# PQ-VAE: Efficient Recommendation Using Quantized Embeddings

**Jan Van Balen**
Apple, Inc.
London, UK
jvanbalen@apple.com

**Mark Levy**
Apple, Inc.
London, UK
mark_levy@apple.com

## ABSTRACT

Large neural recommendation models can be a challenge to deploy at scale. For recommendation services with a large number of users, the most powerful models may require an impractical amount of space to store the large dense vectors encoding each of the users' tastes. Combining ideas from auto-encoder-based recommender systems, neural discrete representation learning (VQ-VAE), and product quantization (PQ), we propose PQ-VAE, a recommendation model that learns compact, discrete embeddings at only a small cost in accuracy.

## KEYWORDS

Recommender systems; auto-encoders; representation learning; product quantization.

## INTRODUCTION

Neural recommendation models typically learn one or more sets of user or item embeddings, which together tend to make up the vast majority of the network's weights. Recent work in neural recommender systems suggest that, given enough regularization, large embedding dimensions can be an effective way to achieve accurate model predictions: the state-of-the-art variational auto-encoder (VAE) in [3] has an embedding dimensions of 600; meanwhile, experiments in [4] show even better results for a model that does away with the VAE's low-rank bottleneck altogether.

Large embeddings of course, come with a large memory footprint. The uncompressed memory footprint of $N$ $D$-dimensional user or item vectors is $N \times D \times P$, where $D$ is the model's embedding

*ACM RecSys 2019 Late-breaking Results, 16th-20th September 2019, Copenhagen, Denmark*

dimension, and $P$ is the precision in bits (typically 32). With tens of millions of users, and sometimes multiple vectors per user, that could mean upward of 100G of embeddings.

In this work, we propose PQ-VAE: a neural recommendation model that uses quantization to learn compact user embeddings at only a small accuracy cost. Our approach builds on three ideas: the auto-encoder recommenders proposed in [3], neural discrete representation learning [5], and product quantization [1].

## RELATED WORK

Experiments in [3] have shown how *denoising auto-encoders* (DAE) and *variational auto-encoders* (VAE) can be used for recommendation. The proposed DAE model is essentially a simple fully-connected neural network with a few hidden layers. Each input example is a user's history of item interactions, as a sparse $M$-dimensional vector of counts. Its objective is to reconstruct these inputs after noise is added in the form of dropout. The $M$ reconstructed values are the recommender's item scores.

We combine this model with discrete representations. VQ-VAE, the model proposed in [5], learns discrete representations of audio and image data, by trying to reconstruct its input from an intermediate representation to which vector quantization is applied. By reducing an image to a small number of integer codes, VQ-VAE generally makes for a good tool for data compression.

The third innovation on which we built PQ-VAE is *product quantization* [1]. This is a technique that is used primarily for approximate nearest neighbor search. The idea is that, by applying piece-wise $k$-means vector quantization to the candidate vectors, only $O(k)$ distance computations need to be performed per query. The resulting distances can be relatively cheaply combined to obtain the approximate distance to each of the neighbors. To the best of our knowledge, the ideas in [5] and [1] have been previously only been combined in the context of image retrieval [6].

## PQ-VAE

PQ-VAE, like the state-of-the-art DAE recommender from [3], is a neural network that takes a user's item interaction counts as inputs and produces item scores on the output side. The model's objective during training is to reconstruct its inputs. The network itself consists of one or more fully-connected layers with tanh activations, as shown in the top half of Figure 1. Unlike the DAE, we apply quantization to the activations of the last hidden layer, similarly to how VQ-VAE learns discrete representations. This is equivalent to the way vectors are quantized in product quantization.

Concretely, the vector of activations corresponding to each example is cut into chunks of around length 10, and each of these chunks is quantized using a learned codebook. This presents a few challenges at training time. First, the quantization operation is not differentiable, making it impossible to optimize the above model exactly via gradient descent. As in [5], we work around this by using the unquantized instead of the quantized vector during the gradient computation, as if the quantization

step simply never happened.[1] Second, we would like to learn codes efficiently as part of model training. To this end, we use *exponential moving average k-means* to learn codes in a batch-wise fashion, again following [5]; this way we avoid having to process the entire dataset on every update, as in the classic $k$-means algorithm.

In a trained PQ-VAE, the quantized representations in the last hidden layer represent the user, as illustrated in Figure 1. When these user representations are frozen for serving, we can convert them to discrete vectors, allowing us to store them with just 10–100 bytes each, 1-2 orders of magnitudes less that the the embeddings learned by a large DAE.

Finally, the PQ structure of the embeddings alternatively allows for efficient retrieval. At prediction time, the PQ structure of the embeddings allows us to swap out the model's final layer for PQ-based similarity scoring, replacing most of the multiply-adds with a lookup into precomputed scores.[2] However, we noticed that without a highly optimized PQ implementation (see e.g. [2]) it is difficult to make this approach surpass the performance of brute-force similarity search on real-world datasets.

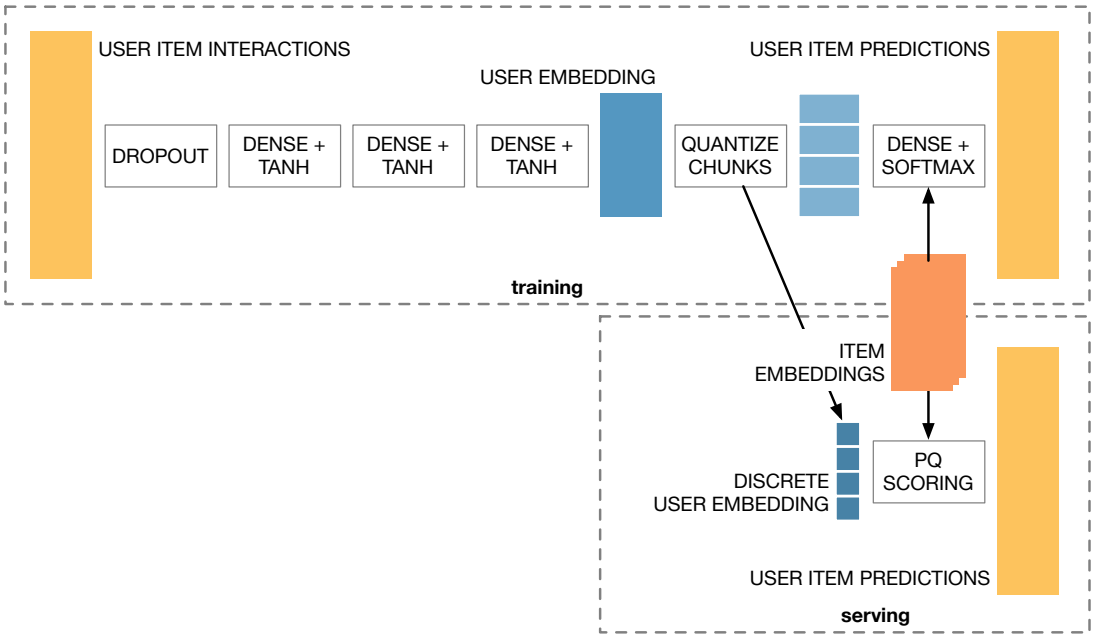


**Figure 1: Schematic of the proposed PQ-VAE model.**

[1]This is sometimes referred to as *straight-through estimation*. In a way, it is not too different from how a DAE learns: the quantization effectively produces additive (quantization) noise to which the model must learn to develop some amount of invariance.

[2]In a typical recommendation scenario, user embeddings function as query vectors, and item embeddings as candidates to be retrieved. In this paradigm, PQ-VAE quantizes queries as opposed to (more usually in PQ) the candidate vectors. Therefore, we cannot use PQ-based nearest neighbor search directly. We found that is possible to reduce the complexity of the similarity computation between a query and a collection of candidate vectors regardless of which of the two are quantized. However, efficient distance computations on quantized user embeddings require the similarity scores between the query codebook and all candidates to be precomputed, trading memory footprint for speed.

**Table 1: Memory footprint and performance on *MovieLens 20M*, and one internal dataset, for a selection of models. Model names are the architecture followed by the dimensions of the hidden layers. The DAE + PQ models are DAE's of which the embeddings were quantized after training. Dimensions with × refer to the number (40) and length (5) of chunks after piece-wise vector quantization.**

| | | *ranking performance (NDCG @ 100)* | |
|---|---|---|---|
| *Model* | *size in memory of 50M vectors* | *MovieLens 20M* | *internal 66M* |
| VAE 600-200-600 | 120G | 0.426 | |
| DAE 200 | 40G | 0.420 | 0.386 |
| PQ-VAE 200-200-40×5 | 2G | 0.412 | 0.362 |
| PQ-VAE 200-200-20×10 | 1G | 0.401 | 0.360 |
| DAE 200-200-200 + PQ 40×5 | 2G | 0.340 | |
| DAE 200-200-200 + PQ 20×10 | 1G | 0.333 | |

## EXPERIMENTS

Table 1 shows the results of an experiment on the *MovieLens 20M* dataset and, for some of the models, one internal dataset. The task is to predict a set of 20% "unseen" interactions from the other 80% as in [3]. We report NDCG@100 for 10,000 held-out users not seen during training, i.e., we evaluate so-called strong generalization. All DAE and PQ-VAE models were trained using the Adam optimizer, a selection of learning rates between $10^{-5}$ and $10^{-3}$, L2-regularization between $10^{-2}$ and $10^{-1}$ and for a maximum of 50 epochs.

The first two models, included for reference, were shown to perform well in [3]. For the remaining models, we focused on networks with a three 200-dimensional hidden layers. Next, the PQ-VAE models are shown. We compare quantization with 20 and 40 chunks of length 10 and 5, respectively. Experiments with 100 and 400-dimensional hidden layers did not show improvements over the performance shown in the table, nor did experiments with less than 20 chunks. Finally, the two rows labeled "DAE + PQ" show results for experiments in which a DAE was trained first, and product quantization was applied the model's learned embeddings after training.

Results for *MovieLens 20M* show that the difference in ranking performance due to quantization is small: less than 1% for a 10× decrease in memory footprint. Smaller models appear to come with a bigger drop in performance, e.g. 4.6% for 20×. We observe however that PQ-VAE predictions are much better than those from DAE user embeddings to which we apply product quantization after the fact:

ranking performance drops by 19%. Preliminary results on the larger internal dataset are consistent with these findings: a 40× user vector compression can be achieved with a relative performance drop of less than 5%.

## CONCLUSION

We show that is possible to learn high-quality, memory-efficient user representations by implementing product quantization inside a neural recommender. In future work we hope to further leverage the discrete nature of the learned user representations to speed up prediction.

## REFERENCES

[1] Hergé Jégou, Matthijs Douze, and Cordelia Schmid. 2011. Product Quantization for Nearest Neighbor Search. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 33, 1 (Jan 2011), 117–128. https://doi.org/10.1109/TPAMI.2010.57

[2] Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2017. Billion-scale similarity search with GPUs. *arXiv preprint arXiv:1702.08734* (2017).

[3] Dawen Liang, Rahul G. Krishnan, Matthew D. Hoffman, and Tony Jebara. 2018. Variational Autoencoders for Collaborative Filtering. In *Proceedings of the 2018 World Wide Web Conference (WWW '18)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, Switzerland, 689–698. https://doi.org/10.1145/3178876.3186150

[4] Harald Steck. 2019. Embarrassingly Shallow Autoencoders for Sparse Data. *arXiv preprint arXiv:1905.03375* (2019).

[5] Aaron van den Oord, Oriol Vinyals, and koray kavukcuoglu. 2017. Neural Discrete Representation Learning. 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.). Curran Associates, Inc., 6306–6315. http://papers.nips.cc/paper/7210-neural-discrete-representation-learning.pdf

[6] Hanwei Wu and Markus Flierl. 2018. Learning Product Codebooks using Vector Quantized Autoencoders for Image Retrieval. *arXiv preprint arXiv:1807.04629* (2018).