Vector Quantized Bayesian Neural Network Inference for Data Streams

Namuk Park, Taekyu Lee, Songkuk Kim
Yonsei University
{namuk.park, taekyulee, songkuk}@yonsei.ac.kr

Abstract

Bayesian neural networks (BNN) can estimate the uncertainty in predictions, as opposed to non-Bayesian neural networks (NNs). However, BNNs have been far less widely used than non-Bayesian NNs in practice since they need iterative NN executions to predict a result for one data, and it gives rise to prohibitive computational cost. This computational burden is a critical problem when processing data streams with low-latency. To address this problem, we propose a novel model VQ-BNN, which approximates BNN inference for data streams. In order to reduce the computational burden, VQ-BNN inference predicts NN only once and compensates the result with previously memorized predictions. To be specific, VQ-BNN inference for data streams is given by temporal exponential smoothing of recent predictions. The computational cost of this model is almost the same as that of non-Bayesian NNs. Experiments including semantic segmentation on real-world data show that this model performs significantly faster than BNNs while estimating predictive results comparable to or superior to the results of BNNs.

1 Introduction

While deterministic neural networks show high accuracy in many areas, they cannot estimate reliable uncertainty. Predictions cannot be perfect and some incorrect predictions might bring about fatal consequences in areas such as medical analysis and autonomous vehicles control. Therefore, estimating uncertainty as well as predictions is crucial for the safer application of machine learning based systems.

Bayesian neural network (BNN) uses probability distributions to model neural network (NN) weights and estimates not only predictive results but also uncertainties. This allows computer systems to make better decisions by combining prediction with uncertainty. Moreover, BNNs can achieve high performance in a variety of fields, e.g. image recognition (Kendall, Badrinarayanan, and Cipolla 2015; Kendall and Gal 2017), language modeling (Fortunato, Blundell, and Vinyals 2017), reinforcement learning (Kahn et al. 2017; Osband, Aslanides, and Cassirer 2018), meta-learning (Yoon et al. 2018; Finn, Xu, and Levine 2018), and multi-task learning (Kendall, Gal, and Cipolla 2018).

Despite these merits, BNNs have a major disadvantage that make it difficult to use as a practical tool; the predictive inference speed of BNNs is dozens of times slower than that of deterministic NNs. It has held back BNNs from wide applications. Particularly, this problem is a significant barrier for processing data streams with low-latency. This will be further elaborated below.

BNN inference. Let $p(\boldsymbol{w}|\mathcal{D})$ be a posterior probability of NN weights $\boldsymbol{w}$ with respect to training dataset $\mathcal{D}$, and $p(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{w})$ be a probability distribution parameterized by NN's result for an input data vector $\boldsymbol{x}$ and a weight $\boldsymbol{w}$. Then, the inference result of BNN is a predictive distribution:

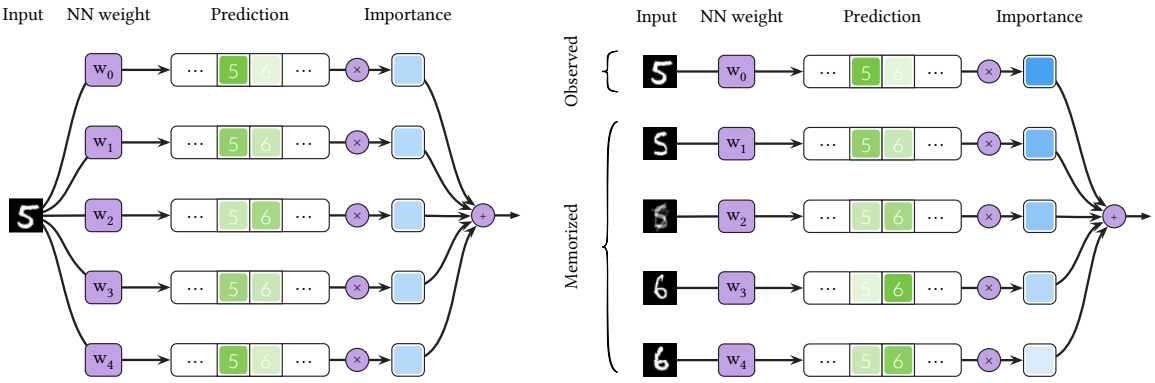
$$p(\boldsymbol{y}|\boldsymbol{x}_0, \mathcal{D}) = \int p(\boldsymbol{y}|\boldsymbol{x}_0, \boldsymbol{w})\, p(\boldsymbol{w}|\mathcal{D}) d\boldsymbol{w} \tag{1}$$

where $\boldsymbol{x}_0$ is observed input data vector and $\boldsymbol{y}$ is output vector. Since this equation cannot be solved analytically, we use the MC estimator to approximate it:

$$p(\boldsymbol{y}|\boldsymbol{x}_0, \mathcal{D}) \simeq \sum_{\boldsymbol{w}_i} \frac{1}{N_{\boldsymbol{w}}}\, p(\boldsymbol{y}|\boldsymbol{x}_0, \boldsymbol{w}_i) \tag{2}$$

where $\boldsymbol{w}_i \sim p(\boldsymbol{w}|\mathcal{D})$ and $N_{\boldsymbol{w}}$ is the number of the samples. The MC estimator implies that NN needs to be executed iteratively to calculate the predictive distribution. As many real-world data is large and practical NNs are deep, multiple NN execution cannot be fully parallelized (Kendall and Gal 2017). Consequently, the computation speed is significantly decreased. For example, according to Appendix B and (Kendall and Gal 2017), BNN requires up to fifty predictions to obtain high predictive performance in computer vision tasks, which means that the data processing speed of BNN could be fifty times lower than that of deterministic NN.

VQ-BNN inference. Suppose we have access to memorized input dataset $\{\boldsymbol{x}_0, \boldsymbol{x}_1, \cdots\}$ consisting of similar data and corresponding predictions with different weights $\{p(\boldsymbol{y}|\boldsymbol{x}_0, \boldsymbol{w}_0), p(\boldsymbol{y}|\boldsymbol{x}_1, \boldsymbol{w}_1), \cdots\}$ in the process of inference. Then, the predictive distribution of BNN can be approximated by combining these predictions. Based on this idea, we propose novel predictive distribution called *vector quantized BNN* (VQ-BNN) inference that approximates



(a) BNN Inference

(b) VQ-BNN Inference

Figure 1: Comparison of BNN inference and VQ-BNN inference. The predictive distribution of BNN inference is the sum of the probabilities $\{p(\boldsymbol{y}|\boldsymbol{x}_0, \boldsymbol{w}_i)\}$ parameterized by NN's results—e.g. for classification tasks, $p(\boldsymbol{y}|\boldsymbol{x}_0, \boldsymbol{w}_i) = \texttt{Softmax}(\text{NN}(\boldsymbol{x}_0, \boldsymbol{w}_i))$ where $\text{NN}(\cdot)$ is logit of NN—for the same observed input data and different NN weights. The predictive distribution of VQ-BNN inference is the importance weighted sum of one prediction $p(\boldsymbol{y}|\boldsymbol{x}_0, \boldsymbol{w}_0)$ for the observed data and the previously memorized predictions $\{p(\boldsymbol{y}|\boldsymbol{x}_i, \boldsymbol{w}_i)\}$ for different inputs and weights. The importance is defined as the similarity between the observed data and memorized data. VQ-BNN inference for continuously changing data streams is temporal smoothing of recent predictions with exponentially decaying importances because we assume that the similarity between the latest data and the past data decreases exponentially over time. In this figure, the inputs are toy examples.

BNN inference by using the quantized vectors to speed up calculating the predictive distribution.

In order to reduce the computational burden, VQ-BNN inference performs NN prediction for the observed input $\boldsymbol{x}_0$ only once. Then, it compensates the result with previously memorized predictions. We expect that the predictive distribution of VQ-BNN is analogous to that of BNN, since NN produces similar predictions for similar inputs. For a more sophisticated approximation, the importance of the prediction in the predictive distribution is determined based on the similarity between the observed input $\boldsymbol{x}_0$ and the prediction's input $\boldsymbol{x}_i$. To sum up, VQ-BNN inference is as follows:

$$p(\boldsymbol{y}|\boldsymbol{x}_0, \mathcal{D}) \simeq \sum_{(\boldsymbol{x}_i, \boldsymbol{w}_i)} \pi(\boldsymbol{x}_i|\boldsymbol{x}_0)\, p(\boldsymbol{y}|\boldsymbol{x}_i, \boldsymbol{w}_i) \tag{3}$$

where $\boldsymbol{w}_i \sim p(\boldsymbol{w}|\mathcal{D})$ and $\pi(\boldsymbol{x}_i|\boldsymbol{x}_0)$ is an importance of $\boldsymbol{x}_i$ with respect to $\boldsymbol{x}_0$. To estimate this predictive distribution, only $p(\boldsymbol{y}|\boldsymbol{x}_0, \boldsymbol{w}_0)$ needs to be calculated, since remainders are obtained from memorized predictions. This makes the computational performance of VQ-BNN comparable to that of deterministic NN.

VQ-BNN inference requires memorizing input vectors, similar to the observed input data $\boldsymbol{x}_0$, and corresponding predictions. To obtain them, we suppose that most of the time-varying data streams are continuously changing. Based thereupon, we prepare the proximate data sequence for VQ-BNN inference on data streams by memorizing the last few data and NN predictions. Also, we propose the importance of a previous data that decreases exponentially over time, i.e., $\pi(\boldsymbol{x}_i|\boldsymbol{x}_0) = \exp(-\Delta t_i/\tau)/\sum_i \exp(-\Delta t_i/\tau)$ where τ is hyperparameter and Δt_i is the time difference between $\boldsymbol{x}_i$ and $\boldsymbol{x}_0$. In conclusion, VQ-BNN inference for data streams is *temporal smoothing* with exponentially decaying importance of recent predictions. We summarize VQ-BNN inference in Figure 1.

Results. We evaluate VQ-BNN with computer vision tasks namely semantic segmentation and depth estimation on a variety of high-dimensional video sequence datasets. The results show that VQ-BNN has almost no degradation in computational performance compared to deterministic NNs. The predictive performance of VQ-BNN is comparable to or superior to that of BNN in various situations.

Contributions. The main contributions of this work are as follows.

- We propose vector quantized Bayesian neural network (VQ-BNN) inference as an approximation of Bayesian neural network inference to enhance the computational performance.
- We propose temporal smoothing of predictions with exponentially decaying importance by applying VQ-BNN inference to data streams.
- We empirically show that the computational performance of VQ-BNN is almost the same as that of deterministic NN and the predictive performance is comparable to or better than that of BNN on real-world data streams.

2 Vector Quantized Bayesian Neural Network Inference

Let $\mathcal{S}$ be a set of data points $\{\boldsymbol{x}_0, \cdots, \boldsymbol{x}_K\}$ generated by a source and $p(\boldsymbol{x}|\mathcal{S})$ be an estimated probability distribution of the set of data. The data points are also known as *prototypes* because they represent the probability. When the source is stationary, the estimated probability represents the observation noise.

We propose a predictive distribution for $\mathcal{S}$ as an alternative to the predictive distribution of BNN for one data point $\boldsymbol{x}_0$:

$$p(\boldsymbol{y}|\mathcal{S},\mathcal{D}) = \int p(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{w})\, p(\boldsymbol{x}|\mathcal{S})\, p(\boldsymbol{w}|\mathcal{D})\, d\boldsymbol{x} d\boldsymbol{w} \quad (4)$$

$$= \int p(\boldsymbol{y}|\boldsymbol{z})\, p(\boldsymbol{z}|\mathcal{S},\mathcal{D})\, d\boldsymbol{z} \quad (5)$$

For simplicity, we introduce $\boldsymbol{z} = (\boldsymbol{x}, \boldsymbol{w})$ and $p(\boldsymbol{z}|\mathcal{S},\mathcal{D}) = p(\boldsymbol{x}|\mathcal{S})\, p(\boldsymbol{w}|\mathcal{D})$ in this expression. We call $p(\boldsymbol{x}|\mathcal{S})$ *data uncertainty* and $p(\boldsymbol{w}|\mathcal{D})$ *model uncertainty*.

In general, Eq. (5) cannot be solved analytically. We obtain *VQ-BNN inference*, i.e.,

$$p(\boldsymbol{y}|\mathcal{S},\mathcal{D}) \simeq \sum_{\boldsymbol{z}_i} \pi(\boldsymbol{x}_i|\mathcal{S})\, p(\boldsymbol{y}|\boldsymbol{z}_i) \quad (6)$$

by approximating $p(\boldsymbol{z}|\mathcal{S},\mathcal{D})$. In this equation, we use the following quantized vector samples with importances:

$$(\boldsymbol{z}_i, \pi(\boldsymbol{z}_i|\mathcal{S},\mathcal{D})) \sim p(\boldsymbol{z}|\mathcal{S},\mathcal{D}) \quad (7)$$

where $\boldsymbol{z}_i$ is a joint of a prototype $\boldsymbol{x}_i \in \mathcal{S}$ and a random NN weight sample $\boldsymbol{w}_i \sim p(\boldsymbol{w}|\mathcal{D})$, i.e., $\boldsymbol{z}_i = (\boldsymbol{x}_i, \boldsymbol{w}_i)$. Then, $p(\boldsymbol{y}|\boldsymbol{z}_i)$ is a NN prediction for $\boldsymbol{x}_i$ with a random weight. $\pi(\boldsymbol{z}_i|\mathcal{S},\mathcal{D})$ is the importance of $\boldsymbol{z}_i$ with $\sum_{i=0}^{K} \pi(\boldsymbol{z}_i|\mathcal{S},\mathcal{D}) = 1$. In Eq. (6), we assume that $\pi(\boldsymbol{z}_i|\mathcal{S},\mathcal{D}) \simeq \pi(\boldsymbol{x}_i|\mathcal{S})$ because $\boldsymbol{w}_i$ is i.i.d.. VQ-BNN inference implies that importances and predictions are required to obtain the predictive distribution. Equation (6) is equivalent to Eq. (3) except that the set of prototypes is denoted by $\boldsymbol{x}_0$ instead of $\mathcal{S}$.

Consider the case where the prototypes are given from a noiseless stationary source, i.e., $p(\boldsymbol{x}|\mathcal{S}) = \delta(\boldsymbol{x} - \boldsymbol{x}_0)$. In this case, all prototypes and importances are the same, and all predictions $p(\boldsymbol{y}|\boldsymbol{x}_i, \boldsymbol{w}_i)$ become $p(\boldsymbol{y}|\boldsymbol{x}_0, \boldsymbol{w}_i)$. As a result, VQ-BNN inference which is Eq. (6) reduces to BNN inference which is Eq. (2). In the same manner, when $\mathcal{S}$ consists of data proximate to $\boldsymbol{x}_0$, the predictive distribution of VQ-BNN is similar to that of BNN.

We can improve the computational performance of calculating predictive distribution by using VQ-BNN inference. Without loss of generality, let $\boldsymbol{x}_0$ be the observed input data. Also, suppose that we have access to memorized prototypes $\{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_K\}$ and the corresponding predictions $\{p(\boldsymbol{y}|\boldsymbol{z}_1), \cdots, p(\boldsymbol{y}|\boldsymbol{z}_K)\}$. To calculate the predictive distribution of VQ-BNN, $\pi(\boldsymbol{x}_i|\mathcal{S})$ for all prototypes and only $p(\boldsymbol{y}|\boldsymbol{z}_0)$ for $\boldsymbol{z}_0$ are required since the remainders are obtained from the memorized predictions. Because the time to calculate importances and to aggregate memorized predictions are negligible, it takes almost the same amount of time to perform VQ-BNN inference and to perform NN prediction once.

The case of data stream. In order to use VQ-BNN as the approximation theory of BNN, we have to take the proximate dataset as prototypes and derive the importances of the prototypes. To calculate the predictive distribution for data streams, VQ-BNN exploits the fact that most real-world data streams change continuously.

Thanks to the temporal proximity of data stream, we take the latest data and the recent subsequence from the data stream as proximate prototypes as follows:

$$\mathcal{S} = \{\boldsymbol{x}_t |\, 0 \geq t \geq -K\} \quad (8)$$

where t is integer timestamp and K is non-negative number of prototypes from old data streams. In most cases, we need to derive the NN predictions for every data points from the data stream, and it is easy to memorize the sequence of NN predictions $\{p(\boldsymbol{y}|\boldsymbol{z}_t)\}$.

We define the importance in a similar way as above. Temporal proximity of data stream implies that older data contributes less to the estimated probability distribution for a data stream. Based on this idea, we propose a model in which the importance decreases exponentially over time as follows:

$$\pi(\boldsymbol{x}_t|\mathcal{S}) = \frac{\exp(-|t|/\tau)}{\sum_{t=0}^{-K} \exp(-|t|/\tau)} \quad (9)$$

where τ is given non-negative parameter. τ is determined experimentally depending on the characteristics of model and data stream. As τ approaches 0, the latest prototypes will mainly contribute to the results. As τ approaches ∞, old prototypes also will equally contribute to the results. In summary, VQ-BNN inference for data stream is temporal smoothing of recent predictions of BNN with exponentially decaying importances.

We have to mention that if the input vector is high-dimensional, VQ-BNN might need a very large number of prototypes to represent the probability of a dataset. The more prototypes are required, the more memory is required, which makes VQ-BNN inference impractical. Despite these concerns, VQ-BNN achieve high prediction performance by using a very small number of prototypes. This is because the relevant data in the data stream are concentrated in a short time interval. Appendix C shows that the semantic segmentation task on a real-world video sequence requires the recent 5 frames to obtain high predictive performance.

There are more complex algorithms that can be used to estimate prototypes from a data stream. For example, (Xu, Shen, and Zhao 2012; Frezza-Buet 2014; Ghesmoune, Lebbah, and Azzag 2016) proposed algorithms that change prototypes depending on data stream. However, these algorithms are not suitable for VQ-BNN since they are too complicated and slow. It is an important drawback because VQ-BNN is developed to achieve high computational performance to process data. The experimental results shows that this simple importance model can achieve high predictive performance.

Implementation. In order to calculate VQ-BNN inference, we have to determine the prediction $p(\boldsymbol{y}|\boldsymbol{z}_i)$ parameterized by NN. For classification tasks, we set $p(\boldsymbol{y}|\boldsymbol{z}_i)$ as a

categorical distribution parameterized by the softmax of NN logit:

$$p(\boldsymbol{y}|\mathcal{S},\mathcal{D}) \simeq \sum_{t=0}^{-K} \pi(\boldsymbol{x}_t|\mathcal{S})\,\texttt{Softmax}(\text{NN}(\boldsymbol{x}_t,\boldsymbol{w}_t)) \quad (10)$$

where $\text{NN}(\cdot)$ is logit of NN, $\boldsymbol{x}_t$ is given by Eq. (8), $\boldsymbol{w}_t \sim p(\boldsymbol{w}|\mathcal{D})$, and $\pi(\boldsymbol{x}_t|\mathcal{S})$ is given by Eq. (9). For regression tasks, $p(\boldsymbol{y}|\boldsymbol{z}_i)$ are usually modeled to have a Gaussian distribution with the mean of the NN's result:

$$p(\boldsymbol{y}|\mathcal{S},\mathcal{D}) \simeq \sum_{t=0}^{-K} \pi(\boldsymbol{x}_t|\mathcal{S})\,\mathcal{N}(\boldsymbol{y}|\text{NN}(\boldsymbol{x}_t,\boldsymbol{w}_t),\sigma^2) \quad (11)$$

where σ is a given parameter.

For stream processing, we further simplify the VQ-BNN inference with exponentially decaying importance. Let $q_{t'}(\boldsymbol{y}|\mathcal{S},\mathcal{D})$ be the predictive distribution for prototypes in $t' \geq t \geq -\infty$. Then, we rewrite $q_0(\boldsymbol{y}|\mathcal{S},\mathcal{D})$ as follows:

$$q_0(\boldsymbol{y}|\mathcal{S},\mathcal{D}) = \sum_{t=0}^{-\infty} \alpha \exp(-|t|/\tau)\, p(\boldsymbol{y}|\boldsymbol{z}_t) \quad (12)$$

$$= \alpha\, p(\boldsymbol{y}|\boldsymbol{z}_0) + (1-\alpha)\, q_{-1}(\boldsymbol{y}|\mathcal{S},\mathcal{D}) \quad (13)$$

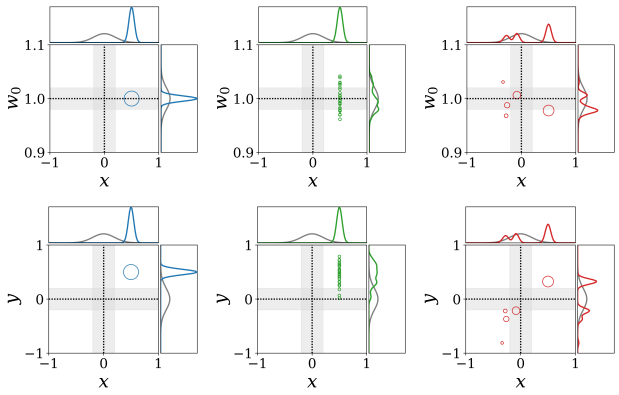
where $\alpha = \left(\sum_{t=0}^{-\infty} \exp(-|t|/\tau)\right)^{-1}$. According to this equation, the predictive distribution is the mixture of the latest prediction and the previous predictive distribution.

Training. The loss function of a BNN, such as evidence lower bound (ELBO) or negative log-likelihood (NLL), depends on a predictive distribution. Therefore, we can calculate the loss function by using VQ-BNN inference instead of by using BNN inference when training NNs.

However, *we obtain the posterior distribution in the same way as BNN training* for some practical limitations. First, VQ-BNN inference depends on the order of the input data stream. It increases the implementation complexity of training with VQ-BNN inference. Next, many training datasets do not have all the labels corresponding to the input data stream. To derive the predictive distribution for an input with a label, VQ-BNNs have to predict the result for the previous inputs without a label. It significantly increases the time required for the NN training process. Experiments show that VQ-BNN inference achieves high predictive performance even though it uses the posterior distribution by BNN training.

3 Experiments

This section evaluates the performance of VQ-BNN in three sets of experiments. The first experiment visualizes the characteristics of VQ-BNN with simple linear regression on synthetic data. The second experiment performs semantic segmentation on high-dimensional real-world video sequences. This classification task compares the performances of VQ-BNN with other baselines of deep NNs in a practical situation. The last experiment performs monocular depth estimation on high-dimensional real-world video sequences. This experiment compares the performance of VQ-BNN in a regression task.



(a) DNN (b) BNN (c) VQ-BNN

Figure 2: Visualization of VQ-BNN with simple linear regression. The top are approximated distributions of input and NN weight prototypes $p(x, w_0|\mathcal{S},\mathcal{D})$ and the bottom are approximated distributions of output prototypes with data $p(x, y|\mathcal{S},\mathcal{D})$ at $t = 0$. The sizes of the circles indicate the importances of each prototype. They also show marginal distributions $p(x|\mathcal{S})$, $p(w_0|\mathcal{D})$, and $p(y|\mathcal{S},\mathcal{D})$. In Figure 2c, data points at $x < 0$ are memorized prototypes from the past data stream. The black dotted lines and gray distributions represent true values. The error is 80% confidence interval.

Baselines. We compare the following three methods in the experiments:

- **DNN.** Let $\texttt{Softmax}(\hat{\boldsymbol{y}})$ be predictive probability of deterministic NN (DNN) where $\hat{\boldsymbol{y}}$ is NN logits for classification tasks. It is easy to implement, but it deviates from the true classification probability when the NN is deepened, broadened, and regularized well (Guo et al. 2017).
- **BNN.** BNNs use the MC estimator Eq. (2) to calculate a predictive distribution. It is difficult to analytically determine the sufficient number of NN weight samples to converge predictive distribution. Instead, we experimentally set the number of the samples to 30—i.e., BNNs with MC dropout (Gal and Ghahramani 2016) layers predict results with 30 forward passes in Section 3.2 and Section 3.3—so that the negative log-likelihood (NLL) converge. Appendix B shows the predictive performance of BNN for different numbers of forward passes.
- **VQ-BNN.** As explained in Section 2, VQ-BNN inference uses the same model and weight distribution as BNN. In all experiments, we use the same hyperparameters, $K = 5$ and $\tau = 1.25$, which implies that VQ-BNN is not overly sensitive to hyperparameter selection. See Appendix C for performance changes according to hyperparameters.

3.1 Simple Linear Regression

This experiment uses a linear regression model $y = w_0 x + w_1$ to find out the characteristics of VQ-BNN. The posteriors for BNN and VQ-BNN are given by $p(w_0|\mathcal{D}) = \mathcal{N}(1.0, 0.02^2)$ and $p(w_1|\mathcal{D}) = \mathcal{N}(0.0, 0.2^2)$. The weights for DNN are expected values of the posteriors, i.e., $w_0 = 1.0$

METHOD	BAT THR (IMG/SEC)	STR THR (IMG/SEC)	NLL	ACC (%)	ACC_{90} (%)	UNC_{90} (%)	IOU (%)	IOU_{90} (%)	FREQ_{90} (%)	ECE (%)
DNN	**27.5**	**10.5**	0.314	91.1	96.1	61.3	66.1	77.7	86.4	4.31
BNN	0.824	0.788	0.276	91.8	96.5	63.0	68.1	79.9	**86.8**	3.71
VQ-BNN	25.5	9.41	**0.253**	**92.0**	**97.4**	**72.4**	**68.6**	**83.7**	83.1	**2.24**

Table 1: Computational and predictive performance with semantic segmentation for each method.

and $w_1 = 0.0$. The distribution of time-varying input data streams is given by $p(x|t) = \mathcal{N}(x|vt, 0.1^2)$ where t is integer timestamp from -10 and $v = 0.01$.

Results. Figure 2 shows the probability distributions approximated by prototypes at $t = 0$. In this figure, the upper row displays approximated distributions of input and NN weight prototypes, i.e., $p(x, w_0|\mathcal{S}, \mathcal{D})$, and the lower row shows approximated distributions of output prototypes with data, i.e., $p(x, y|\mathcal{S}, \mathcal{D})$. The sizes of the circle indicate the importances of each prototype. It also show the three kinds of marginal distributions: the probability distribution of data $p(x|\mathcal{S}, \mathcal{D})$, the posterior distribution of NN weight $p(w_0|\mathcal{S}, \mathcal{D})$, and the predictive distribution $p(y|\mathcal{S}, \mathcal{D})$. w_1 is omitted from $\boldsymbol{w}$ in these figures, but it behaves like w_0.

To make a prediction, DNN uses a data point and a point-estimated NN weight. BNN uses a data point and a NN weight distribution, instead of point-estimated weight. VQ-BNN estimates predictive distribution by using the NN weight distribution and the set of data from the past to now that represents the probability distribution of data. In other words, VQ-BNN without distribution of $\boldsymbol{x}$ is equivalent to BNN, and BNN without distribution of $\boldsymbol{w}$ is equivalent to DNN.

In this experiment, the most recent data sample is $x = 0.4$. It is a noisy value because the expected value of x at $t = 0$ is 0. Since DNN and BNN only use the most recent data point to predict results, their predictive distributions are highly dependent on the noise of the data. As a result, an unexpected data makes the predictive distributions of DNN and BNN inaccurate. In contrast, VQ-BNN smoothen the predictive distribution by using predictions with respect to past data. Therefore, the predictive distributions of VQ-BNN are robust to the noise of data and its prediction.

Implications. These results imply that VQ-BNN may give a more accurate predictive result than BNN when the input and its prediction are noisy. Also, VQ-BNN is less likely to be overconfident than BNN since VQ-BNN uses both NN weight distribution and a probability distribution of data. For this reason, VQ-BNN might be better calibrated than BNN.

3.2 Semantic Segmentation

Semantic segmentation experiment, which is a pixel-wise classification, evaluates the computational and predictive performance of VQ-BNN with a modern deep NN in practical situation. We use the CamVid dataset (Brostow, Fauqueur, and Cipolla 2009) consisting of 360×480 pixels 30 frame-per-second (fps) video sequences of real-world day and dusk road scenes. We use U-Net (Ronneberger, Fischer, and Brox 2015) as the backbone architecture. Bayesian U-Net, similar to (Kendall, Badrinarayanan, and Cipolla 2015), contains six MC dropout layers. For more information about experimental settings, see Appendix A.1. See Appendix D.1 for experiments on a different dataset and model.

Computational performance. The throughput (BAT THR, ↑[1]) column of Table 1 shows the number of video frames processed by each model per second in batch processing. In this table, VQ-BNN processes 25.5 images per second, which is only 7% slower than DNN, and 33× faster than BNN. Likewise, the throughput column for stream processing (STR THR, ↑) shows that VQ-BNN processes 9.41 images per second in stream processing, which is only 10% slower than DNN, and 12× faster than BNN.

In conclusion, *the computational performance of VQ-BNN is comparable to that of DNN and significantly better than that of BNN.* See Appendix D.1 for more information.

Predictive performance. We use global pixel accuracy (ACC, ↑) and mean Intersection over Union (IOU, ↑) to evaluate predictive results. We also use NLL (↓), Expected Calibration Error (ECE, ↓) (Naeini, Cooper, and Hauskrecht 2015; Guo et al. 2017), and the following metrics to measure predictive uncertainty:

- **Accuracy-90 (ACC_{90}, ↑).** If NN is confident in its prediction, it must be accurate. Therefore, we select predictions with confidence higher than 90% and measure the accuracy, i.e., $p(\textbf{accurate}|\textbf{confident})$. Likewise, we measure IoU for the confident predictions (IOU_{90}, ↑).
- **Unconfidence-90 (UNC_{90}, ↑).** If the prediction of NN is incorrect, NN should not be confident in it. Therefore, we measure the probability of prediction which is not 90% confident for incorrect prediction, i.e., $p(\textbf{unconfident}|\textbf{inaccurate})$.
- **Frequency-90 (FREQ_{90}, ↑).** Even if NN derives reliable uncertainty, the model is ineffectual if it rarely predicts high-confidence results. Therefore, we measure the percentage of predictions with 90% confidence, i.e., $p(\textbf{confident})$.

The NLL to ECE columns of the Table 1 show the quantitative comparison of the predictive performance for each

[1]We use arrows to indicate which direction is better.

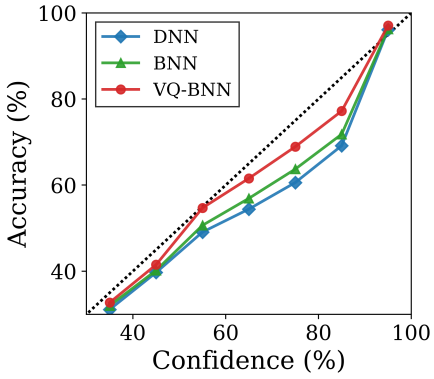


Figure 3: Reliability diagram with semantic segmentation. The black dotted line shows the accuracy we expect for each confidence.

method. This table shows that the predictive performance of BNN is better than that of DNN. Also, according to uncertainty metrics, VQ-BNN predict uncertainty better than BNN. Moreover, ACC and IOU show that VQ-BNN predicts more accurate results than BNN, which is beyond our expectations.

Figure 3 shows the reliability diagram (Niculescu-Mizil and Caruana 2005; Naeini, Cooper, and Hauskrecht 2015; Guo et al. 2017). As shown in this figure, DNN is miscalibrated; there is significant discrepancy between confidence and accuracy. In contrast, VQ-BNN is better calibrated than DNN, and surprisingly better than BNN.

According to these results, *VQ-BNN is the most appropriate method not only to distinguish uncertain predictions but also to predict accurate results*. Table 4 in Appendix D.1 shows the predictive performance of VQ-BNN in various situations. Appendix D.1 also evaluates VQ-DNN, which is the temporal smoothing of DNN's predictions. Its predictive performance is better than that of DNN, but worse than that of VQ-BNN.

Analysis. The results of Section 3.1 imply that VQ-BNN is effective in compensating for noisy predictions. For semantic segmentation task, the data that derives inaccurate results mainly correspond to the edges of objects.

VQ-BNN smoothens the predictive distribution by using past predictions. Because objects move slowly every frame, the uncertainty predicted by VQ-BNN is located at the edges of the objects. Even if VQ-BNN accidentally predicts a wrong result for the most recent frame, the past predictions compensate for this error. Figure 4 shows that predictive uncertainty of VQ-BNN is mainly located at the edge of the car, and past predictions correct the most recent incorrect predictions for the car.

We quantitatively show that VQ-BNN achieves higher predictive performance than other methods at the edges of objects. We measure predictive performance for pixels representing the object edges, and we call it *edge predictive performance*. The results of this experiment show that the edge predictive performance lags behind the predictive performance for all pixels. It implies that there are many inaccurate predictions on object edges. In addition, the difference in the edge predictive performance between VQ-BNN and BNN is greater than the difference in the predictive performance between VQ-BNN and BNN. This implies that VQ-BNN works well for edge pixels. See Appendix D.1 and Table 6 for more details on the edge predictive performance.

VQ-BNN relies on temporal consistency of data streams. Appendix D.1 evaluate the sensitivity to the temporal consistency, and the result show that the predictive performance is degraded when temporal consistency decreases.

METHOD	BAT THR (IMG/SEC)	STR THR (IMG/SEC)	NLL	RMSE (M)
DNN	**54.0**	**14.5**	1.55	0.804
BNN	1.59	1.61	1.10	0.705
VQ-BNN	50.8	13.6	**1.09**	**0.700**

Table 2: Computational and predictive performance with depth estimation for each method.

3.3 Depth Estimation

Monocular depth estimation experiment shows the performance of VQ-BNN with a deep NN in a regression task on a real-world dataset. We use the NYUDv2 dataset (Nathan Silberman and Fergus 2012), which consists of 240×320 pixels 20-30 fps video sequences from a variety of indoor scenes. As in Section 3.2, we use U-Net and Bayesian U-Net as backbone architectures. For more information about experimental settings, see Appendix A.2.

Computational performance. The throughput for batch processing (BAT THR, ↑) of Table 2 shows the number of video frames processed by each model per second. In this table, VQ-BNN processes 54.0 images per second, which is only 6% slower than DNN, and 32× faster than BNN. Similarly, the throughput for stream processing (STR THR, ↑) shows that VQ-BNN processes 13.6 images per second in stream processing, which is only 6% slower than that of DNN, and 8× faster than that of BNN. These results are consistent with the results in Section 3.2; the computational performance of VQ-BNN is significantly better than that of BNN, and is similar to that of DNN. See Appendix D.2 for more information.

Predictive performance. We use root-mean-square error (RMSE, ↓) to evaluate predictive results for depth estimation. We use NLL to evaluate predictive uncertainty.

RMSE and NLL columns of Table 2 show predictive performances for each method. This table shows that both RMSE and NLL of VQ-BNN are the lowest among those of the three methods. In conclusion, VQ-BNN is the most appropriate method for predicting accurate predictive results as well as reliable uncertainty in the regression task. See Appendix D.2 for more information on depth regression experiment.

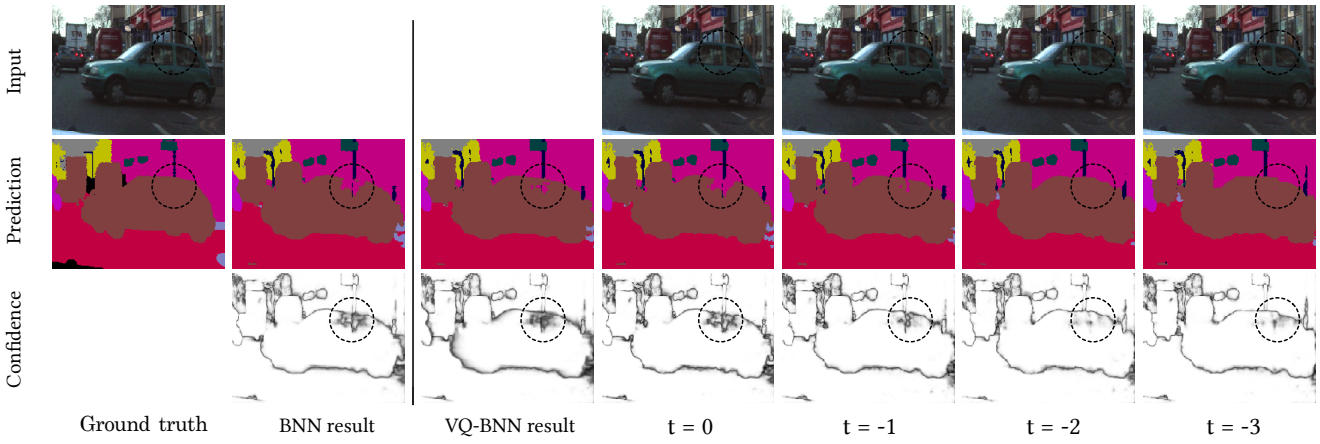


Figure 4: Qualitative Analysis of VQ-BNN with semantic segmentation. The first row is (cropped and adjusted) input image, the second row is prediction for the image, and the last row is confidence. A whiter background corresponds to higher confidence. VQ-BNN predicts the result once for the most recent image ($t = 0$). Then, it derives the predictive distribution by adding the latest prediction ($t = 0$) and past predictions ($t \in \{-1, -2, \cdots\}$), with exponentially decaying importances. Since the objects in the video sequence move slowly, the predictive distribution of VQ-BNN has high uncertainty at the edges of the objects. Even if VQ-BNN accidentally predicts a wrong result for the most recent frame, the past predictions compensate for this error. In this figure, the predictive uncertainty of VQ-BNN is mainly located at the edge of the car, and VQ-BNN derives more accurate predictive result for the car than BNN does.

4 Related Work

Several sampling-free BNNs, e.g. (Hernández-Lobato and Adams 2015; Wang, Xingjian, and Yeung 2016; Wu et al. 2018), were proposed recently and they might be a solution to the problem that BNNs require multiple NN predictions. Sampling-free BNNs approximate the posterior and the probability of layer's outputs using a simple type of parametric distribution such as Gaussian distribution or exponential family. Therefore, they predict results with one or two forward passes.

However, neural networks used in real world situations have dozens or more of layers, and sampling-free BNNs are not suitable for deep NNs. To the best of our knowledge, most sampling-free BNN have been evaluated only with a couple of layers on low-dimensional data such as UCI datasets. One of the reasons is that the Gaussian approximation in sampling-free BNNs can be inaccurate to represent real-world probabilities. Since the discrepancy between true values and approximate values accumulates in every NN layer, the error of deep sampling-free BNNs becomes not negligible. Moreover, in many cases, sampling-free BNN can not use variational inference to obtain a posterior because ELBO is not amenable. For these reasons, we mainly consider sampling-based BNNs for comparison in this paper. Recently, (Gast and Roth 2018) and (Haußmann, Hamprecht, and Kandemir 2020) applied sampling-free BNNs to LeNet. (Postels et al. 2019) applied it to SegNet; however, the neural network does not predict well-calibrated results. See Appendix E for more details on the sampling-free BNN.

(Riquelme, Tucker, and Snoek 2018) and (Kohl et al. 2018), which utilize Bayesian methods only in the last layer, predict results efficiently. However, this approach generally achieves poor predictive performance and are not robust to corrupted inputs (Ovadia et al. 2019).

A temporal smoothing has been widely used to reduce noise for accurate time-series forecasting (Pai and Lin 2005; Ediger and Akar 2007; Benvenuto et al. 2020). (Zhang 2003; Khashei and Bijari 2010; Chan et al. 2011) combined it with NN to improve accuracy for forecasting tasks on low-dimensional data streams. In this paper, we show that the temporal smoothing can significantly improve the computational performance of BNNs on high-dimensional data streams.

5 Conclusion

We present VQ-BNN inference, which is a novel approximation of BNN inference, to improve the computational performance of BNN inference for data streams. BNN inference iteratively executes NN prediction for a data, which makes it dozens of times slower. In contrast, VQ-BNN inference executes NN prediction only once for the latest data from the data stream, and compensate the result with previously memorized predictions. Specifically, VQ-BNN inference for data streams is temporal smoothing of recent predictions with exponentially decaying importance, and it is easy to implement. This method results in an order of magnitude times improvement in computational performance compared to BNN. Experiments with computer vision tasks such as semantic segmentation on various real-world datasets show that the computational performance of VQ-BNN is almost the same as that of deterministic NN, and the predictive performance is comparable to or even superior to that of BNN. Since the computational performance of deterministic NN is the best we can expect, VQ-BNN is an efficient method to estimate uncertainty.

Acknowledgements

This work was supported by Samsung Research Funding Center of Samsung Electronics under Project Number SRFC-IT1801-10.

References

Abadi, M.; Agarwal, A.; Barham, P.; Brevdo, E.; Chen, Z.; Citro, C.; Corrado, G. S.; Davis, A.; Dean, J.; Devin, M.; et al. 2016. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467* .

Badrinarayanan, V.; Kendall, A.; and Cipolla, R. 2017. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on pattern analysis and machine intelligence* 39(12): 2481–2495.

Benvenuto, D.; Giovanetti, M.; Vassallo, L.; Angeletti, S.; and Ciccozzi, M. 2020. Application of the ARIMA model on the COVID-2019 epidemic dataset. *Data in brief* 105340.

Brostow, G. J.; Fauqueur, J.; and Cipolla, R. 2009. Semantic object classes in video: A high-definition ground truth database. *Pattern Recognition Letters* 30(2): 88–97.

Chan, K. Y.; Dillon, T. S.; Singh, J.; and Chang, E. 2011. Neural-network-based models for short-term traffic flow forecasting using a hybrid exponential smoothing and Levenberg–Marquardt algorithm. *IEEE Transactions on Intelligent Transportation Systems* 13(2): 644–654.

Cordts, M.; Omran, M.; Ramos, S.; Rehfeld, T.; Enzweiler, M.; Benenson, R.; Franke, U.; Roth, S.; and Schiele, B. 2016. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 3213–3223.

Ediger, V. Ş.; and Akar, S. 2007. ARIMA forecasting of primary energy demand by fuel in Turkey. *Energy policy* 35(3): 1701–1708.

Finn, C.; Xu, K.; and Levine, S. 2018. Probabilistic model-agnostic meta-learning. In *Advances in Neural Information Processing Systems*, 9516–9527.

Fortunato, M.; Blundell, C.; and Vinyals, O. 2017. Bayesian recurrent neural networks. *arXiv preprint arXiv:1704.02798* .

Frezza-Buet, H. 2014. Online computing of non-stationary distributions velocity fields by an accuracy controlled growing neural gas. *Neural Networks* 60: 203–221.

Gal, Y.; and Ghahramani, Z. 2016. Dropout as a Bayesian Approximation - Representing Model Uncertainty in Deep Learning. *ICML* .

Gast, J.; and Roth, S. 2018. Lightweight probabilistic deep networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 3369–3378.

Ghesmoune, M.; Lebbah, M.; and Azzag, H. 2016. A new growing neural gas for clustering data streams. *Neural Networks* 78: 36–50.

Guo, C.; Pleiss, G.; Sun, Y.; and Weinberger, K. Q. 2017. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, 1321–1330. JMLR. org.

Haußmann, M.; Hamprecht, F. A.; and Kandemir, M. 2020. Sampling-free variational inference of bayesian neural networks by variance backpropagation. In *Uncertainty in Artificial Intelligence*, 563–573. PMLR.

Hernández-Lobato, J. M.; and Adams, R. P. 2015. Probabilistic Backpropagation for Scalable Learning of Bayesian Neural Networks. *NIPS* stat.ML.

Kahn, G.; Villaflor, A.; Pong, V.; Abbeel, P.; and Levine, S. 2017. Uncertainty-aware reinforcement learning for collision avoidance. *arXiv preprint arXiv:1702.01182* .

Kendall, A.; Badrinarayanan, V.; and Cipolla, R. 2015. Bayesian segnet: Model uncertainty in deep convolutional encoder-decoder architectures for scene understanding. *arXiv preprint arXiv:1511.02680* .

Kendall, A.; and Gal, Y. 2017. What uncertainties do we need in bayesian deep learning for computer vision? In *Advances in neural information processing systems*, 5574–5584.

Kendall, A.; Gal, Y.; and Cipolla, R. 2018. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 7482–7491.

Khashei, M.; and Bijari, M. 2010. An artificial neural network (p, d, q) model for timeseries forecasting. *Expert Systems with applications* 37(1): 479–489.

Kohl, S.; Romera-Paredes, B.; Meyer, C.; De Fauw, J.; Ledsam, J. R.; Maier-Hein, K.; Eslami, S.; Jimenez Rezende, D.; and Ronneberger, O. 2018. A probabilistic u-net for segmentation of ambiguous images. *Advances in neural information processing systems* 31: 6965–6975.

Naeini, M. P.; Cooper, G.; and Hauskrecht, M. 2015. Obtaining well calibrated probabilities using bayesian binning. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*.

Nathan Silberman, Derek Hoiem, P. K.; and Fergus, R. 2012. Indoor Segmentation and Support Inference from RGBD Images. In *ECCV*.

Niculescu-Mizil, A.; and Caruana, R. 2005. Predicting good probabilities with supervised learning. In *Proceedings of the 22nd international conference on Machine learning*, 625–632. ACM.

Osband, I.; Aslanides, J.; and Cassirer, A. 2018. Randomized prior functions for deep reinforcement learning. In *Advances in Neural Information Processing Systems*, 8617–8629.

Ovadia, Y.; Fertig, E.; Ren, J.; Nado, Z.; Sculley, D.; Nowozin, S.; Dillon, J.; Lakshminarayanan, B.; and Snoek, J. 2019. Can you trust your model's uncertainty? Evaluating predictive uncertainty under dataset shift. In *Advances in Neural Information Processing Systems*, 13991–14002.

Pai, P.-F.; and Lin, C.-S. 2005. A hybrid ARIMA and support vector machines model in stock price forecasting. *Omega* 33(6): 497–505.

Postels, J.; Ferroni, F.; Coskun, H.; Navab, N.; and Tombari, F. 2019. Sampling-free Epistemic Uncertainty Estimation Using Approximated Variance Propagation. In *Proceedings of the IEEE International Conference on Computer Vision*, 2931–2940.

Riquelme, C.; Tucker, G.; and Snoek, J. 2018. Deep bayesian bandits showdown: An empirical comparison of bayesian deep networks for thompson sampling. *arXiv preprint arXiv:1802.09127* .

Ronneberger, O.; Fischer, P.; and Brox, T. 2015. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, 234–241. Springer.

Simonyan, K.; and Zisserman, A. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* .

Wang, H.; Xingjian, S.; and Yeung, D.-Y. 2016. Natural-parameter networks: A class of probabilistic neural networks. In *Advances in Neural Information Processing Systems*, 118–126.

Wu, A.; Nowozin, S.; Meeds, E.; Turner, R. E.; Hernandez-Lobato, J. M.; and Gaunt, A. L. 2018. Deterministic variational inference for robust bayesian neural networks. *arXiv preprint arXiv:1810.03958* .

Xu, Y.; Shen, F.; and Zhao, J. 2012. An incremental learning vector quantization algorithm for pattern classification. *Neural Computing and Applications* 21(6): 1205–1215.

Yoon, J.; Kim, T.; Dia, O.; Kim, S.; Bengio, Y.; and Ahn, S. 2018. Bayesian model-agnostic meta-learning. In *Advances in Neural Information Processing Systems*, 7332–7342.

Zhang, G. P. 2003. Time series forecasting using a hybrid ARIMA and neural network model. *Neurocomputing* 50: 159–175.