Intro to Creativity in Diffusion Models

Guiding Question: What does it mean for a model to be "creative"?

- Diffusion models are the best tool we have for generating images right now, but the gap between the behavior observed and practice and the standard theory is enormous.
  - Theory predicts perfect memorization of the training data (under specific regiemes)
  - “Under idealized score-matching, the network would only learn to regenerate training data —so in this sense, the model isn’t being ‘creative’ at all.”
- Suppose I live in a world of 32 x 32 grayscale images of cats and that I have a dataset of 10k of them
  - Each picture looks like this and is easily flattened to a 1 by 1024 vector
  - Imagine if we plot them in $\mathbb{R}^{1024}$ and construct a density function which we'll visualize in $\mathbb{R}^{1025}$
  - This distribution is what we're trying to learn
  - The higher the density near an image X, the more images in our dataset resemble image X
- This distribution is complicated and high dimensional, but if we were able to learn the probability surface that "connects the dots" then we'd be able to sample from the distribution at will and generate new cat images
- A diffusion model is a model that approximates this "underlying image distribution" so that we can sample from it
  - We'll call this model p_θ where θ tells us that we are learning parameters that will help us approximate underlying image distribution
  - For simplicity and because Gaussian mixtures are extremely versatile, we're going to work with multivariate gaussian distributions to try to approximate what's going on here
- Theory tells us that if we learn this image distribution properly, we'll always arrive back at one of our training images -- we won't even have interpolation!
  - "Critically, a network that learns the exact score function of the training set—i.e., a mixture of delta functions centered on training data—can only reproduce training examples, and hence cannot generalize. This observation suggests that neural networks optimize for a balance between learning the score function of the training set and capturing aspects of the underlying data manifold. However, the precise characteristics of the learned score function, and how the development of this balance proceeds over the course of training, remain largely unexplored."
    - The Unreasonable Effectiveness of Gaussian Score Approximation for Diffusion Models and its Applications Binxu and John here @ Kempner

Part 1: What is Diffusion?

- I'll give a quick overview of diffusion model basics here, but for a more complete picture, I'd highly recommend the book Probabilistic Machine Learning: Advanced Topics by Kevin Murphy. I also gave a talk on this in January of this year, and the notes are available on the LnL slack!
- We start with the fairly obvious observation that it's easier to turn good data into noise than noise into good data
  - We have a stochastic encoder which we call $q(x_t|x_{t-1})$ which adds more noise at each time step. Eventually, after enough steps we have $x_T \sim \mathcal{N}(0,1)$
  - We then use a neural network to help us learn the REVERSE process (the decoder), which we call $p_\theta(x_{t-1}|x_t)$. We start with some random noise x_T which we apply this decoder to iteratively until we generate a sample x_0 from the underlying data distribution
- There are many, many different and deep down equivalent ways of thinking of diffusion, but because I find it notationally clearest, we'll use the stochastic differential equation perspective on diffusion here!
- We can think of the forward process as an SDE

$$d\boldsymbol{x} = \underbrace{\boldsymbol{f}(\boldsymbol{x}, t)}_{\text{drift}} dt + \underbrace{g(t)}_{\text{diffusion}} d\boldsymbol{w}$$

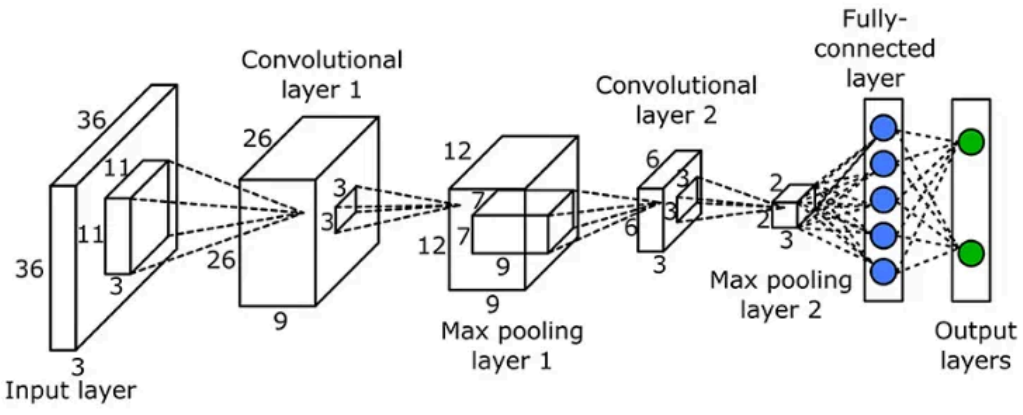
  - where dw is Brownian motion and g(t) tells us how much noise we add at each time t
- We can (with a fair amount of mathematical trouble) reverse this SDE and get a "reverse-time SDE" given by
- $dx = [f(x_t, t) - g^2(t)\nabla_{x_t} \log q_t(x_t)]dt + \sqrt{\beta(t)}d\bar{w}_t$ where $\bar{w}_t$ is a time reversed Weiner process and $\nabla_{x_t} \log q_t(x_t)$ is the SCORE FUNCTION (you can either learn the score or predict the noise, they're equivalent since seeing the sample and predicting the noise or the score will give you the same information since sample = noise + signal)
- This is the crucial step here, this is what we parametrize with a neural network and LEARN from our data
- $\nabla_{x_t} \log q_t(x_t) = s_\theta(x_t, t)$
- Why Score functions?
  - Before we dive deeper into diffusion, it’s worth taking a moment to understand what we actually mean by a **score function**, and why it's useful.
  - The **score function** of a probability distribution $p(x)$ is defined as the gradient of the log-density:
    - $s(x) := \nabla_x \log p(x) s(x)$

This vector field points in the direction where the log-probability increases most rapidly — in other words, **it tells you how to "climb up" the density landscape of your data distribution**. This is useful for a number of reasons:

- **In generative modeling**, the score function encodes *local geometric information* about the distribution. If we know $\nabla_x \log p(x)$ for every x, we can construct algorithms to sample from $p(x)$.
- But remember, if we have exact equality in the above, we just MEMORIZE the training data
  - This tells us paradoxically that creativity must come from NOT learning the ideal score!
  - Where do we make mistakes? How do we make mistakes? Why are these mistakes so "good"??
- One answer is that the neural network we use to parametrize the score is NOT perfectly expressive. In other words, the inductive biases of our neural network prevent us from learning the score exactly, but those inductive biases are creatively productive and allow us to generate a more diverse set of examples

Part 2: What's the simplest score network? A CNN!

- There was an awesome paper out of Surya Ganguli's lab at Stanford that started to answer this question!
- Their idea was to start with a simple way of parametrizing the score (which is not used in practice very often), using a CNN
  - A CNN is a neural network which uses convolutional layers
    - At its heart a CNN is just a bunch of discrete convolutions—each one “looks” only at a small neighborhood of the input and re-uses the same weights everywhere.



- A CNN has two key inductive biases (in other words, limits on the kinds of functions it can learn)
  - Locality
  - Translational Equivariance

- Definition: Discrete Convolution
  - Let $X : \mathbb{Z}^2 \to \mathbb{R}$ be an image and $K : \{-r, \dots, r\}^2 \to \mathbb{R}$ a convolutional kernel of radius r.
  - Their **discrete convolution** $Y = X * K$ is defined by $(X * K)[u, v] \;=\; \sum_{i=-r}^{r} \sum_{j=-r}^{r} K[i, j]\, X[u + i,\; v + j]$.
- For any integer shift $(a, b) \in \mathbb{Z}^2$, the **translation** $T_{(a,b)}$ acts on X by $\big(T_{(a,b)} X\big)[u, v] \;=\; X\big[u - a,\; v - b\big]$
- Translation Equivariance means $T_{(a,b)}\big(X * K\big) \;=\; \big(T_{(a,b)} X\big) * K$
- CNNs are Translation Equivariant
  - $\big((T_{(a,b)} X) * K\big)[u, v] = \sum_{i=-r}^{r} \sum_{j=-r}^{r} K[i, j]\; \big(T_{(a,b)} X\big)[u + i, v + j]$.
  - Substitute the definition of $T_{(a,b)}$ to see that the above $= \sum_{i=-r}^{r} \sum_{j=-r}^{r} K[i, j]\; X\big[(u + i) - a,\; (v + j) - b\big]$
  - Re-index: $= \sum_{i=-r}^{r} \sum_{j=-r}^{r} K[i, j]\; X\big[(u - a) + i,\; (v - b) + j\big] = (X * K)[\,u - a,\; v - b\,]$
  - Recognize the result: $(X * K)[\,u - a,\; v - b\,] = \big(T_{(a,b)}(X * K)\big)[u, v]$
  - Thus, we have $T_{(a,b)}(X * K) \;=\; (T_{(a,b)} X) * K$
- So a convolutional layer is translation equivariant and so too (roughly) is a CNN.

- They were able to get pretty fantastic results with these two inductive biases! In particular, if you parametrize your score function as something of the form
  - $M_t[\phi](x) = f[\phi_{\Omega_x}]$ where we find that the score of a particular pixel x in an image ϕ is simply a function of the patch of the image around x called ϕ_{Ω_x}
  - The same function is used at every pixel x. There is no explicit dependence on the absolute coordinate x.
  $$
  \mathcal{L}
  ;=;
  \sum*{x}
  \mathbb{E}{\phi \sim \pi*t}
  \Bigl\lVert
  f\bigl[\phi{\Omega_x}\bigr]
  ;-;
  s_t\phi
  \Bigr\rVert^2

  We then optimize this loss function to find the best function f *to minimize the loss :"To find the fun*

(a) Theory (left) vs. ResNet (right)

But, their assumptions aren't quite representative of reality. For one, CNNs with large enough "receptive fields" are not actually local:
a receptive field is the region of the input image that can influence that neuron's activation—i.e. the set of input pixels that feed (directly or via intervening layers) into that neuron.

Part 3: What do people actually use?

- Most state-of-the-art image diffusion models rely on **convolutional U-Net** architectures as their score function approximator (denoiser). These U-Nets are typically augmented with residual connections and **attention mechanisms** for high fidelity generation.
- OpenAI's DALL·E 2 (unCLIP) image generator similarly employs a large U-Net (≈3.5 billion parameters) with integrated attention layers
- This is not to say that this is the only way to do it
  - Meta AI's Diffusion Transformer (DiT) (Peebles et al. 2022) demonstrates that a Vision Transformer (ViT) architecture can serve as the denoiser, challenging the need for CNN down/up-sampling blocks
  - Sora also is kinda a DiT, built on a **Diffusion Transformer** backbone instead of a U-Net
- "Across these diffusion models, several architectural trends stand out. First, **multi-scale U-Net backbones with attention** have become a de facto standard due to their proven performance (as seen in Stable Diffusion, DALL·E 2, Imagen, etc.)"

Part 4: Attention + My Workshop Paper

- Most state of the art implementations involve and include attention as part of the backbone of their score approximation neural network
- Attention strongly violates this locality assumption, because it encodes global relationships
- 1. **Feature extraction**
  Pass the noisy image x_t through convolutional (or other) layers to get feature maps $h \in \mathbb{R}^{H\times W\times C}$. These retain local structures but don't yet capture long-range dependencies.

2. **Project to queries, keys, and values**
   Compute three linear projections of h: $Q = W_Q\,h, \quad K = W_K\,h, \quad V = W_V\,h,$
   each of shape $H \times W \times d$.
   1. **Why?** Queries and keys let every location "ask" about—and "listen" to—every other location, and values carry the information to be mixed in.
3. **Compute attention weights**
   For each spatial position i, measure similarity of its query Q_i with every key K_j:
   $a_{ij} = \frac{\exp(Q_i \cdot K_j/\sqrt{d})}{\sum_k \exp(Q_i \cdot K_k/\sqrt{d})}$

- **Global context:** Because j runs over all $H \times W$ positions, even distant patches influence the weights.
- **Result:** Each position learns which other patches are most relevant for denoising. Consider generating faces and seeing one eye and then right next to it another eye

4. **Recombine values** Form new features by weighted summation: $(\mathrm{Attn}(h))_i = \sum_j a_{ij} V_j$
   Now each output position i contains a mixture of information from across the entire image, not just its original neighborhood—this is what injects global interactions.
5. **Residual + feed-forward**
   Add a skip-connection and layer-norm, then pass through a small MLP:
   $h' = \mathrm{MLP}\big(\mathrm{LayerNorm}(h + \mathrm{Attn}(h))\big)$
   This stabilizes training and lets the model refine how global information affects each location.
6. **Score prediction**
   Continue processing h' (possibly interleaving more attention blocks) until you output either:
   - A noise prediction $\epsilon_\theta(x_t, t)$, or
   - A direct estimate of $\nabla_{x_t} \log p_t(x_t)$.
     Because attention has injected global information at every layer, your score network can denoise in a way that respects—and enforces—coherent structure across the whole image.

We apply the same strategy as Ganguli, but instead enforce a functional form that describes a very simple attention based model:

$$\tilde{g}[\phi](x) \;=\; g(\phi_{\Omega_x}) \;+\; \sum_y \alpha_{xy}\, g(\phi_{\Omega_y}), \quad \text{where} \quad \alpha_{xy} \;=\; \frac{\exp\big(\langle g(\phi_{\Omega_x}), g(\phi_{\Omega_y})\rangle\big)}{\sum_{y'} \exp\big(\langle g(\phi_{\Omega_x}), g(\phi_{\Omega_{y'}})\rangle\big)},$$

While solving for the optimal $\tilde{g}$ here is hard, we are able to understand intuitively the role that attention has in some simpler cases, for example top-1 attention.

For simplicity, we assume that our attention is a winner-take-all regime, meaning that only the most attended to patch contributes to the sum. In particular, we have

$$\sum_y \alpha_{xy}\, g(\phi_{\Omega_y}) \;\longrightarrow\; g\big(\phi_{\Omega_{y^*(x)}}\big), \quad y^*(x) = \arg\max_y \langle g(\phi_{\Omega_x}),\, g(\phi_{\Omega_y})\rangle.$$

We also assume patch-independence under the distribution π_t for all t, so that conditioning on $\phi_{\Omega_x} = \Phi$ does not change the distribution of ϕ_{Ω_y} for $y \neq x$ and that our embedding g is mean-centered over patches (ie, $\mathbb{E}_{\phi \sim \pi_t}\big[g(\phi_{\Omega_x})\big] = 0 \quad \forall\, x$)

$$\tilde{g}[\Phi] = \nabla_{\Phi(0)} \log \sum_x \big[\pi_t(\phi_{\Omega_x} = \Phi) + \pi_t(\phi_{\Omega_{y^*(x)}} = \Phi)\big]$$

We can solve this to get a normal mixture solution, assuming that π_t is a normal density, which tells us that attention is enforcing some kind of consistency constraint

Part 5: What is creativity? What is consistency?

- I think the really cool paper out of Ganguli's lab indicates that there's a lot more to our notion of creativity than we might initially imagine
- Also has some interesting implications for data-privacy etc
- What kinds of biases encourage "creative" generations?
  - How do attention mechanisms shape inductive biases?
  - Can we quantify "consistency" or "novelty" in outputs?
- "Can we trade off locality vs. global consistency to control creativity?"
- "What does this mean for memorization and privacy?"

Image 1 from CNNs for Dummies

Analytic Theory of Creativity for Convolutional Diffusion Models by Mason Kamb and Surya Ganguli
Probabilistic Machine Learning: Advanced Topics by Kevin Murphy
Rombach et al., "High-Resolution Image Synthesis with Latent Diffusion Models" (CVPR 2022)
OpenAI technical report on DALL·E 2 (unCLIP)
Saharia et al., "Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding" (Imagen, NeurIPS 2022)
Unite.AI, "Midjourney vs Stable Diffusion" (2023)
Humza Naveed, "Diffusion Transformer: Architecture behind Sora" (2024)
Liu et al., "Sora: A Review on … Large Vision Models" (arXiv 2024)
Peebles et al., "Scalable Diffusion Models with Transformers (DiT)" (arXiv 2022) – demonstrates high-performing image diffusion using a pure Transformer backbone with adaptive layer norm conditioning