Performance Comparison of Deep Neural Network Quantizers to Continuous ASR Systems

Tobias Watzel, Lujun Li, Ludwig Kürzinger, Gerhard Rigoll
Technical University of Munich, Chair of Human-Machine Communication, Email: tobias.watzel@tum.de

Abstract
Continuous approaches where a Hidden Markov Model (HMM) is combined with a Gaussian Mixture Model (GMM)/Deep Neural Network (DNN) are still one of the most popular architectures in Automatic Speech Recognition (ASR). They performed well on several challenging databases. Discrete approaches, i.e., models with a Discrete Hidden Markov Model (DHMM), require a discretization of the input data whereby information is getting lost. Several discrete approaches tried in the past to compete in a discrete fashion, however, they were not able to achieve an equally good Word Error Rate (WER) as in continuous systems. In our approach we return to discrete models. We build up a Deep Neural Network Quantizer (DNNQ), propose a novel training technique and demonstrate how such a system performs compared to a continuous system. In our experiments we reveal that the DNNQ provides a Word Error Rate (WER) reduction of 23.0 % on the dev and of 22.8 % on the test set, respectively, compared to a continuous HMM/Gaussian Mixture Model (GMM) system.

Introduction
Automatic Speech Recognition (ASR) systems have been gaining a lot of attention over the past years. Until 2012, mostly HMM-GMM systems in different variations were used in speech recognition [4]. Then slowly, DNNs were getting popular, caused by an increase of computational power in computer systems. Usually, these DNNs are combined with an HMM into an HMM-DNN system, more precisely a hybrid approach. Nowadays, besides hybrid approaches, end-to-end ASR systems are used in different approaches, e.g. [3]. Despite of several improvements in end-to-end models, hybrid models are still one of the best systems with the lowest WERs. Recent architectures can be found in [10, 6]. All hybrid approaches have in common that the DNN models a continuous posterior distribution $p(\boldsymbol{y}|\boldsymbol{x})$ based on a feature $\boldsymbol{x}$ which is combined with a time-variant component, e.g. an HMM [2].
Several years ago, another system category was proposed which combined DNN with a DHMM to a so-called discrete model. However, it has been disregarded since it performed worse than a conventional continuous HMM-GMM model. The idea of a discrete system is to cluster the input data $\boldsymbol{x}$ in different clusters j, e.g. by applying the k-means algorithm. As a result we receive a codebook which can be used by a Vector Quantizer (VQ) to assign $\boldsymbol{x}$ to a nearest cluster j. By doing so, we retrieve a label stream based on $\boldsymbol{x}$. This label stream can be used to train a DHMM. Differently as in an HMM-GMM, in a DHMM the emission probability is modeled by a histogram which is optimized by maximum likelihood (ML).
In order to further improve a discrete system, another cluster algorithm has to be chosen since the k-means algorithm can only cluster the data by linear class borders. Therefore, Neukirchen and Rigoll proposed the NNVQ [11]. The NNVQ is a shallow neural network which is trained to quantize the data into different classes. The highest activation of the neuron in the output layer represents the cluster j to which the input $\boldsymbol{x}$ belongs to. Despite of the plain Resource Management (RM) database, their model actually outperformed the traditional k-means system, however, it was only able to perform equally well as a continuous system does.
In another approach [12] they examined the performance of the NNVQ on a large-vocabulary speech recognition (LVSR) task. They evaluated the model on the Wall Street Journal (WSJ) database [8] and performed better than a k-means system. However, the continuous system still outperformed their discrete model.
In the following work, we return to discrete ASR. We want to apply the current state-of-the-art developments in neural networks and enhance the ideas of [11, 12]. We take the NNVQ and add several layers to receive a DNNQ. The resulting model is compared to a continuous GMM system.

Proposed Method
Let $\boldsymbol{D} = \{(\boldsymbol{x}_i, \hat{y}_i)\}_{i=1}^{N}$ be a dataset of size N with feature vectors $\boldsymbol{x}_i$ and its corresponding ground-truth label $\hat{y}_i$. In our approach, we use a DNNQ to act as a VQ to create the function $g_\theta : \boldsymbol{X} \mapsto \hat{\boldsymbol{m}}$, where θ represents the weights of the network and $\hat{\boldsymbol{m}} = \{\arg\max_j \boldsymbol{m}_i^j\}_{i=1}^{N}$ is the maximal activation j in the output layer $\boldsymbol{m}_i$ of size N_{clu}. By changing N_{clu}, we are able to increase or decrease the number of emitting labels $\hat{\boldsymbol{m}}$, thus, in range of $1 \leq \hat{m}_j \leq N_{\text{clu}}$, whereas the ground-truth labels are in a range of $1 \leq \hat{y}_k \leq N_K$. The dimension of the label space is defined by N_K.
For classification tasks in machine learning, the cross-entropy loss $\mathcal{L}_{CE}$ is popular. It is created in every mini-batch {b} to optimize the weights, i.e., for a single sample $\{i\}$

$$\mathcal{L}_{\text{CE}}(\boldsymbol{m}_i; \hat{y}_i) = -\sum_{j=1}^{N_{\text{clu}}} \delta(\hat{y}_i, j) \log m_i^j, \tag{1}$$

where $\delta(\hat{y}_i, j)$ is the Kronecker delta. Based on the $\mathcal{L}_{\text{CE}}$ we want to maximize the MI criterion $I(\hat{\boldsymbol{Y}}; \hat{\boldsymbol{M}})$, i.e.,

$$I(\hat{\boldsymbol{Y}}; \hat{\boldsymbol{M}}) = H(\hat{\boldsymbol{Y}}) - H(\hat{\boldsymbol{Y}}|\hat{\boldsymbol{M}}), \tag{2}$$

where $H(\hat{\boldsymbol{Y}})$ is the entropy of $\hat{\boldsymbol{Y}}$ and $H(\hat{\boldsymbol{Y}}|\hat{\boldsymbol{M}})$ represents the entropy of $\hat{\boldsymbol{Y}}$ conditioned $\hat{\boldsymbol{M}}$. Here, $\hat{\boldsymbol{M}} = \hat{m}_j$ and $\hat{\boldsymbol{Y}} = \hat{y}_k$ are the discrete random variables produced by the DNNQ and the ground-truth labels. Note that the entropy $H(\hat{\boldsymbol{Y}})$ is fixed during training since the ground-truth labels $\hat{\boldsymbol{Y}}$ are fixed, meaning that we can only minimize $H(\hat{\boldsymbol{Y}}|\hat{\boldsymbol{M}})$. Furthermore, we are only able to reduce $H(\hat{\boldsymbol{Y}}|\hat{\boldsymbol{M}})$ to certain limit until we need to increase the number of emitting labels $\hat{\boldsymbol{y}}$ by raising N_{clu} to further decrease the conditioned entropy. However, if we increase the number of emitting labels, we need to vary the size of the output layer of the DNNQ i.e. vary N_{clu}. For a larger output layer size we do not have suitable labels which we could use for training since we need identical dimensions for using $\mathcal{L}_{\text{CE}}$.

Due to the variable output layer size $\boldsymbol{m}$, we propose a novel training method: We begin by creating the joint probability $P_{\text{b}}(\hat{\boldsymbol{y}}, \boldsymbol{m})$ of the ground-truth labels $\hat{\boldsymbol{y}}$ and the DNNQ output $\boldsymbol{m}(\boldsymbol{x})$ taking the input data $\boldsymbol{x}$ in every mini-batch. Next, we condition on $\boldsymbol{m}$

$$P_{\text{b}}(\hat{y}_k|m_j) \approx \frac{\varepsilon + \sum_{i=1}^{N_{\text{b}}} \delta(\hat{y}_i, k) m_i^j}{\varepsilon N_{\text{clu}} + \sum_{i=1}^{N_{\text{b}}} m_i^j} \tag{3}$$
$$\forall \quad 1 \leq j \leq N_{\text{clu}} \quad \forall \quad 1 \leq k \leq N_K.$$

Here, ε is a small constant, N_{b} is the size of the mini-batch and N_{K} is the dimension of the one-hot-encoded ground-truth labels $\hat{\boldsymbol{y}}$. In order to force the DNNQ to produce a spiky output, we apply a scaled *softmax* output $\boldsymbol{m}$

$$m_j = \frac{\exp^{\boldsymbol{a}_j T_{\text{sca}}}}{\sum_{l=1}^{N_{\text{clu}}} \exp^{\boldsymbol{a}_l T_{\text{sca}}}} \quad \forall \quad 1 \leq j \leq N_{\text{clu}} \tag{4}$$

where T_{sca} represents a variable for scaling and $\boldsymbol{a}_j$ are the activations of the layer before. By applying a scaling layer we are able to model the *argmax* operation in a way that the DNNQ acts as a VQ but has still valid gradients which enables us to perform a training.

The theory above contains similarities to [11]. However, we simplify the process of creating $P(\hat{\boldsymbol{y}}|\boldsymbol{m})$ by only taking the data of the mini-batches to create $P_{\text{b}}(\hat{\boldsymbol{y}}|\boldsymbol{m}) \approx P(\hat{\boldsymbol{y}}|\boldsymbol{m})$. We achieve this by ensuring that the mini-batch size is large enough. With $P_{\text{b}}(\hat{\boldsymbol{y}}|\boldsymbol{m})$ and the output $\boldsymbol{m} = P_{\text{b}}(\boldsymbol{m})$ of the DNNQ we can marginalize out $\boldsymbol{m}$ with

$$\boldsymbol{m}_{\text{b,tra}} = P_{\text{b}}(\hat{\boldsymbol{y}}|\boldsymbol{m}) P_{\text{b}}(\boldsymbol{m}), \tag{5}$$

where $\boldsymbol{m}_{\text{b,tra}}$ are the transformed outputs of dimension N_K. Finally, we can apply the transformed labels in $\mathcal{L}_{\text{CE}}(\boldsymbol{m}_{\text{b,tra}}; \hat{\boldsymbol{y}}_{\text{b}})$. Thus, we are able to train the DNNQ by creating suitable labels for arbitrary output layer sizes N_{clu} based on the ground-truth labels $\hat{\boldsymbol{y}}$.

Experimental Setup

We evaluate our models on the publicly available TEDLIUMv2 [13] dataset. The dataset is already split into training, test and dev set. The training set contains audio data of 207 h. We train the DNNQ in tensorflow [1] and use kaldi [9] for pre-processing the dataset and for decoding and evaluating the final model. We begin by extracting 12-dimensional MFCCs and the log-energy for every 25 ms signal frame in each utterance. Then, we apply a cepstral mean normalization for every utterance and add delta and delta-delta features. The resulting features are used to train a basic HMM-GMM with ML. Next, we cluster the monophone states to build up a triphone model. Based on the triphone model, we perform a forced alignment procedure on the entire dataset which returns state-based triphone labels. We are mapping these triphone states back to monophone states which are used for training the DNNQ.

The DNNQ consists of four fully-connected (FC) layers with 512 neurons and ReLU activations followed by a batch normalization (BN) [5] layer respectively. We regularize the weights of the DNNQ by a L2 regularization scaled with 10^{-8} to avoid exploding weights and to reduce the complexity of the system. The output layer is a fully-connected layer with sigmoid activations. It consists of N_{clu} neurons which symbolize the clusters to which a feature vector can be assigned to. In order to produce spiky outputs, we scale the output layer by a factor of T_{sca} and feeding it into the *softmax* function. The parameter T_{sca} lets us decide if we want to force smoother or spikier outputs. In our experiments we found $T_{\text{sca}} = 15$ as suitable value. For the optimization of the DNNQ we apply the Adam optimizer [7] with a learning rate of 0.01.

The frame-wise CE training is executed as mentioned in the theoretic part. We begin by sampling $P_{\text{b}}(\hat{\boldsymbol{y}}|\boldsymbol{m})$ in each mini-batch and create $P_{\text{b}}(\hat{\boldsymbol{y}}|\boldsymbol{m})$ with $\varepsilon = 0.01$ using Equation 3. Note that we require a large mini-batch size for retrieving a representative statistic $P_{\text{b}}(\hat{\boldsymbol{y}}|\boldsymbol{m})$. For this reason we set a high mini-batch size to 15 000. Then, we can use the obtained statistic $P_{\text{b}}(\hat{\boldsymbol{y}}|\boldsymbol{m})$ to apply the label mapping with Equation 5. For a faster and stable training, we perform a label smoothing [14].

Results

We evaluate our approach on monophone states. Therefore, we train one HMM-GMM model on monophone and one on triphone states using the entire training set. We map the triphone states to monophone states to create accurate labels for the DNNQ. The separately HMM-GMM trained on monophone states is used for our comparison. After retrieving the ground-truth labels, we train the DNNQ on the different cluster sizes $N_{\text{clu}} \in \{300, 600, 900, 1200\}$. The DNNQ is trained for

Table 1: WERs (%) for $N_{\text{clu}} \in \{300, 600, 900, 1200\}$ taking the entire training set.

Monophone					
	DNNQ				GMM
N_{clu}	300	600	900	1200	-
dev	**41.8**	42.3	44.1	44.7	54.3
test	44.7	**43.8**	45.6	46.6	56.7

10 epochs and we halve the learning rate after every epoch. If the performance on the validation set is increasing, we save the model.
The results depicted in Table 1 demonstrate that our approach outperformed a conventional continuous HMM-GMM. The DNNQ was able to obtain a better performance on the dev set for $N_{\text{clu}} = 300$ and on the test set for $N_{\text{clu}} = 400$. Mainly, the improvement is a result of a deeper network architecture compared to [12]. Deeper layers are assisting the DNNQ to generalize in a better way. Moreover, we are applying state-of-the-art layers like batch normalization. These layers are speeding up the training and improving the generalization process. We observe that increasing N_{clu} does not correspond to a lower WER. On the dev set for $N_{\text{clu}} = 300$ and on the test set for $N_{\text{clu}} = 400$ the DNNQ achieved a final WER of 41.8 % and 43.8 %, respectively. Compared to the HMM-GMM, which achieved a WER of 54.3 % on the dev and 56.7 % on the test set, the DNNQ relatively decreases the WER by 23.0 % and 22.8 %. It seems that the process of discretization is not hurting the performance as one can observe in Table 1.

Conclusion

In our work, we returned to discrete ASR by applying a DNNQ. We demonstrate, even though we quantize the data, we are still able to return a smaller WER compared to a continuous model. This profound way based on sampling in the mini-batch to train the DNNQ makes it possible to use an arbitrary output layer size without loosing the flexibility to scale the architecture of the DNNQ. For future approaches, we will combine our model with a classical vanilla DNN for ASR. Since we think that our DNNQ learns to process features differently than a traditional vanilla DNN, we will focus on an ensemble approach where we combine the outputs of the DNNQ and DNN.

References

[1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al. Tensorflow: a system for large-scale machine learning. In *OSDI*, vol. 16, pp. 265–283, 2016.

[2] H. A. Bourlard and N. Morgan. *Connectionist speech recognition: a hybrid approach*, vol. 247. Springer Science & Business Media, 2012.

[3] A. Graves and N. Jaitly. Towards end-to-end speech recognition with recurrent neural networks. In *International Conference on Machine Learning*, pp. 1764–1772, 2014.

[4] G. Hinton, L. Deng, D. Yu, G. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, B. Kingsbury, et al. Deep neural networks for acoustic modeling in speech recognition. *IEEE Signal processing magazine*, 29, 2012.

[5] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[6] N. Kanda, Y. Fujita, and K. Nagamatsu. Lattice-free state-level minimum bayes risk training of acoustic models. In *Proc. INTERSPEECH*, 2018.

[7] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[8] D. B. Paul and J. M. Baker. The design for the wall street journal-based csr corpus. In *Proceedings of the workshop on Speech and Natural Language*, pp. 357–362. Association for Computational Linguistics, 1992.

[9] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz, et al. The kaldi speech recognition toolkit. In *IEEE 2011 workshop on automatic speech recognition and understanding*, number EPFL-CONF-192584. IEEE Signal Processing Society, 2011.

[10] D. Povey, V. Peddinti, D. Galvez, P. Ghahremani, V. Manohar, X. Na, Y. Wang, and S. Khudanpur. Purely sequence-trained neural networks for asr based on lattice-free mmi. In *Interspeech*, pp. 2751–2755, 2016.

[11] G. Rigoll, C. Neukirchen, and J. Rottland. A new hybrid system based on mmi-neural networks for the rm speech recognition task. In *1996 IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 2, pp. 865–868. IEEE, 1996.

[12] J. Rottland, C. Neukirchen, D. Willett, and G. Rigoll. Large vocabulary speech recognition with context dependent mmi-connectionist/hmm systems using the wsj database. In *Fifth European Conference on Speech Communication and Technology*, 1997.

[13] A. Rousseau, P. Deléglise, and Y. Esteve. Enhancing the ted-lium corpus with selected data for language modeling and more ted talks. In *LREC*, pp. 3935–3939, 2014.

[14] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2818–2826, 2016.