

Soft Convex Quantization: Revisiting Vector Quantization with Convex Optimization

Tanmay Gautam^{1,2}

TGAUTAM23@BERKELEY.EDU

Reid Pryzant¹

REIDPRYZANT@MICROSOFT.COM

Ziyi Yang¹

ZIYIYANG@MICROSOFT.COM

Chenguang Zhu¹

CHEZHU@MICROSOFT.COM

Somayeh Sojoudi²

SOJOURDI@BERKELEY.EDU

¹Microsoft Azure AI, ²Department of Electrical Engineering and Computer Sciences at UC Berkeley

Abstract

Vector Quantization (VQ) is a well-known technique in deep learning for extracting informative discrete latent representations. VQ-embedded models have shown impressive results in a range of applications including image and speech generation. VQ operates as a parametric K-means algorithm that quantizes inputs using a single codebook vector in the forward pass. While powerful, this technique faces practical challenges including codebook collapse, non-differentiability and lossy compression. To mitigate the aforementioned issues, we propose Soft Convex Quantization (SCQ) as a direct substitute for VQ. SCQ works like a differentiable convex optimization (DCO) layer: in the forward pass, we solve for the optimal convex combination of codebook vectors to quantize the inputs. In the backward pass, we leverage differentiability through the optimality conditions of the forward solution. We then introduce a scalable relaxation of the SCQ optimization and demonstrate its efficacy on the CIFAR-10, GTSRB and LSUN datasets. We train powerful SCQ autoencoder models that significantly outperform matched VQ architectures, observing an order of magnitude better image reconstruction and codebook usage with comparable quantization runtime.

Keywords: Differentiable convex optimization, vector quantization, representation learning

1. Introduction

Over the past years, architectural innovations and computational advances have both contributed to the spectacular progress in deep generative modeling (Razavi et al., 2019; Esser et al., 2020; Rombach et al., 2021). Key applications driving this field include image (van den Oord et al., 2017; Esser et al., 2020; Rombach et al., 2021) and speech (Dhariwal et al., 2020) synthesis.

State-of-the-art generative models couple autoencoder models for compression with autoregressive (AR) or diffusion models for generation (van den Oord et al., 2017; Chen et al., 2017; Esser et al., 2020; Rombach et al., 2021; Gu et al., 2022). The autoencoder models are trained in the first stage of the generation pipeline and aim to extract compressed yet rich latent representations from the inputs. The AR or diffusion models are trained in the second stage using latents obtained from the pre-trained autoencoder and are used for generation. Throughout this work, we refer to the autoencoder models as *first-stage* models while the actual generative models trained on the latent space are referred to as *second-stage* models. Therefore, the effectiveness of the entire generation approach hinges upon the extraction of informative latent codes within the first stage. One of

the pervasive means of extracting such latent codes is by embedding a vector quantization (VQ) bottleneck (van den Oord et al., 2017; Razavi et al., 2019) within the autoencoder models.

Motivated by the domain of lossy compression and techniques such as JPEG (Wallace, 1992), VQ is a method to characterize a discrete latent space. VQ operates as a parametric online K-means algorithm: it quantizes individual input features with the “closest” learned codebook vector. Prior to VQ, the latent space of variational autoencoders (VAEs) was continuous and regularized to approximate a normal distribution (Kingma et al., 2014; Kingma and Welling, 2019). The VQ method was introduced to learn a robust discrete latent space that doesn’t suffer the posterior collapse drawbacks faced by VAEs regularized by Kullback-Leibler distance (van den Oord et al., 2017). Having a discrete latent space is also supported by the observation that many real-world objects are in fact discrete: images appear in categories and text is represented as a set of tokens. An additional benefit of VQ in the context of generation is that it accommodates learning complex latent categorical priors. Due to these benefits, VQ underpins several image generation techniques including vector-quantized variational autoencoders (VQVAEs) (van den Oord et al., 2017; Razavi et al., 2019), vector-quantized generative adversarial networks (VQGANs) (Esser et al., 2020), and vector-quantized diffusion (VQ Diffusion) (Gu et al., 2022). Notably it also has application in text-to-image (Ramesh et al., 2021) and speech (Dhariwal et al., 2020) generation.

While VQ has been applied successfully across many generation tasks, there still exist shortcomings in the method. One practical issue pertains to backpropagation when learning the VQ model: the discretization step in VQ is non-differentiable. Currently, this is overcome by approximating the gradient with the straight-through estimator (STE) (Bengio et al., 2013). The VQ method is also plagued by the “codebook collapse” problem, where only a few codebook vectors get trained due to a “rich getting richer” phenomena (Kaiser et al., 2018). Here codebook vectors that lie closer to the distribution of encoder outputs get stronger training signals. This ultimately leads to only a few codebook vectors being used in the quantization process, which impairs the overall learning process. Another limitation with VQ is that inputs are quantized with exactly one (nearest) codebook vector (van den Oord et al., 2017). This process is inherently lossy and puts heavy burden on learning rich quantization codebooks. Several works have aimed at mitigating the aforementioned issues with heuristics (Jang et al., 2017; Maddison et al., 2017; Zeghidour et al., 2021; Dhariwal et al., 2020; Huh et al., 2023; Lee et al., 2022). While the recent works have demonstrated improvements over the original VQ implementation, they are unable to fully attain the desired behavior: exact backpropagation through a quantization step that leverages the full capacity of the codebook.

In view of the shortcomings of existing VQ techniques, we propose a technique called soft convex quantization (SCQ). Rather than discretizing encoder embeddings with exactly one codebook vector, SCQ solves a convex optimization in the forward pass to represent each embedding as a convex combination of codebook vectors. Thus, any encoder embedding that lies within the convex hull of the quantization codebook is exactly representable. Inspired by the notion of differentiable convex optimization (DCO) (Amos and Kolter, 2017; Agrawal et al., 2019), this approach naturally lends itself to effectively backpropagate through the solution of SCQ with respect to the entire quantization codebook. By the means of this implicit differentiation, stronger training signal is conveyed to all codebook vectors: this has the effect of mitigating the codebook collapse issue.

We then introduce a scalable relaxation to SCQ amenable to practical codebook sizes and demonstrate its efficacy with extensive experiments: training 1) VQVAE-type models on CIFAR-10 (Krizhevsky, 2009) and German Traffic Sign Recognition Benchmark (GTSRB) (Stallkamp et al., 2012) datasets, and 2) VQGAN-type models (Esser et al., 2020) on higher-resolution LSUN (Yu

et al., 2015) datasets. SCQ outperforms state-of-the-art VQ variants on numerous metrics with faster convergence. More specifically, SCQ obtains up to an order of magnitude improvement in image reconstruction and codebook usage compared to VQ-based models on the considered datasets while retaining a comparable quantization runtime. We also highlight SCQ’s improved performance over VQ in low-resolution latent compression, which has the potential of easing the computation required for downstream latent generation.

2. Background

2.1. Vector Quantization Networks

Vector quantization (VQ) has risen to prominence with its use in generative modeling (van den Oord et al., 2017; Razavi et al., 2019; Rombach et al., 2021). At the core of the VQ layer is a codebook, i.e. a set of K latent vectors $\mathcal{C} := \{c_j\}_{j=1}^K$ used for quantization. In the context of generative modeling, an encoder network $E_\phi(\cdot)$ with parameters ϕ maps input x into a lower dimensional space to vector $z_e = E(x)$. VQ replaces z_e with the closest (distance-wise) vector in the codebook:

$$z_q := Q(z_e) = c_k, \quad \text{where } k = \arg \min_{1 \leq j \leq K} \|z_e - c_j\|, \quad (1)$$

and $Q(\cdot)$ is the quantization function. The quantized vectors z_q are then fed into a decoder network $D_\theta(\cdot)$ with parameters θ , which aims to reconstruct the input x . Akin to standard training, the overarching model aims to minimize a task specific empirical risk:

$$\min_{\phi, \theta, \mathcal{C}} \mathbb{E}_{x \sim \mathcal{P}_{dist}} [\mathcal{L}_{\text{task}}(D(Q(E(x))), x)] \quad (2)$$

where $\mathcal{L}_{\text{task}}$ could be a reconstruction (van den Oord et al., 2017; Razavi et al., 2019) or perceptual loss (Esser et al., 2020; Rombach et al., 2021) and \mathcal{P}_{dist} is the underlying data distribution. Training is performed using standard first-order methods via backpropagation (Rumelhart et al., 1986). As differentiation through the discretization step is ill-posed, the straight-through-estimator (STE) (Bengio et al., 2013) is used as a gradient approximation.

To ensure an accurate STE, a commitment loss is introduced to facilitate learning the codebook:

$$\mathcal{L}_{\text{commit}}(E_\phi, \mathcal{C}) = (1 - \beta)d(sg[z_e], z_q) + \beta d(z_e, sg[z_q]), \quad (3)$$

where $d(\cdot, \cdot)$ is a distance metric, $\beta > 0$ is a hyperparameter and $sg[\cdot]$ is the *stop gradient* operator. The first term brings codebook nearer to the encoder embeddings, while the second term optimizes over the encoder weights and aims to prevent fluctuations between the encoder outputs and its discretization. Combining $\mathcal{L}_{\text{task}}$ and $\mathcal{L}_{\text{commit}}$ yields a consolidated training optimization:

$$\min_{\phi, \theta, \mathcal{C}} \mathbb{E}_{x \sim \mathcal{P}_{dist}} [\mathcal{L}_{\text{task}}(D_\theta(Q(E_\phi(x))), x) + \mathcal{L}_{\text{commit}}(E_\phi, \mathcal{C})]. \quad (4)$$

A concrete example of this framework is the loss used to train the VQVAE architecture (van den Oord et al., 2017; Razavi et al., 2019):

$$\mathcal{L}_{\text{VQ}}(E_\phi, D_\theta, \mathcal{C}) = \|x - \hat{x}\|_2^2 + (1 - \beta)\|sg[z_e] - z_q\|_2^2 + \beta\|z_e - sg[z_q]\|_2^2. \quad (5)$$

where $\hat{x} := D_\theta(Q(E_\phi(x)))$ is the reconstruction.

2.2. Vector Quantization Challenges

Gradient Approximation. As mentioned in 2.1, differentiation through the discretization step is required to backpropagate through a VQ-embedded network. Taking the true gradient through the discretization would yield zero gradient signal and thus deter any useful model training potential. To this end, the STE is used to approximate the gradient. From the perspective of the discretization function, the upstream gradient is directly mapped to the downstream gradient during backpropagation, i.e. the non-differentiable discretization step is effectively treated as an identity map. While prior work has shown how a well-chosen coarse STE is positively correlated with the true gradient (Yin et al., 2019), further effort has been put into alleviating the non-differentiability issue. In Jang et al. (2017); Maddison et al. (2017), the Gumbel Softmax reparameterization method is introduced. This method reparameterizes a categorical distribution to facilitate efficient generation of samples from the underlying distribution. Let a categorical distribution over K discrete values have associated probabilities π_i for $i \in [K]$. Then we can sample via the reparameterization

$$\text{sample} \sim \arg \max_i \{G_i + \log \pi_i\}, \quad (6)$$

where $G_i \sim \text{Gumbel}(0, 1)$ are samples from the Gumbel distribution. Since the $\arg \max$ operator is not differentiable, the method approximates it with a Softmax operator during backpropagation.

Codebook Collapse. In the context of VQ, codebook collapse refers to the phenomenon where only a small fraction of codebook vectors are used in the quantization process (Kaiser et al., 2018). While the underlying cause is not fully understood, the intuition behind this behavior is that codebook vectors that lie nearer to the encoder embedding distribution receive more signal during training and thus get better updates. This causes an increasing divergence in distribution between embeddings and underused codebook vectors. This misalignment is referred to as an *internal codebook covariate shift* (Huh et al., 2023). Codebook collapse is an undesired artefact that impairs the overarching model’s performance as the full codebook capacity is not used. Thus, there have been many concerted efforts to mitigate this issue. One line of work targets a codebook reset approach: replace the dead codebook vectors with a randomly sampled replacement vector (Zeghidour et al., 2021; Dhariwal et al., 2020). This approach requires careful tuning of iterations before the replacement policy is executed. Another direction of work aims to maintain stochasticity in quantization during the training process (Kaiser et al., 2018; Takida et al., 2022). This body of work is based on observations that the quantization is stochastic at the beginning of training and gradually converges to deterministic quantization (Takida et al., 2022). In (Huh et al., 2023), authors introduce an affine reparameterization of the codebook vectors to minimize the divergence of the unused codebook vectors and embedding distributions.

Lossy quantization. As mentioned previously, VQ-embedded networks are trained with the STE that assume the underlying quantization function behaves like an identity map. Therefore, effective training relies on having a good quantization function that preserves as much information as possible of the encoder embeddings. Given an encoder embedding z_e , the quantized output can be represented as $z_q = z_e + \epsilon$ where ϵ is a measure of the residual error. Since STE assumes the quantization is an identity map, the underlying assumption is that $\epsilon = 0$. In practice, however, the quantization process with a finite codebook is inherently lossy and we have $\epsilon > 0$. Therefore, the underlying quantization function should make the quantization error as small as possible to guarantee loss minimization with the STE. For large residuals, no loss minimization guarantees can be made for the STE. Recent work has proposed an alternating optimization scheme that aims to

reduce the quantization error for VQ ([Huh et al., 2023](#)). In ([Lee et al., 2022](#)), authors introduce residual quantization (RQ) which performs VQ at multiple depths to recursively reduce the quantization residual. While RQ has shown improved empirical performance, it is still plagued with the same core issues as VQ and trades-off additional computational demands for executing VQ multiple times within the same forward pass.

2.3. Differentiable Convex Optimization (DCO) Layers

DCO is an instantiation of implicit layers ([Amos and Kolter, 2017](#)) that enables the incorporation of constrained convex optimization within deep learning architectures. The notion of DCO layers was introduced in [Amos and Kolter \(2017\)](#) as quadratic programming (QP) layers with the name OptNet. QP layers were formalized as

$$\begin{aligned} z_{k+1} := \arg \min_{z \in \mathbb{R}^n} \quad & z^\top R(z_k) z + z^\top r(z_k) \\ \text{s.t.} \quad & A(z_k) z + B(z_k) \leq 0, \\ & \bar{A}(z_k) z + \bar{B}(z_k) = 0 \end{aligned} \tag{7}$$

where $z \in \mathbb{R}^n$ is the optimization variable and layer output, while $R(z_k)$, $r(z_k)$, $A(z_k)$, $B(z_k)$, $\bar{A}(z_k)$, $\bar{B}(z_k)$ are optimizable and differentiable functions of the layer input z_k . Such layers can be naturally embedded within a deep learning architecture and the corresponding parameters can be learned using the standard end-to-end gradient-based training approach prevalent in practice. Differentiation with respect to the optimization parameters in equation 7 is achieved via implicit differentiation through the Karush-Kuhn-Tucker (KKT) optimality conditions ([Amos and Kolter, 2017](#); [Amos et al., 2017](#)). On the computational side, [Amos and Kolter \(2017\)](#) develop custom interior-point batch solvers for OptNet layers that are able to leverage GPU compute efficiency.

3. Methodology

In this section, we introduce the soft convex quantization (SCQ) method as an instantiation of a DCO. SCQ acts as an improved drop-in replacement for VQ that addresses many of the challenges introduced in Section 2.2.

3.1. Soft Convex Quantization with Differentiable Convex Optimization

SCQ leverages convex optimization to perform soft quantization. As mentioned previously, SCQ can be treated as a direct substitute for VQ and its variants. As such, we introduce SCQ as a bottleneck layer within an autoencoder architecture. The method is best described by decomposing its workings into two phases: the forward pass and the backward pass. The forward pass is summarized in Algorithm 1 and solves a convex optimization to perform soft quantization. The backward pass leverages differentiability through the KKT optimality conditions to compute the gradient with respect to the quantization codebook.

Forward pass. Let $X \in \mathbb{R}^{N \times F \times H \times W}$ denote an input (e.g. of images) with spatial dimension $H \times W$, depth (e.g. number of channels) F and batch size N . The encoder $E_\phi(\cdot)$ takes X and returns $Z_e := E_\phi(X) \in \mathbb{R}^{N \times \tilde{F} \times \tilde{H} \times \tilde{W}}$ where \tilde{F} is the embedding dimension and $\tilde{H} \times \tilde{W}$ is the latent resolution. Z_e is the input to the SCQ. The SCQ method first runs VQ on Z_e and stores the resulting

Algorithm 1 Soft Convex Quantization Algorithm**Design choices:** Quantization regularization parameter $\lambda > 0$, embedding dimension \tilde{F} , codebook size K **Input:** Encodings $Z_e \in \mathbb{R}^{N \times \tilde{F} \times \tilde{H} \times \tilde{W}}$ **Return:** Convex quantizations $Z_q \in \mathbb{R}^{N \times \tilde{F} \times \tilde{H} \times \tilde{W}}$ **Parameters:** Randomly initialize codebook $C \in \mathbb{R}^{\tilde{F} \times K}$ **begin forward**

1. $Z_e^{\text{flattened}} \in \mathbb{R}^{\tilde{F} \times N \tilde{H} \tilde{W}} \leftarrow \text{Reshape}(Z_e)$

2. $\tilde{P} \in \mathbb{R}^{K \times N \tilde{H} \tilde{W}} \leftarrow \text{VQ}(\text{detach}(Z_e^{\text{flattened}}))$

3. $P^* := \arg \min_{P \in \mathbb{R}^{K \times N \tilde{H} \tilde{W}}} \|Z_e^{\text{flattened}} - CP\|_F^2 + \lambda \|P - \tilde{P}\|_F^2 : P \geq 0, 1_K^\top P = 1_{N \tilde{H} \tilde{W}}$

4. $Z_q^{\text{flattened}} \in \mathbb{R}^{\tilde{F} \times N \tilde{H} \tilde{W}} \leftarrow CP^*$

5. $Z_q \in \mathbb{R}^{N \times \tilde{F} \times \tilde{H} \times \tilde{W}} \leftarrow \text{Reshape}(Z_q^{\text{flattened}})$

end

one-hot encoding as $\tilde{P} \in \mathbb{R}^{K \times N \tilde{H} \tilde{W}}$. The codebook used to obtain \tilde{P} is the same codebook used throughout the SCQ process; \tilde{P} represents the output of the VQ method given the SCQ codebook. \tilde{P} is detached from the computational graph and treated as a constant, i.e. no backpropagation through \tilde{P} . Then Z_e is passed into a DCO of the form

$$\begin{aligned} P^* := \arg \min_{P \in \mathbb{R}^{K \times N \tilde{H} \tilde{W}}} & \|Z_e^{\text{flattened}} - CP\|_F^2 + \lambda \|P - \tilde{P}\|_F^2 \\ \text{s.t. } & P \geq 0, \\ & P^\top 1_K = 1_{N \tilde{H} \tilde{W}}, \end{aligned} \quad (8)$$

where $Z_e^{\text{flattened}} \in \mathbb{R}^{\tilde{F} \times N \tilde{H} \tilde{W}}$ is a flattened representation of Z_e , $C \in \mathbb{R}^{\tilde{F} \times K}$ represents a randomly initialized codebook matrix of K latent vectors, $\lambda > 0$ is a regularization parameter and $P \in \mathbb{R}^{K \times N \tilde{H} \tilde{W}}$ is a matrix we optimize over. SCQ solves convex optimization 8 in the forward pass: it aims to find weights P^* that best reconstruct the columns of $Z_e^{\text{flattened}}$ with a convex combination of codebook vectors. The regularization term in the objective biases the SCQ solution towards the one-hot VQ solution, i.e. we observe that $\lim_{\lambda \rightarrow \infty} P^* = \tilde{P}$. This shows that VQ is a particular instantiation of SCQ for $\lambda \rightarrow \infty$. When keeping λ finite, we obtain a balance between the one-hot VQ solution compatible with downstream autoregressive generative processes and the improved backpropagation, codebook collapse prevention capabilities of the DCO.

The constraints in optimization 8 enforce that the columns of P lie on the unit simplex, i.e. they contain convex weights. The codebook matrix C is a parameter of the DCO and is updated with all model parameters to minimize the training loss. It is randomly initialized before training and is treated as a constant during the forward pass. The SCQ output is given by $Z_q^{\text{flattened}} := CP^*$. This is resolved to the original embedding shape and passed on to the decoder model.

Backward pass. During the forward pass SCQ runs VQ and then solves optimization 8 to find a sparse, soft convex quantization of Z_e . The underlying layer parameters C are treated as constants during the forward pass. C is updated with each backward pass during training. As C is a parameter of a convex optimization, DCO enables backpropagation with respect to C via implicit differentiation through the KKT conditions (Agrawal et al., 2019).

Improved backpropagation and codebook coverage with SCQ. During the forward pass of SCQ, multiple codebook vectors are used to perform soft quantization on Z_e . Optimization 8 selects a convex combination of codebook vectors for each embedding in Z_e . Therefore, SCQ is inclined to

better utilize the codebook capacity over VQ where individual codebook vectors are used for each embedding. Owing to the DCO structure of SCQ, we can also backpropagate effectively through this soft quantization step, i.e. training signal is distributed across the entire codebook.

Improved quantization with SCQ. We consider how the quantization error is incurred for VQ: the error is measured between the input feature and the “closest” codebook vector. For SCQ, any individual input feature can be exactly reconstructed if it lies within the convex hull of the set of codebook vectors. This is a consequence of the formulation of optimization 8. Thus, for any input feature that does not coincide exactly with a codebook vector and lies within the convex hull of codebook vectors, we incur no quantization error in SCQ whereas we incur nonzero error in the VQ method. For input features outside the convex hull of the set of codebook vectors, we again incur smaller error for SCQ as we measure the error with respect to the projection onto the convex hull. This intuitively suggests SCQ’s propensity for low quantization errors during the forward pass as compared to VQ variants that are inherently more lossy.

3.2. Scalable Soft Convex Quantization

As proposed in (Amos and Kolter, 2017), optimization 8 can be solved using interior-point methods which give the gradients for free as a by-product. Existing software such as CVXPYLayers (Agrawal et al., 2019) is readily available to implement such optimizations. Solving 8 using such second-order methods incurs a cubic computational cost of $O((NK\tilde{H}\tilde{W})^3)$. However, for practical batch sizes of $N \approx 100$, codebook sizes $K \approx 100$ and latent resolutions $\tilde{H} = \tilde{W} \approx 50$, the cubic complexity of solving 8 is intractable.

To this end we propose a scalable relaxation of the optimization 8 that remains performant whilst becoming efficient. More specifically, we approximate 8 by decoupling the objective and constraints. We propose first solving the regularized least-squares objective with a linear system solver and then projecting the solution onto the unit simplex. With this approximation, the overall complexity decreases from $O((NK\tilde{H}\tilde{W})^3)$ for the DCO implementation to $O(K^3)$. In practice ($K \approx 10^3$) this linear solve adds negligible overhead to the wall-clock time as compared to standard VQ. This procedure is outlined in our revised scalable SCQ method shown in Algorithm 2. The projection onto the unit simplex is carried out by iterating between projecting onto the nonnegative orthant and the appropriate hyperplane.

4. Experiments

This section examines the efficacy of SCQ by training autoencoder models in the context of generative modeling. Throughout this section we consider a variety of datasets including CIFAR-10 (Krizhevsky, 2009), the German Traffic Sign Recognition Benchmark (GTSRB) (Stallkamp et al., 2012) and higher-dimensional LSUN (Yu et al., 2015) Church and Classroom. We run all experiments on 48GB RTX 8000 GPUs. Details on hyperparameter configurations and convergence plots can be found in our extended online technical report (Gautam et al., 2023).

4.1. Training VQVAE-Type Models

We consider the task of training VQVAE-type autoencoder models with different quantization bottlenecks on CIFAR-10 (Krizhevsky, 2009) and GTSRB (Stallkamp et al., 2012). This autoencoder architecture is still used as a first stage within state-of-the-art image generation approaches such as

Algorithm 2 Practical Soft Convex Quantization Algorithm

Design choices: Regularization parameter $\lambda > 0$, number of projection steps m , embedding dimension \tilde{F} , codebook size K

Input: Encodings $Z_e \in \mathbb{R}^{N \times \tilde{F} \times \tilde{H} \times \tilde{W}}$

Return: Convex quantizations $Z_q \in \mathbb{R}^{N \times \tilde{F} \times \tilde{H} \times \tilde{W}}$

Parameters: Randomly initialize codebook $C \in \mathbb{R}^{\tilde{F} \times K}$

begin forward

1. $Z_e^{\text{flattened}} \in \mathbb{R}^{\tilde{F} \times N\tilde{H}\tilde{W}} \leftarrow \text{Reshape}(Z_e)$

2. $\tilde{P} \in \mathbb{R}^{K \times N\tilde{H}\tilde{W}} \leftarrow \text{VQ}(Z_e^{\text{flattened}})$

3. $P \in \mathbb{R}^{K \times N\tilde{H}\tilde{W}} \leftarrow \text{LinearSystemSolver}(C^\top C + \lambda I, C^\top Z_e^{\text{flattened}} + \lambda \tilde{P})$

for $i \in [m]$ **do**

4. $P \leftarrow \max(0, P)$

5. $P_{:,k} \leftarrow P_{:,k} - \frac{\sum_j P_{j,k}-1}{K} \mathbf{1}_K, \quad \forall k \in [N\tilde{H}\tilde{W}]$

end for

6. $P^* \leftarrow P$

7. $Z_q^{\text{flattened}} \in \mathbb{R}^{\tilde{F} \times N\tilde{H}\tilde{W}} \leftarrow CP^*$

8. $Z_q \in \mathbb{R}^{N \times \tilde{F} \times \tilde{H} \times \tilde{W}} \leftarrow \text{Reshape}(Z_q^{\text{flattened}})$

end

VQ Diffusion (Gu et al., 2022). The autoencoder structure is depicted in Gautam et al. (2023) and is trained with the standard VQ loss 5.

We compare the performance of SCQ against existing methods VQVAE (van den Oord et al., 2017), Gumbel-VQVAE (Jang et al., 2017), RQVAE (Lee et al., 2022), VQVAE with replacement (Zeghidour et al., 2021; Dhariwal et al., 2020), VQVAE with affine codebook transformation and alternating optimization (Huh et al., 2023). For reference, we also include results for a non-quantized autoencoder; this model is not compatible with downstream generative applications due to the missing latent structure provided by a quantization bottleneck. The autoencoder and quantization hyperparameters used for each dataset are detailed in Gautam et al. (2023). The performance is measured using the reconstruction mean square error (MSE) and quantization error. The reconstruction error measures the discrepancy in reconstruction at the pixel level, while the quantization error measures the incurred MSE between the encoder outputs Z_e and quantized counterpart Z_q . We also measure the perplexity of each method to capture the quantization codebook coverage. Larger perplexity indicates better utilization of the codebook capacity. In this experiment, the results on the test datasets were averaged over 5 independent training runs for 50 epochs. Table 1 presents the results.

SCQVAE outperforms all baseline quantization methods across all metrics on both datasets: the model yields significantly improved quantization errors and perplexity measures. The improved quantization error suggests better information preservation in the quantization process, while improved perplexity indicates that SCQ enables more effective backpropagation that better utilizes the codebook’s full capacity. These improvements were attained whilst maintaining training wall-clock time with the VQ baselines. The RQVAE method, on the other hand, did incur additional training time (approximately 2 \times) due to its invoking of multiple VQ calls within a single forward pass.

Figures 1 (a) and (b) illustrate SCQVAE’s improved convergence properties over state-of-the-art RQVAE (Lee et al., 2022) and VQVAE with replacement, affine transformation and alternating optimization (Huh et al., 2023). For both datasets, SCQVAE is able to converge to a lower reconstruction MSE on the test dataset (averaged over 5 training runs). We next considered the higher-dimensional (256 \times 256) LSUN (Yu et al., 2015) Church dataset. Figure 2 visualizes the reconstruction of SCQVAE in comparison with VQVAE (van den Oord et al., 2017) and Gumbel-VAE (Jang et al., 2017; Maddison et al., 2017) on a subset of test images after 1, 10 and 20 training epochs. This visualiza-

Table 1: Comparison between methods on an image reconstruction task for CIFAR-10 and GTSRB over 5 independent training runs. All metrics are computed and averaged on the test set.

Method	MSE (10^{-3}) \downarrow	Quant Error \downarrow	Perplexity \uparrow	Avg Quant Time (ms) \downarrow
CIFAR-10	VQVAE	41.19	70.47	6.62
	VQVAE + Rep	5.49	4.13×10^{-3}	106.07
	VQVAE + Affine + OPT	16.92	25.34×10^{-3}	8.65
	VQVAE + Rep + Affine + OPT	5.41	4.81×10^{-3}	106.62
	Gumbel-VQVAE	44.5	23.29×10^{-3}	10.86
	RQVAE	4.87	44.98×10^{-3}	20.68
	SCQVAE	1.53	0.15×10^{-3}	124.11
Non-Quantized AE (Reference)				
GTSRB	VQVAE	39.30	70.16	8.89
	VQVAE + Rep	3.91	1.61×10^{-3}	75.51
	VQVAE + Affine + OPT	11.49	13.27×10^{-3}	5.94
	VQVAE + Rep + Affine + OPT	4.01	1.71×10^{-3}	72.76
	Gumbel-VQVAE	56.99	47.53×10^{-3}	4.51
	RQVAE	4.96	38.29×10^{-3}	10.41
	SCQVAE	3.21	0.24×10^{-3}	120.55
Non-Quantized AE (Reference)				

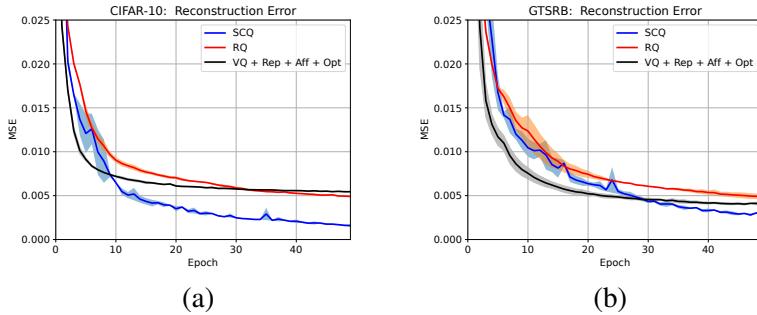


Figure 1: SCQVAE’s improved reconstruction convergence on CIFAR-10 (a) and GTSRB (b).

tion corroborates previous findings and further showcases the rapid minimization of reconstruction MSE with SCQVAE. A similar visualization for CIFAR-10 is given in [Gautam et al. \(2023\)](#).

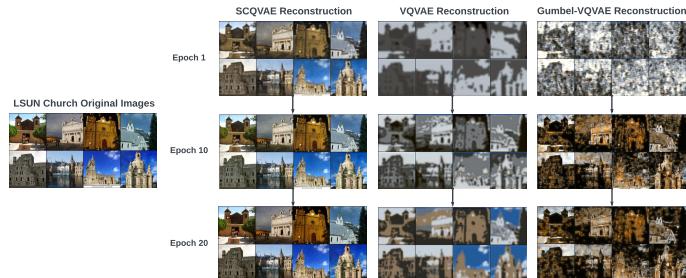


Figure 2: Comparison of LSUN ([Yu et al., 2015](#)) Church reconstruction on the test dataset.

4.2. Training VQGAN-Type Models

In this section, we focus on training first-stage models used within image synthesis methods such as unconditional latent diffusion models (LDM). We use the LSUN ([Yu et al., 2015](#)) Church and Classroom datasets and train VQGAN-type architectures trained with the associated VQGAN loss ([Esser et al., 2020](#)). We refer to the SCQ-embedded architectures as SCQGAN models.

The hyperparameter configurations used for both the VQ and SCQGAN architectures are summarized in [Gautam et al. \(2023\)](#). The performance of both architectures was measured on the test set with the VQGAN loss ([Esser et al., 2020](#)) referred to as $\mathcal{L}_{\text{VQGAN}}$, and LPIPS ([Zhang et al., 2018](#)). To examine the efficacy of both methods at different latent compression resolutions we train different architectures that compress the 256×256 images to 64×64 , 32×32 and 16×16 dimensional latent resolutions. Table 2 summarizes the results. SCQGAN outperforms VQGAN on both datasets across both metrics on all resolutions. This result highlights the efficacy of SCQ over VQ in preserving information during quantization - especially at smaller resolution latent spaces (greater compression). This result is particularly exciting for downstream generation tasks that leverage latent representations. More effective compression potentially eases the computational burden on the downstream tasks whilst maintaining performance levels. In [Gautam et al. \(2023\)](#), we include plots that further illustrate faster convergence for SCQGAN on both metrics across all resolutions.

Table 2: Comparison between SCQGAN and VQGAN on LSUN image reconstruction tasks. The same base architecture is used for all methods and metrics are computed on the test set.

Method	LSUN Church		LSUN Classroom	
	$\mathcal{L}_{\text{VQGAN}}(10^{-1}) \downarrow$	LPIPS ($10^{-1} \downarrow$)	$\mathcal{L}_{\text{VQGAN}}(10^{-1}) \downarrow$	LPIPS ($10^{-1} \downarrow$)
VQGAN (64-d Latents)	4.76	4.05	3.50	3.28
SCQGAN (64-d Latents)	3.93	3.88	3.29	3.23
VQGAN (32-d Latents)	6.60	6.22	8.01	7.62
SCQGAN (32-d Latents)	5.53	5.48	6.76	6.53
VQGAN (16-d Latents)	8.32	8.18	9.87	9.68
SCQGAN (16-d Latents)	7.86	7.84	9.19	9.15

5. Conclusion

This work proposes soft convex quantization (SCQ): a novel soft quantization method that can be used as a direct substitute for vector quantization (VQ). SCQ is introduced as a differentiable convex optimization (DCO) layer that quantizes inputs with a convex combination of codebook vectors. SCQ is formulated as a DCO and naturally inherits differentiability with respect to the entire quantization codebook. This enables overcoming issues such as inexact backpropagation and codebook collapse that plague the VQ method. SCQ is able to exactly represent inputs that lie within the convex hull of the codebook vectors, which mitigates lossy compression. Experimentally, we demonstrate that a scalable relaxation of SCQ facilitates improved learning of autoencoder models as compared to baseline VQ variants on CIFAR-10, GTSRB and LSUN datasets. SCQ gives up to an order of magnitude improvement in image reconstruction and codebook usage compared to VQ-based models on the considered datasets while retaining comparable quantization runtime. In future work, we aim to couple the improved SCQ autoencoder models with latent generative processes and investigate how SCQ can be used to enhance the performance of downstream applications.

References

- A. Agrawal, B. Amos, S. Barratt, S. Boyd, S. Diamond, and Z. Kolter. Differentiable convex optimization layers. In *Advances in Neural Information Processing Systems*, 2019.
- Brandon Amos and J. Zico Kolter. OptNet: Differentiable optimization as a layer in neural networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 136–145. PMLR, 2017.
- Brandon Amos, Lei Xu, and J. Zico Kolter. Input convex neural networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 146–155. PMLR, 2017.
- Yoshua Bengio, Nicholas Léonard, and Aaron C. Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *ArXiv*, abs/1308.3432, 2013.
- Xi Chen, Nikhil Mishra, Mostafa Rohaninejad, and Pieter Abbeel. Pixelsnail: An improved autoregressive generative model, 2017.
- Prafulla Dhariwal, Heewoo Jun, Christine Payne, Jong Wook Kim, Alec Radford, and Ilya Sutskever. Jukebox: A generative model for music, 2020.
- Patrick Esser, Robin Rombach, and Björn Ommer. Taming transformers for high-resolution image synthesis, 2020.
- Tanmay Gautam, Reid Pryzant, Ziyi Yang, Chenguang Zhu, and Somayeh Sojoudi. Soft convex quantization: Revisiting vector quantization with convex optimization. *Technical report*, 2023. URL <https://arxiv.org/pdf/2310.03004.pdf>.
- Shuyang Gu, Dong Chen, Jianmin Bao, Fang Wen, Bo Zhang, Dongdong Chen, Lu Yuan, and Baining Guo. Vector quantized diffusion model for text-to-image synthesis. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10686–10696, 2022. doi: 10.1109/CVPR52688.2022.01043.
- Minyoung Huh, Brian Cheung, Pulkit Agrawal, and Phillip Isola. Straightening out the straight-through estimator: Overcoming optimization challenges in vector quantized networks. In *International Conference on Machine Learning*. PMLR, 2023.
- Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. In *International Conference on Learning Representations*, 2017. URL <https://openreview.net/forum?id=rkE3y85ee>.
- Lukasz Kaiser, Samy Bengio, Aurko Roy, Ashish Vaswani, Niki Parmar, Jakob Uszkoreit, and Noam Shazeer. Fast decoding in sequence models using discrete latent variables. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 2390–2399. PMLR, 10–15 Jul 2018. URL <https://proceedings.mlr.press/v80/kaiser18a.html>.
- Diederik P. Kingma and Max Welling. An introduction to variational autoencoders. *Foundations and Trends® in Machine Learning*, 12(4):307–392, 2019. ISSN 1935-8237. doi: 10.1561/2200000056. URL <http://dx.doi.org/10.1561/2200000056>.

- Diederik P. Kingma, Danilo J. Rezende, Shakir Mohamed, and Max Welling. Semi-supervised learning with deep generative models. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, page 3581–3589, Cambridge, MA, USA, 2014. MIT Press.
- Alex Krizhevsky. Learning multiple layers of features from tiny images. 2009. URL <https://api.semanticscholar.org/CorpusID:18268744>.
- Doyup Lee, Chiheon Kim, Saehoon Kim, Minsu Cho, and Wook-Shin Han. Autoregressive image generation using residual quantization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11523–11532, 2022.
- Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. In *International Conference on Learning Representations*, 2017. URL <https://openreview.net/forum?id=S1jE5L5gl>.
- Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 8821–8831. PMLR, 18–24 Jul 2021. URL <https://proceedings.mlr.press/v139/ramesh21a.html>.
- Ali Razavi, Aäron van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with vq-vae-2. In *Neural Information Processing Systems*, 2019.
- Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models, 2021.
- David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 1986.
- J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural Networks*, 32:323–332, 2012. ISSN 0893-6080. doi: <https://doi.org/10.1016/j.neunet.2012.02.016>. URL <https://www.sciencedirect.com/science/article/pii/S0893608012000457>. Selected Papers from IJCNN 2011.
- Yuhta Takida, Takashi Shibuya, WeiHsiang Liao, Chieh-Hsin Lai, Junki Ohmura, Toshimitsu Uesaka, Naoki Murata, Shusuke Takahashi, Toshiyuki Kumakura, and Yuki Mitsufuji. SQ-VAE: Variational bayes on discrete representation with self-annealed stochastic quantization. In *International Conference on Machine Learning*, 2022.
- Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6309–6318, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.
- G.K. Wallace. The jpeg still picture compression standard. *IEEE Transactions on Consumer Electronics*, 38(1):xviii–xxxiv, 1992. doi: 10.1109/30.125072.

Penghang Yin, Jiancheng Lyu, Shuai Zhang, Stanley J. Osher, Yingyong Qi, and Jack Xin. Understanding straight-through estimator in training activation quantized neural nets. In *International Conference on Learning Representations*, 2019. URL <https://openreview.net/forum?id=Skh4jRcKQ>.

Fisher Yu, Yinda Zhang, Shuran Song, Ari Seff, and Jianxiong Xiao. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*, 2015.

Neil Zeghidour, Alejandro Luebs, Ahmed Omran, Jan Skoglund, and Marco Tagliasacchi. Sound-stream: An end-to-end neural audio codec. *IEEE/ACM Trans. Audio, Speech and Lang. Proc.*, 30:495–507, nov 2021. ISSN 2329-9290. doi: 10.1109/TASLP.2021.3129994. URL <https://doi.org/10.1109/TASLP.2021.3129994>.

Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 586–595, 2018. doi: 10.1109/CVPR.2018.00068.