INTERNATIONAL JOINT CONFERENCE ON NEURAL NETWORKS

IJCNN2025

30 JUNE - 5 JULY 2025 | ROME, ITALY

INTERNATIONAL NEURAL NETWORK SOCIETY

Practical Guide to Developing Flow-Based Generative Models

Scott H. Hawley

Practical Guide to *(Understanding and)* Developing *(Latent)* Flow-Based Generative Models

Scott H. Hawley
Belmont University, Nashville TN USA
Hyperstate Music, Inc.
@drscotthawley

Tutorial @ IJCNN, June 30, 2025

Orientation for this Tutorial

- Why this guide: "for Coders"
  - Many a tutorial/"guide" only shows math
  - Coding tutorials often too simple ("2D dots")
  - More tech/coding needed for bigger problems
  - For new/custom problems: Training, not just Inference
- Tutorial Scope: "medium sized" "image" problems
  - 2D dots < 64^2 ≤ **Us** (~128^2,256^2) < Much Bigger(=$compute$)
  - ***Use latent representations***
  - Train on workstation, not cluster
  - Models fit on one GPU

My bias: Fast inference for image/MIDI model

Speaker Context

PhD 2000 + 6 yrs postdoc: Astrophysics, numerical relativity, HPC
Professor 2006-now: teaching physics to audio engineering undergrads @ Belmont U.
First exposure to ML: audio source separation (via NMF!) @ AES 2013
2023-now: building AI songwriting coach w/ Hyperstate Music AI!

Research includes...

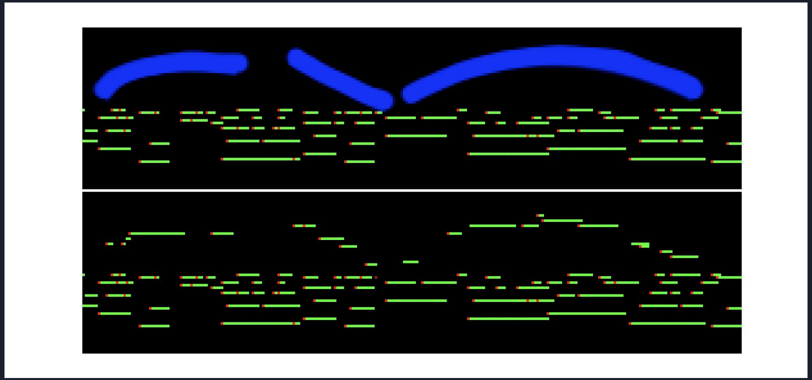
- audio sample library search app (Vibrary, w/ Art+Logic, 2017-2018)
- philosophy & ethics of AI (2018+)
- audio effects modeling (SignalTrain, 2019-20)
- object detection for musical acoustics (espiownage, 2021-22)
- text-to-music diffusion models (Stable Audio, 2023)
- composable semantic+geometric ops in latent space (OpLaS, 2024)
- controllable music generation (PicturesOfMIDI, 2024)

Teaching:

- "Deep Learning & AI Ethics" course at Belmont U: github.com/drscotthawley/DLAIE
- Tutorial blog posts: drscotthawley.github.io/blog/: PCA, RVQ, Attention, BYOL, Flow Models...

Speaker Context 2

Diffusion inpainting in pixel space with HDiT was *musically-aware*!

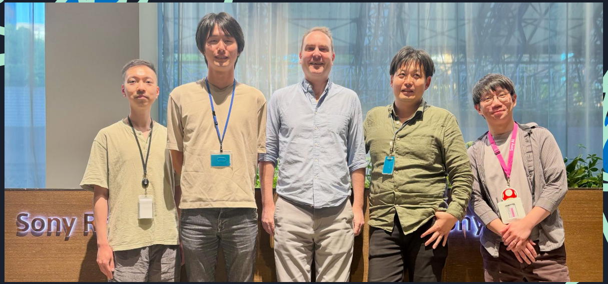


…But computationally "expensive": slow and compute intensive.

How to make it lightweight & faster…even run on CPU?

- Latent space: smaller, less compute
- Flow model: Fewer integration steps via *smooth* flow
- Bonus: Use custom quantized encoder for *interpretable latent reps!*

"Flowing" from diffusion tutorial by Sony AI...



Diffusion ➟ Flows*

"Integration" / "Sampling"

$$dx = v(x,t)dt + \cancel{g(t)dW}$$

Training (flip *t* axis *t*: 0→1)

$$x = \alpha(t)x_1 + \sigma(t)x_0 \qquad x_0 = \epsilon \sim \mathcal{N}$$

Flows (FM & RF)

$$\alpha(t) = t, \quad \sigma(t) = 1 - t, \quad g(t) = 0$$

AKA "linterp"

Mathematicians: "They're the same"
Coders: "Flows are simpler"

*Flow Matching = Rectified Flows = ___, all @ ICLR 2023
=> "Flows" in this tutorial
!= "Normalizing Flows"; we ditch the normalizing property.

For those of us more comfortable with *analysis* than statistics…

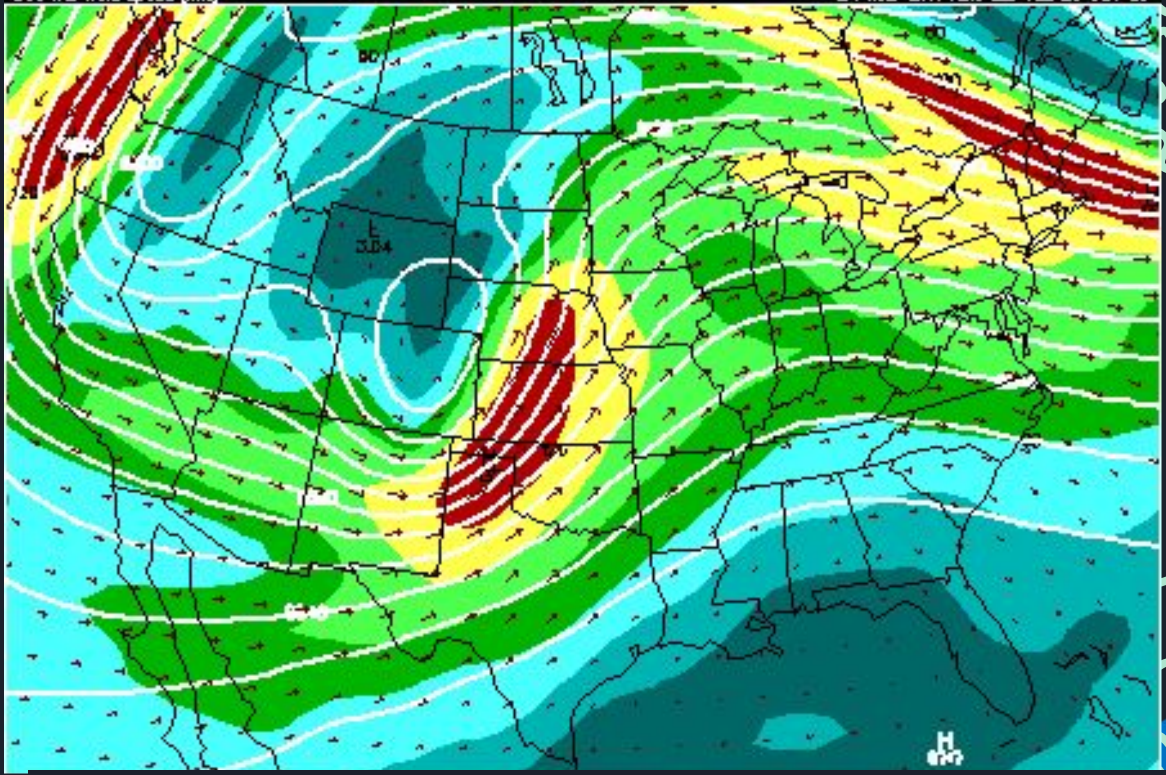
Flows, we "know"...

Vector field:
Every location *x* has an associated velocity vector.

Velocities can change over time

"Streamlines":

- smooth
- don't cross (=invertible)



Wind flow map image from the University of Illinois WW2010 Project

Quickstart: Diffusion ➟ Flow

Given a working diffusion code,

1. Simplify the training objective
2. Simplify the sampling noise schedule
3. Optional: Upgrade your sampling integrator

All finished!

Let's take a break! ;-)

✅ [My "flow" fork]() of
[@crowsonkb]()/`k-diffusion`

...*Given* a *working* code?

"Large-scale **training** of latent diffusion models (LDMs) has enabled unprecedented quality in image generation. However, **the key components** of the best performing LDM **training recipes** are oftentimes not available to the research community..."

— Barrada et al 2024, "On Improved Conditioning Mechanisms and Pre-training Strategies for Diffusion Models", AKA "Three Things Everyone Should Know About Training Diffusion Models"

...Hence this tutorial. We'll try to build "key components" but for flows (for simplicity)

Let's Back Up
to 1st Principles…

Familiar Idea: Hard Problem ➞ Tiny Steps

Often in science, we cast a "hard" problem in terms of tractable "small **changes**": e.g., calculus, differential equations

Full solutions found by accumulating over many steps: integration

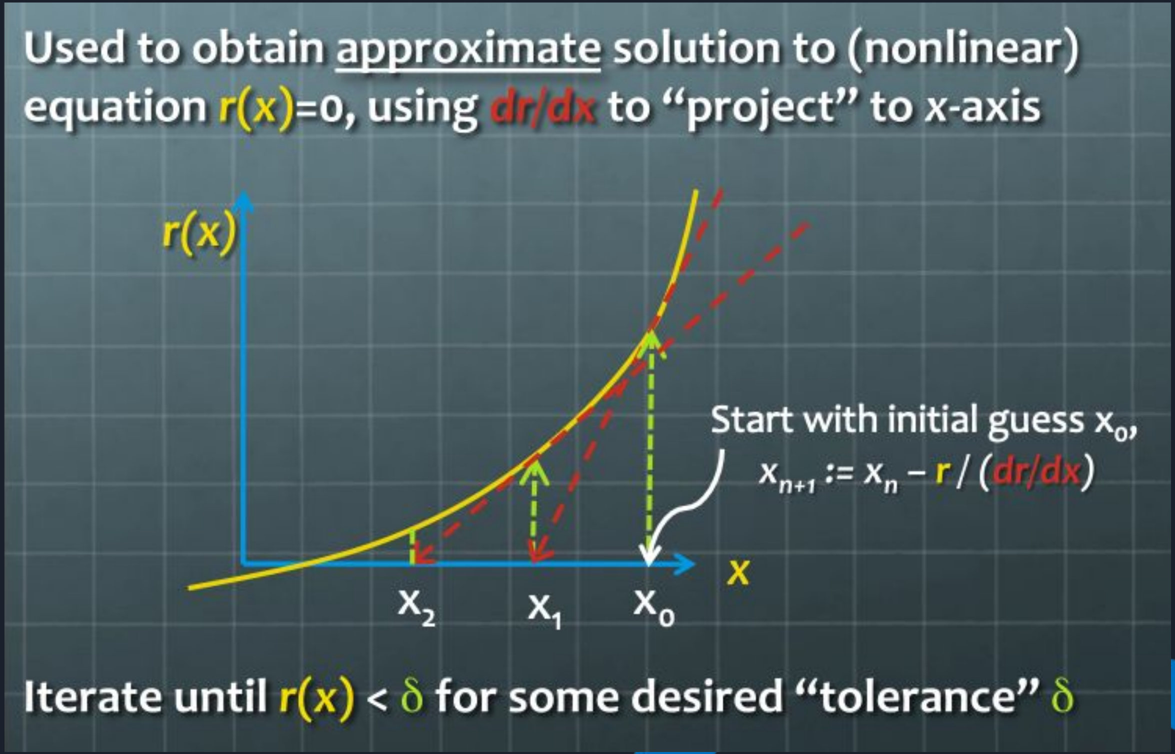
cf. ML architectures: most NN ops in ResNets, UNets, & Transformers compute ***changes***

Similarly, in numerical methods (in general), we often turn solution-finding into a *process*, akin to time-evolution.

Examples…

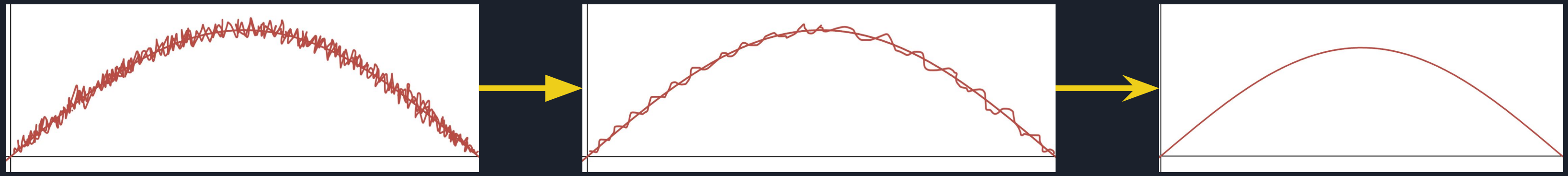
Familiar Idea: Hard Problem ➟ Tiny Steps

Example: Newton's Method:



Familiar Idea: Hard Problem ➟ Tiny Steps

... Relaxation (elliptic PDE -> *parabolic* PDE),



(= diffusion!)

...Minimization, e.g. gradient descent

Familiar Idea: Hard Problem ➡ Tiny Steps

Two main starting points for this:
#1. Solving forward from "*known*" starting point (IVP)
#2. Refinement (of entire solution) from initial "*guess*"

Generative models in these paradigms:
- (Markov chains? - #1 &/or # 2?)
- Autoregression: #1
- Diffusion/Flow: #2 – though "phrased" as #1!

❌ GANs

cf. ML architectures: most NN ops in ResNets, UNets, & Transformers compute ***changes***

Recent work re. this:
"**Auto-Regressive vs Flow-Matching**: a Comparative Study of Modeling Paradigms for Text-to-Music Generation",
Tal, Kreuk, & Adi, arXiv:2506.08570, **10 June 2025**

Familiar Idea: Hard Problem ➞ Tiny Steps

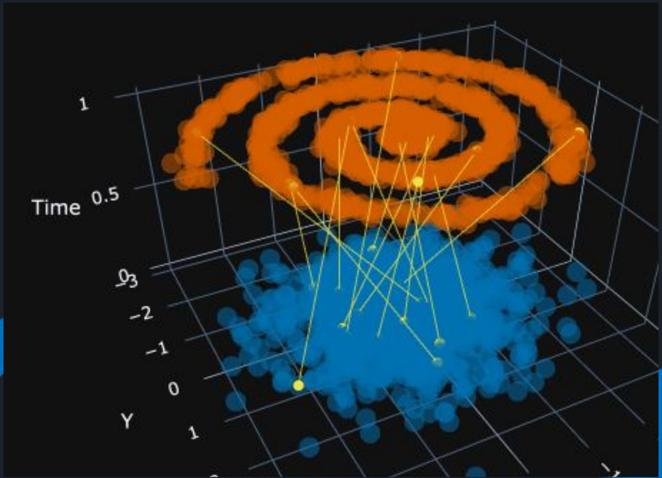
For our flow models, the tiny steps will be incremental motion along a streamline of the velocity field, e.g.

x += v(x,t) dt

Where v(x,t) is the output of some (nonlinear multidimensional) curve-fitting system
– a neural network!

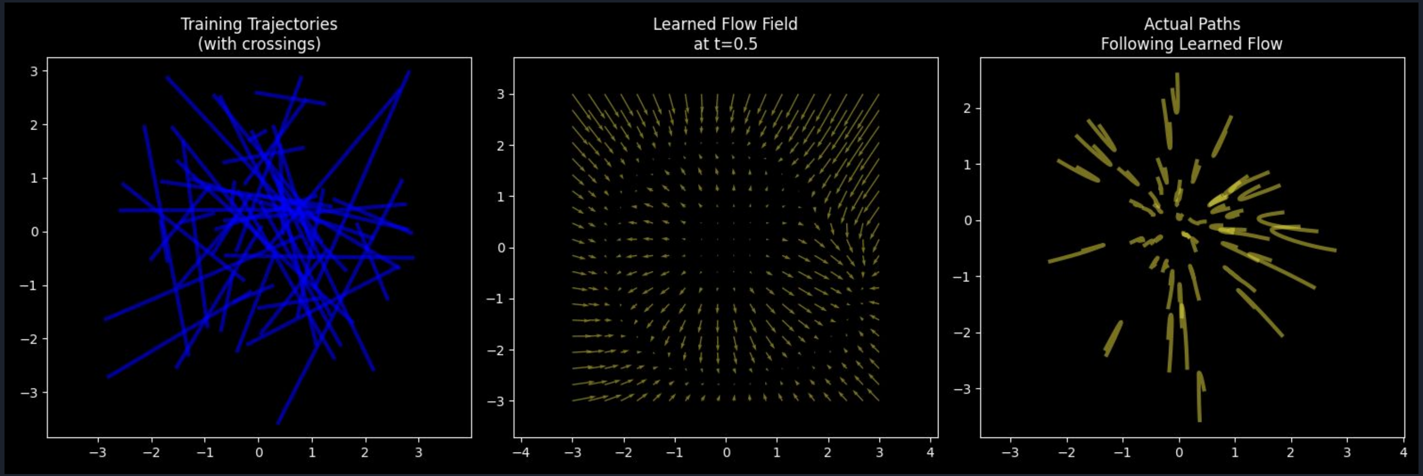
Review / Basic Tutorial on "Flows" - 1

- "Flow With What You Know": ICLR 2025 Blogpost Track Won "Best"!
  Many others ~same time, including (math) guide by Meta
- Basic idea:
  - Randomly pair samples from Source & Target distributions
  - "Guess" a *constant velocity* - i.e. linear interpolation
  - Train v(x,t) vs. guess with MSE



Review / Basic Tutorial on "Flows" - 2

Learned velocity model predicts curved trajectories:



Trajectories are smooth (unlike most diffusion models), so can be integrated via higher-order ODE solvers (e.g. RK4)
=> fewer integration steps => faster sampling.

Review / Basic Tutorial on "Flows" - 3

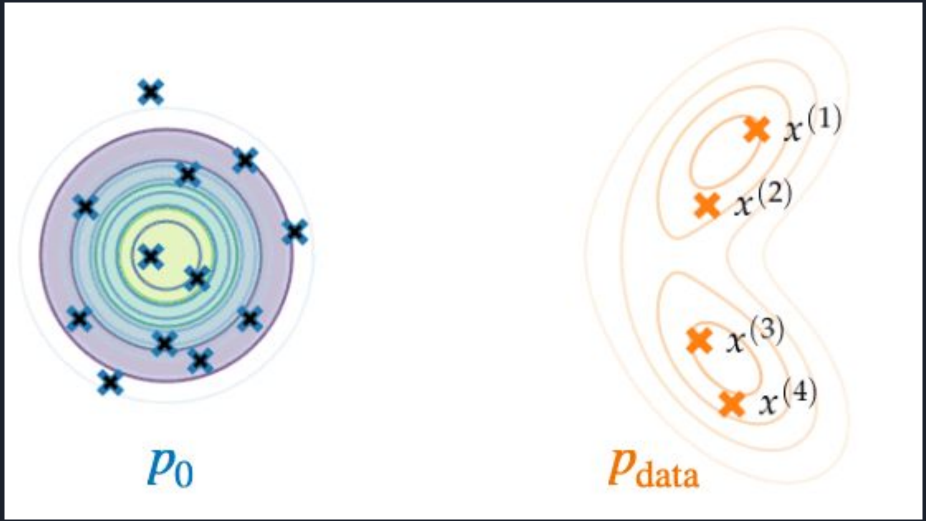
Training v_{model} != "supervised learning" in usual sense, because v_{linear} is *always "very wrong"*. (Loss flattens out >> 0)

You're just using the NN to aggregate/"marginalize" many "streamlines" and provide for "interpolation" at untried (x,t) pairs

Turns out that scaling up from 2D dots isn't easy

…Hence this tutorial!

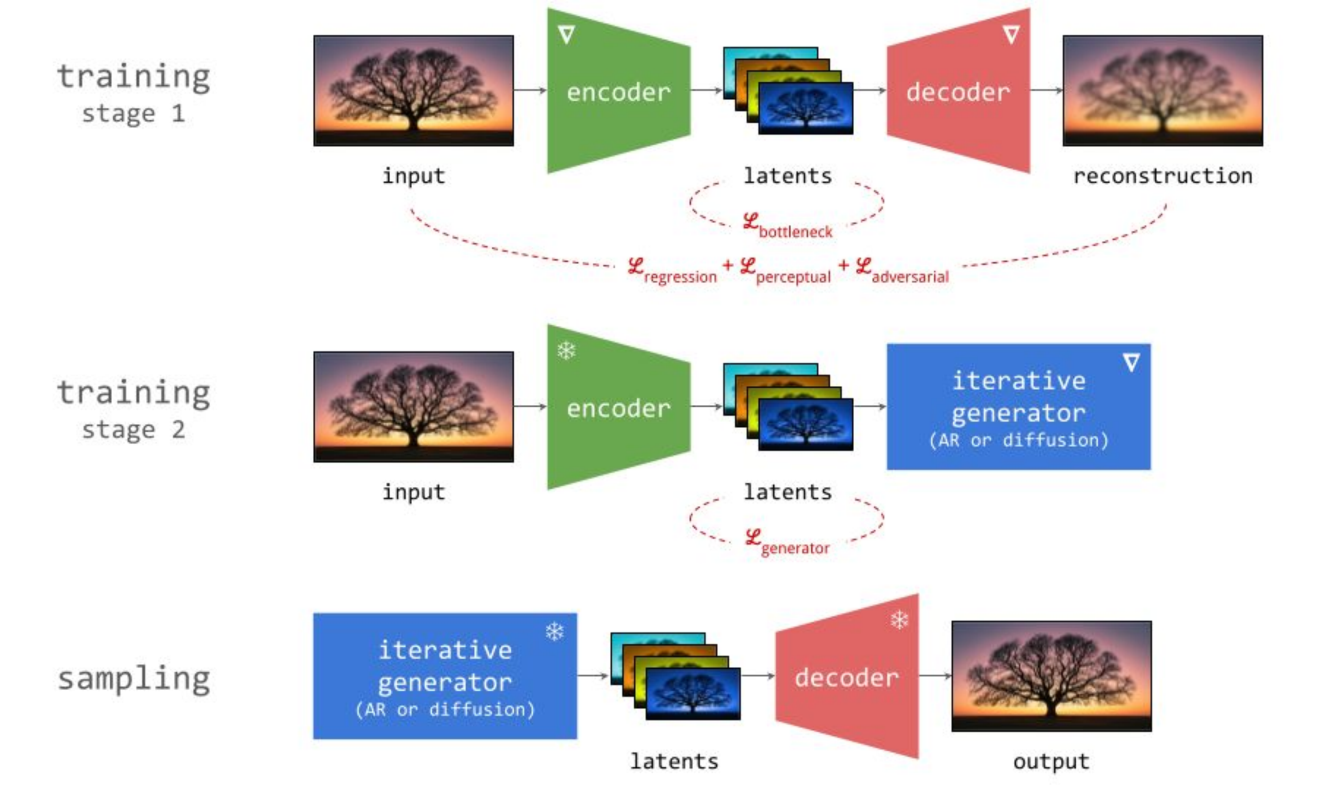
Generalization? Flow vs. Diffusion Endpoints



GIF from Anne Gagneaux, cf. "**On the Closed-Form of Flow Matching: Generalization Does Not Arise from Target Stochasticity**" by Bertrand et al, arXiv:2506.03719 (4 June 2025)

Latent Gen Paradigm

🔥🔥[Sander Dieleman: "Generative modelling in latent space"] (Apr 15 2025)



"Understanding and" Developing

- Could just grab other people's code(s), assemble components, & quickly move on without "deep understanding", *assuming...*
    - your problem of interest is similar enough to theirs
    - can find their code (and maybe weights)
    - the LICENSE is permissive enough.
- For me, *building from scratch* is needed to truly understand, and/or to build for "custom" problems. *And yet...*
    - code won't be as sophisticated/optimized as "theirs"
    - will struggle with *mistakes/bugs*!

Made a repo for this tutorial....

Teaching + Research repo:

Contributors wanted!

flocoder

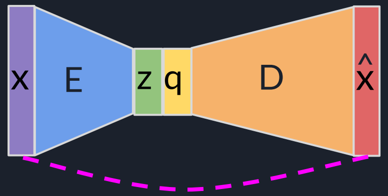
github.com/drscotthawley/flocoder

Teaching Tool vs. Performance?

`flocoder` started as ***teaching*** project, writing code from ***scratch*** so we understand it all

But maybe better to repurpose others' codes (SD3, Meta's,..) for ***performance***?

Latents Obtained via AutoEncoder



Orange =
AutoEncoder stuff

Many people start with a pretrained AutoEncoder

- Images: SD, SDXL, *DC-AE
- Audio: (SoundStream), DAC, Music2Latent…
- ___ for video

Training your own AE can be intensive. There's a lot of data, crafting & "sweat" involved

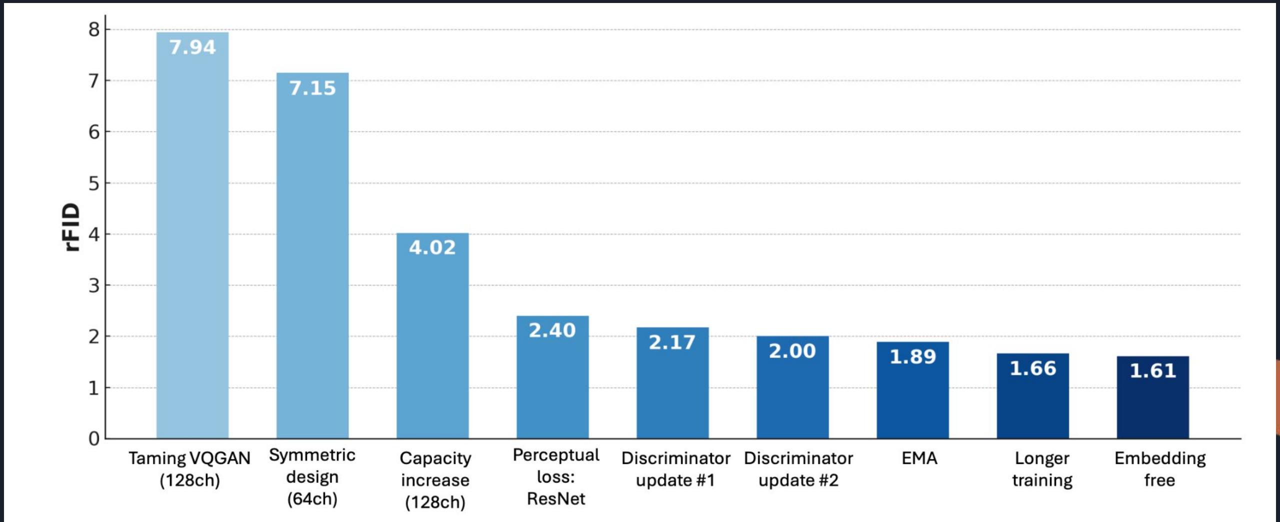
VQGAN training codes & notes:

https://gist.github.com/madebyollin/ff6aeadf27b2edbc51d05d5f97a595d9

2025: Significant AE training updates this year! Next 2 slides…

Open Source VQGAN(+) Progress...

"Although stronger, closed-source VQGAN variants exist..., their details are not fully shared in the literature, creating a significant performance gap for researchers without access to these advanced tokenizers. While the community has made attempts to reproduce these works, none have matched the performance (for both reconstruction and generation) reported in [the closed source variants].." — "MaskBit" aka VQGAN+ paper by Weber et al, TMLR 2024 , arXiv:2409.16211, GitHub: markweberdev/maskbit (TF), lucidrains/maskbit-pytorch



"Deep Compression AutoEncoder (DC-AE)" Chen et al, *ICLR 2025*

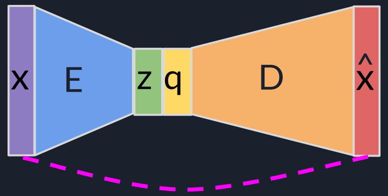
https://github.com/mit-han-lab/efficientvit/

```
from diffusers import AutoencoderDC
dc_ae=AutoencoderDC.from_pretrained("mit-han-lab/dc-ae-f64c128-in-1.0-diffusers",..
```

Orange = AutoEncoder stuff

ImageNet 256×256	Latent Shape	Autoencoder	rFID ↓	PSNR ↑	SSIM ↑	LPIPS ↓
f32c32	8×8×32	SD-VAE [30]	2.64	22.13	0.59	0.117
		DC-AE	**0.69**	**23.85**	**0.66**	**0.082**
f64c128	4×4×128	SD-VAE [30]	26.65	18.07	0.41	0.283
		DC-AE	**0.81**	**23.60**	**0.65**	**0.087**
ImageNet 512×512	Latent Shape	Autoencoder	rFID ↓	PSNR ↑	SSIM ↑	LPIPS ↓
f64c128	8×8×128	SD-VAE [30]	16.84	19.49	0.48	0.282
		DC-AE	**0.22**	**26.15**	**0.71**	**0.080**
f128c512	4×4×512	SD-VAE [30]	100.74	15.90	0.40	0.531
		DC-AE	**0.23**	**25.73**	**0.70**	**0.084**
FFHQ 1024×1024	Latent Shape	Autoencoder	rFID ↓	PSNR ↑	SSIM ↑	LPIPS ↓

Table 2: **Image Reconstruction Results.**

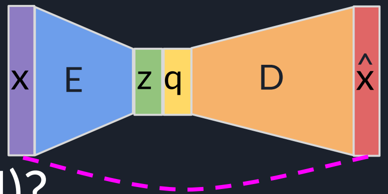


Orange =
AutoEncoder stuff

Grab a pretrained first, then train "custom" later

Probably should skip AE training for now.
=>Grab a pretrained AE (e.g. SD, DC-AE), go on to Flow
…then come back to (custom) AE training if time

Choose your Latent Space



Continuous (VAE) or Quantized (VQVAE / VQGAN)?

- Continuous: Stable Diffusion (images) incl. SD3 ("Scaling Rectified Flow...")
- Quantized: Residual VQ popular for audio, others
  - Tutorial on RVQ: drscotthawley.github.io/blog/posts/2023-06-12-RVQ.html
    - The verdict: Don't write your own quantizer,
      Just use @lucidrains/`vector_quantize_pytorch`

`Many recent models use VAE, not seeing as much VQ.`

- My choice: RVQGAN, Flow in *continuous* (pre-quantizer) latents z
  - Quantized flows exist but... on RVQ point sets??

Generic problems: download a pre-trained AE 🤗

Custom problems / new ideas (e.g. RepL): How to build/train?

- Major GitHub repo's may omit AE training code
- But you may find unofficial implementations

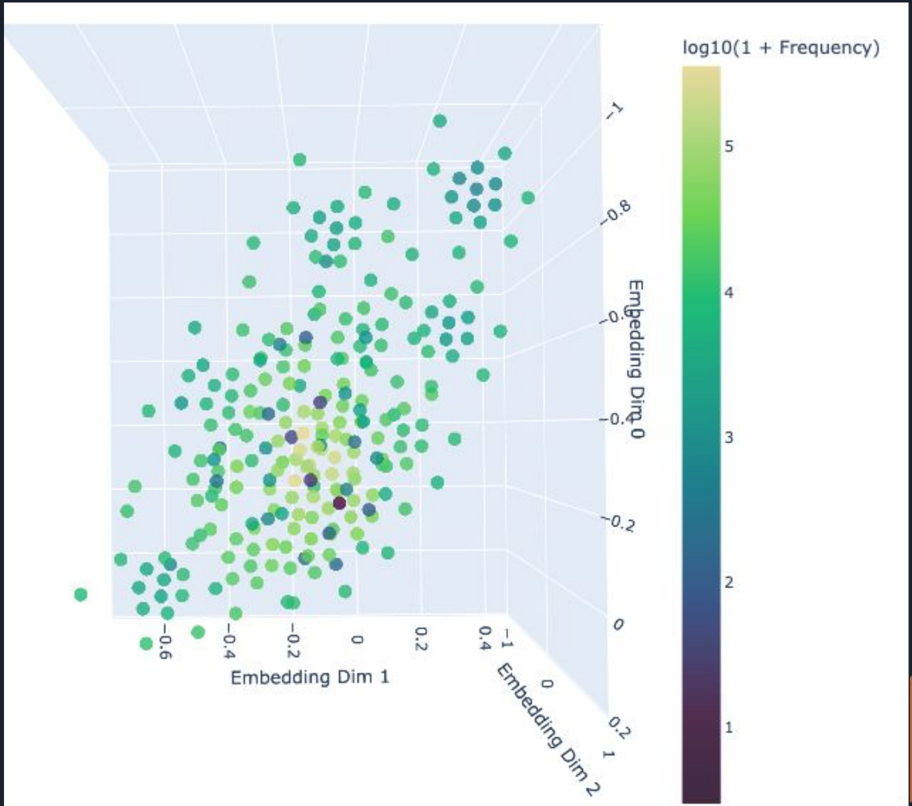
Orange =
AutoEncoder topic

Sample RVQGAN Latent Space

Non-Gaussian in both overall shape (non-oval, clusters), and frequencies

Gives more "options" than VAE but can be "a pain."

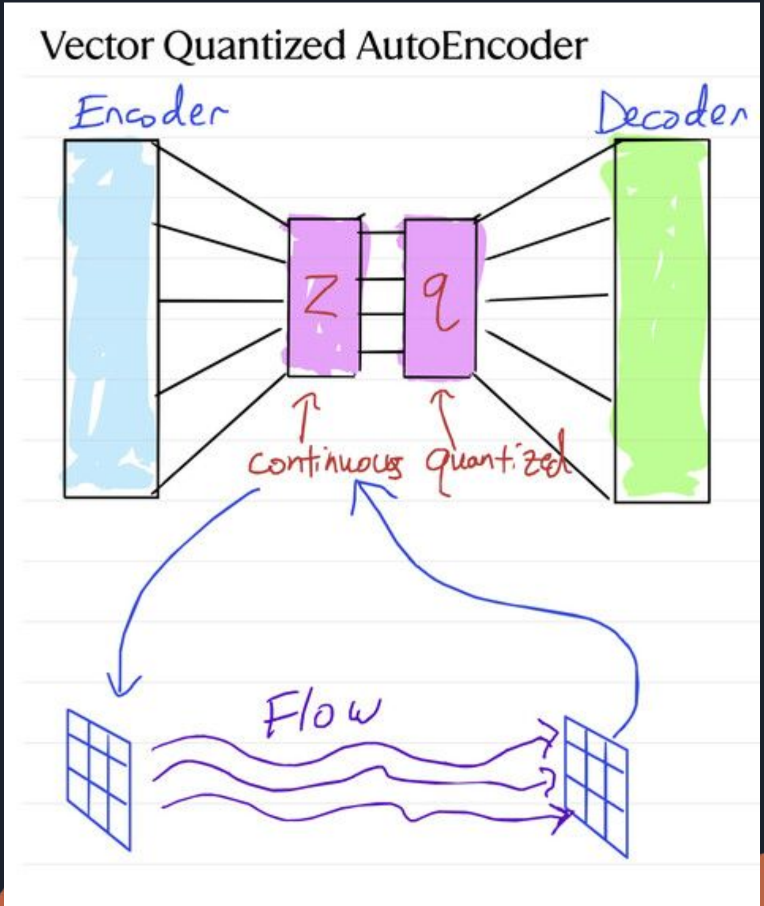
Recc: Start with VAE!



A Picture

(or if we just use Stable Diffusion VAE, nevermind)

TODO: "What does quantization get you?"



AutoEncoder Training Tips

`flocoder/scripts/train_vqgan.py`

VQGAN

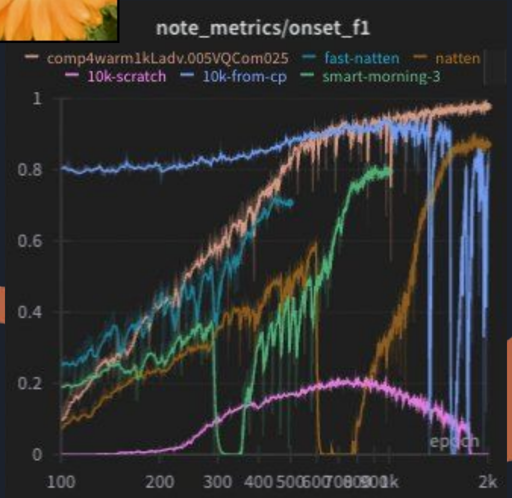
- Resources:
  - github.com/dome272/VQGAN-pytorch
  - github.com/cloneofsimo/vqgan-training
- Keys & tips:
  - Images: Perceptual loss (LPIPS) to help learn textures
  - Avoid instabilities: Gradient clipping, No mixed precision
  - Save VRAM: gradient checkpointing
  - Decoder: use PixelShuffle instead of Upsample or Transposed Conv.
  - Use *non-local* attention at end of Encoder & start of Decoder
    - to separate "what" (e.g. textures) from "where"
  - After a while, freeze Encoder

Trained for X days on RTX A6000…

AutoEncoder Results

- [WandB: Oxford Flowers](#) (128x128x3->16x16x4, 4cb, 64 each)

  - Maybe redo with less compression?
- MIDI (tan line) ------------------------------->



...and "scene."

End of AutoEncoder notes

"Throughput"

Before training flow model,

Pre-Encode the Data

`flocoder/scripts/preencode_data.py`

i.e., train flow model with *frozen* AutoEncoder, not "end to end"

Note: probably want to do augmentation here.

Flow Training (Velocity Model)

- Basic FM setup (Review):
  - Gaussian source data (doesn't need to be but often used)
  - Guess v := Random pairing source<-> target, constant velocity
  - Model learns curvy trajectories
    - Hence ***each guess is very wrong, always & forever!***
    - Optional: train "Reflow" model later for straighter (~OT) trajectories
- Improvements/Options:
  - Time warping for better coverage of time/space domain
  - Better than Random pairing (Source <-> Target)?
    - Minibatch OT (Tong et al 2023), TorchCFM package
      - fast Approx via greedy NN ;-)
  - Add "straightness" loss?

(Training) Metrics - How's It Going?

- Loss is a poor indicator, can nearly flatten out 👎
    - remember why? ;-)
- (Integrated) Output quality
    - FID / FAD score, if applicable
        - Note: For latent gen models, such metrics apply only to the *decodings of* the gen'd latents
- Distribution comparison (of latents): gen'd vs. target
    - Wasserstein, Sinkhorn, Jensen-Shannon, KL,...
        - These may be slow, so use small validation set
- Other statistical comparisons
    - mean, stdev, min, max...
    - Not ideal but fast

Chizat et al NeurIPS 2020: "Faster Wasserstein via Sinkhorn..." cf. `geomloss`

Terminology:
"Learned" == "Marginal"

Coding/Architecture Tips

Velocity Model: UNet, ViT, DiT,...

- Grab someone else's model code…if you can?
- Attention:
  - NATTEN or flash attention
  - RoPE for positional encoding
  - Note *time embedding* may "prefer" OG sinusoidal PE
- Conditioning (i.e., train with extra inputs): via embeddings
  - Make sure conditioning values stay normalized.
  - Image-wide cond signals often merged w/ time emb.
    - class (MNIST, CIFAR,...) number -> embedding
    - text (T5?) embeddings
    - …?

Results, Flowers

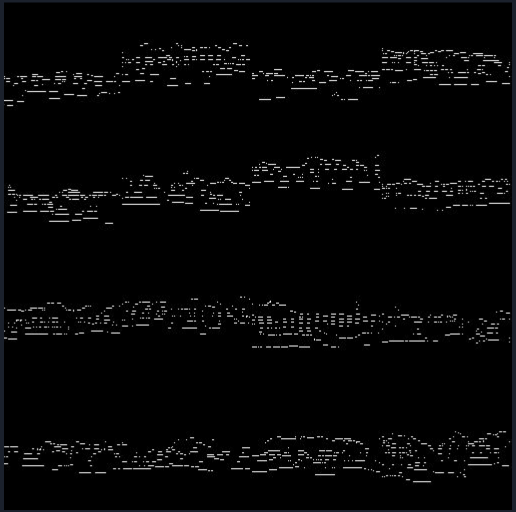
Target

Pred (Unconditional)

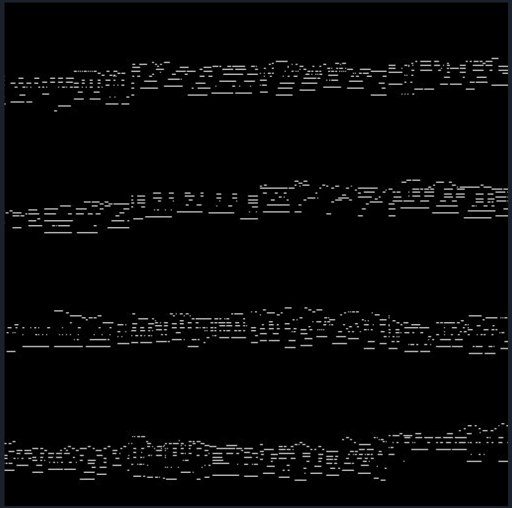
Images: 3x128x128
Latents: 3x16x16
(Custom RVQGAN)
1500 Epochs

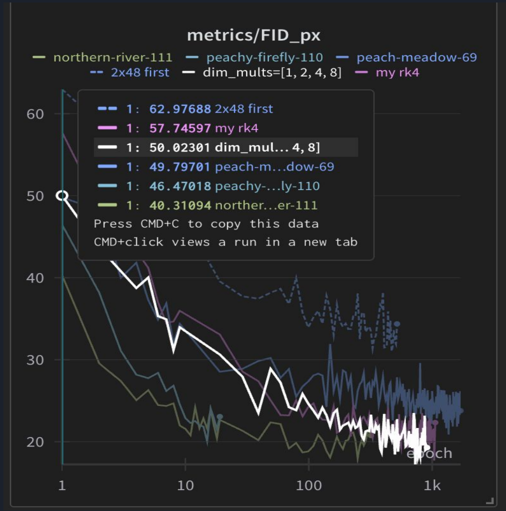
Results, MIDI

Pred (decoded)



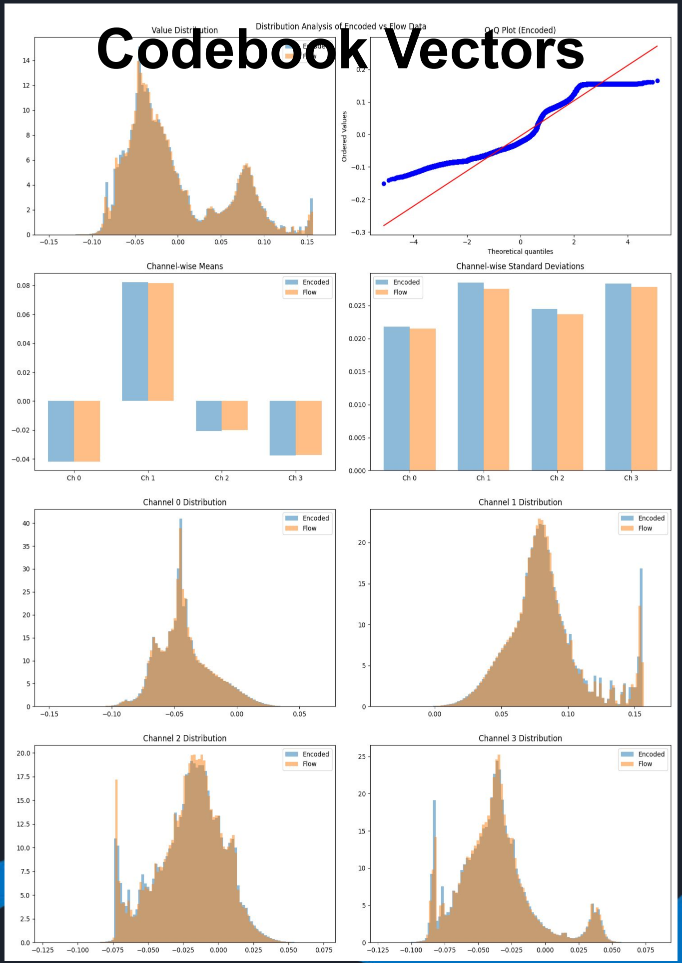
Target (decoded)





Results, MIDI

Flow model learns to generate very non-normal distribution of codebook vectors:



Optimizations

- Training:
  - Faster convergence: Minibatch OT / Greedy NN
  - Better quality(/stability): Use EMA weights
- Inference (speed):
  - Reflow?
  - Higher-order integrator?
  - Quantization?

Further Topics

Inference Tricks:

- Guidance
- Inverse Probs: Deblurring, Super-Res, Inpainting

Scaling Even Bigger

Guidance

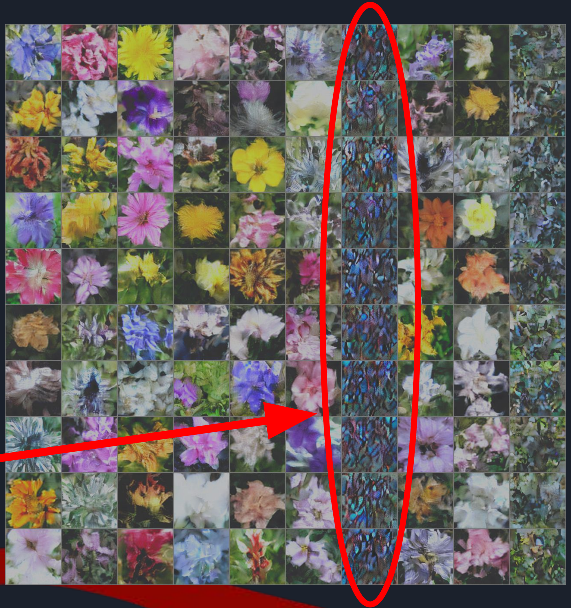
A "plugin" at *inference time*, altering the velocity via linear combination.

Classifier-Free Guidance:

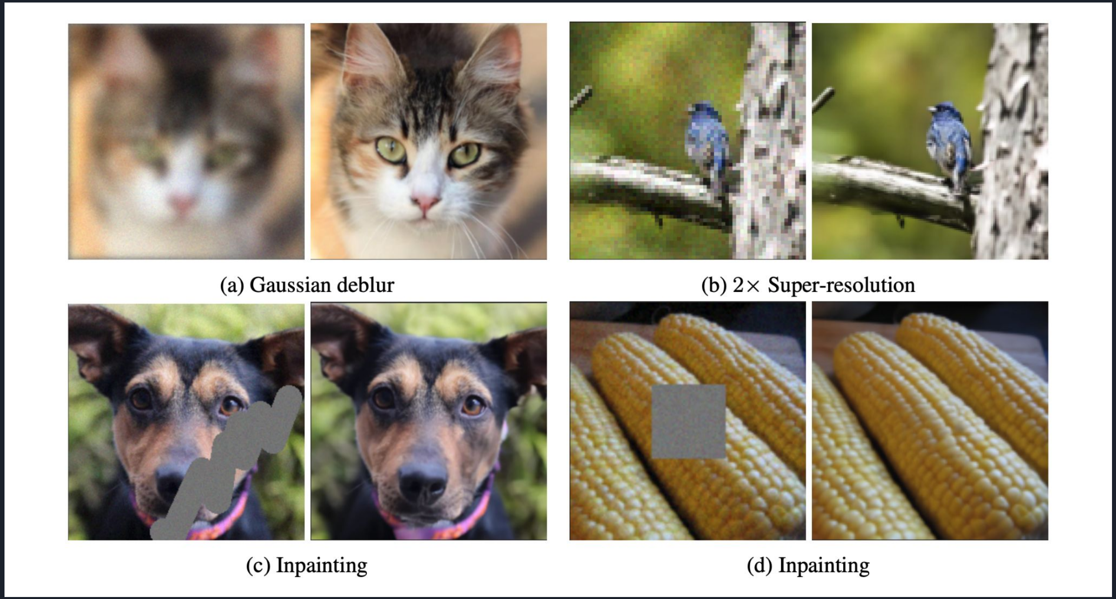
- Training: randomly turn off conditioning
- Inference:

$$v_{cond} = \text{model}(x, t, \text{class})$$
$$v_{uncond} = \text{model}(x, t, \text{None})$$
$$v = v_{uncond} + cfg * (v_{uncond} - v_{cond})$$

- *Otherwise*, you may get class-compliance but poor output quality, e.g.



Inference-Time Inverse Problems



(a) Gaussian deblur (b) 2× Super-resolution

(c) Inpainting (d) Inpainting

"Training Free *Linear* Image Inverses with Flows", Pokle et al 2024 arXiv:2310.04432

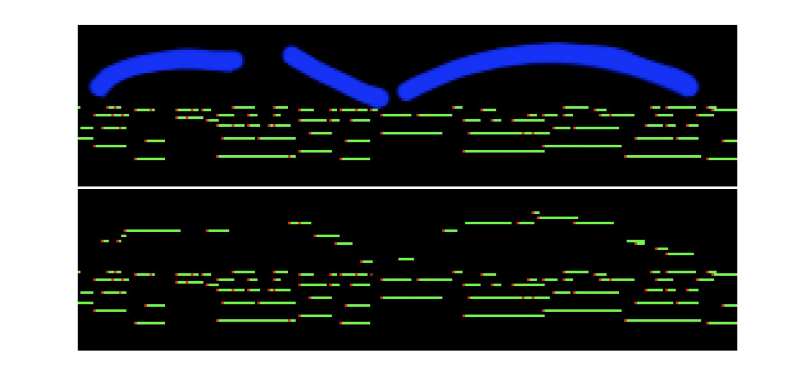
For noisy observations $y = Ax_1 + \varepsilon$
with degradation matrix A

Focus: Inpainting

- Some inpainting you get for free
  - Maximum Likelihood: most (normal) photos in training data don't have blank areas: e.g., face images have noses!
- 2 Papers, Mar. 2024: TFLIIF (prev slide), & PnP-Flow (Martin et al)
- Could also try training a *conditional* Inpainting model
  - in pixel space
  - in pixel-structured latent space
  - in abstract latent space: train a mask encoder too?...

Focus: Inpainting

Motivation: Image from PicturesOfMIDI…



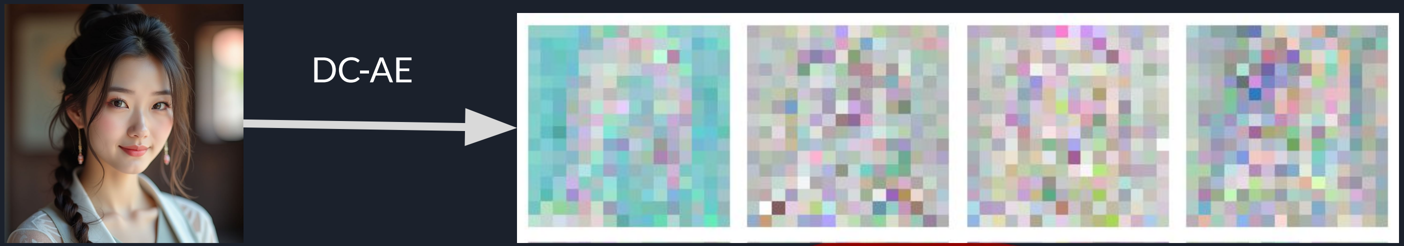
Algorithms for inpainting with diffusion models: CRASH, RePaint, …

...In Latent Space?

$y = Ax_1 + \varepsilon$ where A is a *linear* degradation operator *in pixel space*

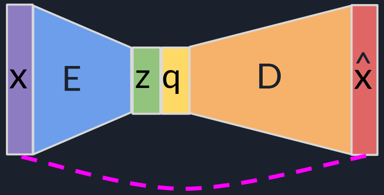
But given a *nonlinear* encoder E to latents $y_L = E(y)$, $x_{1L} = E(x_1)$
in general *there is no* A_L s.t. $y_L = A_L x_{1L} + \varepsilon_L$ (e.g. 67% error!)
Two main ways to handle this:

1. Ensure that E *preserves spatial structure* (e.g. no global attention)



2. Continually project back & forth to "check":
   latents <-> pixels

...In Latent Space?



"Representation learning and reconstruction are two separate tasks, and while it is convenient to do both at once, this might not be optimal."-

–-Sander Dieleman's blog post (https://sander.ai/2025/04/15/latents.html)

> "Bonus: Use custom quantized encoder for *interpretable latent reps!* " ❌

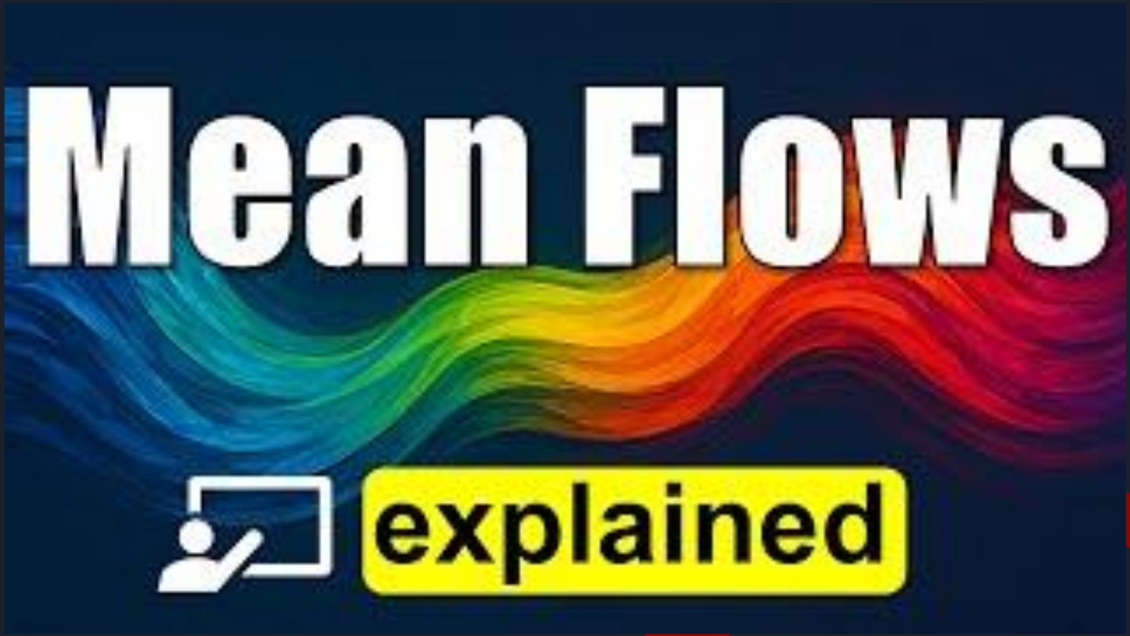
Could do interp & gen, but not inpainting b/c custom encoder used non-local attn which destroyed spatial structure.

Scaling Even Bigger...? ($$)

- Data:
  - Get it ethically, e.g. licensed. (Days of mass scraping are gone)
  - Storage/delivery: maybe WebDataset, cf. works by LAION
- Model(s) still probably fit on one GPU, so:
  - Distributed Data Parallel
  - 🤗 Accelerate / LightingAI / RAPIDS / DeepWhatev / FastWhatev
- Compute: SLURM, NCCL
- Text conditioning: play with prompts(/metadata) a lot
- 🧙 RLHF: pre-release, loop in user feedback
  - May paper Flow-GRPO 👀

Single-Step: Mean Flow Models?

Watch Jia-Bin Huang's Video!
https://www.youtube.com/watch?v=swKdn-qT47Q



References

- My blog post & others, and references therein
- Meta's guide & code, Dec 2024
- Sander D's blog: "Generative modelling in latent space"
- Barrada et al: arxiv.org/abs/2411.03177v2
- TorchCFM repo (Tong et al, Meta)

Thanks! / Acknowledgements

- Raymond Fan (Ph.D., U. Toronto): great discussions
- Tadao Yamoaka: permission to include/use/mod code(s)
  - UNet, MNIST, CIFAR10. (I ported CIFAR10 -> Flowers)
- IJCNN Organizers, esp. Danilo Comminiello 👏👏👏 and Eleonora Grassucci 👏👏👏

Attribution: Curvy shapes from Canva AI free trial, recolored & moved.

–@drscotthawley

Repo

Supplemental Materials

Link to these Google Slides:
https://docs.google.com/presentation/d/1tL3IRDIkK2vAvagCkPXxMMCSTnxoSsiPzgkiHLIY2t8/edit?usp=sharing

aka https://tinyurl.com/IJCNNHawley

Supplementary/ Colab:

https://tinyurl.com/IJCNN2025HawleyColab :

Colab