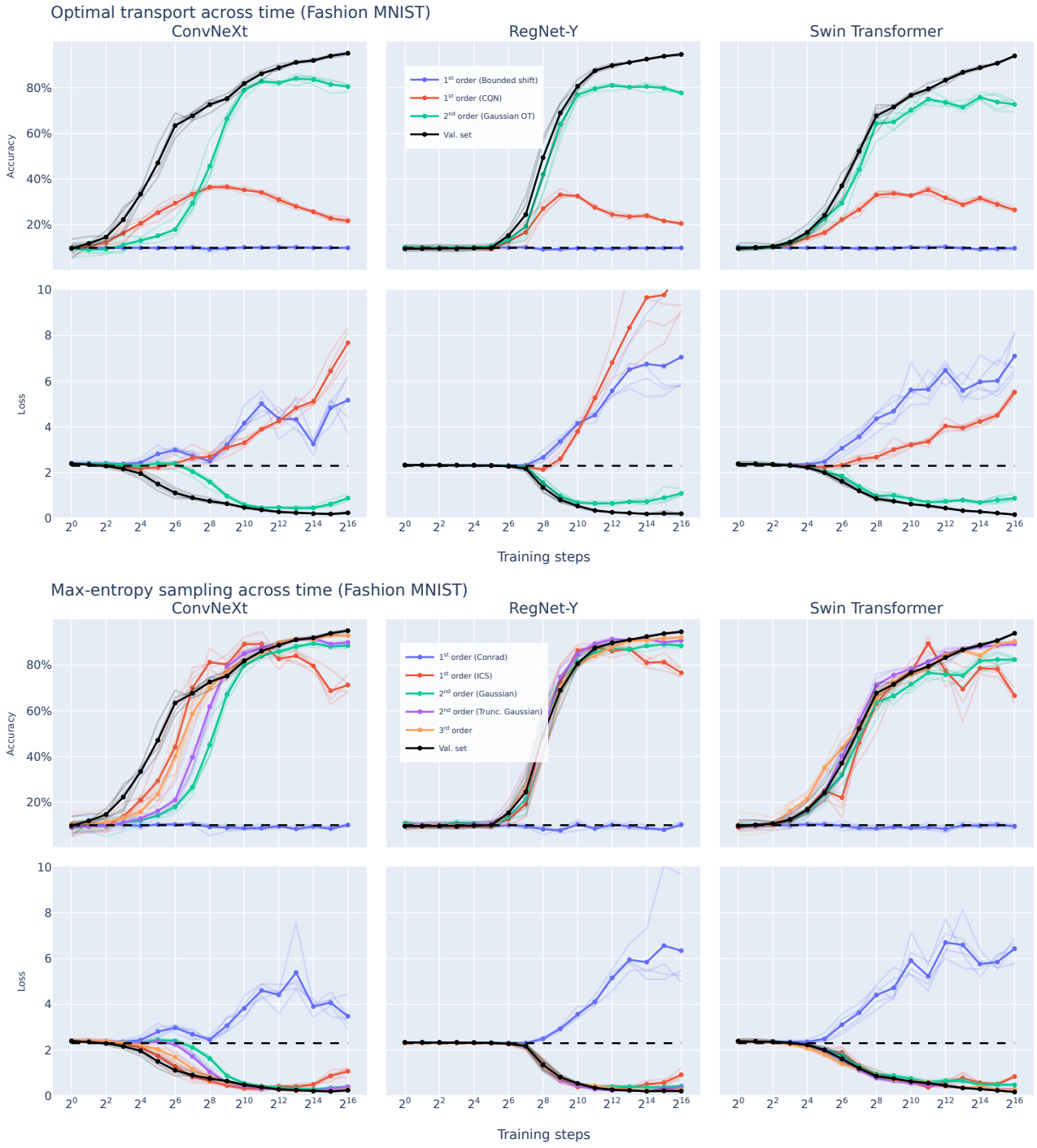


Figure 9. Fashion MNIST learning curves exhibit only a modest degree of non-monotonicity, likely because the first and second moments of the data distribution are sufficient to produce very realistic-looking samples (Fig. 2)

C.4. MNIST

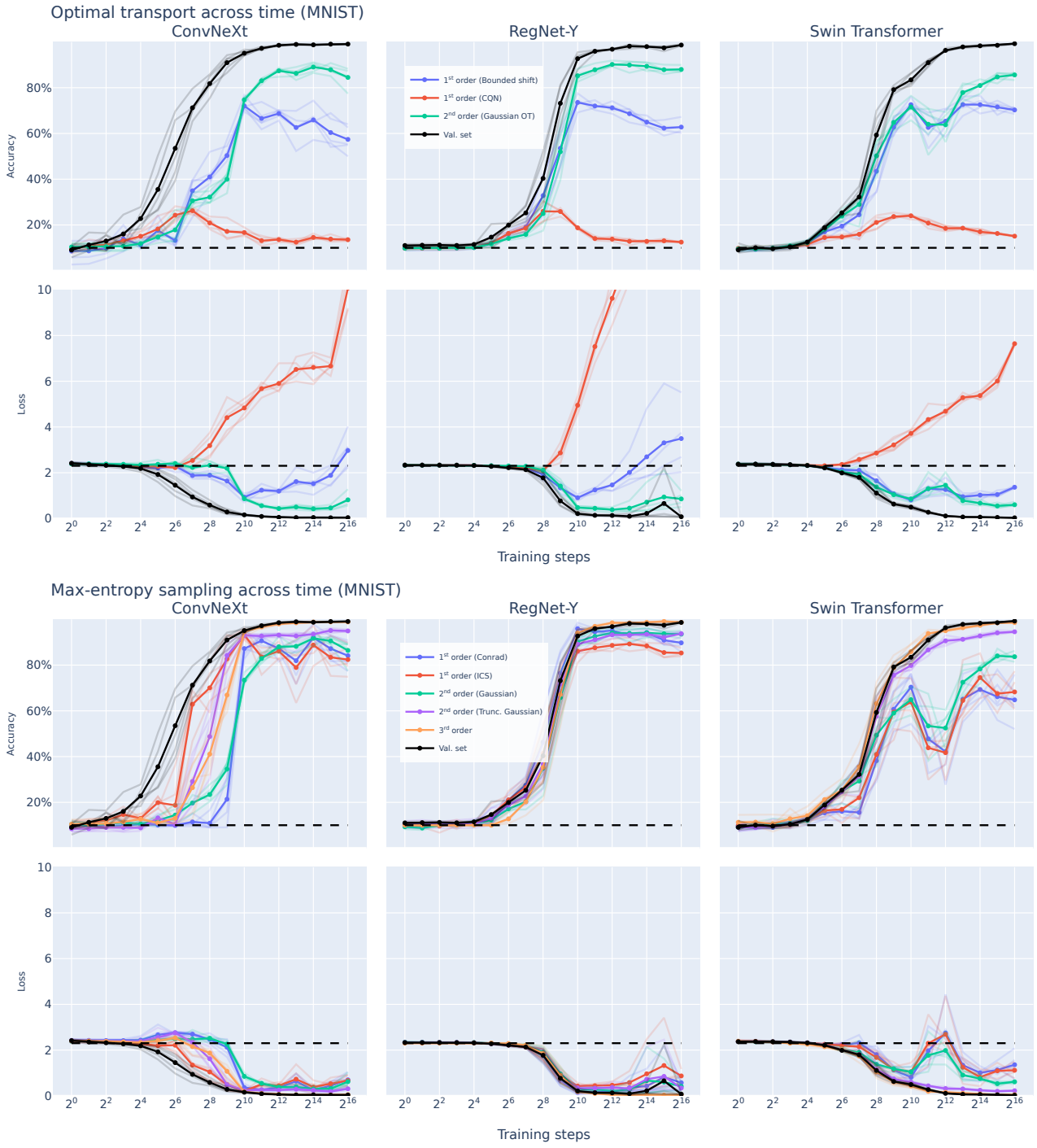


Figure 10. MNIST learning curves exhibit only a modest degree of non-monotonicity, likely because the first and second moments of the data distribution are sufficient to produce very realistic-looking samples (Fig. 2)

D. Note on third-order results

We tried to use a variant of the algorithm described in Appendix H for 3rd order max entropy sampling. However, the samples invariably failed to match the desired 3rd order statistics sufficiently to trigger the gradient 10^{-7} gradient tolerance stopping condition. We believe this is due to the very large number of entries in the coskewness tensor leading to more complex constraints on the acceptable values of the synthetic dataset. In comparison, we have a far smaller number of free parameters in the synthetic dataset. For example, for CIFAR-10 we optimized one thousand samples, each with $3 \times 32 \times 32 = 3072$ coordinates, for a total of 3×10^6 parameters. By comparison, the coskewness tensor has $3072^3 \approx 2.9 \times 10^{10}$ entries, each one placing an additional constraint on the dataset. It may be possible to generate higher quality samples by increasing the number of synthetic datapoints, but this would further increase the computational burden, which is already quite high in the 3rd order case (Appendix I).

Nevertheless, in the interest of completeness and transparency, we report our 3rd order results in Appendix C. These results are difficult to interpret because they are generated using synthetic data whose 3rd order statistics only *approximately* match the desired ones. We thus place little weight on them. However, we find it interesting that even partially matching the coskewness tensor will improve model performance on the synthetic data in most settings, with the notable exceptions of the SVHN dataset and RegNet-Y model, where the 3rd order datasets either consistently underperform their 2nd order counterparts on all models (in the case of the SVHN dataset) or selectively underperform on the CIFAR10 and CIFARNet datasets (in the case of the RegNet-Y model). We thus see that a partial match of 3rd order statistics improves performance in 10 out of the 15 model and dataset pairs we investigate.

D.1. Street View Housing Numbers

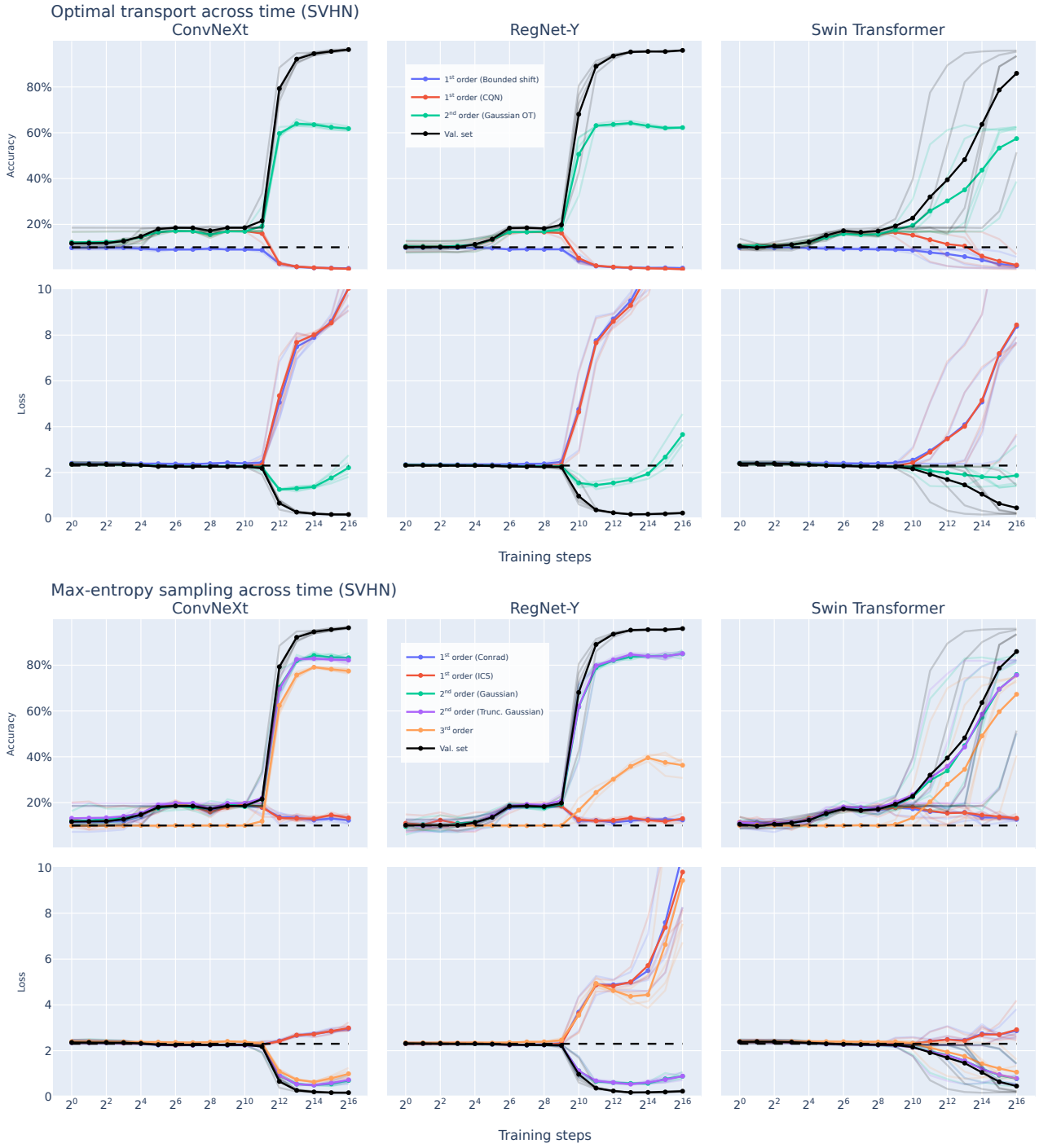


Figure 11. The Street View Housing Numbers dataset is somewhat of an outlier in that none of the models ever exceed random baseline accuracy on 1st order synthetic images. We hypothesize this is because of the extreme diversity of colors, fonts, and background textures in SVHN, which make "simple" first order features less discriminative for classifying digits. We also found it necessary to use a smaller learning rate to achieve convergence on this dataset (Footnote 14).

D.2. Pythia Language Models

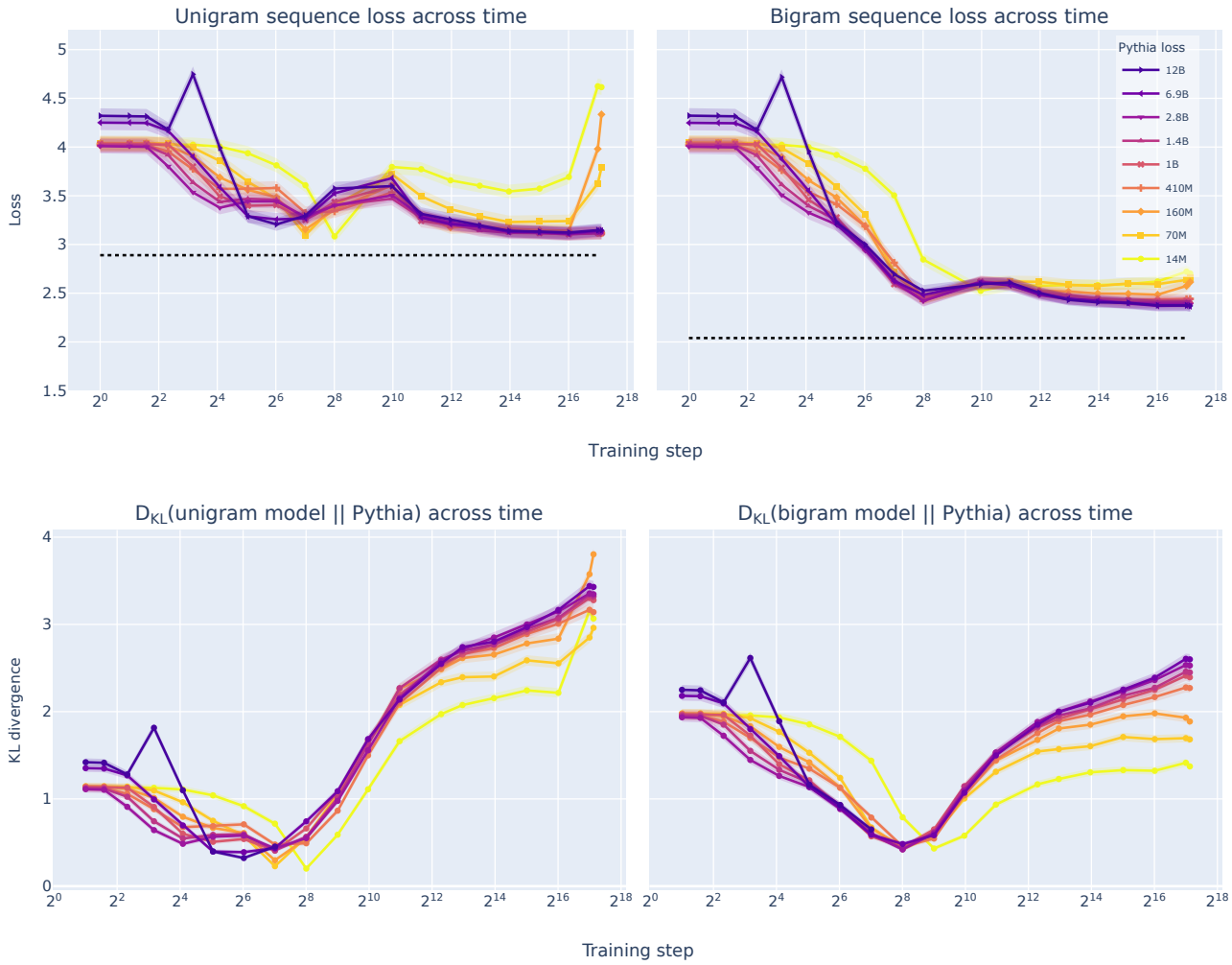


Figure 12. **(top)** Average cross entropy loss of Pythia models evaluated on unigram and bigram sequences, **(bottom)** KL divergence between the predictions of our n-gram language models and the predictions of Pythia checkpoints when evaluated on chunks of tokens sampled from the Pile ($N = 1024$.)

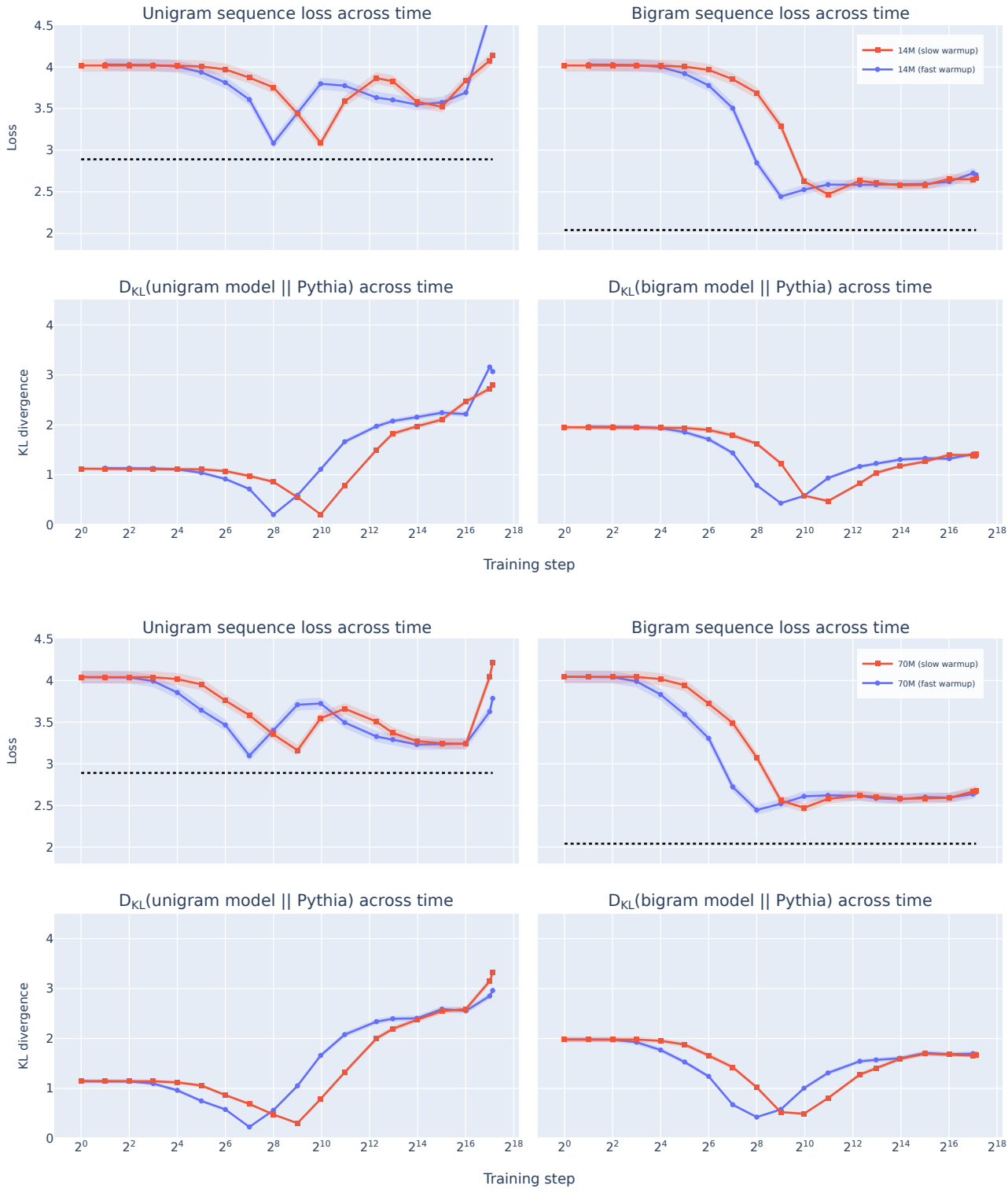


Figure 13. Effects of fast and slow learning rate warmup on n-gram sequence loss and KL divergence, Pythia 14M and 70M.

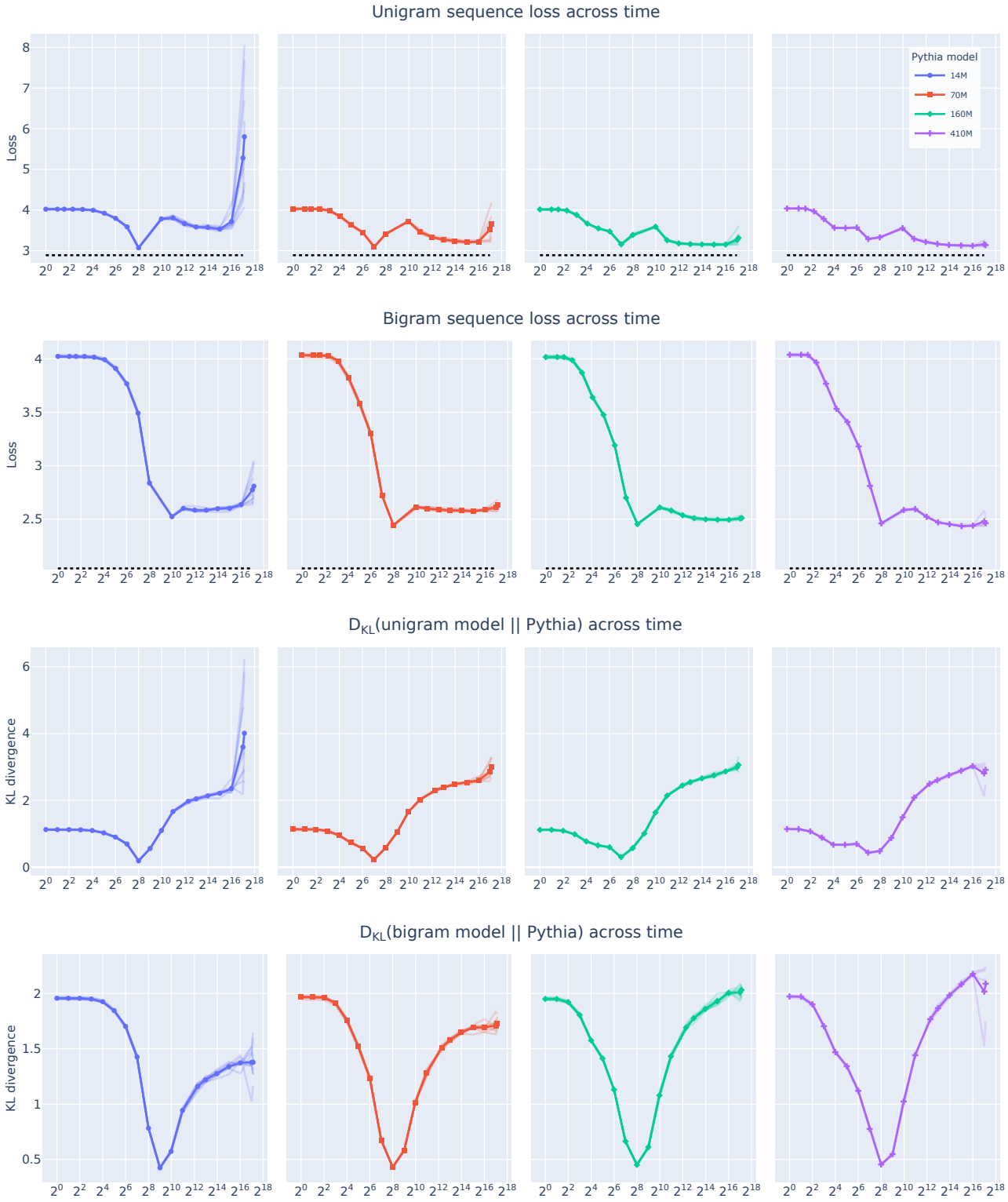


Figure 14. N-gram sequence loss and KL divergence over Pythia sizes and random seeds (1024 sequences sampled at each step, 9 seeds for Pythia 14M, 70M and 160M, 4 seeds for Pythia 410M.)

E. First order optimal transport under a boundary constraint

We would like to surgically change the mean of a set of images while keeping their pixel intensities constrained to the range $[0, 1]$. The least-squares optimal algorithm for this task is described in Alg. 1, and we prove its correctness in the following theorem.

Theorem E.1. *Let* $\mathbf{x}$ *be a vector in* $[0, 1]^n$ *and let* $m \in [0, 1]$ *be a desired mean. Then the optimization problem*

$$\min_{\mathbf{y}\in[0,1]^n} \|\mathbf{x} - \mathbf{y}\|_2^2 \quad \text{s.t. } \frac{1}{n}\sum_{i=1}^{n} y_i = m$$

has a unique solution given by Algorithm 1.

Proof. Let $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$. If $\bar{x} = m$, we immediately have the optimal solution $\mathbf{y}^* = \mathbf{x}$, because our constraints are already satisfied and we achieve zero loss by leaving $\mathbf{x}$ unchanged.

Algorithm 1 Optimal constrained mean shift

Require: Input vector $\mathbf{x} \in [0, 1]^n$
Ensure: Desired mean $m \in [0, 1]$
Sort the coordinates of $\mathbf{x}$
$\bar{x} \leftarrow \sum_{i=1}^{n} x_i$
$\mathbf{y} \leftarrow \mathbf{0}_n$
for $i \in 1 \dots n$ **do**
  $y_i \leftarrow x_i + m - \bar{x}$
  if $y_i > 1$ **then**
    $\bar{x} \leftarrow \sum_{j=i}^{n} x_j$
    $y_i \leftarrow 1$
    $m \leftarrow \frac{nm-i}{n-i}$
Put coordinates of $\mathbf{y}$ in their original order
return $\mathbf{y}$

We can reduce the case where $m < \bar{x}$ to the case where $\bar{x} < m$ (or vice versa) by exploiting the reflection-symmetry of the problem. Specifically, if the solution to the analogous problem in $\mathbf{x}'$ and m', where $\mathbf{x}' = \frac{1}{2} - \mathbf{x}$ and $m' = \frac{1}{2} - m$, is $\mathbf{y}'$, then the solution to the original problem is $\mathbf{y}^* = \frac{1}{2} - \mathbf{y}'$. This is due to the reflection-invariance of the Euclidean distance, the linearity of our mean constraint, and the fact that reflecting about $\frac{1}{2}$ cannot move points in $[0, 1]^n$ outside $[0, 1]^n$. Because of this symmetry, in what follows we will assume without loss of generality that $\bar{x} < m$.

Note also the optimal solution must have the property that $\forall i : x_i \leq y_i$. Assume for the sake of contradiction that $x_i > y_i$ for some i. Then we can improve upon this solution by increasing y_i by some $\epsilon > 0$, and compensating for this by moving another entry y_j for which $x_j < y_j$ closer to its original value by ϵ.

Setting up the Lagrangian. Using the Karush-Kuhn-Tucker conditions, we encode the problem with the Lagrangian

$$\mathcal{L}(\mathbf{y}, \lambda, \boldsymbol{\mu}, \boldsymbol{\nu}) = \sum_{i=1}^{n}(y_i - x_i)^2 + \underbrace{\lambda\Big(\frac{1}{n}\sum_{i=1}^{n} y_i - m\Big)}_{\text{mean constraint}} - \underbrace{\sum_{i=1}^{n}\mu_i y_i + \sum_{i=1}^{n}\nu_i(y_i - 1)}_{\text{inequality constraints}}. \tag{4}$$

Differentiating $\mathcal{L}$ with respect to y_i yields the stationarity condition

$$\frac{\partial \mathcal{L}}{\partial y_i} = 2(y_i - x_i) + \frac{\lambda}{n} - \mu_i + \nu_i = 0. \tag{5}$$

The KKT complementary slackness condition requires that $\mu_i y_i = 0$ and $\nu_i(y_i - 1) = 0$ for each i. This implies that μ_i must be zero if $y_i > 0$, and ν_i must be zero if $y_i < 1$. For each i where $y_i < 1$, we can use Eq. 5 and complementary slackness to write y_i as $x_i - \frac{\lambda}{2n}$.

Putting it all together. Assume $\mathbf{x}$ and $\mathbf{y}$ are written in a basis that ensures the coordinates of $\mathbf{x}$ are sorted in descending order, so that $x_1 \geq x_2 \geq \ldots \geq x_n$. Our problem is invariant to permutation of indices, so this does not affect the solution.

We can now solve for y_1, the final position of the largest coordinate, in the following way. Suppose that $y_1 < 1$. Then we have $\forall i : y_i < 1$, and the mean constraint can be written as $\frac{1}{n}(\sum_{i=1}^{n} x_i) - \frac{\lambda}{2n} = m$. This allows us to solve for all y_i:

$$y_i = x_i - \frac{\lambda}{2n} = x_i + m - \frac{1}{n}(\sum_{i=1}^{n} x_i). \tag{6}$$

Note Eq. 6 may "overshoot" and violate the inequality constraint $y_i \leq 1$. If it does, then we know our supposition is false and $y_1 = 1$. If it does not violate the constraint, then it must be optimal because it is also the solution to the relaxed version of this problem without the $[0, 1]$ constraint. In the latter case, we are done.

Given that $y_1 = 1$, the subproblem of solving for $y_2, \ldots, y_n$ is a smaller instance of the original problem: the target mean for these coordinates is $m' = \frac{nm-1}{n-1}$. We can recursively apply this reasoning to solve for all other y_i. This procedure coincides with Algorithm 1. □

F. Derivation of the Dury distribution

Theorem F.1. *Among all distributions supported on* $[0,1]$ *with desired mean* $m \neq \frac{1}{2}$*, the Dury distribution with density* $p(x) = \frac{b\exp(-bx+b)}{\exp(b)-1}$ *has maximum entropy, where the parameter* $b \neq 0$ *is chosen to satisfy the equation* $m = -\frac{b-\exp(b)+1}{b(\exp(b)-1)}$*. In the special case of* $m = \frac{1}{2}$*, the maximum entropy distribution is* $\mathrm{Unif}(0,1)$*.*

Proof. Consider the density function $p(x) = \exp(-a - bx)$, where a and b are selected to satisfy normalization $\int_{[0,1]} p(x)dx = 1$ and mean $\int_{[0,1]} p(x)xdx = m$ constraints, and another arbitrary density $q(x)$ which satisfies the same constraints. We will show that the entropy of q can be no greater than the entropy of p.

$$\begin{aligned} H(q) &\leq H(q,p) && \text{(inequality of entropy and cross-entropy)} \\ &= \mathbb{E}_q[\log p(x)] && \text{(definition of cross-entropy)} \\ &= -a - bm && \text{(definition of } p(x) \text{ and linearity)} \\ &= H(p) && \text{(QED)} \end{aligned}$$

We can now analytically solve for a in terms of b. Integrating $p(x)$ from 0 to 1 yields $-\frac{\exp(-a-b)}{b} + \frac{\exp(-a)}{b} = 1$. Solving for a we get $a = -b + \log\left(\frac{\exp(b)-1}{b}\right)$, which when plugged back into the original formula gives us $p(x) = \frac{b\exp(-bx+b)}{\exp(b)-1}$.

Integration by parts yields the following formula for the mean: $\int_{[0,1]} p(x)xdx = -\frac{b-\exp(b)+1}{b(\exp(b)-1)}$. We can use a root-finding algorithm such as Newton's method to solve this expression for b given a desired mean m. Note, however, that there is a singularity in the mean formula where $b = 0$. Applying l'Hôpital's rule twice yields the limit:

$$\lim_{b\to 0} -\frac{b-\exp(b)+1}{b(\exp(b)-1)} = \lim_{b\to 0} \frac{\exp(b)-1}{b\exp(b)+\exp(b)-1} = \lim_{b\to 0} \frac{\exp(b)}{b\exp(b)+2\exp(b)} = \frac{1}{2}. \tag{7}$$

The maximum entropy distribution supported on $[0,1]$ with no mean constraint is known to be $\mathrm{Unif}(0,1)$. Since it happens to have the mean $\frac{1}{2}$, we may conclude that the Dury distribution approaches $\mathrm{Unif}(0,1)$ as b approaches 0. □

G. Discrete domain proofs

Please refer to Section 2.5 for context pertinent to this section.

Definition G.1 (n-gram statistic)**.** Let $\mathcal{V}^N$ be the set of token sequences of length N drawn from a finite vocabulary $\mathcal{V}$. Given some distribution over $\mathcal{V}^N$, an ***n*-gram statistic** is the probability that an n-tuple of tokens $(v_1, \ldots, v_n) \in \mathcal{V}^n$ will co-occur at a set of unique indices $(t_1, \ldots t_n) \in \mathbb{N}^n$.

Theorem 2.1. *[n-gram statistics are moments] Let $\mathcal{V}^N$ be the set of token sequences of length N drawn from a finite vocabulary $\mathcal{V}$, let P be a distribution on $\mathcal{V}^N$, and let $f : \mathcal{V}^N \to \{0,1\}^{N\cdot|\mathcal{V}|}$ be the function that encodes a length-N sequence of tokens as a flattened concatenation of N one-hot vectors of dimension $|\mathcal{V}|$. Let $f_\sharp P$ be the pushforward of P through this one-hot encoding, i.e. its analogue in $\{0,1\}^{N\cdot|\mathcal{V}|}$.*

Then every moment of $f_\sharp P$ is equal to an n-gram statistic of P and vice versa.

Proof. While it is natural to view one-hot sequences as Boolean matrices of shape $N \times |\mathcal{V}|$, where each row corresponds to a sequence position, we instead consider *flattened* one-hot encodings in order to make use of the standard mathematical machinery for moments of random vectors.

In this flattened representation, the component at index i indicates whether the token at $t(i)$ is equal to the token $v(i)$, where $(t(i), v(i)) := \mathrm{divmod}(i, |\mathcal{V}|)$. For example, if $\mathcal{V} = \{$"apple", "pear"$\}$ and $N = 3$, the sequence "apple apple pear" will be encoded as the vector $(1, 0 \mid 1, 0 \mid 0, 1)$:

$$\underbrace{\text{apple}}_{(1,0)}\ \underbrace{\text{apple}}_{(1,0)}\ \underbrace{\text{pear}}_{(0,1)} \xrightarrow{f} (1, 0, 1, 0, 0, 1).$$

Now consider the moment corresponding to some arbitrary multi-index $\alpha \in \mathbb{N}^{N\cdot|\mathcal{V}|}$. For illustration, let $\alpha = (0, 2 \,|\, 0, 0 \,|\, 1, 0)$. Then the corresponding moment is

$$\mathbb{E}[f(x)^\alpha] = \mathbb{E}[\underbrace{\cancel{f(x)_1^0} f(x)_2^2}_{(0,2} \underbrace{\cancel{f(x)_3^0} \cancel{f(x)_4^0}}_{0,0} \underbrace{f(x)_5^1 \cancel{f(x)_6^0}}_{1,0)}], \tag{8}$$

where $f(x)_1^0$ denotes the first component of $f(x)$ raised to the power 0. Since each component of $f(x)$ is a Boolean indicator for the presence or absence of a vocabulary item at a given position, we can rewrite it with Iverson brackets:

$$= \mathbb{E}\big[[x(1) = \text{"pear"}]^2 \cdot [x(3) = \text{"apple"}]\big] \tag{9}$$
$$= \mathbb{P}[x(1) = \text{"pear"} \wedge x(3) = \text{"apple"}], \tag{10}$$

or the probability that the first and third tokens will be "pear" and "apple" respectively. Note that the exponent on the $[x(1) = \text{"pear"}]$ makes no difference here as long as it is nonzero. We can always binarize α, replacing all nonzero values with 1, and the moment will be unchanged since any nonzero power of $\{0, 1\}$ is still $\{0, 1\}$.

In general, since the coordinates are all Booleans in $\{0, 1\}$, multiplication corresponds to logical conjunction and expectation corresponds to probability:

$$\mathbb{E}[f(x)^\alpha] = \mathbb{E}\Big[\prod_{i=1}^{N} f(x)_i^{\alpha_i}\Big] = \mathbb{P}\Big[\bigwedge_{i\in A} x(t_i) = v_i\Big], \tag{11}$$

where A is the set of indices in $1 \ldots |\mathcal{V}| \times N$ where α is nonzero. By Def. G.1, this probability is an n-gram statistic of order $k = |A|$.

Conversely, we can convert any an n-gram statistic with tokens in $\mathcal{V}^n$ and sequence positions in $\mathbb{N}^n$ into a moment of $f_\sharp P$ by first flattening the indices, then plugging them into Eq. 11. The sequence positions correspond to rows, and the tokens correspond to columns, of a one-hot matrix representation of a sequence. Here we need to multiply the row and column indices together to yield indices into the flattened vector.

There will be infinitely many moments which correspond to any given n-gram, because multi-indices with components larger than one are redundant. □

Theorem 2.2. *[Equal embedding moments] Let $\mathbf{E} \in \mathbb{R}^{|\mathcal{V}| \times d}$ be an embedding matrix, and let P and Q be two distributions over $\mathcal{V}^N$. Then if P and Q have the same n-gram statistics up to order $k \geq 1$, their embeddings under $\mathbf{E}$ have the same moments up to order k.*

Proof. By Thm. 2.1, we know that the one-hot analogues of P and Q, i.e. $f_\sharp P$ and $f_\sharp Q$, have equal moments up to order k. That is, for every multi-index $\alpha \in \mathbb{N}^{N \cdot |\mathcal{V}|}$ where $|\alpha| \leq k$,

$$\mathbb{E}_{\mathbf{x} \sim f_\sharp P}[\mathbf{x}^\alpha] = \mathbb{E}_{\mathbf{x} \sim f_\sharp Q}[\mathbf{x}^\alpha]. \tag{12}$$

Now let $g : \{0,1\}^{|\mathcal{V}| \times N} \to \mathbb{R}^{d \times N}$ be the function that multiplies each one-hot vector in a sequence by $\mathbf{E}$, returning a sequence of embedding vectors. Each side of Eq. 12 is the expectation of a polynomial in the components of $\mathbf{x}$, and since g is a linear map, $g(\mathbf{x})^\alpha$ is also a polynomial with the same degree.

Now consider $(g \circ f)_\sharp P$ and $(g \circ f)_\sharp Q$, the analogues of P and Q in embedding space. Its moments take the form

$$\mathbb{E}_{\mathbf{x} \sim (g \circ f)_\sharp P}[\mathbf{x}^\alpha] = \mathbb{E}_{\mathbf{x} \sim f_\sharp P}[g(\mathbf{x})^\alpha]. \tag{13}$$

Because $g(\mathbf{x})^\alpha$ is a polynomial with degree $|\alpha| \leq k$, the expectation $\mathbb{E}_{\mathbf{x} \sim f_\sharp P}[g(\mathbf{x})^\alpha]$ must be a linear combination of moments of $f_\sharp P$ with order no greater than k. But by Eq. 12, all of these moments are equal between P and Q, and hence all the moments of $(g \circ f)_\sharp P$ and $(g \circ f)_\sharp Q$ up to order k must be equal. □

H. Sampling from maximum entropy distributions with complex constraints

We used the following code to generate maximum entropy samples subject to a mean, covariance, and hypercube constraint. 3rd order sampling code is similar but more complex, as it includes a form of "minibatching" for the entries of the coskewness tensor in order to reduce memory usage.

```
from torch import nn, optim, Tensor
import torch


def koleo(x: Tensor) -> Tensor:
    """Kozachenko-Leonenko estimator of entropy."""
    return torch.cdist(x, x).kthvalue(2).values.log().mean()


def psd_sqrt(A: Tensor) -> Tensor:
    """Compute the unique p.s.d. square root of a positive semidefinite matrix."""
    L, U = torch.linalg.eigh(A)
    L = L[..., None, :].clamp_min(0.0)
    return U * L.sqrt() @ U.mH


def hypercube_sample(
    n: int,
    mean: Tensor,
    cov: Tensor,
    *,
    koleo_weight: float = 1e-3,
    max_iter: int = 100,
    seed: int = 0,
):
    """Generate `n` samples from max-ent distribution on [0, 1]^d with given moments."""
    d = mean.shape[-1]
    eps = torch.finfo(mean.dtype).eps
    rng = torch.Generator(device=mean.device).manual_seed(seed)

    # Initialize with max-ent samples matching `mean` and `cov` but without hypercube
    # constraint. We do so in a way that is robust to singular `cov`
    z = mean.new_empty([n, d]).normal_(generator=rng)
    x = torch.clamp(z @ psd_sqrt(cov) + mean, eps, 1 - eps)    # Project into hypercube

    # Reparametrize to enforce hypercube constraint
    z = nn.Parameter(x.logit())
    opt = optim.LBFGS([z], line_search_fn="strong_wolfe", max_iter=max_iter)

    def closure():
        opt.zero_grad()
        x = z.sigmoid()

        loss = torch.norm(x.mean(0) - mean) + torch.norm(x.T.cov() - cov)
        loss -= koleo_weight * koleo(x)
        loss.backward()
        return float(loss)

    opt.step(closure)
    return z.sigmoid().detach()
```

I. Computational requirements

At the scales of the datasets we use in this study, both maximum entropy second order hypercube-constrained sampling and Gaussian optimal transport are extremely cheap to run. In the most expensive configuration (generating around 200K 64×64 CIFARNet images), the optimization loop takes roughly 65 seconds on a single NVIDIA L40 GPU, while requiring approximately 29 gigabytes of GPU memory. Based on hourly pricing of $1.10 per hour from vast.ai, this will cost around

$0.02 to generate a full set of synthetic CIFARNet images, with all other first and second order methods described in this paper requiring fewer computational resources than that.

However, the memory required for both hypercube-constrained sampling and Gaussian optimal transport rise with the square of the number of image features (meaning the fourth power of the image size). The compute requirements of the hypercube sampling also rise with the fourth power of the image size, while the requirements for Gaussian optimal transport rise with the sixth power. This means our methods can quickly become computationally infeasible with larger image sizes, which is why we limit ourselves to at most 64×64 images in this study.

Additionally, the maximum entropy *third order* hypercube-constrained sampling is much more expensive than the second order methods, since the size of the statistic tensor grows as $O(d^{order})$. This means the coskewness tensor for CIFARNet images has dimensions $12288 \times 12288 \times 12288$. This would require nearly eight terabytes to store in full precision, which exceeds the memory capacity of our computing hardware by a significant degree.

We therefore want to generate fake data that matches CIFARNet's coskewness statistics, without ever computing those statistics in full. Each step of our optimization process for generating third order fake thus only computes the coskewness statistics along matching length l slices of coskewness tensors of the fake and real data, meaning we only need to store two $12288 \times 12288 \times l$ tensors at each step of optimization.

CIFARNet is the most expensive dataset to imitate with maximum entropy third order hypercube-constrained sampling, as matching its first, second, and third order statistics at the same time is challenging. We used 10,000 optimization steps per class, taking a total of 36 hours on a single NVIDIA A40 GPU. Using an hourly price of $0.403 from vast.ai, this would cost roughly $14.5.