Robust Training of Vector Quantized Bottleneck Models

Adrian Łańcucki, Jan Chorowski, Guillaume Sanchez, Ricard Marxer, Nanxin Chen, Hans J G A Dolfing, Sameer Khurana, Tanel Alumäe, Antoine Laurent

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 [1]. Bottleneck layers, which induce this behavior, are especially important in feature learning. Simple bottlenecks 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 $\partial\mathcal{L}/\partial e(x) \approx \partial\mathcal{L}/\partial q(x)$ [11], and has additional loss terms for training the codebook [5]:

$$\begin{aligned}\mathcal{L} = L(q(x)) &+ ||\text{sg}[e(x)] - q(x)||^2 \\ &+ \gamma||e(x) - \text{sg}[q(x)]||^2,\end{aligned} \tag{2}$$

where $\text{sg}[\cdot]$ 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

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

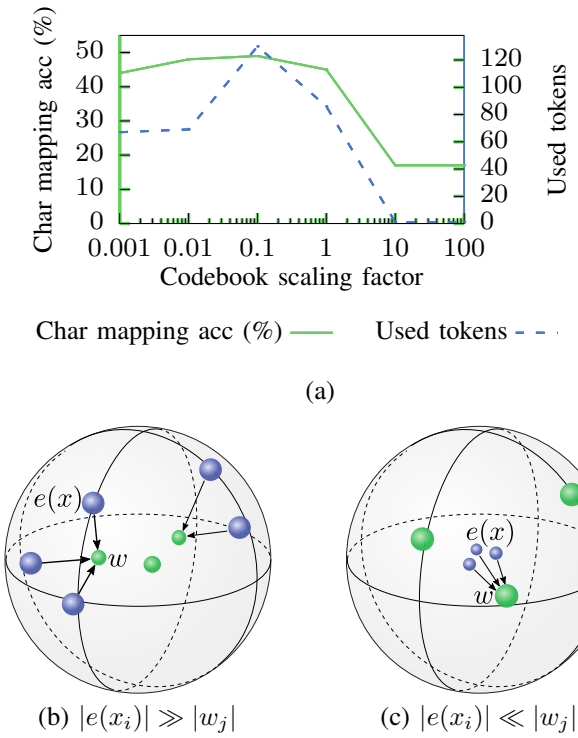

Fig. 1. The impact of scale of encoder outputs relative to the scale of codebook words shown in 3-D. **(a)** Relative scale of codewords w and encoder outputs $e(x)$ impacts performance of mapping bottleneck features to symbols on a subset of ScribbleLens (see Section IV-B for details). **(b)** If the encoder outputs are larger, multiple codewords are likely to be used. **(c)** If the encoder outputs are smaller, they tend to cluster and map to fewer codewords.

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.

Algorithm 1 Data-dependent Reestimated Vector Quantization

for $it \in 1, \dots iterations$ **do**
  $x = \text{encoder}(input)$
  $reservoir \leftarrow \text{update\_reservoir}(reservoir, x)$
  if $it < M_{init}$ **then**
    $quantized = x$
  else
    if $it \ \% \ r_{reestim} == 0$ **and** $it < M_{init} + M_{reestim}$ **then**
      $codebook = \text{k-means++}(reservoir)$
    $quantized = \text{nearest\_codes}(codebook)$
  ▷ Continue forward pass

A. Importance of proper scaling

We empirically observe that in order to maximize codebook usage, the codewords have to be smaller in norm than the encoder outputs $e(x)$ (Fig. 1a). During quantization, every output of the encoder $e(x)$ is matched to the nearest codeword w. If the codewords are smaller in norm, the selection depends on the angular distance of the encoded representation and the codeword (Fig. 1b), and many codewords are used. However, if the codewords are larger in norm, all encoder outputs are likely to be assigned to the smallest codeword (Fig. 1c). This blocks the training signal from reaching remaining codewords, and leads to underutilization of the codebook. Since the scale of feature activations can change during network training, we investigate the use of batch normalization to enforce a magnitude ratio between $e(x)$ and w.

B. Batch data-dependent codebook updates

Optimal k-means codebook initialization is data-dependent [15], and most usage scenarios of k-means assume stationarity of the data. However, during training of the encoder, the representations $e(x)$ changes with every iteration. These changes might be too rapid for the codebook to adapt, especially in the light of sparse gradient updates, that influence only the selected codewords.

For this reason, we propose to perform periodic data-dependent codebook reestimation during a number of initial training steps (Algorithm 1). During training, we continually maintain a sample of outputs from the encoder using reservoir sampling [16]. During the first M_{init} warm-up iterations we perform no quantization, letting the network to initially stabilize outputs of the encoder.

After M_{init} iterations, we initialize the codebook by applying k-means++ clustering [15] to samples collected in the reservoir. This ensures that all codewords are initially likely to be used. Afterward, we periodically reestimate the codebook during the next $M_{reestim}$ iterations, to help it keep up with changes to the distribution of encoder outputs. By doing so, we treat the non-stationary problem of clustering rapidly changing encoder outputs as a sequence of stationary problems.

After this prolonged data-dependent initialization, we train the model using the regular Vector Quantization (VQ) algorithm. This fine-tunes the codebook and allows the model to learn an efficient quantization which discards unimportant information. One of the advantages of data-dependent initialization is its robustness to the choice of the $M_{reestim}$ parameter, leading to better results than the vanilla training algorithm.

C. EMA: an alternative training rule

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 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 $x_1, \dots, x_{n_i}$ the training samples and by $e(x_1), \dots, e(x_{n_i})$ 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:

$$\begin{aligned} N_i &\leftarrow N_i \cdot \gamma + n_i(1-\gamma), \\ m_i &\leftarrow m_i \cdot \gamma + \sum_j^{n_i} e(x_j)(1-\gamma), \\ w_i &\leftarrow \frac{m_i}{N_i}, \end{aligned} \tag{4}$$

where γ 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 codebook word $q(x) = w_i$. The SGD gradient update for a codeword w_i and the ordinary VQ loss (2) is

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial w_i} = \frac{\partial}{\partial w_i} \Big[& L(q(x)) + ||\text{sg}[e(x)] - q(x)||^2 \\ & + \gamma ||e(x) - \text{sg}[q(x)]||^2 \Big]. \end{aligned} \tag{5}$$

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

$$\frac{\partial \mathcal{L}}{\partial w_i} = 2\left(w_i - e(x_j)\right), \tag{6}$$

which inserted into the gradient update

$$w_i \leftarrow w_i - \alpha \frac{\partial \mathcal{L}}{\partial w_i} \tag{7}$$

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

$$w_i \leftarrow w_i \cdot \gamma + e(x_j)(1-\gamma). \tag{8}$$

■

TABLE I
HIGHER CODEBOOK USAGE (PERPLEXITY) LEADS TO LOWER PHONEME ERROR RATE (PER) ON THE SUPERVISED WSJ TASK

Model	Dev PER	Code Perplexity
vanilla VQ	16.9	16
+ BN	17.6	16.5
+ BN + codebook LR	13.1	151
+ BN + EMA	10.5	61.8
+ BN + data-dep reest	10.4	393
+ BN + data-dep reest + codebook LR	**9.8**	**574**
no bottleneck	11.6	*N/A*

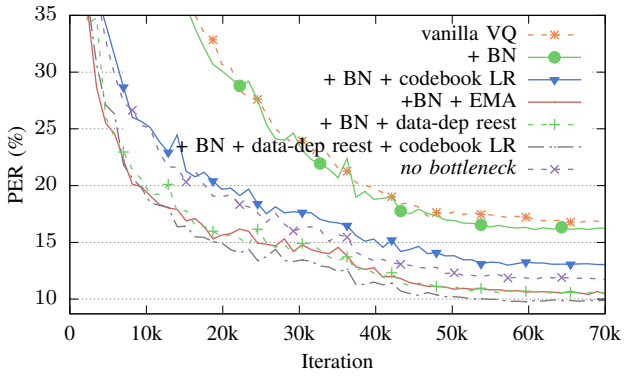

Fig. 2. Wall Street Journal Dev93 supervised phoneme error rate (PER)

Values reported for γ in [5] indicate that higher learning rates for the codebook might be beneficiary for ordinary VQ training. The γ hyperparameter should be set carefully in accordance with learning rate. In the next section, we investigate the effect of increasing the learning rate $10\times$.

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].

[1] Source code for replication available at https://github.com/distsup/DistSup

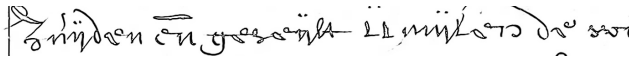

Fig. 3. A sample training line from the ScribbleLens corpus [23], [19]

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 Δ, Δ-Δs, 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 γ 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 $10\times$ 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. We notice that enabling the vanilla VQ bottleneck hurts the performance of the model. Batch normalization together with separate codebook learning rate help, but best results also require data-dependent codebook reestimation. This model not only offers the highest codebook usage (indicated by highest perplexity), but also reaches the error rate below the continuous baseline, demonstrating the regularizing effect of the information bottleneck.

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 be 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]:

$$\begin{aligned}\log p(x) &= \log \sum_z p(x|z)p(z) \geq ELBO = \\ &= \log \mathbb{E}_{z\ q(z|x)} p(x|z) - KL(q(z|x)||p(z)) = \\ &= \log(p(x|z = q(x)) + \log p(z = q(x)).\end{aligned} \tag{9}$$

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 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 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 $5\text{e}{-4}$. 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].

TABLE II
ON SCRIBBLELENS, DATA-DEPENDENT CODEBOOK REESTIMATION RESULTS IN THE FULLEST CODEBOOK USAGE, LEADING TO THE BEST RECONSTRUCTION MEASURED IN BITS PER DIM. UNIFORM AND UNIGRAM ELBOS SHOW THAT INCREASED BOTTLENECK BANDWIDTH RESULTS IN A SMALLER RECONSTRUCTION LIKELIHOOD IMPROVEMENT.

Model	Rec. BPD	Pplx	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

TABLE III
UNSUPERVISED BITS/DIM (BPD) ON CIFAR-10 TEST SET FOR 8×128-CODEWORD AND 5×1024-CODEWORD VQ BOTTLENECKS (MEDIAN OF FIVE RUNS)

Model	Reconstruction BPD 8×128	5×1024
VQ-VAE	4.679	4.811
+ BN	4.677	4.809
+ BN + codebook LR	4.677	**4.796**
+ BN + EMA	4.683	4.798
+ BN + data-dep reest	4.665	4.807
+ BN + data-dep reest + codebook LR	**4.659**	4.802
VQ-VAE [5]	4.67 (10×512)	

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) for the models trained 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.

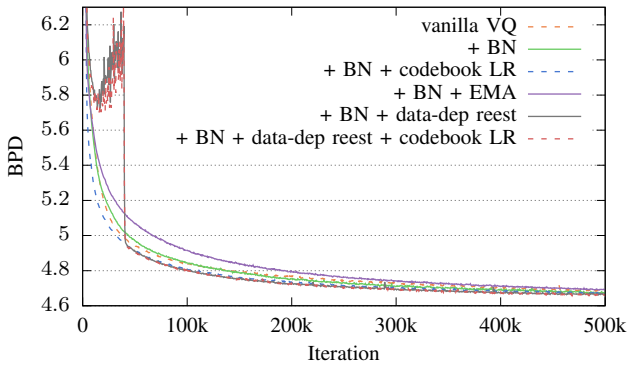

Fig. 4. Unsupervised bits/dim (BPD) on CIFAR-10 test set for 8×128-codeword model during training with checkpoint averaging. During initial data-dependent reestimation iterations, BPD is higher for checkpoint-averaged models, because each codebook re-initialization breaks the averages. However, when the reestimation period is over, these models converge faster than others.

The Variational Autoencdoer [4] can be seen as an auto-encoder whose bottleneck limits the expected number of bits used to encode each sample. It therefore implements the information bottleneck [1], [29] principle. The limitation on encoding bitrate is made explicit in discrete latent variable models, such as the VQ-VAE [30]. These have been shown to extract meaningful latent representations of speech in an unsupervised fashion [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 [10], which yields a low-variance, but high bias estimator for the gradient. The gradient can also be estimated using the REINFORCE algorithm [31], which is unbiased but has a high variance. The two approaches can be successfully combined [32].

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

Deterministic quantization with k-means can be seen as hard Expectation Maximization (EM) [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 [4]. Probabilistic quantization can be applied only to improve training, and disabled during inference [33]. In addition, [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 [34]. Training of these systems requires Gibbs sampling or Variational Inference [35]. Recently, they have been composed with neural acoustic models [36], [37] by extending the VAE with a structured prior [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 [12].

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.

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 experiments[2].

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.

References

[1] Naftali Tishby, Fernando C Pereira, and William Bialek, "The information bottleneck method," in *The 37th annual Allerton Conference on Communication, Control, and Computing*, 1999, pp. 368–377.

[2] Karel Veselỳ, Martin Karafiát, František Grézl, Miloš Janda, and Ekaterina Egorova, "The language-independent bottleneck features," in *Proc. Spoken Language Technology Workshop (SLT)*, 2012, pp. 336–341.

[3] Dong Yu and Michael L Seltzer, "Improved bottleneck features using pretrained deep neural networks," in *Interspeech*, 2011.

[4] Diederik P. Kingma and Max Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[5] Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu, "Neural Discrete Representation Learning," *arXiv:1711.00937 [cs]*, Nov. 2017.

[6] J. Chorowski, R. J. Weiss, S. Bengio, and A. van den Oord, "Unsupervised speech representation learning using wavenet autoencoders," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 27, no. 12, pp. 2041–2053, Dec 2019.

[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 *Proc. Interspeech 2019*, 2019, pp. 1088–1092.

[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 *Proc. Interspeech 2019*, 2019, pp. 1118–1122.

[2]https://github.com/bshall/VectorQuantizedVAE

[9] Ryan Eloff, André Nortje, Benjamin van Niekerk, Avashna Govender, Leanne Nortje, Arnu Pretorius, Elan van Biljon, Ewald van der Westhuizen, Lisa van Staden, and Herman Kamper, "Unsupervised Acoustic Unit Discovery for Speech Synthesis Using Discrete Latent-Variable Neural Networks," in *Proc. Interspeech 2019*, 2019, pp. 1103–1107.

[10] Eric Jang, Shixiang Gu, and Ben Poole, "Categorical Reparameterization with Gumbel-Softmax," Nov. 2016.

[11] Yoshua Bengio, "Estimating or Propagating Gradients Through Stochastic Neurons," *arXiv:1305.2982 [cs]*, May 2013.

[12] Jan Chorowski, Nanxin Chen, Ricard Marxer, Hans Dolfing, Adrian Łańcucki, Guillaume Sanchez, Tanel Alumäe, and Antoine Laurent, "Unsupervised neural segmentation and clustering for unit discovery in sequential data," *Perception as Generative Reasoning Workshop, NeurIPS 2019*, 2019.

[13] Sergey Ioffe and Christian Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," *arXiv:1502.03167 [cs]*, Feb. 2015.

[14] Hasim Sak, Andrew W. Senior, and Françoise Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," in *INTERSPEECH 2014, 15th Annual Conference of the International Speech Communication Association, Singapore, September 14-18, 2014*, Haizhou Li, Helen M. Meng, Bin Ma, Engsiong Chng, and Lei Xie, Eds. 2014, pp. 338–342, ISCA.

[15] David Arthur and Sergei Vassilvitskii, "K-means++: The advantages of careful seeding," in *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, Philadelphia, PA, USA, 2007, SODA '07, pp. 1027–1035, Society for Industrial and Applied Mathematics.

[16] Jeffrey S. Vitter, "Random sampling with a reservoir," *ACM Trans. Math. Softw.*, vol. 11, no. 1, pp. 37–57, Mar. 1985.

[17] Aurko Roy, Ashish Vaswani, Arvind Neelakantan, and Niki Parmar, "Theory and Experiments on Vector Quantized Autoencoders," *arXiv:1805.11063 [cs, stat]*, May 2018.

[18] Ali Razavi, Aaron van den Oord, and Oriol Vinyals, "Generating Diverse High-Fidelity Images with VQ-VAE-2," *arXiv:1906.00446 [cs, stat]*, June 2019.

[19] Hans J.G.A. Dolfing, Jerome Bellegarda, Jan Chorowski, Ricard Marxer, and Antoine Laurent, "The "ScribbleLens" Dutch historical handwriting corpus," in *Under review for: International Conference on Frontiers of Handwriting Recognition (ICFHR)*, 2020, http://www.openslr.org/84/.

[20] Dario Amodei, Sundaram Ananthanarayanan, and Rishita Anubhai et al., "Deep speech 2 : End-to-end speech recognition in english and mandarin," in *Proceedings of The 33rd International Conference on Machine Learning*, Maria Florina Balcan and Kilian Q. Weinberger, Eds., New York, New York, USA, 20–22 Jun 2016, vol. 48 of *Proceedings of Machine Learning Research*, pp. 173–182, PMLR.

[21] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber, "Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks," in *Proceedings of the 23rd International Conference on Machine Learning*, New York, NY, USA, 2006, ICML '06, pp. 369–376, ACM.

[22] Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, Jan Silovsky, Georg Stemmer, and Karel Vesely, "The kaldi speech recognition toolkit," in *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*. Dec. 2011, IEEE Signal Processing Society, IEEE Catalog No.: CFP11SRW-USB.

[23] Open Speech and Language Resources, *ScribbleLens*, http://www.openslr.org/84/.

[24] Aäron Van Den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu, "Pixel recurrent neural networks," in *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*. 2016, ICML'16, pp. 1747–1756, JMLR.org.

[25] Diederik Kingma and Jimmy Ba, "Adam: A Method for Stochastic Optimization," *arXiv:1412.6980 [cs]*, Dec. 2014.

[26] Boris T Polyak and Anatoli B Juditsky, "Acceleration of stochastic approximation by averaging," *SIAM journal on control and optimization*, vol. 30, no. 4, pp. 838–855, 1992.

[27] Bruno A. Olshausen and David J. Field, "Emergence of simple-cell receptive field properties by learning a sparse code for natural images," *Nature*, vol. 381, no. 6583, pp. 607–609, June 1996.

[28] Daniel D. Lee and H. Sebastian Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, no. 6755, pp. 788–791, Oct. 1999.

[29] Alexander A. Alemi, Ian Fischer, Joshua V. Dillon, and Kevin Murphy, "Deep Variational Information Bottleneck," Dec. 2016.

[30] Hanwei Wu and Markus Flierl, "Variational Information Bottleneck on Vector Quantized Autoencoders," *arXiv:1808.01048 [cs, stat]*, Aug. 2018.

[31] Ronald J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine Learning*, vol. 8, no. 3-4, pp. 229–256, May 1992.

[32] George Tucker, Andriy Mnih, Chris J Maddison, John Lawson, and Jascha Sohl-Dickstein, "REBAR: Low-variance, unbiased gradient estimates for discrete latent variable models," in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., pp. 2627–2636. Curran Associates, Inc., 2017.

[33] Casper Kaae Sønderby, Ben Poole, and Andriy Mnih, "Continuous relaxation training of discrete latent variable image models," *Beysian DeepLearning workshop*, NIPS 201.

[34] Chia-ying Lee and James Glass, "A Nonparametric Bayesian Approach to Acoustic Model Discovery," in *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Jeju Island, Korea, July 2012, pp. 40–49, Association for Computational Linguistics.

[35] Lucas Ondel, Lukaš Burget, and Jan Černocký, "Variational Inference for Acoustic Unit Discovery," *Procedia Computer Science*, vol. 81, pp. 80–86, 2016.

[36] Janek Ebbers, Jahn Heymann, Lukas Drude, Thomas Glarner, Reinhold Haeb-Umbach, and Bhiksha Raj, "Hidden Markov Model Variational Autoencoder for Acoustic Unit Discovery," in *Interspeech 2017*. Aug. 2017, pp. 488–492, ISCA.

[37] Thomas Glarner, Patrick Hanebrink, Janek Ebbers, and Reinhold Haeb-Umbach, "Full Bayesian Hidden Markov Model Variational Autoencoder for Acoustic Unit Discovery," in *Interspeech 2018*. Sept. 2018, pp. 2688–2692, ISCA.

[38] Matthew J. Johnson, David Duvenaud, Alexander B. Wiltschko, Sandeep R. Datta, and Ryan P. Adams, "Composing graphical models with neural networks for structured representations and fast inference," *arXiv:1603.06277 [stat]*, Mar. 2016.