Modern Methods, Classical Constraints

Anonymous author

Abstract

This paper proposes both discriminative and generative model architectures that are designed to be parameter efficient, whilst requiring few optimisation steps to train. In particular, both architectures employ both depthwise separable convolutions and transformer (encoder) layers as a means of maintaining expressivity at a lower computational cost. Convergence of the discriminator is expedited using a cyclical learning rate, whilst the convergence of the generator is optimised with a non-linear schedule. Experiments are performed on the CIFAR-100 dataset.

1 The Discriminative Model

1.1 The Discriminative Modelling Problem

Discriminative modeling focuses on learning the conditional probability $P(Y|X)$, where Y represents the target labels, and X represents the input features. If $d(\mathbf{x}) = P(Y|X = \mathbf{x})$, the task of our deep neural network $f(\mathbf{x}; \theta)$ is to learn parameters θ such that $f(\mathbf{x}; \theta) = d(\mathbf{x})$. In the following section we attempt to define an $f(\mathbf{x}; \theta)$ that attempts to learn this true distribution, but with the additional constraints that $|\theta| < 100,000$ and where we use no more than 10,000 gradient update steps to train this model.

1.2 Methodology

1.2.1 Model Architecture

We employ depthwise separable convolutions [41] as a parameter-efficient means of extracting and refining low-level features. Subsequently, these features undergo further processing through a series of transformer encoder layers, whose weights are tied [42, 21, 7]. Finally, classification logits are obtained through a pair of linear transformations.

Each convolutional layer is succeeded by a batch normalisation layer [17] and a sigmoid linear unit activation function [10, 26]. We use 2x2 max pooling to reduce spatial dimensions after the first two layers.

The Transformer layers deviate slightly from the original formulation toward a more modern one [40] that replaces the Layer Normalisation [1] with Root Mean Square Layer Normalisation [44], removes the Dropout layers [38], and uses a Gated Sigmoid Linear Unit activation function [6], in the manner proposed in [32].

See Figure 1 for more details.

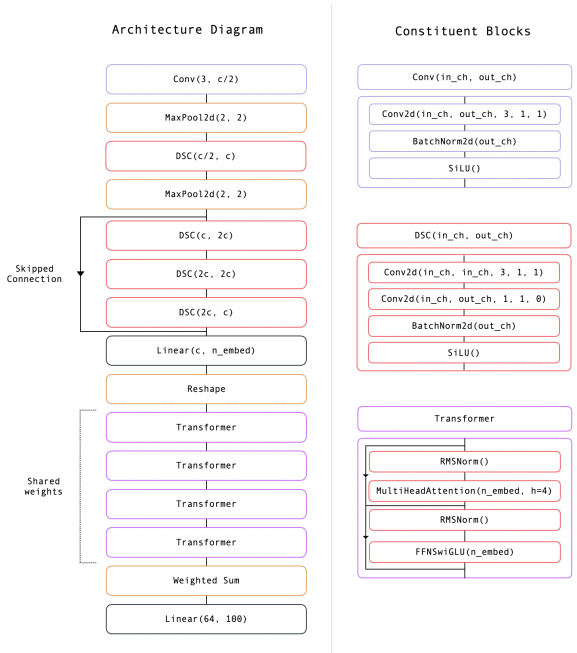

Figure 1: Discriminator Architecture

1.2.2 Discussion

Existing architectures optimised for parameter efficiency exist [14, 15, 13], and could feasibly be directly scaled down to meet our specific requirements. In the spirit of exploration, we opt for a subtly different approach.

In line with the MobileNet family of models [13, 14, 31], we select depthwise separable convolutions over standard convolutions for low-level feature processing since, by effectively factorising the convolution operation, they require fewer parameters to perform an operation of "comparable quality" to that of a standard convolution.

We choose to refine these features further using four shared-weight transformer encoder layers. Transformers, like convolutions, decouple spatial and feature processing. The achieve this by convolving an MLP with the features from each spatial position independently. In our case this allows for a 64x decrease in the number of parameters required, in comparison with a standard MLP. This is in addition to their primary feature, the esteemed self-attention mechanism which introduces dynamic, global, communication that enhances expressivity. Weight sharing increases expressivity further, without increasing parameter count.

The imposed parameter and optimisation step constraints resulted in (relatively) short training times, which enabled us to efficiently iterate using the scientific method. We tested numerous architecture variants such as, what transformer variant to use (LLAMA vs GPT2), which activation function (GeLU vs ReLU vs SiLU), how many transformer layers (0-6), whether to use shared weights, what type of positional encoding (Fourier vs learned), and which spatial reduction technique to use (max pool vs 4,2,1).

In the final network we employ various forms of regularisation beyond the normalisation layers discussed previously. Concretely, we use L2 regularisation, label smoothing, gradient clipping, and data augmentation with RandAugment [5]. We trained the network using ADAMW [18, 22] with a OneCycleLR learning rate scheduler [34], to speed up convergence. We performed a learning rate range test [33] to find optimal values for this scheduler. We trained with a batch size of 1024, primarily as a proxy for increasing the number of training epochs. The optimal hyperparameter values can be found in the provided implementation.

1.3 Results

The network has $99,546$ parameters.

The accuracy on the training set was 65.1% and the accuracy on the validation set was 62.0% (train loss: 1.931, val loss: 1.419, train acc: 0.651 ± 0.014, val acc: 0.620 ± 0.011).

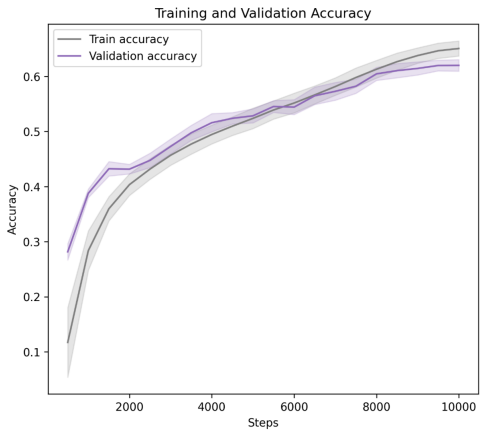

Figure 2: Train & Validation Accuracy

1.4 Limitations

The model has high bias and low variance. The validation accuracy is poor. The validation accuracy is also quite volatile, as we can see from the training graph in Figure 2.

We observed a bias-variance trade-off, as in classical statistics, when trying to regularise the model. Dramatic overparameterisation and much longer training times would allow us to pass the interpolation threshold and leverage the deep double descent phenomenon [23] to obtain both low bias and low variance. We could then smooth the training graph by using a lower learning rate.

Finally, these results say little about the real-world performance of our models, due to the probable overfitting of our hyperparameters on the validation set, and the absence of a test set. Reserving a portion of the training set as a separate test sample is not suitable, as these samples would not represent out-of-distribution data relative to the training distribution.

2 The Generative Model

2.1 Strategy

Among the various classes of deep generative model [27, 9, 25, 27, 8], Generative Adversarial Networks (GANs) and Denoising Diffusion Probabilistic Models (DDPMs) [11, 36] have recently dominated in producing high-quality samples [16]. The primary limitation of GANs is their tendency to mode collapse. In contrast, DDPMs, a subset of Energy Based Models, generate more diverse outputs but require longer to train and sample from.

Torn between a affinity toward to the elegance of DDPMs, and an objective assessment that GANs would perform better under our particular constraints ($<1M$ parameters & $\leq 50,000$ optimization steps), we decided to experiment with both.

Our conditional GAN used depthwise separable convolutions with a classic DCGAN design, and was enhanced with techniques like spectral normalisation and label smoothing. Ultimately however, with extensive tuning, our DDPM was able to achieve a comparable FID score to our GAN and is therefore the model that is officially presented below.

2.2 Methodology

We omit a routine recap of the denoising diffusion process for the sake of brevity. Our particular DDPM is a continuous-time model [2, 36, 20, 37], where we use a variance preserving forward process that is parameterised by the log signal-to-noise ratio (log-SNR): $\lambda_t = log(\frac{\alpha_t^2}{\sigma_t^2})$, where α_t and σ_t are specified by a particular noise schedule. A noise schedule is a monotonically decreasing invertible function that maps the time variable $t \in [0, 1]$ to the corresponding log-SNR λ_t. In particular we use a cosine schedule, inspired its discrete counterpart from [24]. We sample uniformly distributed timesteps t using a low discrepancy quasi-random sequence engine (Sobol engine).

We train with a velocity objective [30], a reparameterisation of the standard $\boldsymbol{\epsilon}$-prediction objective [11], as shown in (3):

Let $\mathbf{v}$ and $\mathbf{x}_{\text{noised}}$ be:

$$\mathbf{v} \equiv \alpha_t \boldsymbol{\epsilon} - \sigma_t \mathbf{x} \tag{1}$$

$$\mathbf{x}_{\text{noised}} \equiv \alpha_t \mathbf{x} + \sigma_t \boldsymbol{\epsilon} \tag{2}$$

where $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I})$ is the noise term (we do not learn covariances), and $\mathbf{x}$ is the real data.

Given the model's prediction of the velocity $\hat{\mathbf{v}} = v_\theta(\mathbf{x}_{\text{noised}}, \lambda_t, c)$, we define the loss as:

$$L_\theta = \mathbb{E}\left[\|\mathbf{v} - \hat{\mathbf{v}}\|_2^2\right] = \mathbb{E}\left[(e^\lambda + 1)\|\mathbf{x} - \hat{\mathbf{x}}\|_2^2\right] = \mathbb{E}\left[(e^\lambda - 1)\|\boldsymbol{\epsilon} - \hat{\boldsymbol{\epsilon}}\|_2^2\right] \tag{3}$$

We employ classifier-free guidance [12] to control the trade-off between the sample fidelity and mode coverage of our model post training. In particular, given a dataset of observations $\mathbf{x}$ drawn from $q(\mathbf{x})$, we train an unconditional generative model with the task of estimating the marginal probability $p(\mathbf{x})$, as well as a conditional model that learns to estimate the conditional densities $p(\mathbf{x}|\mathbf{c})$, given class labels c. Both models are parameterised by the same network. At sampling time, we are then able to control this fidelity-coverage balance by linearly interpolating away from the unconditional velocity estimate, toward the conditional one, using a weighting factor w, as in (4). In our implementation we have $w = 3$, which was determined experimentally.

$$\tilde{v}_\theta(\mathbf{x}_{\text{noised}}, \lambda_t, c) = (1 + w)v_\theta(\mathbf{x}_{\text{noised}}, \lambda_t, c) - wv_\theta(\mathbf{x}_{\text{noised}}, \lambda_t, \varnothing) \tag{4}$$

v_θ is a modernised conditional U-Net [28] augmented with self-attention [42]. We replace the majority of the standard convolutional layers with depthwise separable convolutions. We inject timestep embeddings (sinusoidal) and class embeddings (learned) via an addition at two points in the network. We smooth the trajectory of the model updates using the popular Exponential Moving Average method as in [11, 24]. We omit an architecture diagram since the architecture is relatively standard and since the implementation code is provided.

2.3 Discussion

The code for our model was adapted from an implementation by Katherine Crowson [3]. We refactored and extended this code to suit our purposes. Specifically, we replaced the provided network architecture with our own design, added classifier-free guidance, two non-linear schedules, dynamic clipping in the x-space during sampling [29], and a function to perform spherical latent interpolations [39].

Our initial implementation of v_θ generated samples containing regular wave-like artefacts. We hypothesised that the depthwise separable convolutions, due to their nature of processing input channels separately before combining them, meant that the network was unable to properly assimilate the timestep and class embeddings into the overall representation. We resolved this by replacing the depthwise separable convolutions after each timestep injection, with regular convolutions.

[20] showed that the Evidence Lower Bound Objective (ELBO) is invariant to the choice of noise schedule (except for it's endpoints). However, [19] showed that the choice of noise schedule does directly influence the variance of Monte Carlo estimator for the loss that is used during training. Consequently they showed that lowering the variance of this loss estimator often significantly speeds up optimisation. It is for this reason that we sample timesteps using a Sobol engine during training, and implemented two additional noise schedules to experiment with. In particular, we implemented a cosine schedule and a spliced cosine-linear hybrid [4], alongside the linear schedule already provided.

We use DDPM-style stochastic sampling to generate our primary samples but use DDIM [35] style deterministic sampling to generate our interpolations. This is because the stochastic generative process of DDPMs causes the generations to stray unpredictably from the initial noise, which is of course not the case for deterministic sampling. In our implementation we spherically interpolate between pairs of both latents and class embeddings.

2.4 Results

Our 982,371 parameter model, trained for 50,000 steps, achieves an FID of 31.1.

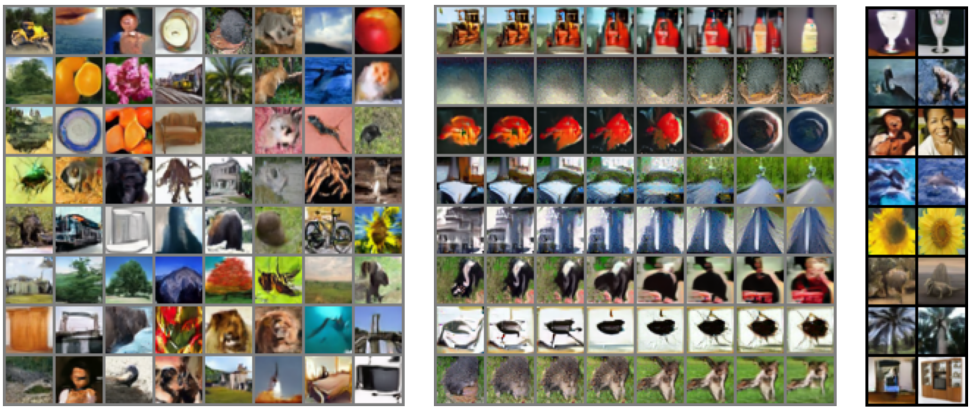

Figure 3: Random samples (left), latent interpolations (middle), nearest neighbours (right).

2.5 Limitations

Despite reasonable shapes and textures, most images lack realistic details. Certain batches contain similar samples which indicates slight mode collapse. Our interpolation technique, being derived from first principles, is rudimentary. A more sophisticated technique like perceptually uniform sampling [43], or interpolation at every noise level, would likely produce more realistic midpoints. Finally, analysis of the nearest neighbors of our samples reveals that the model is quite heavily mimicking the training set.

References

[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. *Layer Normalization*. 2016. arXiv: 1607.06450 [stat.ML].

[2] Nanxin Chen et al. *WaveGrad: Estimating Gradients for Waveform Generation*. 2020. arXiv: 2009.00713 [eess.AS].

[3] Katherine Crowson. *Trains a diffusion model on CIFAR-10*. Online. 2024. URL: https://colab.research.google.com/drive/1IJkrrV-D7boSCLVKhi7t5docRYqORtm3.

[4] Katherine Crowson. *v-diffusion*. Online. 2024. URL: https://github.com/crowsonkb/v-diffusion-pytorch/blob/master/diffusion/utils.py.

[5] Ekin D. Cubuk et al. *RandAugment: Practical automated data augmentation with a reduced search space*. 2019. arXiv: 1909.13719 [cs.CV].

[6] Yann N. Dauphin et al. *Language Modeling with Gated Convolutional Networks*. 2017. arXiv: 1612.08083 [cs.CL].

[7] Mostafa Dehghani et al. *Universal Transformers*. 2019. arXiv: 1807.03819 [cs.CL].

[8] Yilun Du and Igor Mordatch. *Implicit Generation and Generalization in Energy-Based Models*. 2020. arXiv: 1903.08689 [cs.LG].

[9] Ian J. Goodfellow et al. *Generative Adversarial Networks*. 2014. arXiv: 1406.2661 [stat.ML].

[10] Dan Hendrycks and Kevin Gimpel. *Gaussian Error Linear Units (GELUs)*. 2023. arXiv: 1606.08415 [cs.LG].

[11] Jonathan Ho, Ajay Jain, and Pieter Abbeel. *Denoising Diffusion Probabilistic Models*. 2020. arXiv: 2006.11239 [cs.LG].

[12] Jonathan Ho and Tim Salimans. *Classifier-Free Diffusion Guidance*. 2022. arXiv: 2207.12598 [cs.LG].

[13] Andrew Howard et al. *Searching for MobileNetV3*. 2019. arXiv: 1905.02244 [cs.CV].

[14] Andrew G. Howard et al. *MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications*. 2017. arXiv: 1704.04861 [cs.CV].

[15] Forrest N. Iandola et al. *SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and ¡0.5MB model size*. 2016. arXiv: 1602.07360 [cs.CV].

[16] *ImageNet 64x64 Benchmark (Image Generation)*. Papers With Code, 2024. URL: https://paperswithcode.com/sota/image-generation-on-imagenet-64x64.

[17] Sergey Ioffe and Christian Szegedy. *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*. 2015. arXiv: 1502.03167 [cs.LG].

[18] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2017. arXiv: 1412.6980 [cs.LG].

[19] Diederik P. Kingma and Ruiqi Gao. *Understanding Diffusion Objectives as the ELBO with Simple Data Augmentation*. 2023. arXiv: 2303.00848 [cs.LG].

[20] Diederik P. Kingma et al. *Variational Diffusion Models*. 2023. arXiv: 2107.00630 [cs.LG].

[21] Zhenzhong Lan et al. *ALBERT: A Lite BERT for Self-supervised Learning of Language Representations*. 2020. arXiv: 1909.11942 [cs.CL].

[22] Ilya Loshchilov and Frank Hutter. *Decoupled Weight Decay Regularization*. 2019. arXiv: 1711.05101 [cs.LG].

[23] Preetum Nakkiran et al. *Deep Double Descent: Where Bigger Models and More Data Hurt*. 2019. arXiv: 1912.02292 [cs.LG].

[24] Alex Nichol and Prafulla Dhariwal. *Improved Denoising Diffusion Probabilistic Models*. 2021. arXiv: 2102.09672 [cs.LG].

[25] Aaron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. *Pixel Recurrent Neural Networks*. 2016. arXiv: 1601.06759 [cs.CV].

[26] Prajit Ramachandran, Barret Zoph, and Quoc V. Le. *Searching for Activation Functions*. 2017. arXiv: 1710.05941 [cs.NE].

[27] Danilo Jimenez Rezende and Shakir Mohamed. *Variational Inference with Normalizing Flows*. 2016. arXiv: 1505.05770 [stat.ML].

[28] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. *U-Net: Convolutional Networks for Biomedical Image Segmentation*. 2015. arXiv: 1505.04597 [cs.CV].

[29] Chitwan Saharia et al. *Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding*. 2022. arXiv: 2205.11487 [cs.CV].

[30] Tim Salimans and Jonathan Ho. *Progressive Distillation for Fast Sampling of Diffusion Models*. 2022. arXiv: 2202.00512 [cs.LG].

[31] Mark Sandler et al. *MobileNetV2: Inverted Residuals and Linear Bottlenecks*. 2019. arXiv: 1801.04381 [cs.CV].

[32] Noam Shazeer. *GLU Variants Improve Transformer*. 2020. arXiv: 2002.05202 [cs.LG].

[33] Leslie N. Smith. *A disciplined approach to neural network hyper-parameters: Part 1 – learning rate, batch size, momentum, and weight decay*. 2018. arXiv: 1803.09820 [cs.LG].

[34] Leslie N. Smith and Nicholay Topin. *Super-Convergence: Very Fast Training of Neural Networks Using Large Learning Rates*. 2018. arXiv: 1708.07120 [cs.LG].

[35] Jiaming Song, Chenlin Meng, and Stefano Ermon. *Denoising Diffusion Implicit Models*. 2022. arXiv: 2010.02502 [cs.LG].

[36] Yang Song et al. *Score-Based Generative Modeling through Stochastic Differential Equations*. 2021. arXiv: 2011.13456 [cs.LG].

[37] Yuxuan Song et al. *Discriminator Contrastive Divergence: Semi-Amortized Generative Modeling by Exploring Energy of the Discriminator*. 2020. arXiv: 2004.01704 [cs.LG].

[38] Nitish Srivastava et al. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". In: *Journal of Machine Learning Research* 15.56 (2014), pp. 1929–1958. URL: https://www.cs.toronto.edu/~rsalakhu/papers/srivastava14a.pdf.

[39] Xander Steenbrugge. *SLERP defintion*. Online. 2024. URL: https://gist.github.com/karpathy/00103b0037c5aaea32fe1da1af553355.

[40] Hugo Touvron et al. *LLaMA: Open and Efficient Foundation Language Models*. 2023. arXiv: 2302.13971 [cs.CL].

[41] Vincent Vanhoucke. "Learning Visual Representations at Scale". In: *International Conference on Learning Representations (ICLR)*. 2014.

[42] Ashish Vaswani et al. *Attention Is All You Need*. 2023. arXiv: 1706.03762 [cs.CL].

[43] Zhaoyuan Yang et al. *IMPUS: Image Morphing with Perceptually-Uniform Sampling Using Diffusion Models*. 2023. arXiv: 2311.06792 [cs.CV].

[44] Biao Zhang and Rico Sennrich. *Root Mean Square Layer Normalization*. 2019. arXiv: 1910.07467 [cs.LG].