Today: Generative Models: VAE
Start Post-training and test-time compute

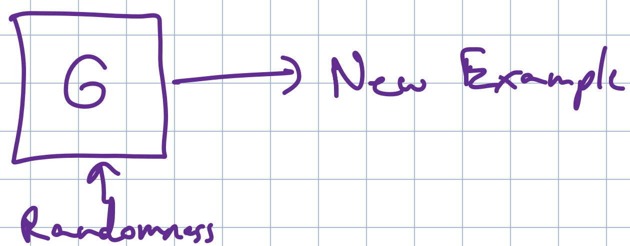
(We'll return to generative models after Post-training interlude)

Announce: Fill out survey
Extra Credit for everyone (3%) IF 75% of class does the survey
Due: Fri of Thanksgiving week.
Submit Draft Reports today & do reviews this weekend

Generative Models

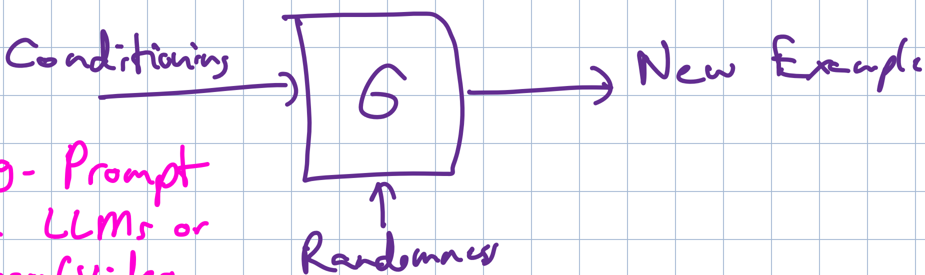
← Understood/Conceptualized as sampling from an unknown distribution.

Unconditional



We'll use these for teaching the core ideas.

Conditional

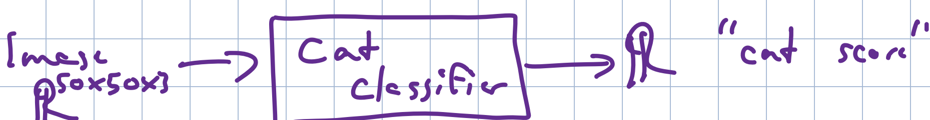


e.g. Prompt in LLMs or Image/Video Generation Models.

This is the most practically useful setting.

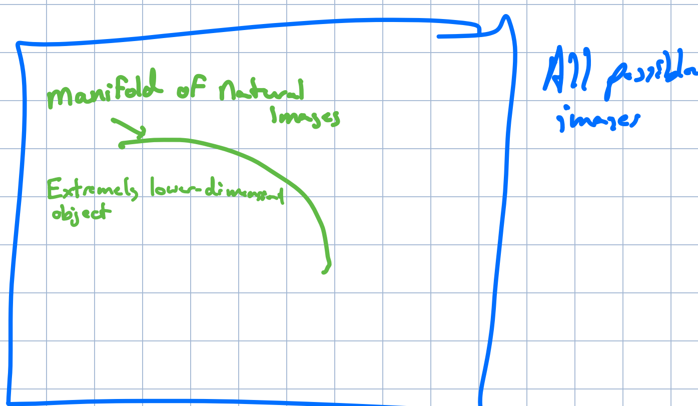
Ideas that don't work

A) Use a classifier



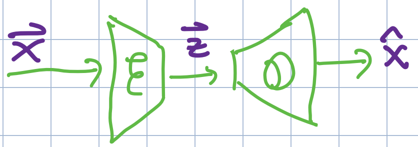
Try: Random Uniform Image followed by Gradient Ascent on the Cat Score

Result: Noise-like image that classifier confidently classifies as a cat.



B) Use an autoencoder

Core Ingredients: Labels are $\vec{x}_i$ itself.
Architecture has an encoder followed by a decoder
Bottleneck in the middle.



Traditional Perspective: Decoder is scaffolding.

Try using $\mathcal{D}$ to generate samples.

Try: Draw $\vec{z}$ from a random distribution. — If $\vec{z}$ too small, get blurry junk
If $\vec{z}$ big, "Noise-like Images."

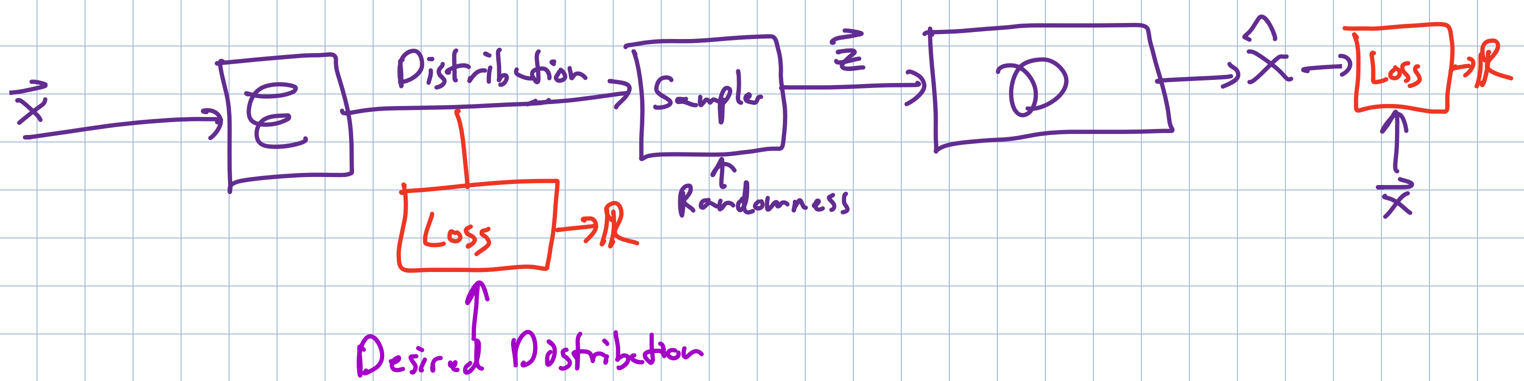
What went wrong? The $\vec{z}$ that was random was nothing like the $\vec{z}$ seen during training.

VAE Approach

3 key ingredients: 1) Make $\vec{z}$ random during training too.

2) Add a loss on distribution of $\vec{z}$

3) Make this work with SGD



Desired Property of Distribution: A) Continuous
B) Easy to sample
C) Easy to compute loss

Loss on Distributions ...

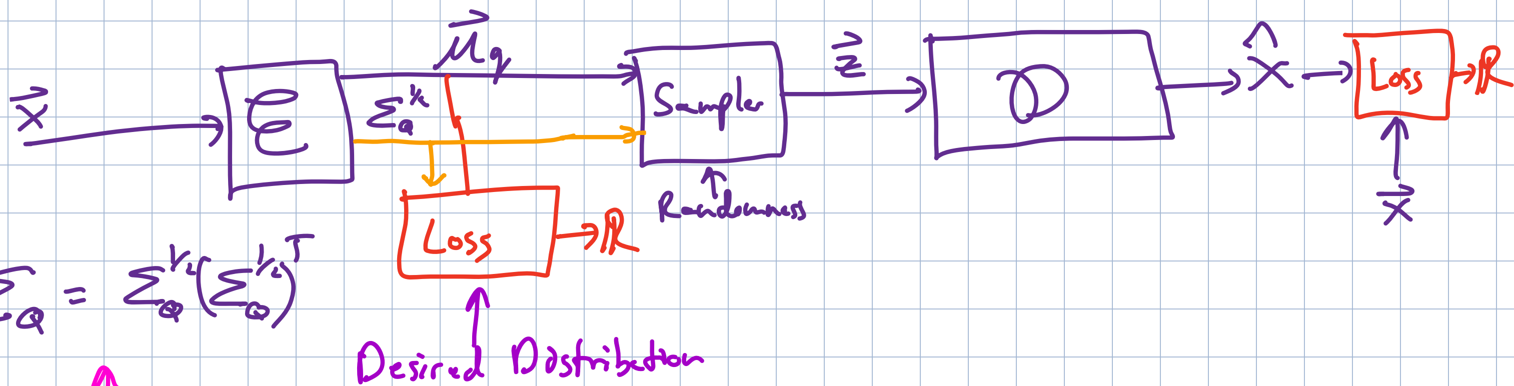
KL Divergence $KL(Q||P) = \int Q(z) \ln \frac{Q(z)}{P(z)} dz$

Asymmetric, but for a good reason.

Note: Put desired distribution in P spot. Q is our distribution.
Why? Don't want Q generating lots of things out-of-distrib.

Our choice for distribution: $N(0, I_k)$

$$KL(N(\vec{\mu}_q, \Sigma_Q) \,||\, N(0, I_k)) = \tfrac{1}{2} Tr(\Sigma_Q) + \vec{\mu}_q^T \vec{\mu}_q - k - \log\det \Sigma_Q$$

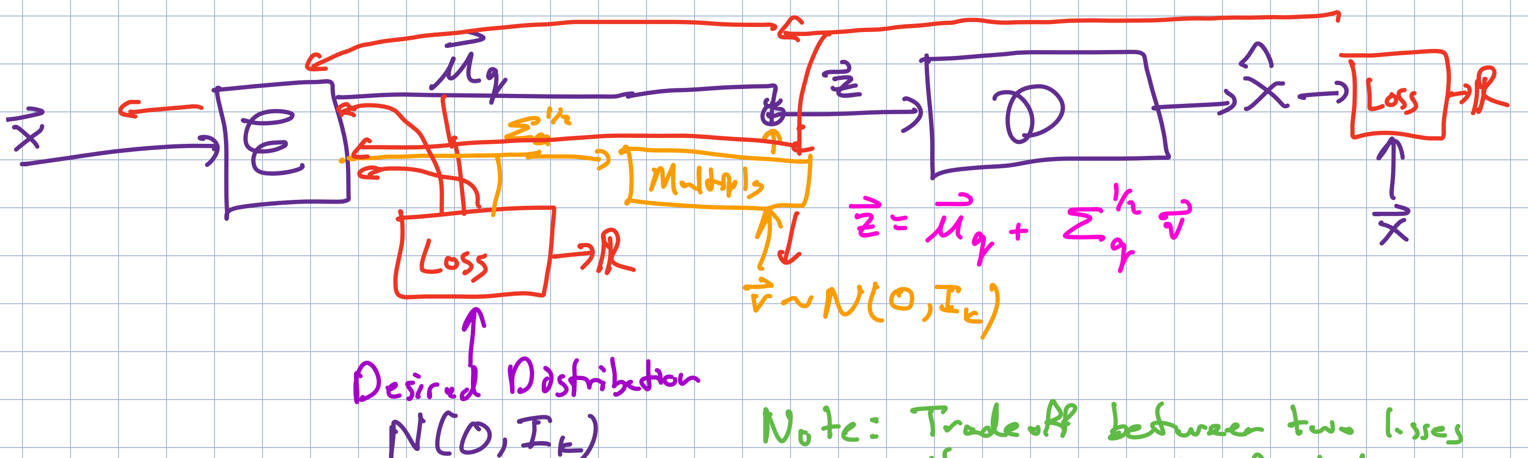


$$\Sigma_Q = \Sigma_Q^{1/2}(\Sigma_Q^{1/2})^T$$

Enforces Positive Semi-Definite

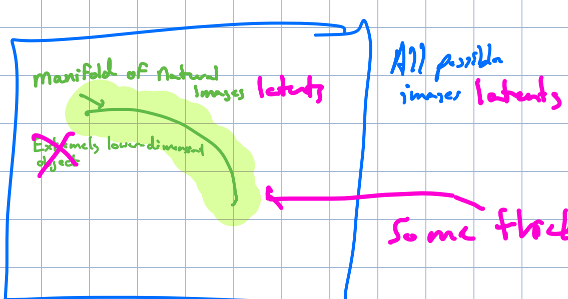
Now Sample...

Gradients make it back to $\mathcal{E}$



Note: Tradeoff between two losses
Hyperparameter for balance.

Distribution loss enforces information bottleneck from $\vec{x}$ to $\vec{z}$



Some thickness due to VAE training.

Key New Tricks: 1) Using a KL Divergence Regularizer on a distribution

2) Treating Sampling in a way that allows gradients to just pass through it.

3) Accepting the stochasticity of random noise in sampling as just more stochasticity in SGD-style optimization.

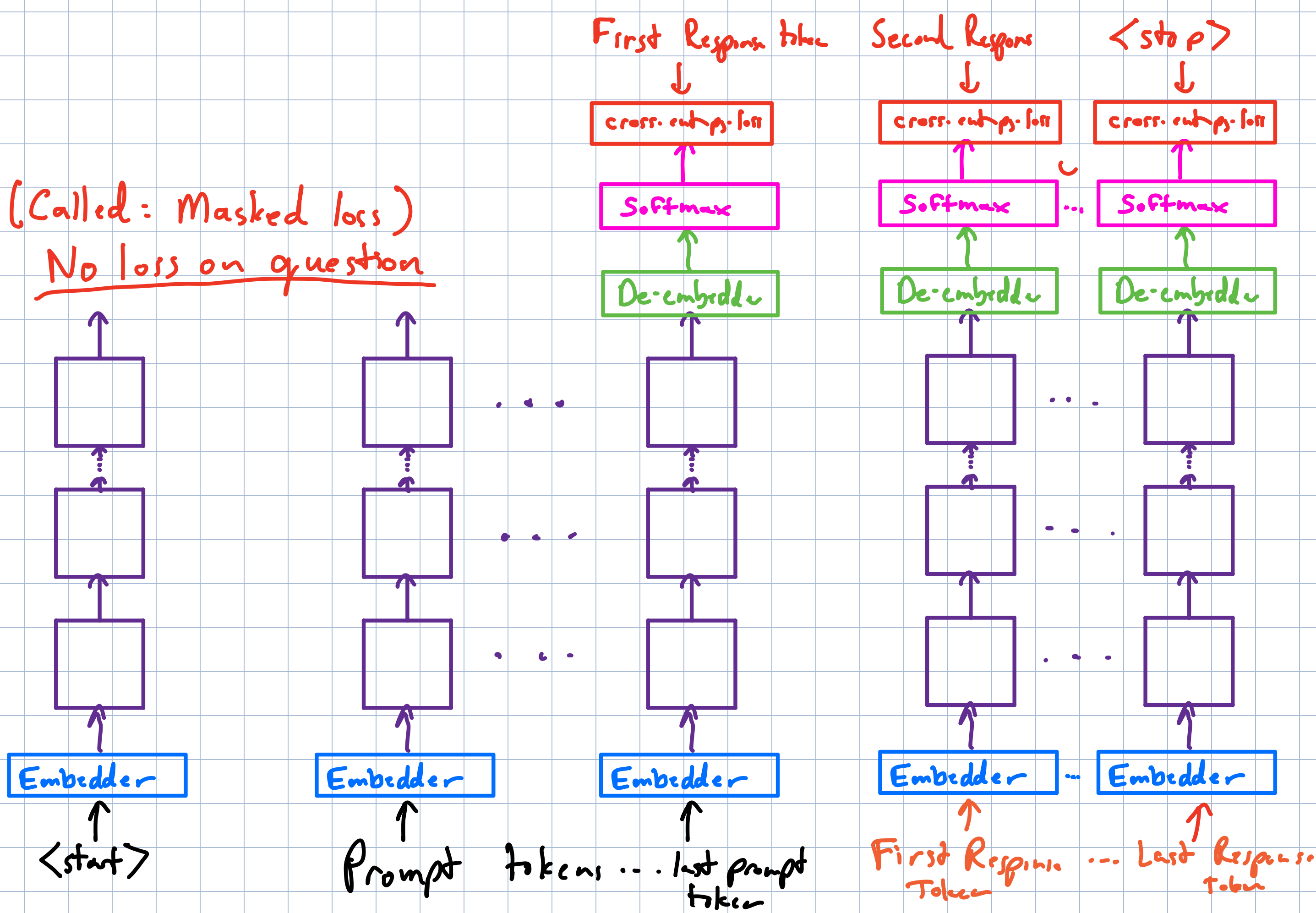
These tricks are useful beyond this VAE setting.

e.g. Quantization-Aware-Training (QAT) uses (2) during training to train a model whose performance does not degrade much when quantized.
Keep weights in high-precision, but quantize in forward-pass.
Ordinarily, quantization is non-differentiable / gives zero-gradients.
But pretend it is added noise and pass gradients thru

You also saw in homework how introducing noise the right way lets you get gradients for some otherwise non-differentiable losses.

Post-training including RLVR.

Recall basic SFT for instruction following.



For LLMs, one key advance was "Chain-of-thought." So it is useful to have examples that show their work while solving a problem.

But how can we make a model better at solving problems?

Two parts to the answer:

A) Be willing to spend more compute while answering.
test time compute

B) Train it to be better.

Test-time compute: 0) Oldest Approach: pure prompting.
"Think step by step. Be careful."

1) Repeated generation. Instead of one try, generate N responses.
At Inference/Test time: A) Take Majority/Plurality Vote
B) Take highest prob

2) Sample better...

Observation (Empirical): RLVR-ed models improved performance and showed generalization/transfer of reasoning to new domains. But people noticed that their outputs were also higher-prob under the base models (without RLVR tuning) as well.
So how much of this gain is distribution shaping?

Recent Paper: Karan & Du, "Reasoning with Sampling: Your base model is smarter than you think" Oct 16, 2025.

We want a good answer. That includes thinking. If we sample from original model, what can go wrong? Moving forward one token at a time, we can make a mistake ... and then flounder.

Older Approach: Beam-Search with some k.

To do what? Sample from higher-likelihood sequences.

Alternative: Sample not from $p(\vec{x})$
but from normalized$(p(\vec{x}))^{\alpha}$ when $\alpha > 1$