Robust Training of Vector Quantized Bottleneck Models

Adrian Łańcucki  
NVIDIA Corporation  
Warsaw, Poland

Jan Chorowski  
University of Wrocław  
Wrocław, Poland

Guillaume Sanchez  
Université de Toulon, LIS  
Toulon, France

Ricard Marxer  
Université de Toulon, LIS  
Toulon, France

Nanxin Chen  
Johns Hopkins University  
Baltimore, USA

Hans J.G.A. Dolfing

Sameer Khurana  
Massachusetts Institute of Technology  
Cambridge, USA

Tanel Alumäe  
Tallinn University of Technology  
Tallinn, Estonia

Antoine Laurent  
Le Mans University  
Le Mans, France

Abstract—In this paper we demonstrate methods for reliable and efficient training of discrete representation using Vector-Quantized Variational Auto-Encoder models (VQ-VAEs). Discrete latent variable models have been shown to learn nontrivial representations of speech, applicable to unsupervised voice conversion and reaching state-of-the-art performance on unit discovery tasks. For unsupervised representation learning, they became viable alternatives to continuous latent variable models such as the Variational Auto-Encoder (VAE). However, training deep discrete variable models is challenging, due to the inherent non-differentiability of the discretization operation. In this paper we focus on VQ-VAE, a state-of-the-art discrete bottleneck model shown to perform on par with its continuous counterparts. It quantizes encoder outputs with on-line $k$-means clustering. We show that the codebook learning can suffer from poor initialization and non-stationarity of clustered encoder outputs. We demonstrate that these can be successfully overcome by increasing the learning rate for the codebook and periodic date-dependent codeword re-initialization. As a result, we achieve more robust training across different tasks, and significantly increase the usage of latent codewords even for large codebooks. This has practical benefit, for instance, in unsupervised representation learning, where large codebooks may lead to disentanglement of latent representations.

Index Terms—VQ-VAE, $k$-means, discrete information bottleneck

I. INTRODUCTION

Automatic extraction of useful features is one of the hallmarks of intelligent systems. Well designed models generalize better by filtering their inputs to retain only the information that is relevant to the task at hand [11]. Bottleneck layers, which induce this behavior, are especially important in feature learning. Simple bottleneck units indirectly reduce the amount of information allowed through by limiting feature dimensionality [2, 3]. The amount of information can be constrained directly, as in variational autoencoders, where a continuous stochastic bottleneck represents each sample with a probability distribution [4].

Discrete latent variable models also offer direct control over the information content of the learned representation. Recently, such models have shown great results in unsupervised learning of speech and image representations. A very popular model, the Vector-Quantized Variational Auto-Encoder (VQ-VAE) [5] encodes a speech signal as a sequence of discrete latent units. Compared with continuous latent variable models, the discrete VQ-VAE units have been empirically shown to correlate well with the phonetic content of the utterance, allowing advanced speech manipulation such as voice conversion [5]. Discrete representations learned by VQ-VAE were shown to achieve state-of-the-art results on unit discovery tasks, yielding a good separation between speaker traits and the phonetic content of speech [6]. Due to this property, in the recently organized ZeroSpeech 2019 “TTS Without T” challenge [7] VQ-VAE was a popular technique employed in many submissions [8, 9] and led to best performance on the unit-discovery task.

However, training deep discrete latent representation models is challenging due to the lack of a derivative function. One solution is to assume that the system is stochastic and optimize the mean output of discrete random units [10]. In contrast, VQ-VAE is a deterministic discrete variable model which operates by solving an approximate on-line $k$-means clustering in which the gradient is approximated using the straight-through estimator [11]. Therefore VQ-VAE can be trained using gradient descent like any other deep learning model. However we observe that it also poses typical $k$-means challenges such as sensitivity to initialization and non-stationarity of clustered neural activations during training. Moreover, the difficulty of $k$-means increases with the number of centroids, and the ability to encode the signal with a broad number of discrete codes is desirable for unsupervised representation learning [12].

In this paper we discuss the training dynamics of VQ-VAE models, compare different learning methods, and propose alternatives. First, we demonstrate the importance of proper initialization scale of the codebook relative to encoder outputs.
Then, we apply a deliberate, data-dependent initialization to the codebook. To address non-stationarity, we treat the problem as a sequence of static clustering problems, and periodically re-initialize the codebook from scratch with off-line clustering of recent encoder outputs. Lastly, we evaluate other possible training improvements: separately tuned codebook learning rates, and active balancing of latent feature magnitudes with codebook entries through batch normalization [13]. We establish that best results are obtained when all three mechanisms are present: batch normalization, data-dependent codebook re-initialization, and separately tuned learning rates.

In the following sections we introduce the Vector Quantized bottleneck, describe the motivation for the proposed enhancements, and their relation to proposed alternations to the learning rate. We then present experiments on three different problem domains. Finally, we put our work in the broader context of representation learning.

II. BACKGROUND

Consider a multilayered neural network trained to solve either a supervised or an unsupervised task. Let one of its hidden layers be called a bottleneck layer. We will refer to the values taken by the bottleneck layer as latent features. The encoder subnetwork takes input \( x \) and computes latent features \( e(x) \), which are then fed to the bottleneck. Its task is to remove spurious information from \( e(x) \).

For instance, auto-encoding neural networks rely on a bottleneck layer placed between the encoder and decoder to prevent learning features which are a trivial transformation of their raw inputs. A common bottleneck choice is e.g. dimensionality reduction used in PCA and SVD. However, bottleneck layers are also useful in supervised learning, e.g. the Projected LSTM architecture for ASR introduces a dimensionality reduction layer between recurrent cells [14].

Vector Quantized VAE [5] introduced a bottleneck that quantizes neural activations \( e(x) \), directly limiting the information content of the extracted features. It learns a codebook of \( K \) codewords \( w_i \in \mathbb{R}^d \), and replaces each activation \( e(x) \) with the nearest codeword:

\[
q(x) = \arg\min_{w_i} \|e(x) - w_i\|. \tag{1}
\]

Regardless of the dimensionality of \( q(x) \), the transmitted information is only about the index \( i \) of the chosen codeword \( w_i \) out of all \( K \) codewords, and thus limited to \( \log_2 K \) bits.

The network is trained using gradient optimization. Since quantization is non-differentiable, its derivative can only be approximated. VQ-VAE relies on the straight-through estimator \( \partial L / \partial e(x) \approx \partial L / \partial q(x) \) [11], and has additional loss terms for training the codebook [5]:

\[
L = L(q(x)) + ||s_g[e(x)] - q(x)||^2_2 + \gamma ||e(x) - s_g[q(x)]||^2_2, \tag{2}
\]

where \( s_g[\cdot] \) denotes the stop gradient operation. The first term \( L(q(x)) \) corresponds to the task loss, for instance the negative log-likelihood of the reconstruction

\[
L(q(x)) = -\log p(x|q(x)) \tag{3}
\]

in an autoencoder. The second term trains the codebook, by moving each codeword closer to the vector it replaced. As the result, it performs on-line \( k \)-means, with the codewords being the centroids. From now on we will use those terms interchangeably. The third term optimizes the encoder to produce representations close to the assigned codewords, with \( \gamma \in \mathbb{R} \) being a scaling hyperparameter.

The quantized signal \( q(x) \) is then processed by the rest of the network. In an auto-encoder, a decoder subnetwork uses \( q(x) \) to reconstruct the input \( x \). However, the quantized signal can also be used to perform supervised classification.

III. INTUITIONS ABOUT VQ TRAINING DYNAMICS

In this section we present practical observations on training of Vector Quantized bottlenecks with the straight-through estimator, as proposed in [5]. First of all, only the second term of the loss (2) affects codewords, implying that training algorithm only modifies the centroids that were selected. As a result, the training is prone to getting stuck in poor local optima, in which only a small subset of all codewords is in use. This permanent low codebook usage prevents learning rich data representations, because too little information is being passed through the bottleneck.
Afterward, we periodically reestimate the codebook during the training of the model using the regular Vector Quantization (VQ) algorithm. This ensures that all codewords are initially likely to be used.

### B. Batch data-dependent codebook updates

To improve the stability of VQ training, an alternative codebook learning rule, based on exponential moving averages (EMA) was recently proposed [5, 17, 18]. Below, we show that it is in fact equivalent to the original VQ-VAE training criterion with a codebook-specific learning rate, and as such does not address the problems of codebook initialization and non-stationarity.

Denote by \( x_1, \ldots, x_n \) the training samples and by \( e(x_1), \ldots, e(x_n) \) the corresponding encoder outputs attracted by a codeword \( w_i \) at training step \( t \) (we omit \( t \) for brevity). It follows that a codeword \( w_i \) is updated with those \( n_i \) closest encoder outputs. Specifically, for every codeword \( w_i \), the mean code \( m_i \) and usage count \( N_i \) is tracked:

\[
N_i \leftarrow N_i \cdot \gamma + n_i (1 - \gamma),
\]

\[
m_i \leftarrow m_i + \sum_j n_i e(x_j)(1 - \gamma),
\]

\[
w_i \leftarrow m_i / N_i,
\]

where \( \gamma \) is a discount factor.

We establish that the difference between the EMA training rule and the ordinary VQ-VAE codebook training algorithm is mostly adaptive rescaling of the codebook learning rate.

**Proposition 1.** The EMA update rule \((4)\) with constant usage counts \( N_i = 1 \) is equivalent to an SGD update for ordinary loss \((2)\) with a rescaling learning rate \( \alpha = (1 - \gamma)/2 \).

**Proof.** Let \( x \) be an input example such that \( e(x) \) maps to a codeword \( q(x) = w_i \). The SGD gradient update for a codeword \( w_i \) and the ordinary VQ loss \((2)\) is

\[
\frac{\partial L}{\partial w_i} = \frac{\partial}{\partial w_i} \left[ L(q(x)) + ||sg[e(x)] - q(x)||^2 \right]
\]

\[
+ \gamma ||e(x) - sg[q(x)]||^2.
\]

Only the second term concerns the codebook, and a given codeword \( w_i \) is only affected by the examples that were mapped to it. Thus the gradient simplifies to

\[
\frac{\partial L}{\partial w_i} = 2 \left( w_i - e(x_j) \right),
\]

which inserted into the gradient update

\[
w_i \leftarrow w_i - \alpha \frac{\partial L}{\partial w_i}
\]

with substitution \( \gamma = (1 - 2\alpha) \) yields

\[
w_i \leftarrow w_i \cdot \gamma + e(x_j)(1 - \gamma).
\]

—
We perform experiments on images (CIFAR-10), speech (Wall Street Journal) with learning rate. In the next section, we investigate the effect of

γ

hyperparameter should be set carefully in accordance with learning rate. In the next section, we investigate the effect of increasing the learning rate 10×.

We now see that when EMA tracks codeword usage running means

\(N_i\)

it effectively rescales the learning rate individually for every codeword according to its demand. If in a given batch there are fewer codewords

\(w_i\)

used than the running average, the learning rate will be lower. Conversely, if the usage increased, the learning rate will be higher. In the next section we compare EMA with ordinary VQ-VAE with rescaled codebook learning rates, and see that they lead to slightly different outcomes.

IV. EXPERIMENTS

We compare the proposed data-dependent codebook reestimation with vanilla codebook learning algorithm through minimization of loss

\([2]\)

and rescaled learning rate for the codebook, either explicitly or through EMA. In light of the scaling mismatch issue between codewords and encoded outputs from Section III-A, we experiment with batch normalization

\([13]\).

In order to test the proposed method in a broader context, we perform experiments on images (CIFAR-10), speech (Wall Street Journal), and handwriting (ScribbleLens [19]). We investigate static and temporal domains in supervised and unsupervised fashion for different modalities

\([1]\).

### A. Speech: Wall Street Journal

We train a supervised speech recognition model and report phoneme error rates (PER). Wall Street Journal (WSJ) is a corpus of news articles read by professional speakers, popular in the domain of speech recognition. Our model is similar to Deep Speech 2

\([20]\)

with an addition of a bottleneck. The encoder is composed of a stack of strided convolutions followed by bidirectional recurrent layers. Encoded features are put through a VQ bottleneck with a single codebook. It quantizes the encoder’s output at every timestep. Finally, the quantized signal is passed through a 1 × 1 convolution and SoftMax followed by the CTC loss

\([21]\).

The inputs to the model are 80 Mel-filterbank bands with energy features extracted every 10ms with \(\Delta\), \(\Delta\Delta\), and global cepstral mean and variance normalization (CMVN). We train the model on 64-frame chunks with targets derived form forced alignments obtained using Tri3b model from Kaldi WSJ recipe

\([22]\). To get a clear picture of the VQ-VAE training process, we refrain from using regularization, which otherwise would be necessary for a small corpus like WSJ.

The model no bottleneck serves as a baseline for quantized models. We contrast six variants of VQ training: base vanilla, with batch normalization (BN) inserted before the quantization, with BN and increased codebook learning rate, using the EMA training rule, and with data-dependent codebook reestimation with and without increased codebook learning rates. We run hyperparameter searches over the learning rate and discount factor

\(\gamma\)

for codebook learning rate and EMA models respectively. For EMA, we report results for the best models. For codebook learning rates, we establish a rule of thumb to set them to 10× the regular learning rates. Wrong adjustment of these parameters could make the models worse than the vanilla baseline.

We report phoneme error rates and codebook perplexity with no external language models in Table I and Figure 2.

\[\text{Fig. 2. Wall Street Journal Dev93 supervised phoneme error rate (PER)}\]

Values reported for

\(\gamma\)

in [5] indicate that higher learning rates for the codebook might be beneficiary for ordinary VQ training. The

\(\gamma\)

hyperparameter should be set carefully in accordance with learning rate. In the next section, we investigate the effect of increasing the learning rate 10×.

We now see that when EMA tracks codeword usage running means

\(N_i\)

it effectively rescales the learning rate individually for every codeword according to its demand. If in a given batch there are fewer codewords

\(w_i\)

used than the running average, the learning rate will be lower. Conversely, if the usage increased, the learning rate will be higher. In the next section we compare EMA with ordinary VQ-VAE with rescaled codebook learning rates, and see that they lead to slightly different outcomes.

### B. Handwriting: ScribbleLens

ScribbleLens

\([23], [19]\)

is a freely available corpus of old handwritten Dutch from late 16th to mid-18th century, the time of Dutch East India company. It is segmented into lines of handwriting (Fig. 3). A subset of ScribbleLens is annotated with transcriptions.
We treat lines of handwriting as multidimensional input features on which we train an unsupervised autoencoder. We use the same encoder architecture as for Wall Street Journal, followed by a VQ bottleneck with a single codebook (size $K = 4096$), and a conditional PixelCNN [24] decoder. Input lines are scaled to 32-pixels high. The encoder extracts a latent vector every four pixel columns.

Used alone, batch normalization has a minor detrimental effect. Increased codebook learning rate and data-dependent codebook reestimation improve latent code usage and lead to lower reconstruction log-likelihoods measured in bits per dimension (BPD; Table II). Again, improvements are related to a higher usage of the bottleneck capacity.

We also report test set evidence lower bound (ELBO) expressed in bits per dimension [4], [5]:

$$\log p(x) = \sum_z p(x|z)p(z) \geq \text{ELBO} =$$

$$= \log \mathbb{E}_z q(z|x)p(x|z) - KL(q(z|x)||p(z)) =$$

$$= \log (p(z|x = q(x)) + \log p(z = q(x)).$$

The last transition follows from the fact that $q(x)$ gives a deterministic encoding, $p(x|z)$ is the reconstruction loss, and $p(z)$ is the prior latent code probability. We consider two formulations of the prior over latent vectors: a uniform prior $p(z) = 1/K$, and a unigram prior of observed codeword frequencies. The negative of ELBO (NELBO) has an intuitive interpretation: to transmit a data sample, we need to transmit the information contained in the latent features plus the information required to reconstruct the data sample from the conditional decoder probability. For ease of comparison with BPD, in Table II we report the NELBOs in bits. The Uniform NELBO is computed as $\text{BPD} + \log_2(4096)/32/4$, i.e. the cost of reconstructing a single pixel augmented by the cost of transmitting a single latent amortized over all pixels this latent corresponds to, while for Unigram NELBO we account for the actual codebook usage by computing $\text{BPD} + \log_2(\text{perplexity})/32/4$.

Ideally, an increase in bottleneck bandwidth should match the increase in reconstruction log-likelihood, keeping the ELBO constant. However, comparing the uniform and unigram NELBOs, we see that improvements in codebook usage translate to smaller likelihood improvements. This may be caused by imperfection of the PixelCNN decoder, unable to fully use the information in the latents.

C. Images: CIFAR-10

On CIFAR-10 we train an unsupervised auto-encoder. The model contains a convolutional encoder with residual connections, followed by a VQ bottleneck and a deconvolutional decoder. Our model follows closely [5], and we describe only the differences. The network models a discrete distribution over 256 output levels for RGB channels with SoftMax. We use the Adam optimizer [25] with learning rate $5e-4$. We also use Polyak checkpoint averaging in which the model is evaluated on parameters that are an exponential moving average of parameters during the most recent training steps [26].

| Model | Reco. BPD | Pppl | Uniform NELBO bits | Unigram NELBO bits |
|-------|-----------|------|---------------------|--------------------|
| vanilla VQ | 0.213 | 322 | 0.307 | 0.278 |
| + BN | 0.216 | 260 | 0.309 | 0.278 |
| + BN + codebook LR | 0.212 | 432 | 0.306 | 0.281 |
| + BN + EMA | 0.207 | 1118 | 0.301 | 0.287 |
| + BN + data-dep reest | 0.200 | 2388 | 0.294 | 0.288 |
| + BN + data-dep reest + codebook LR | **0.200** | **2446** | **0.294** | **0.288** |

As even small images convey a lot of information, we consider two settings with multiple independent codebooks: $8 \times 128$ codewords (56 bits), and $5 \times 1024$ codewords (50 bits; Table III). Each encoded $e(x)$ is split into equally-sized blocks that are quantized independently. This allows us to achieve high bottleneck bitrates with small codebooks, facilitating k-means training, which increases in difficulty with the number of centroids.

Table III summarizes test set bits/dim (BPD) on the CIFAR-10 dataset. For this model, placing a BN layer before the quantization leads to minor improvements in BPD in all settings. All models ultimately converge to similar values. Evaluating BPD during training (Figure 4) shows that data-dependent reestimation improves convergence. The initial high values of BPD seen in the figure is caused by Polyak checkpoint averaging.

V. RELATED WORK

Representation learning is a long-standing problem in machine learning. In auto-encoding systems, trained for input reconstruction a bottleneck prevents the system from learning a trivial data representation, such as a multiplication by an orthonormal matrix. For instance, to prevent trivial encodings classical techniques, such as PCA or classical neural auto-encoders enforce a reduction of the dimensionality of the data. However, other constraints, such as sparsity [27] or non-negativeness [28] have also been explored.
The Variational Autoencoder \cite{10} can be seen as an autoencoder whose bottleneck limits the expected number of bits used to encode each sample. It therefore implements the information bottleneck \cite{1, 29} principle. The limitation on encoding bitrate is made explicit in discrete latent variable models, such as the VQ-VAE \cite{30}. These have been shown to extract meaningful latent representations of speech in an unsupervised fashion \cite{6}.

Deep Learning models are trained by gradient descent on a loss function. Discrete models by definition do not have a gradient. One possible solution is to treat them as stochastic, and optimize the expected value of the loss, which varies smoothly with model parameters and therefore has a well-defined gradient. This approach is employed in the Gumbel-Softmax reparametrization trick for the VAE \cite{10}, which yields a low-variance, but high bias estimator for the gradient. The gradient can also be estimated using the REINFORCE algorithm \cite{31}, which is unbiased but has a high variance. The two approaches can be successfully combined \cite{32}.

In contrast the VQ-VAE \cite{5} operates in a deterministic mode, and computes a descent direction using backpropagation with the straight-through estimator \cite{11}. Despite this approximation, it works well in practice.

Deterministic quantization with \( k \)-means can be seen as hard Expectation Maximization (EM) \cite{17}. It can be replaced with soft EM, which performs probabilistic discrete quantization in order to improve training performance. Such bottleneck is an approximation to a continuous probabilistic bottleneck, similar to VAE \cite{4}. Probabilistic quantization can be applied only to improve training, and disabled during inference \cite{5}. In addition, \cite{33} introduces joint training of a prior on codewords, under which the most frequent codewords can be passed at a lower bit cost.

Classical acoustic unit discovery systems fit a probabilistic model which segments the speech and fits HMMs to each subword unit \cite{34}. Training of these systems requires Gibbs sampling or Variational Inference \cite{35}. Recently, they have been composed with neural acoustic models \cite{36, 37} by extending the VAE with a structured prior \cite{38}. In contrast, the VQ-VAE offers a much simpler approximation to the underlying probabilistic model, which can, however, be extended with an hmm-like segmental prior \cite{12}.

\section{VI. Conclusions}

In this paper we have shown how VQ-VAE codebook learning can benefit from being viewed as on-line \( k \)-means clustering. We propose the following enhancements to the training procedure: increasing the codebook learning rate, enabling batch normalization prior to quantization, and periodic batch codebook reestimations. Together, these three changes stabilize training and lead to a broader codebook usage and improved performance on supervised and unsupervised downstream tasks in image processing, speech recognition, and handwritten unit discovery.

\section{Acknowledgements}

We would like to thank Lucas Pagé-Caccia for help with CIFAR-10 experiments, and Benjamin van Niekerk for open sourcing his VQ-VAE experiment.\cite{32}

The research reported here was conducted at the 2019 Frederick Jelinek Memorial Summer Workshop on Speech and Language Technologies, hosted at L’École de Technologie Supérieure (Montreal, Canada) and sponsored by Johns Hopkins University with unrestricted gifts from Amazon, Facebook, Google, and Microsoft. Authors also thank Polish National Science Center for funding under the Sonata 2014/15/D/ST6/04402 grant the PLGrid project for computational resources on Prometheus cluster.

\section{References}

\begin{thebibliography}{99}
\bibitem{1} Naftali Tishby, Fernando C Pereira, and William Bialek, “The information bottleneck method,” in \textit{The 57th Annual Allerton Conference on Communication, Control, and Computing}, 1999, pp. 368–377.
\bibitem{2} Karel Veselý, Martin Karafiát, František Grézl, Miloš Janda, and Ekaterina Egorova, “The language-independent bottleneck features,” in \textit{Proc. Spoken Language Technology Workshop (SLT)}, 2012, pp. 336–341.
\bibitem{3} Dong Yu and Michael L Seltzer, “Improved bottleneck features using pretrained deep neural networks,” in \textit{Interspeech}, 2011.
\bibitem{4} Diederik P. Kingma and Max Welling, “Auto-encoding variational bayes,” \textit{arXiv} preprint \textit{arXiv:1312.6114}, 2013.
\bibitem{5} Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu, “Neural Discrete Representation Learning,” \textit{arXiv}:1711.00937 [cs], Nov. 2017.
\bibitem{6} J. Chorowski, R. J. Weiss, S. Bengio, and A. van den Oord, “Unsupervised speech representation learning using wavenet autoencoders,” \textit{IEEE/ACM Transactions on Audio, Speech, and Language Processing}, vol. 27, no. 12, pp. 2041–2053, Dec 2019.
\bibitem{7} Ewan Dunbar, Robin Algayres, Julien Karadayi, Mathieu Bernard, Juan Benjumea, Xuan-Nga Cao, Lucie Miskic, Charlotte Dugrain, Lucas Ondel, Alan W. Black, Laurent Besacier, Sakriani Sakti, and Emmanuel Dupoux, “The Zero Resource Speech Challenge 2019: TTS Without T,” in \textit{Proc. Interspeech 2019}, 2019, pp. 1088–1092.
\bibitem{8} Andros Tjandra, Berrak Sisman, Mingyang Zhang, Sakriani Sakti, Haizhou Li, and Satoshi Nakamura, “VQVAE: Unsupervised Unit Discovery and Multi-Scale Code2Spec Inverter for Zerospeech Challenge 2019,” in \textit{Proc. Interspeech 2019}, 2019, pp. 1118–1122.
\end{thebibliography}
