THE CONCRETE DISTRIBUTION: A CONTINUOUS RELAXATION OF DISCRETE RANDOM VARIABLES

Chris J. Maddison, Andriy Mnih & Yee Whye Teh
DeepMind & University of Oxford, United Kingdom

Paper Summarization & Practical Part Report

Raz Biton

Machine Learning 2 - 0970209

Contents

1 Main Topic Of The Paper

Libraries like TensorFlow and PyTorch, commonly used in machine learning, handle automatic differentiation. These libraries represent models using a directed acyclic graph, where the computation follows a forward parametric process. If the components of the graph are differentiable, the gradient of the objective can be automatically calculated using the chain rule through backward computation.

A special class of these graphs is Stochastic Computation Graphs (SCGs), which incorporate random variables, making gradient estimation more complex—particularly when dealing with discrete variables. The paper addresses this challenge by introducing the Concrete Distribution, a continuous relaxation technique that enables gradient estimation for discrete random variables. This is achieved by reparameterizing discrete variables using softmax-transformed Gumbel noise. While the method proves effective, it introduces additional considerations, such as ensuring numerical stability and selecting an appropriate temperature to balance between discreteness and smoothness. The authors also provide practical guidelines for choosing suitable temperature values.

The authors tested the Concrete Distribution on tasks like pattern prediction and density estimation using neural networks. They found that even though the gradients from the Concrete relaxation aren't perfectly accurate compared to the true discrete version, they are stable and still effective for training. Additionally, changing the temperature didn't have a big impact in their experiments, but they suggest it's an area that could be explored further in future research.

To further explore the ideas presented in the paper, I chose the **Practical Track** and implemented a discrete VAE using the Concrete relaxation on the MNIST dataset. I followed three different directions in my experiments. First, I reproduced the main technique from the paper: a dVAE with a fixed temperature. Then, I compared the effects of using different fixed temperature values. Finally, I tested a temperature annealing schedule—a method not included in the original paper.

The results showed that the model using Concrete relaxation was able to learn effectively and produce decent reconstructions. Lower temperature values generally led to better performance. However, applying temperature annealing did not improve the results; in most cases, it actually performed worse than using a fixed temperature.

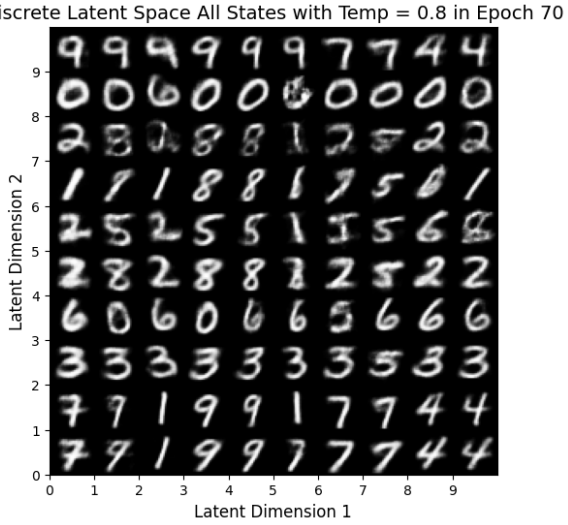


Figure 1: Visualization of all 100 possible latent codes decoded by the model.

2 Summary Of The Paper's Challenges and Proposed Techniques

2.1 Gradient Estimation for Discrete Random Variables in SCG

2.1.1 The Challenge and Approaches to Solve it

This is the main challenge addressed in the article. In Section 2.1, the authors focus on the general problem of optimizing stochastic computation graphs (SCGs).
Suppose we have a stochastic node in the graph $X_i \sim p_\phi(x)$ which samples from a conditional distribution (given its parent nodes), followed by evaluating a deterministic function $f_\theta(x)$ at X. We can think of $f_\theta(X)$ as the objective function. The goal is to optimize the expected value:

$$L(\theta, \phi) = E_{X \sim p_\phi(x)}[f_\theta(X)]$$

with respect to parameters θ, ϕ. Since we want to optimize this function, we need to take the gradient with respect to each parameter.

While the gradient with respect to θ can be estimated easily using Monte Carlo sampling, the gradient with respect ϕ is more difficult. Differentiating the expectation yields:

$$\nabla_\phi L(\theta, \phi) = \nabla_\phi \int p_\phi(x) f_\theta(x)\, dx = \int f_\theta(x) \nabla_\phi p_\phi(x)\, dx,$$

This integral does not have the form of an expectation with respect to x. The gradient is inside the integral, and $\nabla_\phi p_\phi(x)$ is not a probability density—so we can't simply sample x and compute an average as before. This makes it harder to estimate using standard sampling methods.

In Section 2.2, the authors review some state-of-the-art techniques for estimating gradients in stochastic computation graphs (SCGs) and their limitations:

- **Score Function Estimators (SFE):**These can handle both discrete and continuous distributions, but they often suffer from high variance, making gradient estimates unstable—especially with small sample sizes.
- **The Reparameterization Trick:** While effective for continuous variables, it generally cannot be applied to discrete distributions, since they are not differentiable

Because SFE has high variance and the reparameterization trick does not apply directly to discrete variables, the authors introduce a new method based on continuous relaxation.

2.1.2 The Foundations for the Proposed Method

Before diving into the proposed method, Sections 2.4 and 3.1 of the paper provide important foundations. Section 2.4 frames the training of latent variable models, while Section 3.1 introduces the Gumbel-Max trick—a key conceptual step that motivates the construction of the Concrete distribution.

Formulating the Task of Training Latent Variable Models in the SCG Framework
For this formulation, we have:

- **Latent space Z:** A vector of latent variables sampled from the prior distribution $p_\theta(z)$.
- **Observation x:** An observed variable sampled from the conditional distribution $p_\theta(x|z)$ given z.

The marginal probability of the observation x is:

$$p_\theta(x) = \sum_z p_\theta(z) p_\theta(x|z).$$

Computing the log-likelihood $\log p_\theta(x)$ directly is intractable due to the sum over all possible latent configurations. Therefore the authors introduce the multi-sample variational objective with the auxiliary distribution $q_\phi(z|x)$ to approximate the intractable posterior $p_\theta(z|x)$. When $m = 1$, this objective reduces to the widely used Evidence Lower Bound (ELBO). As discussed also in class and lecture notes, the ELBO satisfies:

$$\log p_\theta(x^{(i)}) \geq E_{z\sim q_\phi(z|x^{(i)})}[\log p_\theta(x^{(i)}|z)] - D_{KL}(q_\phi(z|x^{(i)})\|p_\theta(z)).$$

By maximizing the ELBO, we optimize a lower bound on the log-likelihood of the data, $\log p_\theta(x)$, which is our ultimate training objective. This makes learning tractable and allows us to indirectly learn $p_\theta(x)$ by approximating the intractable posterior $p_\theta(z|x)$ with $q_\phi(z|x)$. .

The Gumbel-Max Trick

To motivate the construction of Concrete random variables, the authors review the Gumbel-Max trick—a method for sampling from a discrete distribution using additive noise. Given parameters $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_n)$, where each $\alpha_k > 0$ represents the unnormalized probability of state k, the trick proceeds as follows:

1. Sample $U_k \sim Uniform(0, 1)$ independently for each k,
2. Compute $k^* = \arg\max(\log \alpha_k - \log(-\log U_k))$,
3. Set $D_{k^*} = 1$ and $D_i = 0$ for all $i \neq k^*$ (producing a one-hot vector).

The Gumbel-Max trick alone is not differentiable because of the argmax operation and cannot be directly used in neural networks with backpropagation. That's where the Concrete Distribution (Gumbel-Softmax) comes in.

2.1.3 The Proposed Technique: The Gumbel-Softmax trick (Concrete Distribution)

The Concrete distribution provides a way to approximate discrete random variables with continuous ones, making them suitable for gradient-based optimization. Instead of using the nondifferentiable argmax operation, the Concrete distribution applies a softmax transformation to create a smooth approximation that supports backpropagation. Rather than making a hard choice like with argmax, the softmax gives a smooth probability distribution over all possible outcomes.

The Idea Behind Concrete Random Variables

Discrete variables choose one outcome from many, usually represented by a one-hot vector—a vector with a 1 at one position and 0s elsewhere. These one-hot vectors lie at the corners of a **simplex**, a space of probability vectors where the $(n-1)$-dimensional probability simplex is:

$$\Delta^{n-1} = \left\{ (x_1, x_2, \ldots, x_n) \in R^n \;\middle|\; x_k \geq 0, \sum_{k=1}^{n} x_k = 1 \right\}.$$

Concrete random variables extend this idea by allowing soft assignments—instead of picking one option, they assign probabilities to each.

Steps For Sampling From a Concrete Random Variable

Reviewing the steps, we have:

1. Sample $G_k \sim Gumbel(0, 1)$ for each k (since $G_k = -\log(-\log U_k)$ follows a standard Gumbel distribution).
2. Compute $\tilde{S}_k = \log \alpha_k + G_k$,

3. Instead of selecting the **largest value**, apply a **softmax transformation** with temperature parameter λ:

$$X_k = \frac{\exp\left(\frac{\log \alpha_k + G_k}{\lambda}\right)}{\sum_{i=1}^{n} \exp\left(\frac{\log \alpha_i + G_i}{\lambda}\right)},$$

4. The result $X = (X_1, \ldots, X_n)$ is the probability vector.

The Temperature

Temperature (λ) controls how "soft" or "hard" a selection is in the Concrete distribution. A high temperature increases the level of randomness in making a choice.

Key Properties of Concrete Random Variables

- **Reparameterization:** The distribution of X sampled through the Concrete distribution equation is defined by a closed-form density, and it can be **reparameterized** in terms of Gumbel noise.
- **Rounding:** This property ensures that using the Concrete distribution gives results consistent with true discrete sampling when rounded. (taking the argmax of the soft assignments)

$$P(argmax(X) = k) = \frac{\alpha_k}{\sum_{i=1}^{n} \alpha_i}$$

- **Convexity:** The probability density function (PDF) of the Concrete distribution is log-convex when:

$$\lambda \leq \frac{1}{n-1}$$

  This ensures the distribution remains well-behaved and suitable for optimization.

2.1.4 Concrete Relaxations in Practice

In Section 3.3, the authors show how Concrete random variables can replace discrete ones in a stochastic computation graph. They VAE with a single discrete latent variable as an example. As we saw in the foundations part, the goal is to maximize the ELBO.
In the discrete case, the model uses three main components:

- **Discrete Latent Variable** d: An n-dimensional one-hot discrete variable with unnormalized probabilities a, and a probability mass function $P_a(d)$ representing the prior distribution of the latent space.
- **Likelihood** $p_\theta(x|d)$: A distribution over the data point x, given the latent variable $d \in (0,1)^n$ (this is learned by the decoder to generate data from d).
- **Posterior** $Q_\alpha(d|x)$: An approximating discrete posterior over $d \in (0,1)^n$, with unnormalized probabilities $\alpha(x) \in (0,\infty)^n$ that depend on x (this is learned by the encoder).

Together, the objective to optimize is:

$$L_1(\theta, a, \alpha) = E_{D \sim Q_\alpha(d|x)}\left[\log p_\theta(x|D) \frac{P_a(D)}{Q_\alpha(D|x)}\right]$$

To make this differentiable, The authors propose a relaxed objective by replacing D with a Concrete random variable Z:

$$L_1(\theta, a, \alpha) \to E_{Z \sim q_{\alpha,\lambda_1}(z|x)}\left[\log p_\theta(x|Z) \cdot \frac{p_{a,\lambda_2}(Z)}{q_{\alpha,\lambda_1}(Z|x)}\right]$$

We use different approaches during training and testing to make sure the model learns properly while still working with discrete choices at the end:

- **Training Phase:** Since discrete choices don't allow gradients (which are needed for learning), we replace them with Concrete random variables, which are continuous and differentiable. These Concrete variables act like the discrete ones but can smoothly change, making training possible.
  A temperature parameter controls how "soft" or "sharp" the choices are, and it can be adjusted over time.

- **Test Phase:** Once training is done, we need the model to make actual discrete decisions.
  We convert the Concrete variable back to a discrete one-hot vector using the rounding property (which essentially picks the most likely choice).
  This ensures that at test time, the model behaves like it was originally meant to—with discrete choices.

2.2 Numerical Instability and ExpConcrete

2.2.1 The Challenge

When working with SCGs, like in variational autoencoders, we need to compute log-probabilities of Concrete random variables. However, this can lead to numerical instability due to underflow (When a number is too small, the computer rounds it to zero, losing important information). In Appendix C.3, the authors address this challenge.

2.2.2 The Proposed Technique: Using Log-Space with ExpConcrete

To avoid these issues, the paper introduces a new way to handle calculations using the log-space transformation. This technique is called **ExpConcrete**: Instead of working with raw probabilities, they offer to work with their logarithms. Since the log function is invertible, we can always return to the original values by applying `exp()`. The ExpConcrete variable transformation is defined as:

$$Y_k = \frac{\log \alpha_k + G_k}{\lambda} - L\Sigma E\left(\frac{\log \alpha_i + G_i}{\lambda}\right) = \frac{\log \alpha_k + G_k}{\lambda} - \log \sum_i \exp\left(\frac{\log \alpha_i + G_i}{\lambda}\right)$$

where $G_k \sim Gumbel$ and the LogSumExp function is given by:

$$L\Sigma E(y) = \log \sum_i \exp(y_i)$$

This transformation ensures numerical stability and maintains valid probability distributions.

2.2.3 Applying the Log-Space Technique for Relaxing Discrete Variables

We are revisiting our goal of relaxing $L_1(\theta, a, \alpha)$, which involves working with probabilities of discrete variables in a smooth way.

- We define $Y \sim ExpConcrete(\alpha(x), \lambda_1)$, meaning Y follows an ExpConcrete distribution with parameters $\alpha(x)$ and λ_1. The function $\kappa_{\alpha,\lambda_1}(y \mid x)$ represents its probability density.
- This is linked to the Concrete relaxation $q_{\alpha,\lambda_1}(z \mid x)$, which approximates the true variational posterior $Q_\alpha(d \mid x)$.
- Similarly, we define $\rho_{a,\lambda_1}(y)$, which is the density of another ExpConcrete random variable. This corresponds to the Concrete relaxation $p_{a,\lambda_2}(z)$ of the true distribution $P_a(d)$.

Now, we can express an important expectation equation:

$$E_{Z\sim q_{\alpha,\lambda_1}(z|x)}\left[\log p_\theta(x \mid Z) + \log \frac{p_{a,\lambda_2}(Z)}{q_{\alpha,\lambda_1}(Z \mid x)}\right] = E_{Y\sim \kappa_{\alpha,\lambda_1}(y|x)}\left[\log p_\theta(x \mid \exp(Y)) + \log \frac{\rho_{a,\lambda_2}(Y)}{\kappa_{\alpha,\lambda_1}(Y \mid x)}\right]$$

What this means:

- Instead of working directly with Z, we switch to Y, which is in log-space (ExpConcrete).
- The equation shows that taking the expectation over Z (left side) is equivalent to taking the expectation over Y (right side), but with an exponential transformation.
- This transformation helps maintain stability while preserving the correct probability structure.

In simpler terms, this formulation allows us to work with smoother, more stable values while still accurately representing the original discrete problem.

2.3 Balancing Relaxation: The Temperature Selection Challenge

2.3.1 The Challenge

The performance of Concrete relaxations depends a lot on the choice of the temperature parameter λ during training (as discussed in Section 3.3 and Appendix C.4).
If we don't choose λ properly, the relaxed values might cluster somewhere in between categories instead of aligning with discrete choices and cause the following problems:

- **If λ is too high:** The relaxation becomes too smooth and doesn't approximate a discrete variable well
- **If λ is too low:** The output becomes almost discrete too early, and gradients vanish—making learning difficult.

2.3.2 Technique 1: Guidelines for Choosing a Single Temperature

The Rule of Thumb for Temperature Selection:
FFor general n-class (n-ary) variables, setting $\lambda \leq \frac{1}{n-1}$ helps avoid having probability mass in the middle of the simplex—so the relaxation stays close to discrete behavior.
What Works in Practice:
The paper shows that this rule doesn't always work well in real experiments. In fact, higher temperatures often perform better. For example:

- For $n = 4$, the best λ was found to be 1.
- For $n = 8$, the best λ was $\frac{2}{3}$.
- Across multiple cases $n \in \{2, 4, 8\}$, $\lambda = \frac{2}{3}$ performed well overall.

2.3.3 Technique 2: Using Two Temperatures for KL Divergence

When the loss depends on a KL divergence between two Concrete nodes. Since both the prior and posterior are relaxed using the Concrete distribution, each can get its own temperature parameter:

- λ_1: Posterior temperature (used for learned distributions)
- λ_2: Prior temperature (used for predefined distributions)

Tuning these separately, instead of using just one value, gave much better performance in the authors' experiments.

3 Summary Of The Paper's Main Results

3.1 Experiments

The researchers goal was to evaluate how well Concrete relaxations perform in SCGs with discrete nodes. They focused on two challenging tasks: **density estimation** and **structured output prediction**. The methods were compared against two strong score function baselines: **VIMCO** and **NVIL**.

3.2 Results

The experiments were conducted on MNIST (handwritten digits) and Omniglot (handwritten letters), using Negative Log-Likelihood (NLL) as the main performance metric.

3.2.1 Density Estimation (Generative Modeling)

This task involves fitting a model that captures the underlying data distribution. The key findings include:

- VIMCO performed better for linear models.
- Concrete relaxations outperformed other methods for non-linear models.
- Performance improved when using n-ary layers, where each node can take on one of n possible discrete values (e.g., 2-ary is binary).
- Among n-ary models, 4-ary layers performed better than 8-ary layers, possibly because the added complexity of 8-ary layers did not lead to additional benefits.
- Concrete relaxations also outperformed logistic normal relaxations in some generative settings.

3.2.2 Structured Output Prediction

In this task, the model was trained to predict the bottom half of an MNIST digit given the top half. Results showed that:

- Concrete relaxations consistently outperformed VIMCO.
- Increasing the number of categories (in n-ary layers) led to steady improvements in performance, unlike in the density estimation task.

3.3 Main Conclusions of the Paper

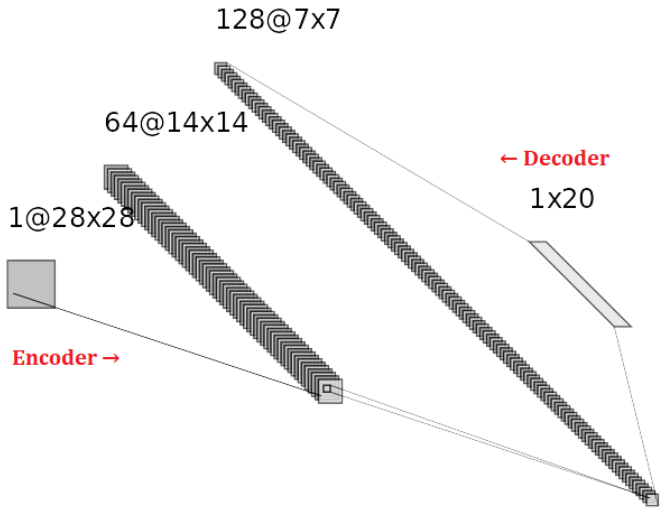
- Concrete distributions provide smooth, differentiable approximations of discrete random variables, enabling gradient-based optimization in models with discrete components.
- Although the gradients produced by Concrete relaxations are biased, they have low variance and serve as unbiased estimators for the relaxed (continuous) objective. This helps the model learn reliably and perform well during training
- Temperature tuning (e.g., annealing schedules) did not lead to significant performance gains in these experiments, though the authors suggest it as a promising direction for future research.

4 Practical Track Report

In this section, I implement the main ideas from the paper, extend them with further exploration, and analyze how the results compare with the findings reported by the authors.

4.1 Implementing The Main Paper Technique

I implemented a discrete VAE (dVAE) using the Concrete (Gumbel-Softmax) relaxation on the MNIST dataset. My architecture closely follows the original paper and differs from the implementation in Tutorial 10. However, I used the same encoder-decoder architecture as the continuous VAE in Tutorial 10:



We recall that the relaxed ELBO objective is:

$$L_1(\theta, a, \alpha) \approx E_{Z \sim q_{\alpha,\lambda_1}(z|x)} \left[\log p_\theta(x|Z) + \log p_{a,\lambda_2}(Z) - \log q_{\alpha,\lambda_1}(Z|x)\right]$$

Model Components:

- **Prior** $P_a(Z)$: Modeled as a relaxed uniform categorical distribution.
- **Posterior** $q_\alpha(Z|x)$: The encoder maps each input image to two categorical variables, each with 10 categories. So the number of possible discrete codes is:

  $$10^2 = 100$$

  However, during training, I applied the Concrete relaxation. Each categorical variable is relaxed to a 10-dimensional probability vector (instead of a one-hot vector), so the full relaxed latent representation is a continuous vector of size:

  $$2 \times 10 = 20$$

- **Likelihood** $p_\theta(x|Z)$: The decoder reconstructs the image from the latent vector using a Bernoulli likelihood (Since MNIST images are grayscale, where each pixel represents a probability of being black or white).

4.1.1 Baseline Model and Training Results

For the baseline, I used $\lambda = \frac{2}{3}$ as recommended in the original paper. The training loss decreases steadily, indicating successful learning:

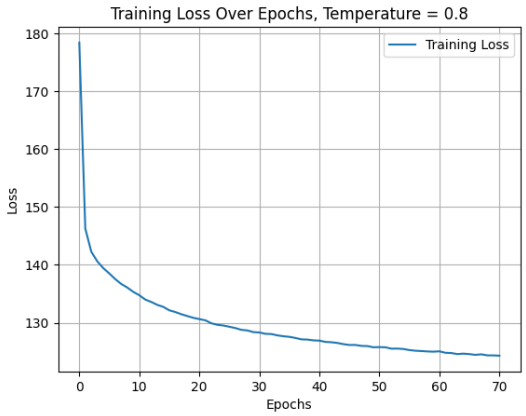


Figure 2: Training loss per epoch. The loss decreases as training progresses, showing that the model is learning.

4.1.2 Visualizing the Discrete Latent Space: Exploring All Possible Codes

Since the latent space consists of only 100 possible codes, I visualized all of them by decoding each one. This allows us to fully explore the discrete latent space:

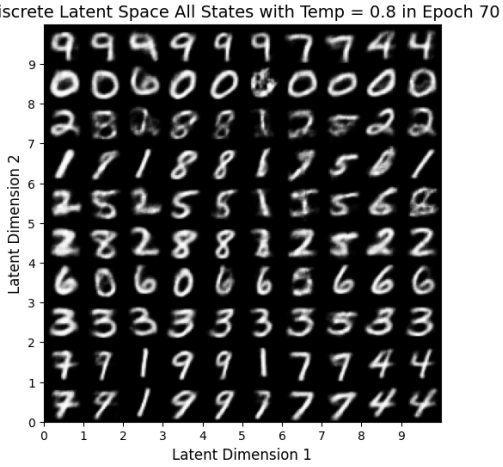


Figure 3: All possible 100 latent codes

4.1.3 Reconstruction Results Over Time

I compared original images with their reconstructions at different training stages. There is a small improvement over time:

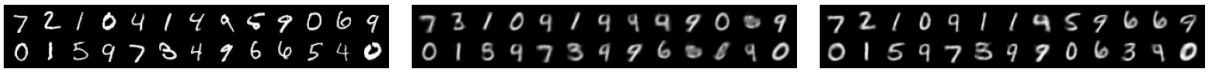


Figure 4: (Left) Original image, (Middle) Reconstruction at epoch 0. (Right) Reconstruction at epoch 70.

4.1.4 Continuous Relaxed Latent Space Visualization Using PCA

I used PCA to reduce the 20D relaxed latent vectors to 2D and plotted them by digit class. We can see a slight improvement in the clustering, though not a significant one. Some digits like 0, 1, 3, and 6 are clustered well; others

like 4 and 9 are mixed due to visual similarity:

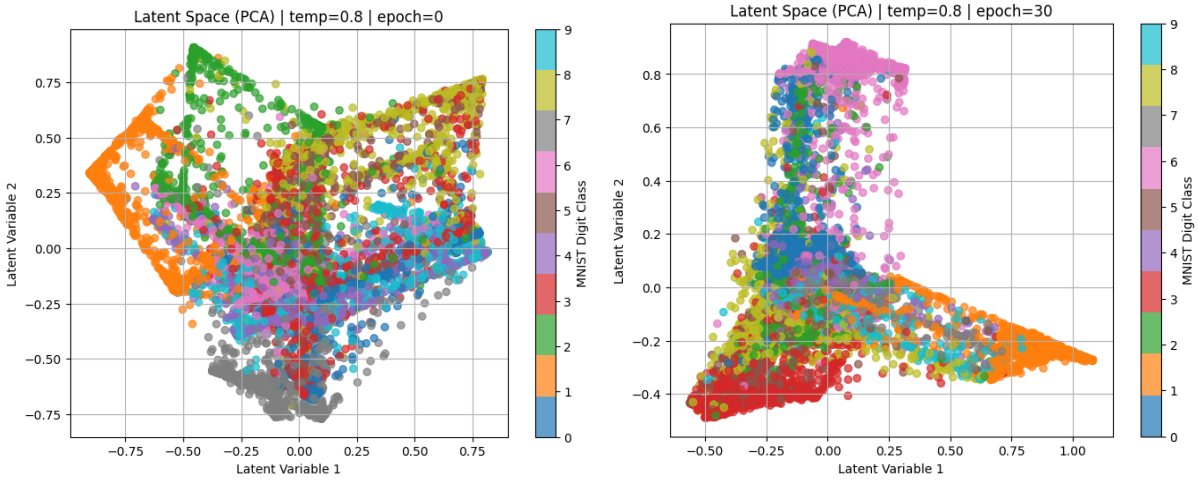


Figure 5: Visualization of the relaxed latent space after PCA: (Left) at epoch 0, (Right) at epoch 70.

4.1.5 Summary and Conclusions

I implemented a dVAE using Concrete relaxation on MNIST, following the approach proposed in the paper.
Key observations:

- The loss decreased smoothly, showing effective optimization.
- The model could explore all 100 possible discrete codes and generate diverse samples.
- Reconstructions improved slightly over time, and most digits were recovered clearly.
- PCA showed meaningful latent structure, though some digit classes (like 4 and 9) overlapped.

Main Conclusion:
The Concrete relaxation enables stable training of models with discrete latent variables. Even with a small latent space, the model learns useful features and produces decent reconstructions. This result is align with the paper.

4.2 Comparison of Different Temperature Values

The paper suggests that the choice of temperature is still an open question. In this section, I experiment with different temperature values and analyze the results. The values tested are: **0.05, 0.3, 0.8, 1, 2, and 5.**

4.2.1 Training Loss

For all the temperature values I tested, the training loss decreased with each epoch. I also noticed a trend: higher temperature values usually led to lower loss. In the figure below, you can see the loss at epoch 0 and at epoch 70 to help visualize this trend.

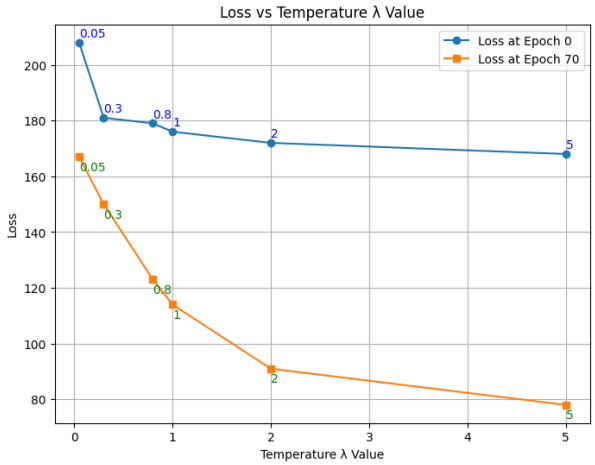


Figure 6: Training loss at epoch 0 and 70 for different temperature values.

4.2.2 Latent Space and Reconstructions

Here, I aim to understand how different temperature values affect the latent space and the quality of the reconstructed images. Since PCA and visualizations alone might not be sufficient for comparison, I also use two additional metrics to evaluate the structure of the latent space and its impact on reconstructions:

- **Intra-class distance:** Measures how close the latent vectors are within the same digit class. A **low** intra-class distance means that similar digits are encoded closely together, which is desirable for consistent reconstructions.
- **Inter-class distance:** Measures how far apart the latent representations of different digit classes are. While a high inter-class distance can help separate classes clearly, it's not always necessary—digits can still be accurately reconstructed even if their latent codes are close, especially when the differences are subtle.

In the following figure, I plot the intra-class and inter-class distances for different temperature values. This helps us observe how the structure of the latent space changes. Alongside the plot, I also show reconstructed image examples for each temperature value, so I can visually compare the quality of reconstructions at different temperatures:

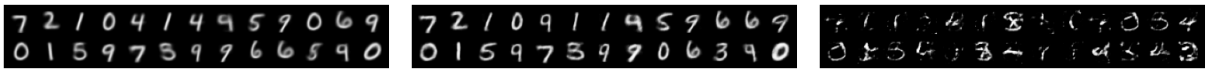


Figure 7: Reconstructions for temperature (Left) 0.05, (Middle) $\frac{2}{3}$, (Right) 5

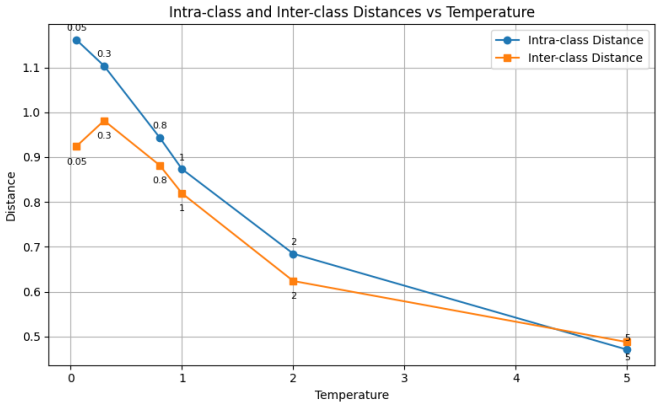


Figure 8: Intra-class and Inter-class Distances vs Temperature

At **low temperatures** (e.g., 0.05), classes are well separated, and reconstructions are usually clear. At **high temperatures** (e.g., 5), classes overlap more, making it harder for the model to reconstruct the digits accurately.

Although higher temperature results in lower intra-class distance (samples of the same class cluster tightly), it also reduces inter-class distance, causing class overlap in latent space. This overlap makes it harder for the decoder to distinguish between digits, leading to worse reconstructions, despite the latent space appearing "smoother" or better clustered in 2D.

4.2.3 Summary and Conclusions

The experiments shows that the temperature parameter plays a significant role in both training and reconstruction. Key observations include:

- Higher temperatures reduce training loss but can blur reconstructions due to class overlap in the latent space.
- Low temperatures (e.g., 0.05) produce sharp, well-separated representations but may hinder optimization.
- A moderate temperature (e.g., $\frac{2}{3}$) strikes the best balance, yielding the highest-quality reconstructions.

Main Conclusions:

- Lower training loss at high temperatures does not necessarily indicate better learning—it often reflects smoother gradients rather than improved latent representations.
- Low temperatures may produce less compact class clusters, but clearer and more accurate reconstructions, due to better class separation.
- High temperatures can cause class overlap in the latent space, which makes decoding more difficult and reduces reconstruction quality.

To summarize, these results highlight the importance of temperature tuning in Concrete relaxation. While the authors found it unimportant in their experiments, my findings show that it significantly affects both model performance and reconstruction quality.

4.3 Temperature Annealing

In the original paper, **the authors noted that annealing the temperature did not lead to significant improvements** in their experiments.
In this section, I test temperature annealing with **a method that does not appear in the original paper** to see if it makes a difference. The idea is to start training with a high temperature λ, which gives smoother gradients, Then, gradually reduce the temperature to make the model behave more like a discrete one. (As seen in earlier sections, higher temperatures often result in lower initial and final losses.)

4.3.1 Annealing Schedule Implementation (A method not from the original paper)

I use a simple exponential decay schedule to reduce the temperature over time:

$$\lambda(t) = \max(\lambda_{min}, \lambda_0 \cdot \exp(-k \cdot t))$$

where:

- λ_0: Initial temperature (e.g., 1.0), $\lambda_{\mathbf{min}}$: Minimum temperature allowed (e.g., 0.5)
- **k**: Decay rate (e.g., 0.03), **t**: Current epoch

At the beginning of each epoch, the temperature is updated using this formula before training continues.

4.3.2 Effect on Training

As mentioned earlier, in theory, using temperature annealing starts the training with smoother gradients. This should make optimization easier at the beginning and lead to relatively lower initial loss.

Experimenting different annealing schedules, I observed that for some models, the training loss actually degraded during training. That is, the loss started to increase at some point after initially decreasing, and only later stabilized again. This unstable behavior was more noticeable with higher annealing rates.
Another result observed, is that models trained with slower annealing rates (temperature decreased more gradually) showed more stable training curves.

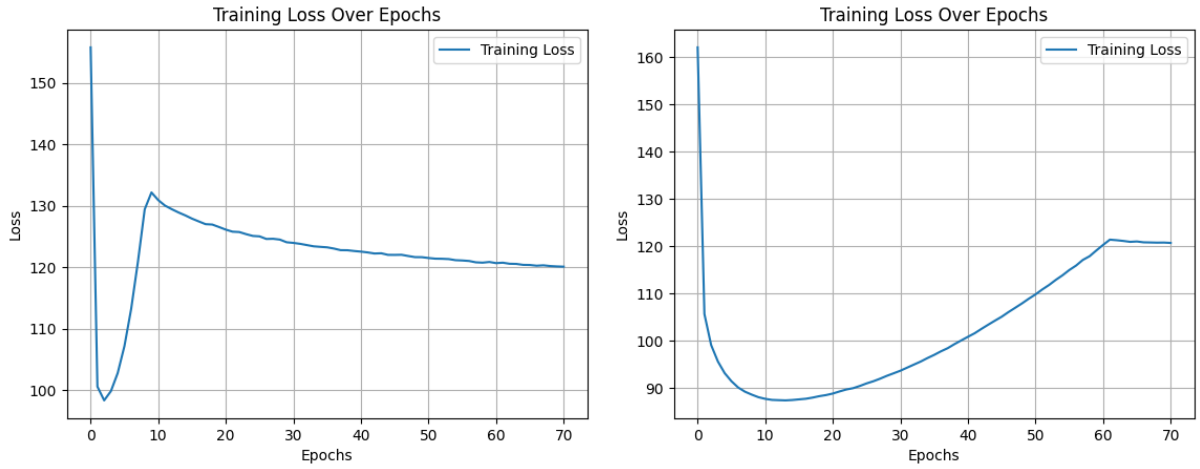


Figure 9: Training loss comparison across different annealing schedules. (Left): Annealing rate = 0.3 with initial temperature = 10. (Middle): Annealing rate = 0.03 with initial temperature = 5.

4.3.3 Effect on Model Performance

To understand how annealing affects the final model quality, I compared different training runs against a baseline model trained with a fixed temperature of $\lambda = \frac{2}{3}$. All annealing-based models were trained with varying initial temperatures and annealing rates, but shared the same minimum temperature (either $\lambda_{min} = \frac{2}{3}$ or 0.3).

Across all experiments, I found that the reconstructions produced by annealed models were worse than those from the baseline. In particular, digits were less clear, and some details were lost.

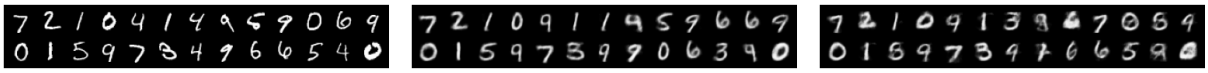


Figure 10: Image reconstructions: (Left) Fixed temperature baseline, (Middle) Anneal rate = 0.003, (Right) Anneal rate = 0.03 with initial temperature = 10 to $\frac{2}{3}$

4.3.4 Summary and Conclusion

Main Conclusions

- Temperature annealing and the rate at which temperature decreases have a clear impact on training. A faster drop (higher annealing rate) can make training unstable.
- For this specific dataset, model, and the annealing settings tested, using a fixed temperature led to better reconstruction quality.

To summarize, my results support the paper's findings: temperature annealing did not improve performance on the reconstruction task. In fact, in most cases, it led to worse results compared to using a fixed temperature.

Resources

- **The Concrete Distribution: A Continuous Relaxation of Discrete Random Variables**
  Chris J. Maddison, Andriy Mnih, Yee Whye Teh
  DeepMind & University of Oxford
- **Variational Autoencoder (VAE) with Discrete Distribution using Gumbel Softmax**
  Alexey Kravets, August 9, 2023