Vali, Mohammadhassan; Bäckström, Tom

NSVQ: Noise Substitution in Vector Quantization for Machine Learning

Published in:
IEEE Access

DOI:
10.1109/ACCESS.2022.3147670

Published: 01/01/2022

Document Version
Publisher's PDF, also known as Version of record

IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received January 11, 2022, accepted January 24, 2022, date of publication January 28, 2022, date of current version February 7, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3147670

NSVQ: Noise Substitution in Vector Quantization for Machine Learning

MOHAMMAD HASSAN VALI AND TOM BÄCKSTRÖM, (Senior Member, IEEE)
Department of Signal Processing and Acoustics, Aalto University, 02150 Espoo, Finland
Corresponding author: Mohammad Hassan Vali (mohammad.vali@aalto.fi)

This work was supported by the Jane and Aatos Erkko Foundation under Contract 700795 AUTHSPKR.

ABSTRACT Machine learning algorithms have been shown to be highly effective in solving optimization problems in a wide range of applications. Such algorithms typically use gradient descent with backpropagation and the chain rule. Hence, the backpropagation fails if intermediate gradients are zero for some functions in the computational graph, because it causes the gradients to collapse when multiplying with zero. Vector quantization is one of those challenging functions for machine learning algorithms, since it is a piece-wise constant function and its gradient is zero almost everywhere. A typical solution is to apply the straight through estimator which simply copies the gradients over the vector quantization function in the backpropagation. Other solutions are based on smooth or stochastic approximation. This study proposes a vector quantization technique called NSVQ, which approximates the vector quantization behavior by substituting a multiplicative noise so that it can be used for machine learning problems. Specifically, the vector quantization error is replaced by product of the original error and a normalized noise vector, the samples of which are drawn from a zero-mean, unit-variance normal distribution. We test our proposed NSVQ in three scenarios with various types of applications. Based on the experiments, the proposed NSVQ achieves more accuracy and faster convergence in comparison to the straight through estimator, exponential moving averages, and the MiniBatchKmeans approaches.

INDEX TERMS Backpropagation, gradient collapse, gradient propagation, noise substitution, vector quantization.

I. INTRODUCTION

Machine learning is one of the most significant and potent technological advancements in recent years [1], [2]. It allows analyzing a massive volume of data and automatically captures intricate and obscure patterns within the data [1], [2]. Machine learning algorithms, especially those based on neural networks, have been shown to be highly efficient and successful in a wide range of real-world applications such as speech enhancement [3], speech recognition [4], [5], natural language processing [6], [7], and computer vision [8]–[11]. With this great potential entailed by these applications, we can expect that machine learning can be used to improve efficiency also in a wide range of future applications.

The learning processes in machine learning algorithms are typically based on propagating gradients in a backward direction. Hence, a mandatory prerequisite for these algorithms is that the mathematical relation between input parameters and objective function (the computational graph) should be smooth and differentiable. In other words, learning is not feasible if there are functions with zero (or undefined) gradient in the computational graph, since this would cause the gradients to collapse when multiplying them with zero (or *None*) based on the chain rule gradient calculation.

Vector quantization (VQ) is a data compression technique which models the probability density function of data by some representative vectors called codebooks [12]. Since VQ renders an abstract high-level discrete representation of the data distribution, it is widely used in a wide range of machine learning-based applications such as image compression [13], image generation [14], speech and audio coding [15]–[18], voice conversion [19], [20], music generation [21], and text-to-speech synthesis [22]–[24]. Despite this broad applicability, VQ is a challenging nonsmooth function for machine learning optimization, since its gradient is zero almost everywhere. In such cases, a standard solution for this challenge is to apply some assumptions or to approximate

The associate editor coordinating the review of this manuscript and approving it for publication was Wei Wang.

the behavior of the quantization function in the backpropagation. Solutions can be categorized, for example, by: 1) straight through estimator, 2) smooth approximation, and 3) stochastic approximation.

The straight through estimator (STE) [25] is a well-recognized approach which avoids the gradient collapse problem by simply copying the gradients over parts of the computational graph which cause this problem [13]–[17], [26]–[28]. However, it has been shown that STE does not consider the influence of quantization and leads to a mismatch between the gradient and true behavior of the quantization at low bitrates [29]. In addition, for the methods which use STE, it is essential to add an additional loss term to the global loss function to make the VQ codebooks to be updated [13], [14], [17], [27], [28]. Therefore, the weighting coefficient for the additional loss term is a new hyper-parameter, which is required to be tuned manually.

Another solution is smooth approximation, which aims to approximate smooth quantization behavior. Such soft quantizers perform soft clustering by adopting the softmax or softmin functions [18], [30]–[32]. In these methods each input is assigned to all quantization levels (codebooks) with a probability, which depends on the distance of the input to the quantization level. Soft quantizers incur additional computational load which is not reasonable in transmission applications. Further, soft quantizers are less accurate and biased on the encoder side, since they apply soft-quantization in training and hard-quantization in the main application [18], [30]–[33]. In addition, the overall performance for most soft quantizers is highly sensitive and dependent on the tuning of hyper-parameters such as annealing speed and temperature in [30] and [33], respectively. In a similar approach [29], the quantization function is approximated with a series of hyperbolic tangent functions which are joined together to model the stair-shape of the quantization operation. The hyper-parameters of this method have to be well-tuned, otherwise it might not work appropriately [29]. Another approach circumvents the gradient collapse by replacing the derivative of the rounding operation with a smooth approximation [34] in the backward pass.

A different solution is recognized as stochastic approximation that performs quantization stochastically or inserts noise (by adding or multiplying) to the parameters (neurons or weights) which are going to be quantized [25], [35]–[39]. For instance in [36], a uniform scalar quantization is applied on all elements of feature vectors in the bottleneck of an autoencoder to allow approximation of the quantization effect by adding noise with a uniform distribution. There are also some other distinct methods which are not related to one of the aforementioned categories such as [40], which estimates quantization gradients in the backward step by optimizing an auxiliary neural network called meta-quantizer. The performance of this method is also sensitive to the choice of hyper-parameters, especially the architecture of the meta-quantizer [40].

Most of the solutions mentioned above are related to scalar quantization, in which the error distribution can be approximated accurately by replacing it with uniformly distributed noise, scaled to match the quantization step size. However, the error distribution for vector quantization does not have a simple form nor does it follow a determined structure. Therefore, the quantization error estimation would be much more complicated in this case.

In this paper, we introduce a novel vector quantization method called noise substitution in vector quantization (NSVQ), in which the vector quantization error is simulated with the product of the original quantization error magnitude with a normalized noise vector, the components of which are drawn from a zero-mean, unit-variance normal distribution. This normalized multiplicative noise does not change the mean or variance of the simulated error and leads it to gain the shape of the original error distribution. By replacing the original vector quantizer with this additive simulated error in this way, not only the proposed NSVQ can pass the gradients in backpropagation and avoid gradient collapse problem, but it also can estimate more accurate gradients for codebooks rather than just simply copying the gradients over VQ module like STE. We also apply weighting to the input vectors so that the importance of the dimensions can be optimized. To validate the efficiency of our proposed NSVQ, we apply it to three different scenarios: 1) to model the spectral envelopes in a speech codec [18], 2) to model the discrete latent representation of a vector quantized variational autoencoder (VQ-VAE) [13], and 3) to model a selection of well-known difficult toy examples.

Our experiments show that, in the training of the vector quantizers, the proposed NSVQ renders higher accuracy and faster convergence than STE, exponential moving averages (EMA) ,[1] and MiniBatchKmeans (the built-in function in the scikit-learn library). Also, in contrast to STE and EMA, our proposed NSVQ behaves consistently with the increase in quantization bitrate. Moreover, NSVQ locates the final optimized codebooks more homogeneously inside the data distribution than MiniBatchKmeans, and contrary to STE, it is not sensitive to the initialization of the codebooks, if we initialize the location of codebooks outside of the data distribution. Furthermore, in contrast to soft quantizers and STE, the proposed NSVQ does not incur any additional hyper-parameters tuning, since it does not need any additional loss term to be added to the global loss function. Finally, our proposed NSVQ performs more deterministic than STE and EMA with smaller variance in the experimental results. We believe that the properties of better accuracy, consistency in behavior, and homogeneity in codebooks locations stem from the fact that the simulated error in NSVQ is able to gain the shape of the original VQ error distribution properly. In addition, the properties of fast convergence and insensitivity to codebooks initialization stem from the codebook replacement function discussed in III-C.

[1] See the appendix in [13].

II. RELATED WORK

To reduce the quantization bias in the backpropagation of machine learning-based optimizations, the gradient collapse problem has been studied in a moderate number of approaches. In an effort to resolve the challenge of propagating gradients in deep learning models with stochastic neurons and hard nonlinearities, four different solutions were investigated in [25]. The first solution is called *Noisy Rectifier*, in which a deterministic function is applied to the activations of the network, which have already been added by a zero-mean noise. In the second solution, *Stochastic Times Smooth*, stochastic neurons are generated by multiplying a binomial noise as a stochastic term with a smooth function of activation values. The third solution is introduced to estimate gradients for stochastic binary neurons by reformulating hard nonlinear function as a stochastic function, the probability of which is a continuous function of parameters to be learned. This approach is referred as a specific case of the reinforce algorithm [35]. The last solution is recognized as *straight through estimator* (STE), which has been shown to be a simple and effective method. The key concept of STE is to set the output gradients of a gradient-problematic function equal to the exact input gradients to that function, i.e. to copy gradients over part of the computational graph which causes the gradient collapse.

The STE solution is applied in some approaches when performing vector quantization (VQ) to the latent representation of autoencoders. A vector quantized variational autoencoder (VQ-VAE) [13] applies vector quantization to the bottleneck of an autoencoder and generates a discrete latent representation capturing abstract high-level features of the input data. This method yields high quality of the reconstructed data at the decoder side for various types of data such as speech, image and video. The VQ-VAE adopts STE [25] to avoid the gradient collapse of vector quantization whereby it copies gradients from the decoder input to the encoder output. The same trick is also employed in [15], which is a low bitrate speech codec based on VQ-VAE [13]. To make VQ-VAE [13] more suitable for speech coding tasks, some architectural modifications are suggested in [15] whereby it preserves the speaker identity and prosody of the utterance. *SoundStream* [16] is an end-to-end optimized low-to-medium bitrate audio codec, which operates in real-time. In *SoundStream*, the input signal passes through a fully convolutional encoder which maps it to a sequence of embeddings (an efficient representation of input signal). Subsequently, the embeddings are quantized using multistage vector quantization and afterwards decoded by a fully convolutional decoder. This approach [16] also employs STE to support gradient propagation in backpropagation for end-to-end training. A different approach [26] trains a quantized neural network, in which activations and weights are quantized with low precision. Two kinds of deterministic and stochastic binarization functions train a binarized neural network, in that both of them cannot propagate gradients in the backward path because of the gradient collapse obstacle. To resolve this problem, a slightly modified version of STE [25] is applied, which ignores gradients of the parameters when the corresponding activations or weights are large, and preserves them otherwise.

The gradient collapse is also addressed by soft clustering which performs a smooth approximation of quantization. An end-to-end optimization of an autoencoder is adopted in [30] which performs image and model compression in a unified way by minimizing a global loss function. The typical model architecture for image compression is an autoencoder, in which the feature representation in the bottleneck should be quantized and entropy coded into a bit stream. To map the bottleneck features to the codebook centers, the encoder uses a softmax function to make this operation smooth enough for differentiation. A parameter controls the ''hardness'' of the assignments, which is set by an initial value and later increased towards infinity to change the assignments gradually from soft to hard. This approach entails a considerable computational load and penalizes the quantization precision while performing soft assignments. Furthermore, tuning the speed of the annealing process (increasing the controlling parameter) is a highly sensitive task, since slow annealing leads to large weights for the network and fast annealing stops the learning process because of the vanishing gradients phenomenon. Another method [31] recasts the K-means clustering algorithm as a neural network optimization, which is called K-meansNet. To this end, the K-means clustering objective is reformulated in a way that it is dependent on a neural network weights. By optimizing these weights using regular machine learning optimizers such as stochastic gradient descent (SGD), the clustering operation can be performed and optimized. To assign data points to clusters, the softmax function is employed in the formulation of K-meansNet, which gives a probability to each cluster based on the distance of input data point to the cluster centers. In a similar way, vector quantization can be used to model the spectral envelopes in a speech codec [18] based on [41] by applying a softmin function. A scalar value is multiplied with the exponents of the softmin function in the numerator and denominator, which acts as a controlling parameter adjusting the ''hardness'' of the vector quantization operation. Similarly to other soft quantizers, the controlling parameter in [18] should also be chosen with specific considerations to make this speech codec works properly.

Quantization behavior is simulated with a smooth approximation in some other approaches as well. To perform network quantization, a smooth quantization function is applied in [29], which approximates the stair-shape of the standard quantization function by linking a series of hyperbolic tangent functions. These functions are gradually updated and capture the shape of standard quantization function during the training process. In another approach, an autoencoder is used for image compression [34], in which there are two gradient-problematic functions: one is the quantization performed by rounding network weights to the nearest integer, and the

other is discretizing the representation of bottleneck features. To solve the gradient collapse challenge for the first function, the derivative of rounding operation is approximated by a smooth function in the backpropagation process. To solve the problem for the latter function, an approximation of the function is suggested by integrating a differentiable probability density function and bounding its intervals to limit the upper bound of the number of bits for the entropy coder.

As another solution for the gradient collapse challenge, some methods apply a stochastic approximation. For example, an autoencoder [38] acts as a variable bitrate image compressor, which contains a module to binarize the bottleneck representation. To solve the gradient collapse problem for the discrete binarization function, in the forward propagation, a binarization function is defined whereby binarization is applied by adding randomized quantization noise to the features. In the backward step, the derivative is taken from the expectation of the binarization function [42], since this expected value equals the original feature value. *BinaryConnect* [39] is a binarized deep neural network, in which the forward and backward propagation steps are performed using binarized weights by applying a stochastic binarization function, but the original full-precision weights are preserved for the parameter update step at each learning iteration. The approach most similar to our proposed NSVQ is presented in [36], which reformulates the global loss function and makes it render nonzero gradients by replacing a uniform noise with the quantization error [35]. This method [36] investigates scalar quantization, whereas our proposed method considers the vector quantization case.

A distinct method [40] reduces the storage space and computational cost of neural network models by quantizing their weights. To this end, an auxiliary neural network is defined as a meta-quantizer whereby the gradient collapse obstacle is obviated and the entire network quantization can be performed in an end-to-end manner. The meta-quantizer network is incorporated in the middle of the chain rule to calculate the gradients of the network weights with respect to the total loss function. It generates loss-aware gradients, which bring a more accurate update for the network parameters in the quantization training phase. Three different architectures are suggested for the meta-quantizer based on fully connected layers and LSTM. The meta-quantizer network is removed for the inference phase.

III. PROPOSED METHOD

A. VECTOR QUANTIZATION

Vector quantization (VQ) is a technique to model the distribution of data with a compressed representation with a fixed number of bits. It performs efficiently for various data distributions, even without sufficient knowledge of the distribution in advance [43]. The VQ methodology defines a set of codebooks and scatters them throughout the data distribution such that the compressed distribution of the data is represented by the codebooks. In other words, each codebook is considered a representative of some data samples. After applying VQ, these data samples are shown by the codebook which is the closest one under a distance measure. Suppose $x \in \mathbb{R}^{D\times 1}$ is a vector from the data distribution and $c_k \in \mathbb{R}^{D\times 1}$ refers to the kth codebook vector $0 \leq k < N = 2^B$, where B is the number of bits for VQ. For each input data sample x, the index of the closest codebook vector is

$$k_{min} = \arg\min_k \; d(x, c_k), \tag{1}$$

where $d(.,.)$ indicates the metric for the distance calculation, like Euclidean distance. The input data sample is then quantized to the nearest codebook vector such that $\hat{x} = c_{k_{min}}$. The principal target of VQ is to find the codebooks which minimize the expected distance of all data samples to the codebooks. In mathematical terms, the objective function for VQ is thus

$$\mathbf{E}[\arg\min_k \; d(x_i, c_k)] \approx \frac{1}{M}\sum_{i=0}^{M-1} \arg\min_k \; d(x_i, c_k), \tag{2}$$

where $\mathbf{E}[.]$ is the expectation operation which is here approximated over M data samples x_i.

B. PROPOSED NSVQ

The main purpose of this study is to enable the use of vector quantization (VQ) in machine learning-based optimizations and to propagate gradients through the VQ model while taking its statistical effect into account. The main problem with VQ is that it is piece-wise constant and its gradient is zero. On the other hand, when using standard backpropagation optimization, gradients are evaluated with the chain rule, so that if any intermediate gradient is zero, then their product will be zero, disabling the optimization process.

A similar problem has already been solved with uniform scalar quantization. Specifically, suppose x is quantized to $\hat{x} = Q[x]$, such that the gradient of the quantization is zero $\frac{\partial}{\partial x}Q[x] \equiv 0$, and the backpropagation collapses. However, the effect of quantization can be simulated with additive noise [36], [37]. Observe that the quantization error $e = \hat{x} - x$ can be assumed to be uniformly distributed (when quantization accuracy is high). Then $\hat{x} = x + e$ and we can replace e with any noise source which has the same distribution, without changing the overall accuracy. In other words, we can replace e with uniformly distributed noise, scaled to match the quantization step size. Then the gradient is $\frac{\partial}{\partial x}(x + e) \equiv 1$ and the backpropagation can be applied without any obstacles.

For VQ we can therefore design a similar simulation of quantization, where the quantization error is replaced by noise of a similar magnitude. However, in difference to scalar quantization, here we must make sure that the codebook of the VQ model can be optimized simultaneously. In VQ, the distribution of the error signal $e = \hat{x} - x$ does not have a

simple form, but we can simulate it with a zero-mean normal distribution $\mathcal{N}(0, \sigma^2)$. Then, we still need to determine the standard deviation.

For the case that x is scalar, we can approximate $\sigma^2 \approx e^2 = |\hat{x} - x|^2$, where $\hat{x}$ is the chosen codebook entry. The simulated quantizer can then be defined as

$$\begin{aligned} \hat{x}' &:= x + \epsilon e = x + \epsilon \cdot (\hat{x} - x) \\ &= x(1-\epsilon) + \epsilon \arg\min_{c_k} |x - c_k|^2. \end{aligned} \tag{3}$$

where ϵ is a normally distributed, unit-variance and zero-mean random variable, and c_ks are the codebook entries.

To characterize the simulated quantizer, note that the error is

$$\begin{aligned} e' &:= \hat{x}' - x = \epsilon e = \epsilon\,(\hat{x} - x) \\ &= -\epsilon x + \epsilon \arg\min_{c_k} |x - c_k|^2. \end{aligned} \tag{4}$$

It is thus the product of ϵ and e. The variance of ϵ is unity, and therefore the simulated quantizer has exactly the same variance as the corresponding vector quantizer $E[e'^2] = E[e^2]$.

We must further verify that we have access to all the gradients we need. For that purpose, let us define a generic loss function which takes the simulated quantization as input $l(\hat{x}')$. This loss function must then admit nonzero gradients with respect to both the input x and the chosen codebook vector c_k such that[2]

$$\begin{aligned} \frac{\partial}{\partial x} l(\hat{x}') &= l'(\hat{x}') \frac{\partial}{\partial x} x' \\ &= l'(\hat{x}') \frac{\partial}{\partial x} \left[x(1-\epsilon) + \epsilon \arg\min_{c_k} |x - c_k|^2 \right] \\ &= l'(\hat{x}') \left[1 - \epsilon + \epsilon \frac{\partial}{\partial x} \arg\min_{c_k} |x - c_k|^2 \right] \\ &= l'(\hat{x}')\,[1 - \epsilon] \end{aligned} \tag{5}$$

and

$$\begin{aligned} \frac{\partial}{\partial c_k} l(\hat{x}') &= l'(\hat{x}') \frac{\partial}{\partial c_k} x' \\ &= l'(\hat{x}') \frac{\partial}{\partial c_k} \left[x(1-\epsilon) + \epsilon \arg\min_{c_k} |x - c_k|^2 \right] \\ &= l'(\hat{x}') \epsilon \frac{\partial}{\partial c_k} \arg\min_{c_k} |x - c_k|^2 \\ &= l'(\hat{x}') \epsilon \frac{\partial}{\partial c_k} c_k \\ &= l'(\hat{x}') \epsilon. \end{aligned} \tag{6}$$

In other words, in both cases we acquire a simple form for the gradient. Although these gradients can be zero sometimes, they are not always zero. To be specific, note that the gradient with respect to c_k can be nonzero only for that codebook with index k, which is the optimal entry for the input x.

[2]Note that the arg min expression can be replaced by the optimal c_k, so that the derivatives are $\frac{\partial}{\partial x} \arg\min_{c_k} |x - c_k|^2 = \frac{\partial}{\partial x} c_k \equiv 0$ and $\frac{\partial}{\partial c_k} \arg\min_{c_k} |x - c_k|^2 = \frac{\partial}{\partial c_k} c_k \equiv 1$.

To consider a similar simulation for the VQ case, suppose that the input vector is N-dimensional $x \in \mathbb{R}^{N\times 1}$. The quantized value is then typically defined by the minimum Euclidean norm $\hat{x} := \arg\min_{c_k} \|x - c_k\|^2$. However, we can also include weighting onto the norm such that some dimensions of x become more important than the others. So, the weighting can be implemented as

$$\hat{x} := \arg\min_{c_k} \|w \odot (x - c_k)\|^2, \tag{7}$$

where $w \in \mathbb{R}^{N\times 1}$ is the weighting vector and $\odot$ represents the component-wise (Hadamard) product.

To integrate the weighted VQ in a machine learning optimization framework, we then have to design a simulated quantizer with similar functionality. For a VQ without weighting, we can assume that the error is uncorrelated and has equal variance over all dimensions, that is, its distribution is $\mathcal{N}(0, \sigma^2 I)$. With weighted VQ, we essentially weight the error such that the diagonal covariance elements are $w_k^{-2}\sigma^2$, where w_k is the kth element of w. The simulated quantizer for the weighted VQ can then be written as

$$\hat{x}' := x + \sigma v \odot w^{-1}, \tag{8}$$

where $\sigma = \sqrt{\frac{1}{N} \|w \odot (x - c_k)\|^2}$ is the error magnitude (viz. standard deviation) of the weighted error and v is a vector the components of which are drawn from a zero-mean, unit-variance normal distribution.

The noise signal v in (8) is defined to follow the normal distribution with zero-mean and unit variance. Observe that this was a choice made without closer inspection. The intention is to model the quantization error, so v should follow the same distribution as the quantization error. The quantization error in turn has a complex structure which is difficult to specify accurately, but we can make some characterizations. In particular, if the codebook is dense in the space and the true distribution is locally uniform, then the optimal codebook would organize itself in a lattice-like structure. In the interior of the lattice, errors would then always be bounded. At the outside border (the surface) of the data, the errors, however, depend on a more accurate description of the data. In any case, we can therefore conclude that a bounded distribution of the error can be useful. If the Voronoi-cells were perfectly (hyper)spherical, then the error distribution would be (at high quantization accuracy) uniformly distributed inside the corresponding hypersphere.

Another issue in (8) is that the noise vector v is multiplied by the error magnitude σ, which itself can be treated as a random scalar. The simulated error term is therefore the product of two random entities, and thus the product will be a product distribution. So, the variance of product distribution will be the product of the variances, when the terms are zero-mean. This leads to an even more accurate solution; if we normalize v such that $v' := \frac{v}{\sqrt{\frac{1}{N}\|v\|^2}}$, then v' will only be a random angle, but always have normalized length. In other words, it has constant variance, such that the product variance

is equal to the variance of the original signal. We can thus define an improved VQ-simulator as

$$\hat{x}' := x + \sqrt{\frac{\|w \odot (x - c_k)\|^2}{\|v\|^2}} v \odot w^{-1} = x + \frac{\|w \odot (x - c_k)\|}{\|v\|} v \odot w^{-1}. \quad (9)$$

Here the term $\sqrt{\|w \odot (x - c_k)\|^2}$ is a scalar which scales v to match the weighted energy of the original error. The vector $\frac{v}{\|v\|}$ is a random direction and when multiplied by the inverse weighted w^{-1}, it gains the shape of the original error distribution. Without weighting, this would then simply correspond to a random rotation of the original error signal. However, with weighting, we move on the surface of an ellipsoid of the same size as the original error signal.

C. CODEBOOK REPLACEMENT

A significant challenge in training codebooks for VQ is *codebook collapse* [44], in which some of the codebook entries are no longer activated in the quantization process. This occurs mainly when optimizing the codebooks using machine learning optimization and when the data distribution has low correlation or high entropy. There exist several works to address this issue by applying a kind of codebook replacement [12], [16], [45], in which the main objective is to replace the codebook entries which do not contribute or make less contribution to the VQ model than more active entries.

In this work, we also resolve the *codebook collapse* problem using a codebook replacement technique, in which during a specified number of training batches, the codebook entries which are used less than a threshold percentage are discarded and replaced with new values. To elaborate, they are replaced with a randomly selected set of more active codebooks (which are used above the threshold percentage) added by a small-magnitude normal noise. The parameters for our codebook replacement are chosen based on the total number of training epochs and the number of batches within each epoch. In other words, the value for these parameters differs for different applications. However, the generic methodology is that we apply the codebook replacement function more frequently in the early stages of the training procedure and as the training goes ahead, we apply it less frequently. Further, the codebook replacement function is stopped in final stages of training process, in order not to introduce new codebooks which are almost the same as the current active ones.

IV. EXPERIMENTS

To assess and compare the performance of our proposed noise substitution in vector quantization (NSVQ) with other methods, we establish three different scenarios: 1) speech coding, 2) image compression with VQ-VAE and 3) classic toy examples. In the first scenario, we apply the proposed NSVQ for a speech coding application introduced in [18],

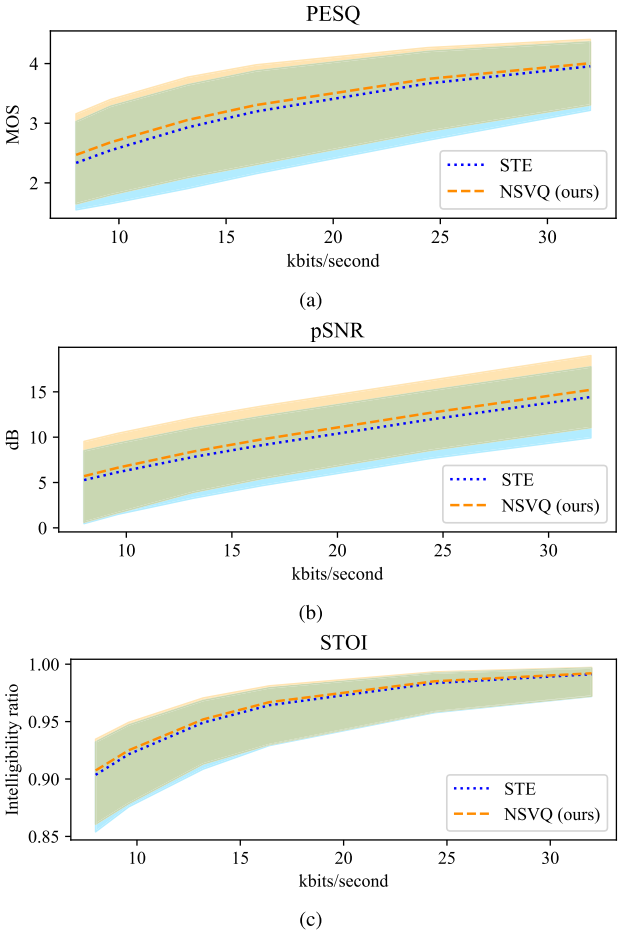

FIGURE 1. Performance of the proposed NSVQ and STE in terms of (a) PESQ, (b) pSNR, and (c) STOI metrics for 12 bit VQ at overall bitrates of 8, 9:6, 13:2, 16:4, 24:4 and 32 kbit/s in the speech coding scenario; dotted and dashed lines refer to the mean values of the STE and proposed NSVQ, respectively, and the corresponding filled areas refer to their 95% quantiles.

in which the spectral envelopes are modeled by a VQ optimized in an end-to-end machine learning framework. The speech codec is trained and tested using 100 h of clean English speech from the LibriSpeech corpus [46] over 5 epochs of training. The experiments are conducted at different bitrates for VQ using the Adam optimizer with a learning rate of 10^{-3}. Finally, we compare the proposed NSVQ and straight through estimator (STE) technique in the same speech coding approach [18] with the same hyper-parameters.

In the second scenario, we use our proposed NSVQ to vector quantize the latent representation of VQ-VAE proposed in [13] for an image compression task. The VQ-VAE acts as a generative model, which renders a latent representation containing abstract high-level representation for various types of data. First the input data passes through the encoder network and afterwards it is vector quantized. Then, these vector quantized variables are decoded by the decoder network. The architecture of the encoder

TABLE 1. Mean and standard deviation of SSIM and Peak SNR values for reconstructed images from the CIFAR10 test set over five individual experiments when applying STE, EMA, and the proposed NSVQ with and without codebook replacement in the image compression scenario (w/o CBR refers to the proposed NSVQ without codebook replacement).

Iterations	Metric	Method	6 bit		7 bit		8 bit		9 bit		10 bit		11 bit	
			Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
5 k	SSIM	STE	0.7397	0.0026	0.7469	0.0072	0.7575	0.0095	0.7634	0.0109	0.7332	0.0348	0.7501	0.0058
		EMA	0.7597	0.0062	0.7454	0.0751	0.7666	0.0380	0.7844	0.0188	0.8083	0.0080	0.7883	0.0257
		NSVQ w/o CBR	0.7619	0.0067	0.7758	0.0063	0.7827	0.0091	0.7916	0.0100	0.7984	0.0098	0.7915	0.0211
		NSVQ (ours)	**0.7680**	**0.0023**	**0.7887**	**0.0023**	**0.8125**	**0.0024**	**0.8329**	**0.0016**	**0.8456**	**0.0034**	**0.8574**	**0.0012**
	Peak SNR	STE	22.3165	0.0975	22.4653	**0.1400**	22.5506	0.3230	22.6390	0.1793	22.2317	0.4157	22.4408	0.2686
		EMA	22.6203	0.1622	22.3902	0.9725	22.7453	0.6605	22.9095	0.3848	23.3771	0.1914	23.0958	0.5225
		NSVQ w/o CBR	22.6992	0.1759	23.0059	0.1734	23.2329	0.2167	23.4041	0.2626	23.6081	0.2668	23.3578	0.5033
		NSVQ (ours)	**22.9018**	**0.0907**	**23.4100**	0.1438	**23.8662**	**0.0594**	**24.3821**	**0.0850**	**24.6717**	**0.1561**	**24.9993**	**0.0709**
10 k	SSIM	STE	0.7697	**0.0031**	0.7866	0.0102	0.7935	0.0098	0.7945	0.0067	0.7942	0.0092	0.7913	0.0093
		EMA	**0.7728**	0.0057	**0.7981**	0.0067	0.8119	0.0112	0.8246	0.0098	0.7839	0.0320	0.8046	0.0182
		NSVQ w/o CBR	0.7658	0.0061	0.7851	0.0098	0.7988	0.0059	0.8056	0.0106	0.8200	0.0036	0.8098	0.0149
		NSVQ (ours)	0.7712	0.0046	0.7971	**0.0027**	**0.8201**	**0.0014**	**0.8377**	**0.0021**	**0.8516**	**0.0016**	**0.8638**	**0.0012**
	Peak SNR	STE	**22.9390**	0.1156	23.2533	0.2059	23.2920	0.1750	23.3583	0.1888	23.3429	0.1652	23.3281	0.2351
		EMA	22.9189	0.2505	23.4358	0.2456	23.5953	0.3195	23.5989	0.3690	22.9720	0.6934	23.3732	0.3285
		NSVQ w/o CBR	22.7998	0.1426	23.2889	0.2248	23.6041	0.1549	23.7214	0.3162	24.0597	0.1365	23.8071	0.3447
		NSVQ (ours)	22.8619	**0.0760**	**23.5967**	**0.0599**	**24.0812**	**0.0762**	**24.5294**	**0.1401**	**24.8967**	**0.0949**	**25.2335**	**0.0891**
15 k	SSIM	STE	**0.7779**	0.0024	0.7981	0.0029	0.8100	0.0071	0.8121	0.0039	0.8139	0.0098	0.8116	0.0049
		EMA	0.7751	0.0070	0.7989	0.0100	0.8202	0.0054	0.8366	0.0040	0.8370	0.0093	0.8452	0.0047
		NSVQ w/o CBR	0.7693	0.0026	0.7829	0.0134	0.7943	0.0048	0.8157	0.0073	0.8158	0.0082	0.8258	0.0140
		NSVQ (ours)	0.7749	**0.0023**	**0.8012**	**0.0022**	**0.8209**	**0.0014**	**0.8408**	**0.0016**	**0.8539**	**0.0014**	**0.8647**	**0.0009**
	Peak SNR	STE	**23.1646**	**0.0433**	23.4630	0.0977	23.5986	0.0923	23.6462	0.1011	23.7396	0.2687	23.7535	0.1237
		EMA	22.8838	0.2454	23.4143	0.2700	23.7823	0.2171	24.1141	0.2348	24.2050	0.2309	24.2770	0.1274
		NSVQ w/o CBR	22.8540	0.0627	23.2028	0.2653	23.3853	0.1224	23.9526	0.1438	23.9067	0.2818	24.1894	0.3623
		NSVQ (ours)	22.9763	0.1117	**23.6077**	**0.0643**	**24.1653**	**0.0493**	**24.6260**	**0.0894**	**24.9842**	**0.0449**	**25.2403**	**0.0476**

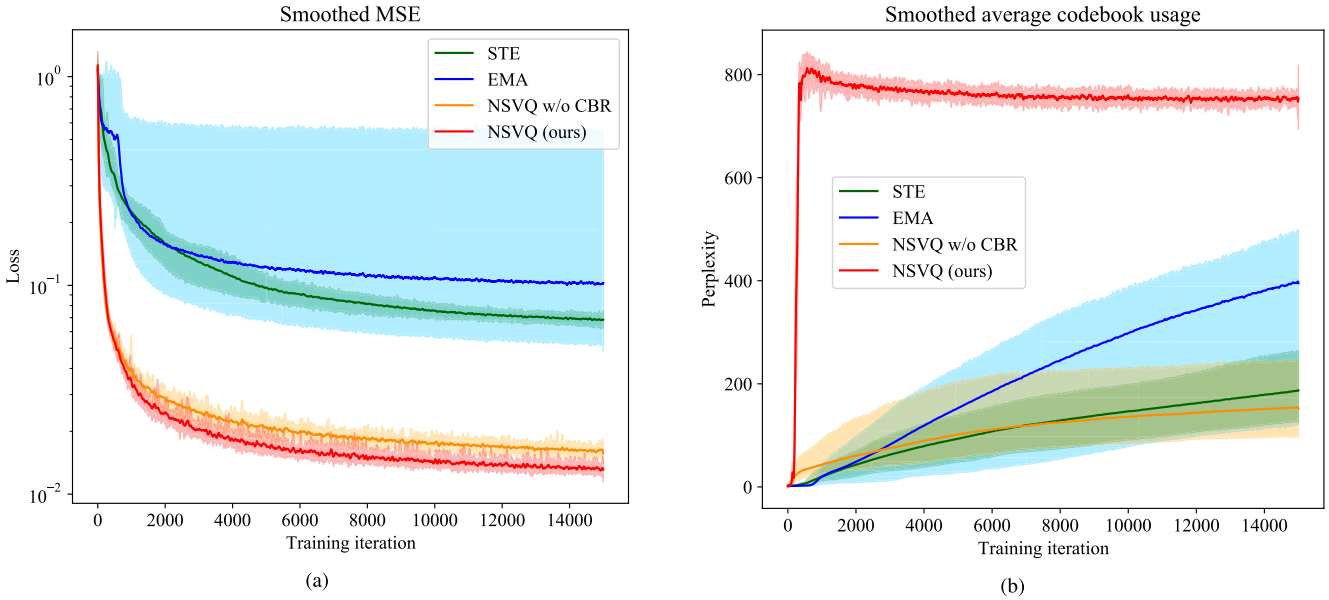

FIGURE 2. **(a) Smoothed training error (MSE) and (b) smoothed average codebook usage (perplexity) of STE, EMA, and the proposed NSVQ with and without codebook replacement for 11 bit VQ after 15 k training updates and over 20 individual experiments in the image compression scenario (w/o CBR refers to the proposed NSVQ without codebook replacement); Solid lines refer to the mean values of the methods, and the corresponding filled areas refer to their 95% quantiles.**

and the decoder is based on *ResNet*, the same way as proposed in the original paper [13], where the encoder comprises two convolutional layers followed by two residual blocks. Similarly to the encoder, the decoder comprises two identical residual blocks followed by two deconvolutional layers.

The VQ-VAE presented in [13] adopts STE to backpropagate gradients for VQ and defines uniform distribution as prior and keeps it constant during the training phase. We train this VQ-VAE using CIFAR10 dataset and set most hyperparameters the same as what is mentioned in the original paper [13]. The coefficient for commitment loss is $\beta = 0.25$, the dimensionality of each latent codebook vector is $D = 64$, and we use the Adam optimizer with a learning rate of 10^{-3}. The main configuration difference to the original paper was the batch size, which is selected as 32 to allow faster convergence [47] for the VQ-VAE. We compare the performance of our proposed NSVQ and STE technique by training the VQ-VAE with different bitrates K (number of codebook vectors for latent representation) and a different number of training updates. In addition to the STE, we also compare the proposed NSVQ with a different method, which adopts exponential moving averages (EMA) to update the VQ codebooks instead of employing an auxiliary loss for the backpropagation. The EMA update is inherited from the K-means algorithm (see the appendix of [13] for more details). We implement the EMA with a decay coefficient of 0.99 ($\gamma = 0.99$), equal to that presented in the original paper [13].

In the third and last scenario, we aim to compress the distribution of four strange-shaped machine learning datasets, including *blobs*, *circles*, *moons*, and *swiss-roll*, using the codebooks learned by the proposed NSVQ, the STE technique, and MiniBatchKmeans (the built-in function in the scikit-learn library). For each data distribution, the initial codebooks are captured from the corresponding data samples added by a normal noise with zero mean and unit variance. We add noise to the initial codebook vectors to more deeply investigate and assess the efficiency of each individual method. For a fair comparison, in the case of each data distribution, the generated data distribution and the corresponding initialization points for the codebooks remain the same for each of these three individual methods. All of the training phases are executed using the Adam optimizer with a learning rate of 10^{-3} over 100 epochs for different bitrates K (number of codebook entries), where the data dimensionality is set at 2 ($D = 2$) for visualization purposes. The generated data distributions contain 10^6 samples, and the experiments are performed with a batch size of 10^4 samples. For more details on the implementation of MiniBatchKmeans, we set *n_init* = 1 to ensure the same initialization with other methods, and *max_no_improvement* = *None* to disable convergence detection and let MiniBatchKmeans to be optimized for the entire 100 epochs.

We provide our NSVQ implementation code in a public webpage for reproducibility purposes.[3] For all three above-mentioned scenarios, we employ the PyTorch machine learning library for optimization and the codebook replacement method described in section III-C for the proposed

[3]https://gitlab.com/speech-interaction-technology-aalto-university/nsvq

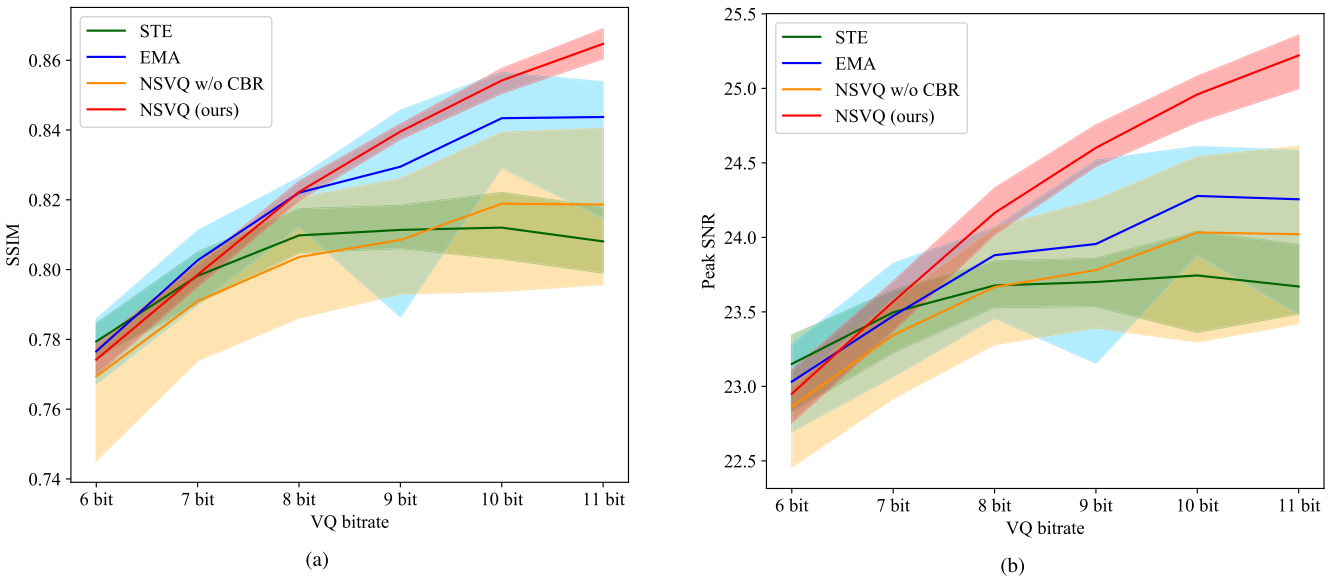

FIGURE 3. (a) SSIM and (b) Peak SNR values of STE, EMA, and the proposed NSVQ with and without codebook replacement for different VQ bitrates after 15 k training updates and over 20 individual experiments in the image compression scenario (w/o CBR refers to the proposed NSVQ without codebook replacement); Solid lines refer to the mean values of the methods, and the corresponding filled areas refer to their 95% quantiles.

NSVQ. In the codebook replacement, we chose the threshold percentage for discarding unused codebooks as 1 % for speech coding, and 10 % for image compression and toy examples scenarios. We chose the values of the discarding thresholds by trial and error, and selected the values which give the best results. The choice of discarding threshold mainly depends on the application and its adopted metrics. For more details, changing this threshold from 10 % to 1 % would make a really slight difference in the final results for both image compression and toy examples scenarios, whereas it would lead to a bit greater difference in the performance of the proposed NSVQ for speech coding scenario. In general, the choice of discarding threshold is not a crucial matter with high importance, since the proposed NSVQ would work properly if we choose the threshold in a reasonable way.

V. RESULTS AND DISCUSSION

As explained in section IV, we analyzed the performance of our proposed noise substitution in vector quantization (NSVQ) and straight through estimator (STE) for three different scenarios. In the speech coding scenario, we apply the perceptual evaluation of speech quality (PESQ) [48], perceptually weighted signal to noise ratio (pSNR) and short-time objective intelligibility (STOI) [49] as the objective metrics to evaluate the quality of the encoded speech signal. The evaluation is carried out at different overall bitrates of 8, 9.6, 13.2, 16.4, 24.4 and 32 kbit/s, matching the operating modes of 3GPP EVS codec [50]. The speech codec presented in [18] employs multistage VQ to allow quantization for higher ranges of bitrates. According to the implemented experiments at different VQ bitrates, the proposed NSVQ and STE performs almost similarly at high bitrates (when performing multistage VQ). Hence, we provide the results for 12 bit VQ in Fig. 1, which can be applied only in one stage of VQ. According to the figure, the proposed NSVQ performs better than STE in terms of PESQ, while obtaining higher PESQ values than STE in 86% of the cases. In terms of pSNR, the proposed NSVQ operates quite clearly better than STE with higher pSNR values in 84% of the cases. From another point of view, our proposed NSVQ achieves on average from 0.43 to 0.79 higher pSNR values in decibels in comparison to STE, when increasing the bitrate from 8 to 32 kbit/s. Regarding the STOI metric, both methods perform comparatively to each other, since the mean values and their 95% quantiles are approximately overlapping. For further details, the proposed NSVQ achieves slightly higher STOI values than STE in 76% of the cases.

In the second scenario, we compare our proposed NSVQ with STE and exponential moving averages (EMA) for image compression application, while performing VQ on the latent representation of the VQ-VAE [13]. The main objective is to train VQ codebooks for discretizing the latent representation of the autoencoder using STE, EMA, and our proposed NSVQ. In the evaluation phase, we reconstruct the images from the learnt VQ codebooks using the trained encoder and decoder.

We train all three approaches for 5, 10 and 15 k training updates using the CIFAR10 dataset. In the evaluation phase, the structural similarity index measure (SSIM) and peak signal to noise ratio (Peak SNR) are employed as objective metrics to evaluate the quality of the reconstructed images from VQ codebooks which are obtained from each of the individual approaches. The experimental results are shown in Table 1. Note that the mean and standard deviation of SSIM and Peak SNR metrics provided in this table refer to their

TABLE 2. Mean and standard deviation of MSE values for vector quantized data over five individual experiments when applying STE, MiniBatchKmeans, and the proposed NSVQ in the toy examples scenario.

Bitrate	Method		blobs	circles	moons	swiss-roll
6 bit	MiniBatchKmeans	Mean	88.35 e-3	101.88 e-3	**57.52 e-3**	11.02 e-2
		Std	**18.88 e-4**	**8.27 e-3**	**29.80 e-4**	87.92 e-4
	STE	Mean	86.07 e-3	193.58 e-3	117.36 e-3	18.23 e-2
		Std	25.82 e-4	32.75 e-3	148.75 e-4	265.58 e-4
	NSVQ (ours)	Mean	**85.01 e-3**	**94.59 e-3**	61.75 e-3	**10.38 e-2**
		Std	19.69 e-4	10.89 e-3	31.25 e-4	**79.45 e-4**
8 bit	MiniBatchKmeans	Mean	23.80 e-3	**12.23 e-3**	83.90 e-4	12.22 e-3
		Std	20.14 e-5	**63.41 e-5**	7.19 e-5	**22.29 e-5**
	STE	Mean	22.80 e-3	37.17 e-3	227.64 e-4	38.02 e-3
		Std	36.67 e-5	631.63 e-5	58.29 e-5	411.05 e-5
	NSVQ (ours)	Mean	**22.36 e-3**	12.54 e-3	**79.13 e-4**	**11.66 e-3**
		Std	**14.35 e-5**	105.38 e-5	**6.50 e-5**	80.45 e-5
10 bit	MiniBatchKmeans	Mean	60.92 e-4	32.47 e-4	24.22 e-4	33.17 e-4
		Std	46.61 e-6	**3.89 e-5**	2.22 e-5	**3.41 e-5**
	STE	Mean	61.91 e-4	83.79 e-4	69.83 e-4	87.87 e-4
		Std	28.10 e-6	21.46 e-5	14.84 e-5	5.49 e-5
	NSVQ (ours)	Mean	**57.17 e-4**	**31.47 e-4**	**23.27 e-4**	**32.07 e-4**
		Std	**8.82 e-6**	9.58 e-5	**1.76 e-5**	7.49 e-5
12 bit	MiniBatchKmeans	Mean	15.56 e-4	85.98 e-5	63.43 e-5	88.21 e-5
		Std	13.99 e-6	**6.63 e-6**	**1.73 e-6**	**6.44 e-6**
	STE	Mean	19.05 e-4	283.98 e-5	227.59 e-5	300.39 e-5
		Std	13.75 e-6	67.74 e-6	36.34 e-6	14.99 e-6
	NSVQ (ours)	Mean	**14.74 e-4**	**85.69 e-5**	**61.48 e-5**	**87.02 e-5**
		Std	**4.19 e-6**	17.47 e-6	5.67 e-6	14.28 e-6

mean and standard deviation calculated over the entire test set of CIFAR10 and over five individual experiments. With regard to the mean SSIM and Peak SNR values in Table 1, the proposed NSVQ (particularly with codebook replacement) performs better than STE and EMA in most cases, especially when training the models for a smaller number of training updates, e.g. 5 and 10 k cases. In other words, our proposed NSVQ converges faster than the STE and EMA methods. The main reason is the codebook replacement function we adopt in the proposed NSVQ, which acts like a trigger in the early steps of the training process and provokes the codebooks to be updated faster.[4] This behavior is shown in Fig. 2. According to this figure, as a consequence of the codebook replacement, the average number of used codebooks (perplexity) in the proposed NSVQ suddenly increases in the early stages of training, which results in a dramatic drop in the training loss value. Additionally, according to Fig. 3 and Table 1, when performing VQ with lower bitrates, the proposed NSVQ performs comparatively to STE and EMA in terms of SSIM and Peak SNR values, but it shows more distinctive results when increasing the VQ bitrate. This is another benefit of the codebook replacement, since it allows the proposed NSVQ to exploit the potential of having more active codebooks at higher bitrates, while replacing less used codebooks with the most significant ones.

To investigate Table 1 and Fig. 3 from another viewpoint, the proposed NSVQ (with codebook replacement) obtains strictly ascending SSIM and Peak SNR values when increasing the VQ bitrate, which confirms that it behaves consistently with the increase in bitrate. However, the STE and EMA methods do not follow the same behavior. Regarding the results in Table 1 and Fig. 3, although the proposed NSVQ (without codebook replacement) behaves more or less consistently with the increase in VQ bitrate, we cannot expect the same behavior for another set of experiments. Because for each experiment, the proposed NSVQ (without codebook replacement) might end up with a different amount of perplexity according to its high perplexity variance in Fig. 2. So, the more perplexity it reaches, the better SSIM and Peak SNR values it achieves. Due to this variance in the performance of various approaches, we plotted the loss, perplexity, SSIM, and Peak SNR values for all approaches in Fig. 2 and Fig. 3 over an average of 20 individual experiments to investigate the variance in their performance. Therefore, with regard to Table 1 and Fig. 3, not only the proposed NSVQ (with codebook replacement) gains higher mean SSIM and Peak SNR values than other approaches, but it also shows less variance in its performance, which statistically confirms its superiority.

[4]Note that for the proposed NSVQ without codebook replacement, it is necessary to apply codebook replacement in the very first step of training only once to allow codebooks to be updated.

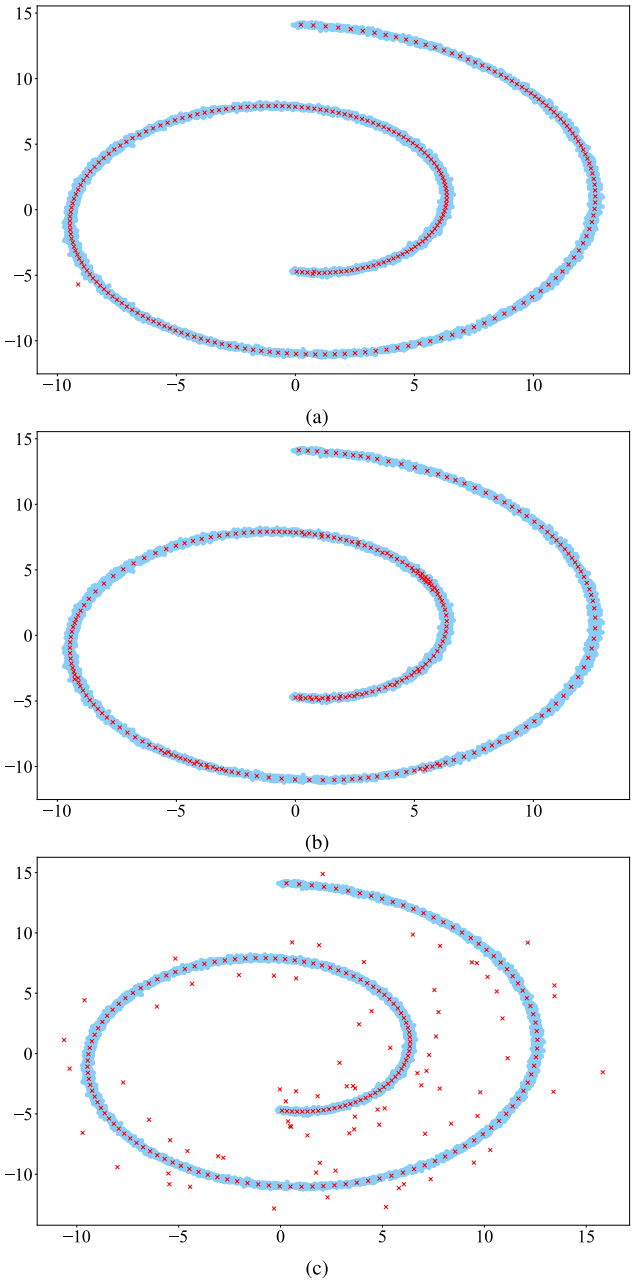

FIGURE 4. Final codebooks for (a) the proposed NSVQ, (b) MiniBatchKmeans, and (c) STE in case of 8 bit VQ for the swiss-roll distribution in the toy examples scenario.

In the third scenario, we quantize the data distributions of *blobs*, *circles*, *moons*, and *swiss-roll* using the proposed NSVQ, STE, and MiniBatchKmeans. To evaluate the performance of the quantization operation, we calculate the mean and standard deviation of mean squared error (MSE) metric between original data distribution and its quantized version over five individual experiments. The related results are shown in Table 2.[5] According to the table, the proposed NSVQ performs better than STE and MiniBatchKmeans in almost all cases regarding the mean of MSE values. With regard to standard deviation of MSE values, our proposed NSVQ performs more deterministic than STE, since it has less variance than STE in all cases. However, when comparing with MiniBatchKmeans, the proposed NSVQ obtains slightly higher variance in some of the experiments. Since the differences for MSE values are not too high for various methods, we plotted the final optimized codebooks found by each of these three methods to attain a better understanding of their performance. Fig. 4 shows the optimized codebooks found by each of these methods for the *swiss-roll* distribution in the case of 8 bit VQ. We chose 8 bit VQ for visualization, since at higher bitrates the figure becomes visually too dense. According to Fig. 4, the proposed NSVQ found the codebooks in a more uniform and homogeneous way in comparison to the MiniBatchKmeans method. Furthermore, the STE method ends up with numerous dead codebooks,[6] whereas the proposed NSVQ has only one dead codebook. Based on our experiments, the proposed NSVQ would not have any dead codebooks when performing VQ with higher bitrates than 8 bit. In contrast to the proposed NSVQ, the STE method would lead to a larger number of dead codebooks when performing VQ with higher bitrates. In other words, the STE method is highly sensitive to initialization and performs poorly when initial codebooks are located out of the data distribution. The same behavior has been shown in our experiments for other similar data distributions with low correlation including *circles* and *moons* datasets. On the other hand, for the distributions with high correlation such as *blobs*, all three methods perform similarly in terms of MSE values without any dead codebooks.

VI. CONCLUSION

Vector quantization (VQ) is a data compression technique which is widely used in many machine learning-based applications, especially in vector quantized variational autoencoders. However, VQ cannot be used as such during training of machine learning models, since its gradients are uniformly zero, which collapses backpropagation. In this paper, we propose a novel method to simulate the VQ behavior by noise substitution so that it can be employed in machine learning optimizations. We evaluate our proposed noise substitution in vector quantization (NSVQ) in three different applications with various types of input data. The experiments demonstrate that the proposed NSVQ compresses the input data with better accuracy, faster convergence, smaller variance in performance, and less sensitivity to the codebooks initialization in comparison to straight through estimator (STE) and exponential moving averages (EMA). Furthermore, contrary to STE and EMA methods, the proposed NSVQ behaves consistently with the increase in VQ bitrate, which is expected from a genuine VQ.

[5]Note that since the data distributions cover different ranges of values in two dimensional space, the scale of MSE values varies for different data distributions.

[6]Dead codebook refers to the codebook which is selected out of the data distribution and would not be used for the decoding phase.

Since scikit-learn library does not support GPU execution, the proposed NSVQ provides an alternative for the conventional K-means algorithm with faster execution time and higher accuracy. The proposed NSVQ provides a non-zero gradient only for the best codebook entry (which has the minimum distance to the input vector) in one training batch. As a future work, we consider defining gradients for the entire codebook entries or a subset of them to yield a more efficient VQ which might also converge faster than the current proposed NSVQ. In addition, as another future work it would be worthwhile to investigate the behavior of the proposed NSVQ by adopting other noise distributions instead of normal distribution.

REFERENCES

[1] I. H. Sarker, "Machine learning: Algorithms, real-world applications and research directions," *Social Netw. Comput. Sci.*, vol. 2, no. 3, pp. 1–21, May 2021.

[2] J. Schmidt, M. R. G. Marques, S. Botti, and M. A. L. Marques, "Recent advances and applications of machine learning in solid-state materials science," *NPJ Comput. Mater.*, vol. 5, no. 1, pp. 1–36, Dec. 2019.

[3] S. Pascual, A. Bonafonte, and J. Serrà, "SEGAN: Speech enhancement generative adversarial network," 2017, *arXiv:1703.09452*.

[4] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-R. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Process. Mag.*, vol. 29, no. 6, pp. 82–97, Oct. 2012.

[5] T. N. Sainath, A.-R. Mohamed, B. Kingsbury, and B. Ramabhadran, "Deep convolutional neural networks for LVCSR," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, May 2013, pp. 8614–8618.

[6] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Proc. 27th Int. Conf. Neural Inf. Process. Syst.*, vol. 2, 2014, pp. 3104–3112.

[7] D. Bahdanau, K. H. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *Proc. 3rd Int. Conf. Learn. Represent.*, 2015, pp. 1–15.

[8] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 1–9.

[9] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. 25th Int. Conf. Neural Inf. Process. Syst.*, vol. 1, 2012, pp. 1097–1105.

[10] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[11] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 4700–4708.

[12] A. Gersho and R. M. Gray, *Vectora Quantization and Signal Compression*, vol. 159. Springer, 2012.

[13] A. van den Oord, O. Vinyals, and K. Kavukcuoglu, "Neural discrete representation learning," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 6309–6318.

[14] A. Razavi, A. van den Oord, and O. Vinyals, "Generating diverse high-fidelity images with VQ-VAE-2," in *Proc. 33th Int. Conf. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 14866–14876.

[15] C. Garbacea, A. V. den Oord, Y. Li, F. S. C. Lim, A. Luebs, O. Vinyals, and T. C. Walters, "Low bit-rate speech coding with VQ-VAE and a WaveNet decoder," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2019, pp. 735–739.

[16] N. Zeghidour, A. Luebs, A. Omran, J. Skoglund, and M. Tagliasacchi, "SoundStream: An end-to-end neural audio codec," 2021, *arXiv:2107.03312*.

[17] Y. Chen, S. Yang, N. Hu, L. Xie, and D. Su, "TeNC: Low bit-rate speech coding with VQ-VAE and GAN," in *Proc. Int. Conf. Multimodal Interact.*, Oct. 2021, pp. 126–130.

[18] M. H. Vali and T. Bäckström, "End-to-end optimized multi-stage vector quantization of spectral envelopes for speech and audio coding," in *Proc. Interspeech*, Aug. 2021, pp. 3355–3359.

[19] S. Ding and R. Gutierrez-Osuna, "Group latent embedding for vector quantized variational autoencoder in non-parallel voice conversion," in *Proc. Interspeech*, Sep. 2019, pp. 724–728.

[20] D.-Y. Wu, Y.-H. Chen, and H.-Y. Lee, "VQVC+: One-shot voice conversion by vector quantization and U-Net architecture," 2020, *arXiv:2006.04154*.

[21] P. Dhariwal, H. Jun, C. Payne, J. W. Kim, A. Radford, and I. Sutskever, "Jukebox: A generative model for music," 2020, *arXiv:2005.00341*.

[22] G. E. Henter, J. Lorenzo-Trueba, X. Wang, and J. Yamagishi, "Deep encoder-decoder models for unsupervised learning of controllable speech synthesis," 2018, *arXiv:1807.11470*.

[23] A. Tjandra, B. Sisman, M. Zhang, S. Sakti, H. Li, and S. Nakamura, "VQVAE unsupervised unit discovery and multi-scale Code2Spec inverter for zerospeech challenge 2019," 2019, *arXiv:1905.11449*.

[24] X. Wang, S. Takaki, J. Yamagishi, S. King, and K. Tokuda, "A vector quantized variational autoencoder (VQ-VAE) autoregressive neural F_0 model for statistical parametric speech synthesis," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 28, pp. 157–170, 2020.

[25] Y. Bengio, N. Léonard, and A. Courville, "Estimating or propagating gradients through stochastic neurons for conditional computation," 2013, *arXiv:1308.3432*.

[26] I. Hubara, M. Courbariaux, and D. Soudry, "Quantized neural networks: Training neural networks with low precision weights and activations," *J. Mach. Learn. Res.*, vol. 18, pp. 1–30, Jan. 2018.

[27] J. Chorowski, R. J. Weiss, S. Bengio, and A. van den Oord, "Unsupervised speech representation learning using WaveNet autoencoders," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 27, no. 12, pp. 2041–2053, Dec. 2019.

[28] Y. Zhao, H. Li, C.-I. Lai, J. Williams, E. Cooper, and J. Yamagishi, "Improved prosody from learned F0 codebook representations for VQ-VAE speech waveform reconstruction," 2020, *arXiv:2005.07884*.

[29] R. Gong, X. Liu, S. Jiang, T. Li, P. Hu, J. Lin, F. Yu, and J. Yan, "Differentiable soft quantization: Bridging full-precision and low-bit neural networks," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 4852–4861.

[30] E. Agustsson, F. Mentzer, M. Tschannen, L. Cavigelli, R. Timofte, L. Benini, and L. Van Gool, "Soft-to-hard vector quantization for end-to-end learning compressible representations," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 1141–1151.

[31] X. Peng, I. W. Tsang, J. T. Zhou, and H. Zhu, "k-meansNet: When k-means meets differentiable programming," 2018, *arXiv:1808.07292*.

[32] T. Yu, J. Yuan, C. Fang, and H. Jin, "Product quantization network for fast image retrieval," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2018, pp. 191–206.

[33] J. Yang, X. Shen, J. Xing, X. Tian, H. Li, B. Deng, J. Huang, and X.-S. Hua, "Quantization networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2019, pp. 7308–7316.

[34] L. Theis, W. Shi, A. Cunningham, and F. Huszár, "Lossy image compression with compressive autoencoders," 2017, *arXiv:1703.00395*.

[35] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Mach. Learn.*, vol. 8, nos. 3–4, pp. 229–256, 1992.

[36] J. Ballé, V. Laparra, and E. P. Simoncelli, "End-to-end optimized image compression," in *Proc. 5th Int. Conf. Learn. Represent.*, 2017, pp. 1–27.

[37] T. Bäckström, "End-to-end optimization of source models for speech and audio coding using a machine learning framework," in *Proc. Interspeech*, Sep. 2019, pp. 3401–3405.

[38] G. Toderici, S. M. O'Malley, S. J. Hwang, D. Vincent, D. Minnen, S. Baluja, M. Covell, and R. Sukthankar, "Variable rate image compression with recurrent neural networks," 2015, *arXiv:1511.06085*.

[39] M. Courbariaux, Y. Bengio, and J.-P. David, "BinaryConnect: Training deep neural networks with binary weights during propagations," in *Proc. 28th Int. Conf. Neural Inf. Process. Syst.*, vol. 2, 2015, pp. 3123–3131.

[40] S. Chen, W. Wang, and S. J. Pan, "MetaQuant: Learning to quantize by learning to penetrate non-differentiable quantization," in *Proc. NIPS*, 2019, pp. 3916–3926.

[41] T. Backstrom, M. B. Mansali, P. P. Zarazaga, M. Ranjit, S. Das, and Z. Lachiri, "PyAWNeS-codec: Speech and audio codec for ad-hoc acoustic wireless sensor networks," in *Proc. 29th Eur. Signal Process. Conf. (EUSIPCO)*, Aug. 2021, pp. 1–11.
[42] T. Raiko, M. Berglund, G. Alain, and L. Dinh, "Techniques for learning binary stochastic feedforward neural networks," 2014, *arXiv:1406.2989*.
[43] T. Bäckström, *Speech Coding with Code-Excited Linear Prediction*. Springer, 2017.
[44] S. Dieleman, A. van den Oord, and K. Simonyan, "The challenge of realistic music generation: Modelling raw audio at scale," in *Proc. 32th Int. Conf. Neural Inf. Process. Syst.*, 2018, pp. 8000–8010.
[45] R. M. Gray, "Vector quantization," *IEEE ASSP Mag.*, vol. 1, no. 2, pp. 4–29, Apr. 1984.
[46] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: An ASR corpus based on public domain audio books," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Apr. 2015, pp. 5206–5210.
[47] R. Kumar, T. Deleu, and E. Racah. (2018). *Reproducing neural discrete representation learning*. Accessed: Oct. 15, 2021. [Online]. Available: https://github.com/ritheshkumar95/pytorch-vqvae/blob/master/final_project.pdf
[48] *Perceptual Evaluation of Speech Quality (PESQ): An Objective Method for End-to-End Speech Quality Assessment of Narrow-Band Telephone Networks and Speech Codecs*, document Rec. ITU-T Recommendation P.862, 2001. [Online]. Available: http://www.itu.int/rec/T-REC-P.862/en
[49] C. H. Taal, R. C. Hendriks, R. Heusdens, and J. Jensen, "An algorithm for intelligibility prediction of time–frequency weighted noisy speech," *IEEE Trans. Audio, Speech, Language Process.*, vol. 19, no. 7, pp. 2125–2136, Sep. 2011.
[50] *EVS Codec Detailed Algorithmic Description; 3GPP Technical Specification (Release 12)*, document Rec. TS 26.445, 2014.

MOHAMMAD HASSAN VALI received the bachelor's and master's degrees from the Babol Noshirvani University of Technology, in 2014 and 2017, respectively. He is currently pursuing the Ph.D. degree with the Department of Signal Processing and Acoustics, Aalto University, Finland. His research interests include signal processing, especially speech coding based on machine learning models.

TOM BÄCKSTRÖM (Senior Member, IEEE) received the master's and Ph.D. degrees from Aalto University, Finland, in 2001 and 2004, respectively, which was then known as the Helsinki University of Technology. He was a Professor with the International Audio Laboratory Erlangen, Friedrich-Alexander University, from 2013 to 2016, and a Researcher at Fraunhofer IIS, from 2008 to 2013. He has been an Associate Professor with the Department of Signal Processing and Acoustics, Aalto University, since 2016. He has contributed to several international speech and audio coding standards and is the chair and a co-founder of the ISCA Special Interest Group "Security and Privacy in Speech Communication." His research interests include technologies for spoken interaction, emphasizing efficiency and privacy and in particular in multi-device and multi-user environments.