Effects of Quantization on Neural Network Learning

by

Marcello Chiesa

Master of Science
Artificial Intelligence and Adaptive Systems
School of Informatics
University of Sussex
2021

Marcello Chiesa

mc858@sussex.ac.uk

Candidate Number: 234716

Dedication

This dissertation is dedicated to my family and friends who supported me along this Master’s degree, and to my dog Ginevra who left us at the end of this journey.

Acknowledgements

I need to thank my supervisor Dr Novi Quadrianto for his insightful feedback and Dr Ian Mackie for his continuous support during this year.

Table of Contents

List of Tables

List of Figures

Abstract

Scattered across the globe there are billions of IoT devices and smartphones, collecting data through their sensors; all this data can be analysed by means of machine learning to improve a wide variety of sectors. Nowadays, models' training and inference are performed on Cloud Computing, but there exists another possibility that offers many advantages, such as faster inference and higher security: Edge Computing. The successful application of ML on the edge requires some adaptation since edge devices have constrained hardware capabilities, thus our quest to find the effects that model compression, an essential characteristic of edge computing, have on learning, in particular on continual learning. We compared the standard multilayer perceptron with its stochastic version on multiple well-known datasets to find out whether stochastic models are better suited for Quantization Aware Training. We concluded that Bayesian Neural Networks offer an advantage when quantized, especially in the context of continual learning.

Chapter 1 Introduction

Every day millions of GB of data are generated through sensors, IoT devices and smartphones and this number is expected to grow by 61% to 175 zettabytes by 2025 (Patrizio, 2021), this immense amount of data offers a world of possibilities for a better integration between devices and the real world. One of the best ways to extract information and make decisions from this data is to feed those to a machine learning (ML) system. Due to technical limitations, data is often offloaded to remote computational infrastructure, most commonly cloud servers, where computations are performed. To be able to fully take advantage of this data, we need to develop a system that is able to operate on the device or close to it, for faster and more secure inference, and update its model using the stream of new data, for a better fit and accuracy. The ability to run machine learning models at the local level is referred to as Edge AI, or more generally Edge Computing (W. Shi, 2016) (Y. Mao, 2017). This new computing paradigm counterposes Cloud Computing (Armbrust, 2010), where the data captured by a device is sent to a remote server to be elaborated and then the results are sent back to the device. This is done because remote servers are more powerful than edge devices and are more easily maintainable but present several disadvantages. For example, the transmission of data from the edge to the cloud makes use of a lot of bandwidth and introduce a large delay between the capture of the data and the receiving of the inference. Moreover, the transmission of sensible data endangers the privacy of the information. On the other hand, Edge Computing is faster and more secure because all the computations are executed locally, on the device that captured the data or local node, thus making the inference faster and protecting the data. Moreover, edge-cloud systems turned out to be 36% less expensive than cloud-only systems and the volume of data required to be transferred was observed to be 96% less, compared to the cloud-only system (Floyer, 2019). Edge Computing presents multiple challenges that make it hard to implement in real-life applications. First, machine learning models are not optimized to be used on device hardware, differently from servers and personal computers, edge devices have limited computation resources, different architectures and small memories, which makes them unfit for training and inference. A new branch of machine learning is rapidly emerging, and it is focused on the optimization of machine learning methodologies for edge computing. In this study, we focus on two aspects central to the successful application of machine learning on the edge: continual learning and model compression. Continual learning is the ability of a system to absorb and retain new information without losing previously learned one, and model compression is the technique to reduce the dimension of a neural network. We exploit the work of (Sayna Ebrahimi, 2020) to study the effects of quantization aware training on multiple learning tasks. We compare the results obtained using the stochastic network with the corresponding pointwise network on both single learning and continual learning tasks and analyse the effects of the quantization of the model on the learning process. We aim to understand the impact that quantization has on continual learning and whether Bayesian Neural Networks are better suited for quantization.

The rest of this paper is organized as follows. First, we review the related literature in Chapter II. Then we explain the methodology used and the performed experiments in Chapter III, finally Chapter IV contains the discussion of the results and the conclusion.

Chapter 2 Literature Review

2.1 Continual Learning

As humans, we are inherently able to accumulate new knowledge, refine it and transfer it across new domains without undermining what was previously learnt. Lifelong learning is achieved through a series of neurophysiological principles that regulate the stability-plasticity balance of the brain. The stability-plasticity dilemma regards the extent to which a system must be prone to integrate and adapt to new knowledge and, importantly, how this adaptation process should be compensated by internal mechanisms that stabilize and modulate the neural activity to prevent catastrophic forgetting (German I. Parisi, 2019). The degree of plasticity changes during our lifetime; the brain is particularly plastic during the early life stage (Hensch, 1998) and becomes gradually more stable, preserving a certain degree of plasticity for its adaptation and reorganization at smaller scales (Quadrato, 2014). Lifelong learning, also called continual learning, in machine learning has been a long-standing challenge for the research community, due to the tendency of neural networks to catastrophically forget existing knowledge when learning a new data distribution. In machine learning, catastrophic forgetting occurs when the new examples to be learned differ significantly from previously observed ones, because this causes the new information to overwrite previously learned knowledge in the shared portion of the neural network (French, 1999). Many approaches have been developed to try to solve this problem. They can be divided into three categories: architectural methods, memory-based methods, and regularization methods.

Architectural methods: The architectural approach to catastrophic forgetting alter the structure of the network to reduce inference between tasks without altering the objective function, for example, the new knowledge can be stored in additional layers, nodes, or modules. The simplest form of architectural regularization is freezing certain weights in the network so that they stay exactly the same (A. S. Razavian, 2014). For instance, (Rusu, 2016) proposed to block any changes to the network trained on previous knowledge and expand the architecture by allocating novel sub-networks with fixed capacity to be trained with the new information. A more relaxed approach reduces the learning rate for layers shared with the original task while fine-tuning to avoid dramatic changes in the parameters (Jason Yosinski, 2014). For example, Dynamically Expandable Network (DEN) (Yue Wu, 2019) expands its network by selecting drifting units and retraining them on new tasks.

Memory-based methods: Augmenting the standard neural network with an external memory is a widely adopted practice (Jason Weston, 2014). (Robins, 1995) proposed a pseudorehearsal method able to minimize catastrophic forgetting without storing requiring any access to the previously learned information itself. More recently, (Lopez-Paz, 2017) proposed the Gradient Episodic Memory (GEM) model that yields a positive transfer of knowledge to previously learned tasks. The main feature of GEM to minimize catastrophic forgetting is an episodic memory used to store a subset of the observed examples from a given task.

Regularization methods: Regularization approaches alleviate catastrophic forgetting by imposing constraints on the update of the neural weights. Each weight is updated based on an importance parameter. (James Kirkpatrick, 2017) proposed to overcome catastrophic forgetting using Elastic Weight Consolidation (EWC), this approach remembers old tasks by selectively slowing down learning on the weights important for those tasks. Similar to EWC, (Han S, 2015) exploits Bayesian Neural Network's intrinsic measure of uncertainty to guide learning.

2.2 Bayesian Neural Networks

Bayesian Neural Networks are a particular type of Artificial Neural Network trained using Bayesian Interference. Classic ANNs are built using a succession of layers made of basic units, the nodes or neurons. A unit takes a weighted average of input values, applies a nonlinear "activation function," and returns a scalar value. The simplest form of ANN is the single-layer perceptron represented in Figure 1. The network architecture is fixed, and it is trained using a process of regression of the parameters θ on some training data D, the standard approach is to approximate a minimal cost point estimate $\hat{\theta}$ using a back-propagation algorithm. Point-estimate approaches are able to predict a class but are not able to express the degree of confidence and frequently result in overconfident predictions. Stochastic neural networks, or Bayesian Neural Network, are a type of ANN built by introducing stochastic components into the network to model multiple networks ϑ, each one with probability $p(\vartheta)$. Thus, they can be considered an ensemble of infinite networks. A BNN does not have single weights defining a layer, rather each weight is substituted by a statistical distribution, as shown in Figure 2. Every time an inference is performed, a new value is sampled from the distribution and use as a weight. The main goal of using a stochastic neural network architecture is to get a better idea of the uncertainty associated with the underlying processes, this is accomplished by comparing the predictions of multiple sampled models. If the different models agree the uncertainty is low. If they disagree, then it is high. Moreover, it is possible to know the uncertainty of each distribution. The goal of the training phase is to learn a probability density over the parameter space based on a set of observations D. Each parameter is initialized with a prior distribution $p(\theta)$. By applying Bayes theorem, and enforcing independence between the model parameters and the inputs, the Bayesian posterior can then be written as $p(\theta|D) = \frac{p(\theta)p(D_y|D_y,\theta)}{p(D)}$. This presents several challenges, sampling from it and computing the evidence $p(D)$ are usually intractable problems. The first methods used to train BNNs were the Markov Chain Monte Carlo algorithms. These methods directly sample the distributions and, despite providing guarantees for finding asymptotically exact samples from the target distribution, they are not suitable for large datasets and/or large models as they are bounded by speed and scalability issues. The most common approach is to use Variational Interference, which learns a variational distribution to approximate the exact posterior.

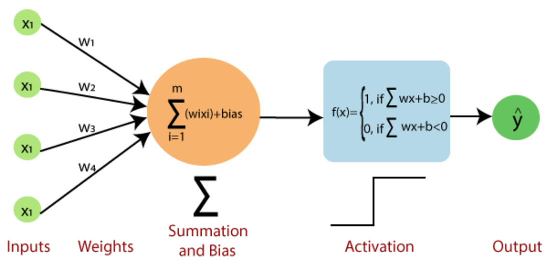

Figure 1 Perceptron representation. Inputs x are weighted and transformed by the activation function to produce the output. Source (Java Point, 2021)

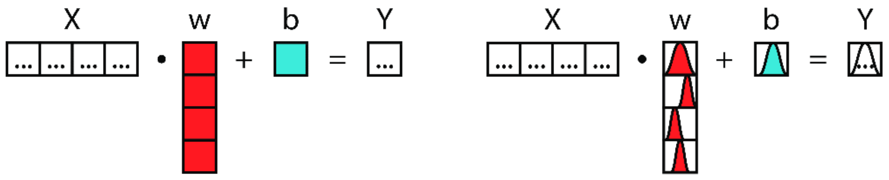

Figure 2 Moving from a point-estimate model to a stochastic model. Source: (Eric J. Ma, 2021)

2.2.1 Bayes by Backprop

In this section, we review the Bayes-by-Backprop (BBB) framework which was introduced by (Charles Blundell, 2015); to learn a probability distribution over network parameters. Let $x \in \mathbb{R}^n$ be a set of observed variables and w be a set of latent variables. A neural network, as a probabilistic model $P(y|x, w)$, given a set of training examples $D = (x, y)$ can output y which belongs to a set of classes by using the set of weight parameters w. Variational inference aims to calculate this conditional probability distribution over the latent variables by finding the closest proxy to the exact posterior by solving an optimization problem. We first assume a family of probability densities over the latent variables w parametrized by θ, i.e., $q(w|\theta)$. We then find the closest member of this family to the true conditional probability of interest $P(w|D)$ by minimizing the Kullback-Leibler (KL) divergence between q and P which is equivalent to minimizing variational free energy or maximizing the expected lower bound:

$$\theta^* = argmin_\theta \, KL(q(w|\theta)||P(w|D)) \text{ (1)}$$

The objective function can be written as:

$$\mathcal{L}_{BBB}(\theta, D) = KL(q(w|\theta)||P(w)) - Eq(w|\theta)[log\big(P(D|w)\big)] \text{ (2)}$$

Eq. 2 can be approximated using N Monte Carlo samples w_i from the variational posterior [j]:

$$\mathcal{L}_{BBB}(\theta, D) \approx \sum_{i=1}^{N} \log q(w_i|\theta) - logP(w_i) - log(P(D|w_i) \text{ (3)}$$

We assume $q(w|\theta)$ to have a Gaussian pdf with diagonal covariance and parametrized by $\theta = (\mu, \rho)$. A sample weight of the variational posterior can be obtained by sampling from a unit Gaussian and reparametrized by $w = \mu + \sigma \circ \epsilon$ where ϵ is the noise drawn from unit Gaussian and $\circ$ is a pointwise multiplication. Standard deviation is parametrized as $\sigma = log(1 + exp(\rho))$ and thus is always positive. For the prior, as suggested by [j], a scale mixture of two Gaussian pdfs are chosen which are zero-centred while having different variances of σ_1^2 and σ_2^2.

2.3 Uncertainty-based Continual Learning with Bayesian Neural Network

A common strategy to perform continual learning is to reduce forgetting by regularizing further changes in the model representation based on parameters' importance. In the uncertainty-based continual learning approach developed by (Sayna Ebrahimi, 2020), the learning rate of each parameter is function of its importance. In particular, the learning rates of σ and ρ are inversely proportional to their importance Ω. By using a Bayesian Neural Network, it is possible to directly measure the uncertainty of each parameter, i.e. its standard deviation σ:

$$\Omega \propto {}^{1}/_{\sigma}$$

$$\sigma_\rho \leftarrow {}^{\alpha_\rho}/_{\Omega_\rho}$$

$$\sigma_\sigma \leftarrow {}^{\alpha_\sigma}/_{\Omega_\sigma}$$

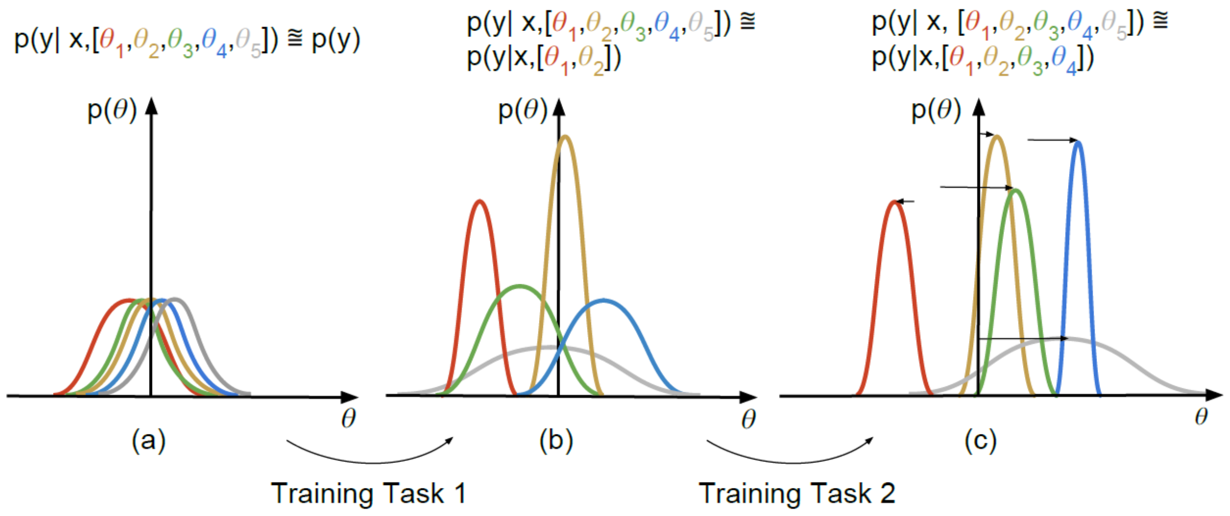

Figure 3 Illustration of the evolution of weight distributions – uncertain weights adapt more quickly – when learning two tasks using UCB. (a) weight parameter initialized by distributions initialized with mean and variance values randomly sampled from N(0; 0:1). (b) posterior distribution after learning task one; while θ1 and θ2 exhibit lower uncertainties after learning the first task, θ3, θ4, and θ5 have larger uncertainties, making them available to learn more tasks. (c) a second task is learned using higher learning rates for previously uncertain parameters (θ1, θ2, θ3, and θ4) while learning rates for θ1and θ2 are reduced. Size of the arrows indicate the magnitude of the change of the distribution mean upon gradient update. Source (Sayna Ebrahimi, 2020)

Figure 1 illustrates how the posterior distribution is modified when learning two consecutive tasks, note how the narrower distribution is not modified while the others are. Intuitively, the more a parameter distribution is uncertain, the more learnable it can be and so a larger learning step can be taken. The key benefit of UCB with learning rate as the regularizer is that it neither requires additional memory, as opposed to pruning technique nor tracking the change in parameters with respect to the previously learned task, as needed in common weight regularization methods. UCB was extensively tested on diverse object classification datasets with short and long sequences of tasks and report superior or on-par performance compared to existing approaches.

2.4 Model Compression Techniques

Compression techniques are fundamental for successfully use machine learning techniques on edge devices. There exist multiple approaches to reduce the dimensions of a model, mainly four: pruning, low-rank factorization, knowledge distillation and quantization. Deep NNs are composed of multiple layers of fully connected layers or convolutional layers, these are computationally and storage intensive (Benoit Jacob, 2017). A common compression technique that can be applied to any type of neural network is quantization, which reduces the number of bits required to store a parameter. Standard model parameters are stored as 32-bit floating-point numbers, reducing the number of bits used to represent the weights and activations can lead to a significant reduction in the number of MAC (multiply–accumulate) operations required and reduction of the size of the trained DNN. It is common to represent parameters as 8-bit integer numbers, thus gaining (Krishnamoorthi, 2021):

- 4x reduction in model size;
- 2-4x reduction in memory bandwidth;
- 2-4x faster inference due to savings in memory bandwidth and faster compute with int8 arithmetic (the exact speed up varies depending on the hardware, the runtime, and the model).

Reducing the number of bits inevitably introduce inaccuracies, which must be taken into account and can lead to lower overall accuracy. Quantization can be applied during or after the training of the neural network; generally, the floating-point NN model is developed and then quantized for efficient inference.

2.4.1 Quantization Aware Training

Quantization-aware training (QAT) (Benoit Jacob, 2017) typically results in the highest accuracy, to achieve this result the training process takes into account the loss of precision due to the lower number of bits. With QAT, all weights and activations are "fake quantized" during both the forward and backward passes of training: that is, float values are rounded to mimic int8 values, but all computations are still done with floating-point numbers (Figure 4). This procedure does not speed up training but takes into account the error introduced by quantization and thus train a better model with respect to post-training quantization approaches. We can define a fake quant as:

$$x_{out} = FakeQuant(x) = DeQuant(Quant(x_{in}))) = s \cdot (Clamp(round(^{x_{in}}/_{s}) - z) + z)$$

where s is the scale factor and z is the zero-point. The scale parameter is used to scale back the low-precision values back to the floating-point values. It is stored in full precision for better accuracy. On the other hand, the zero-point is a low precision value that represents the real value 0. The advantage of the zero-point is that we can have a wider range for integer values even for skewed tensors. The Quant operation transforms the float values of a tensor to low precision integer values. The DeQuant operation approximates the original value with some quantization error that will be optimized by the model.

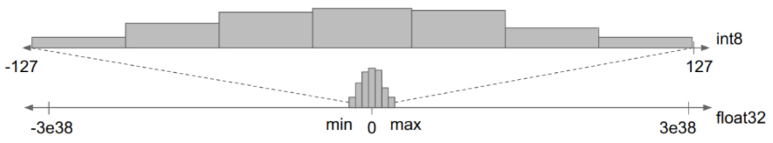

Figure 4 Small range of float32 values mapped to int8 is a lossy conversion since int8 only has 255 information channels. Source (TensorFlow Model Optimization team, 2021)

QAT scored the best results between quantization approaches, Table 1 shows the results of QAT with some of the most popular and complex neural network architectures. We can observe that the accuracy drop is negligible in this mode of quantization.

Table 1 Performance Comparison of Quantization Aware Training. Source: (TensorFlow Model Optimization team, 2021).

Model	**Floating-point Accuracy**	**QAT Accuracy**	**Post-Quantization Accuracy**
MobileNet v1	71.03%	71.06%	69.57%
MobileNet v2	70.77%	70.01%	70.2%
ResNet-50	76.3%	76.1%	75.95%

Chapter 3 Methods

3.1 Experimental setup

Datasets: This set of experiments aims to test the behaviour of Bayesian Neural Networks with respect to ordinary pointwise neural networks when quantized. We first test them on single-task learning and then on continual learning tasks. For the single-task learning, we use the Fashion MNIST dataset (Han Xiao, 2017) and the SVHN dataset (Yuval Netzer, 2011). For continual learning, we evaluate our approach in two common scenarios: 1) class-incremental learning of a single or two randomly alternating datasets, where each task covers only a subset of the classes in a dataset, and 2) continual learning of multiple datasets, where each task is a dataset. We use the Split Fashion MNIST with 5 tasks similar to (Cuong V. Nguyen, 2018) (Hanna Tseran, 2018) and permuted MNIST (Rupesh K Srivastava, 2013) for class incremental learning with similar experimental settings as used in (Hanna Tseran, 2018). We also tested the different approaches on a sequence of 5 datasets with different distributions: MNIST, SVHN, Fashion MNIST, CIFAR10 and notMNIST. No data augmentation of any kind has been used in our analysis.

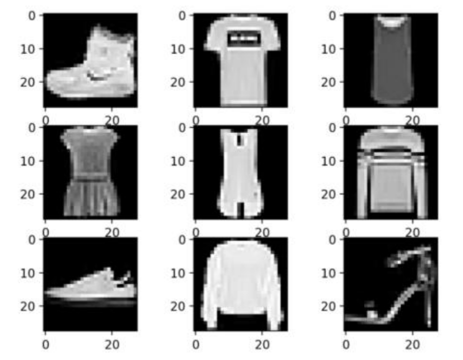

Figure 4 Samples from the Fashion MNIST database.

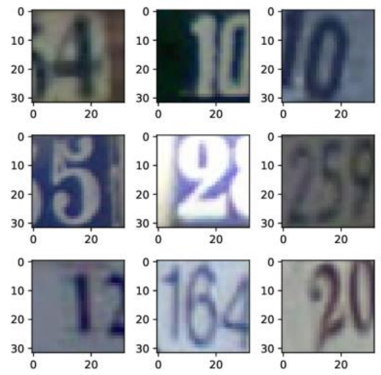

Figure 5 Samples from the Street View House Number database.

Training details: It is important to note that in all our experiments, no pre-trained model is used. We used stochastic gradient descent with a batch size of 64 and a learning rate of 0.01, decaying it by a factor of 0.3 once the loss plateaued. Dataset splits and batch shuffle are identically in all experiments. For the UCB method, we set the importance parameters $\Omega_{\rho} = 1$ and $\Omega_{\mu} = \frac{1}{\sigma}$ since it is the best configuration fund by (Sayna Ebrahimi, 2020). We used a total number of posterior samples equal to 10 to ensure robustness against random noise. Each test was repeated 3 times.

Performance measurement: To understand the effects of quantization over the two approaches, pointwise and Bayesian, we measure the accuracy, size and training time. Moreover, we measure the backward transfer in case of continual learning. For single-task learning, the accuracy is defined as $ACC = \frac{\#\, correct\ classification}{\#\, tot\ samples}$. For continual learning, let n being the total number of tasks, once all are learned, we evaluate our model on all n tasks. The accuracy is the average test classification accuracy across all tasks. To measure catastrophic forgetting, we report the backward transfer: BWT, which indicates how much learning new tasks has influenced the performance of previous ones. While BWT < 0 directly reports catastrophic forgetting, BWT > 0 indicates that learning new tasks has helped with the preceding tasks. Formally, we define BWT and ACC as follows:

$$ACC = \frac{1}{n}\sum_{i=1}^{n} R_{i,n} \qquad BWT = \frac{1}{n}\sum_{i=1}^{n} R_{i,n} - R_{i,i}$$

where $R_{i,j}$ is the test classification accuracy on task i after sequentially finishing learning the j^{th} task.

Quantization Aware Training: For both approaches, the Quant module is inserted before the first layer and the DeQuant module is inserted after the softmax activation layer. We prepared the model for the *fbgemm* backend, which corresponds to the x86 architecture.

Network architectures: Both types of neural networks were composed of two layers with different numbers of nodes depending on the task, the first layer was common to all tasks, while the second one was specific to every task. This type of network is called a multi-head network (Figure 5) and it has been used to avoid the complication of crosstalk between digits at the readout layer due to changes in the label distribution during training. Because there are more parameters in the Bayesian neural network compared to its equivalent regular neural net, we ensured fair comparison by doubling the number of nodes of the regular ones, as in (Sayna Ebrahimi, 2020).

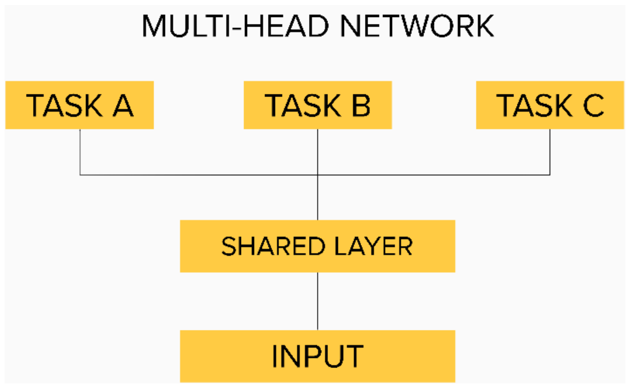

Figure 5 Multi-head network representation.

3.2 Fashion MNIST

The first dataset to be tested was the Fashion MNIST dataset, in which both the standard MLP and the Bayesian MLP had to learn 10 different classes together, each sample represents a 28x28 greyscale image. Both models were tested with and without any quantization applied. Both models contained two hidden layers consisting of 256 units each. Table 2 summarises the results. For the standard MLP, the accuracy degradation, when QAT is applied, was 3.79%, while for the Bayesian version it was 0.01%. The size reduction was x3.86 for the standard model and x1 for the Bayesian one. The cost for the lower accuracy degradation was a much higher training time.

Table 2 Results from the Fashion MNIST single-task learning. Results are divided into Training Time, Size and Accuracy. Training time is higher for BNN, but they performed better.

Approach	Architecture	Quantization	Epochs	Layers	Units	T-Time	Size	ACC
UCB	MLP	No	10	2	256	10.87	1631.91	**86.78%**
ORD	MLP	No	10	2	512	**1.00**	1629.92	84.78%
UCB	MLP	Yes	10	2	256	14.73	1732.22	86.77%
ORD	MLP	Yes	10	2	512	2.77	**421.43**	80.99%

3.3 SVHN

The second dataset to be tested was the SVHN dataset, in which both the standard MLP and the Bayesian MLP had to learn 10 different digits. Each sample represented a 32x32 RGB image and all classes were learned together. Both models were tested with and without any quantization applied. Table 3 summarises the results. For the standard MLP, the accuracy degradation, when QAT is applied, was -12.54%, which means that the quantized model performed better than the non-quantized one, while for the Bayesian version it was 0.04%. The size reduction was x3.97 for the standard model and x1 for the Bayesian one. Again, the cost for the lower accuracy degradation was a much higher training time.

Table 3 Results from the SVHN single-task learning. Results are divided into Training Time, Size and Accuracy. Training time is higher for BNN, but they performed better.

Approach	**Architecture**	**Quantization**	**Epochs**	**Layers**	**Units**	**T-Time**	**Size**	**ACC**
UCB	MLP	No	10	2	256	24.77	6317.57	**76.46%**
ORD	MLP	No	10	2	512	**2.97**	6315.65	52.34%
UCB	MLP	Yes	10	2	256	28.40	6318.02	76.41%
ORD	MLP	Yes	10	2	512	4.53	**1592.88**	64.87%

3.4 5-Split MNIST

The first database to test the class-incremental learning ability of the models was the 5-split MNIST database. For this benchmark, we split the full MNIST training data set into 5 subsets of consecutive digits. The 5 tasks correspond to learning to distinguish between two consecutive digits from 0 to 10. We used a small MLP with only two hidden layers consisting of 256 units each. Finally, we optimized our network using a minibatch size of 64 and trained for 10 epochs. Table 4 summarises the results. The accuracy degradation of the Bayesian MLP was 0.00%, and the backward transfer degradation was 0.00%. The accuracy degradation of the standard MLP was -1.53% and the backward transfer degradation was -0.01%. Both metrics were improved when the model was quantized.

Table 4 Results from the 5-Split MNIST multi-task learning. Results are divided into Training Time, Size, Accuracy and Backward transfer. Training time is higher for BNN, but they performed better.

Approach	**Architecture**	**Quantization**	**Epochs**	**Layers**	**Units**	**T-Time**	**Size**	**ACC**	**BWT**
UCB	MLP	No	10	2	256	21.2	1639.4	**98.03%**	**0.00%**
ORD	MLP	No	10	2	512	**2.1**	1632.1	87.28%	-0.12%
UCB	MLP	Yes	10	2	256	25.6	1639.4	**98.03%**	**0.00%**
ORD	MLP	Yes	10	2	512	5.3	**428.5**	88.81%	-0.11%

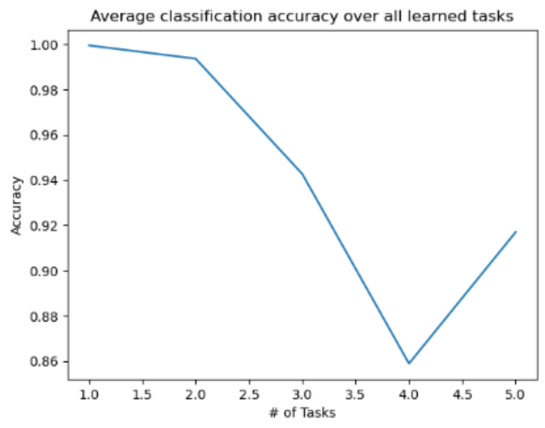

Figure 6 5-Split MNIST training example. Standard MLP trained for 10 epochs.

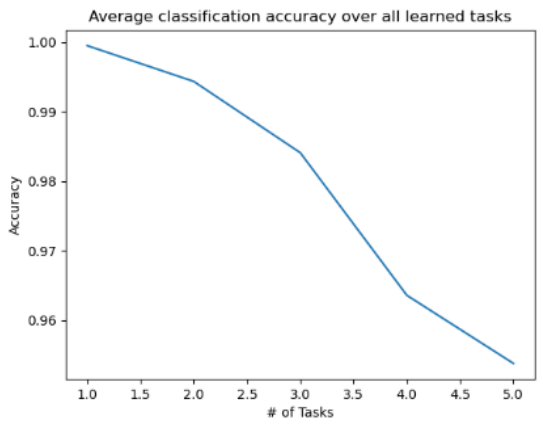

Figure 7 5-Split MNIST training example. Standard MLP trained for 10 epochs with QAT.

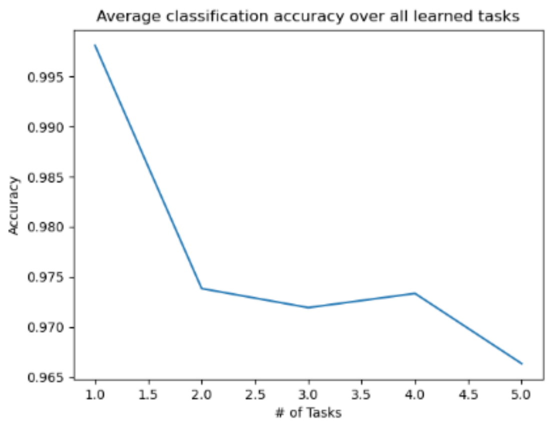

Figure 8 5-Split MNIST training example. Bayesian MLP trained for 10 epochs.

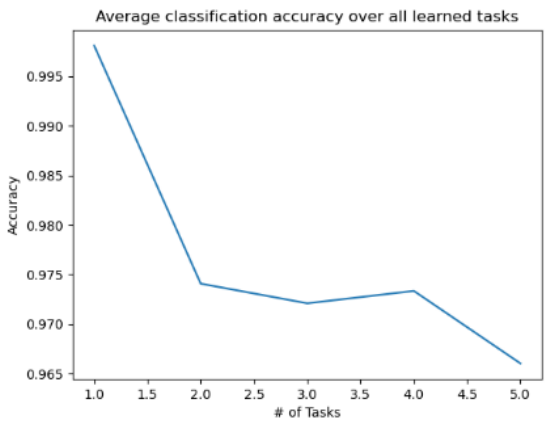

Figure 9 5-Split MNIST training example. Bayesian MLP trained for 10 epochs with QAT.

3.5 Permuted MNIST

In this benchmark, we randomly permute all MNIST pixels differently for each of the 10 tasks. The Permuted MNIST is a popular variant of the MNIST dataset to evaluate continual learning approaches in which each task is considered as a random permutation of the original MNIST pixels (Sebastian Farquhar, 2018). Following the literature, we learn a sequence of 10 random permutations and report average accuracy at the end. The mini-batch size was set to 64 and we trained for 10 epochs. Table 5 summarises the results. The accuracy degradation of the Bayesian MLP was -1.48%, and the backward transfer degradation was 0.00%. The accuracy degradation of the standard MLP was 18.86% and the backward transfer degradation was 0.16%.

Table 5 Results from the Permuted MNIST multi-task learning. Results are divided into Training Time, Size, Accuracy and Backward transfer. Training time is higher for BNN, but they performed better.

Approach	Architecture	Quantization	Epochs	Layers	Units	T-Time	Size	ACC	BWT
UCB	MLP	No	10	2	256	113	1834	90.61%	**-0.01%**
ORD	MLP	No	10	2	512	**18**	1819	67.54%	-0.28%
UCB	MLP	Yes	10	2	256	100	1832	**91.29%**	**-0.01%**
ORD	MLP	Yes	10	2	512	73	**485**	48.67%	-0.45%

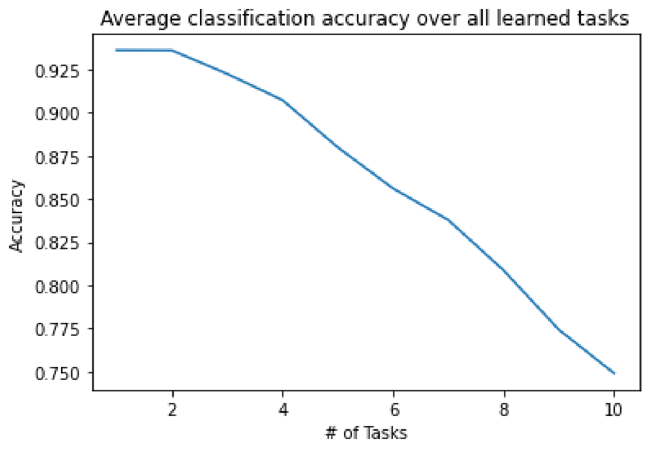

Figure 10 Permuted MNIST training example. Standard MLP trained for 10 epochs.

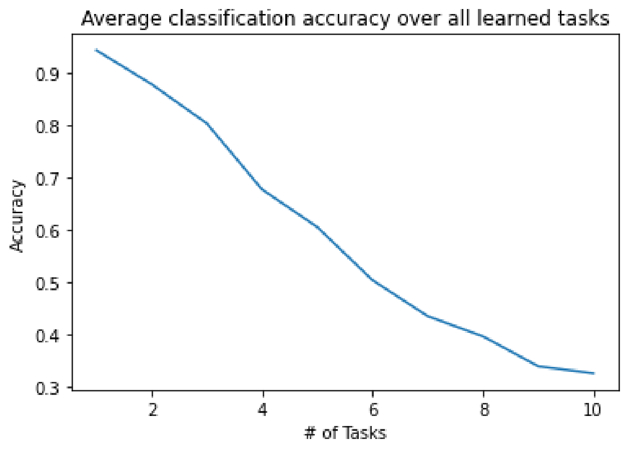

Figure 11 Permuted MNIST training example. Standard MLP trained for 10 epochs with QAT.

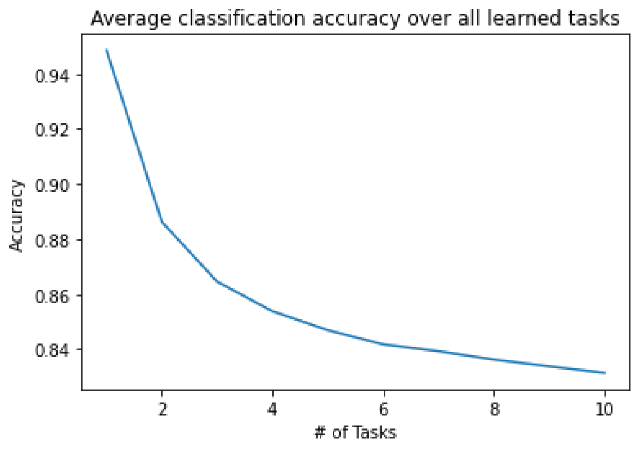

Figure 12 Permuted MNIST training example. Bayesian MLP trained for 10 epochs.

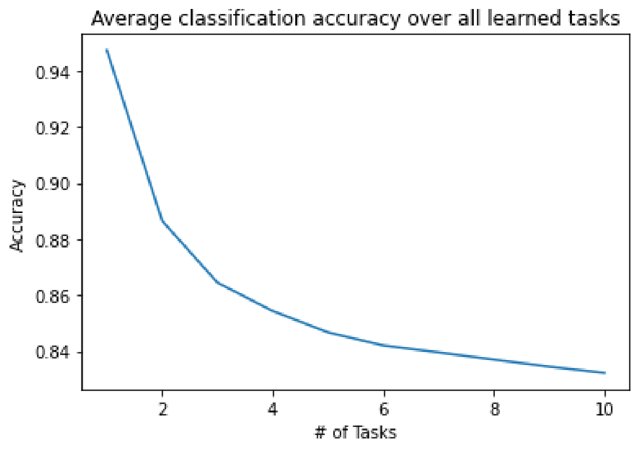

Figure 13 Permuted MNIST training example. Bayesian MLP trained for 10 epochs with QAT.

3.6 Multiple Datasets Learning

The last experiment tested the multimodal learning ability, meaning the models had to acquire and retain knowledge from 5 different datasets: MNIST, SVHN, Fashion MNIST, CIFAR 10 and notMNIST. They contain several types of data, such as numbers, letters and objects. Each dataset contained 10 classes and all samples were black and white 32 by 32 images. Table 6 summarises the results. The accuracy degradation of the Bayesian MLP was 0.07%, and the backward transfer degradation was -0.01%. The accuracy degradation of the standard MLP was $3.06\%\%$ and the backward transfer degradation was 0.09%. Both metrics were improved when the model was quantized.

Table 6 Results from the multimodal learning test. Results are divided into Training Time, Size, Accuracy and Backward transfer. Training time is higher for BNN, but they performed better.

Approach	Architecture	Quantization	Epochs	Layers	Units	T-Time	Size	ACC	BWT
UCB	MLP	No	10	2	256	131.4	4311	64.00%	-0.08%
ORD	MLP	No	10	2	512	**7.3**	2205	31.75%	-0.44%
UCB	MLP	Yes	10	2	256	125	2212	**64.07%**	**-0.07%**
ORD	MLP	Yes	10	2	512	20.3	**572**	28.69%	-0.35%

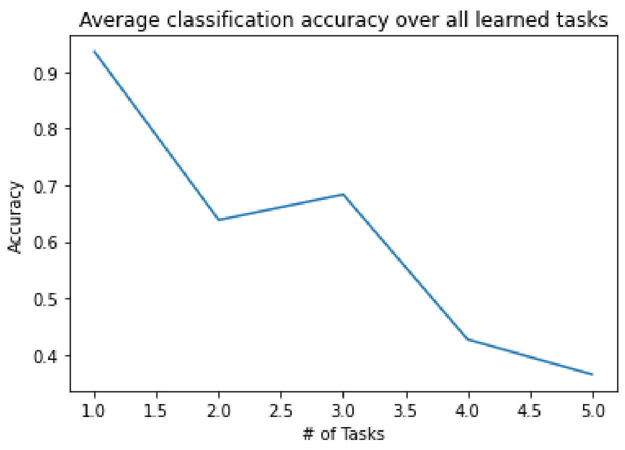

Figure 14 Multiple database training example. Standard MLP trained for 10 epochs.

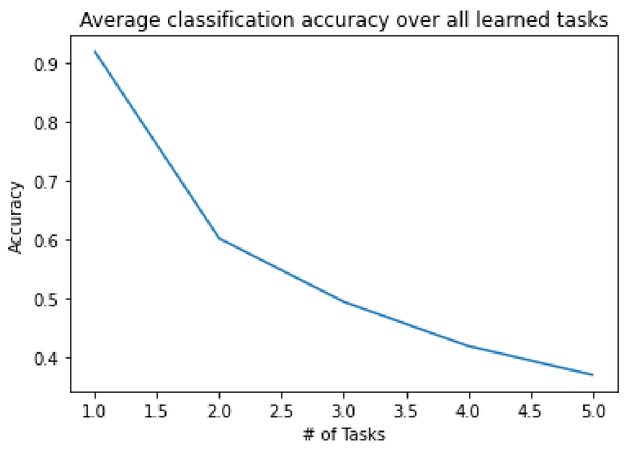

Figure 15 Multiple database training example. Standard MLP trained for 10 epochs with QAT.

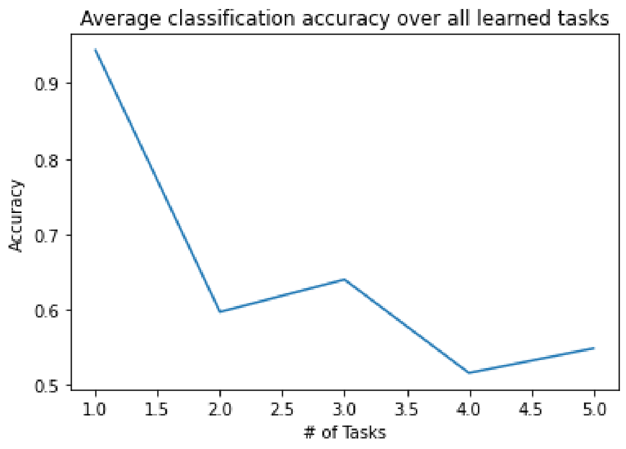

Figure 16 Multiple database training example. Bayesian MLP trained for 10 epochs.

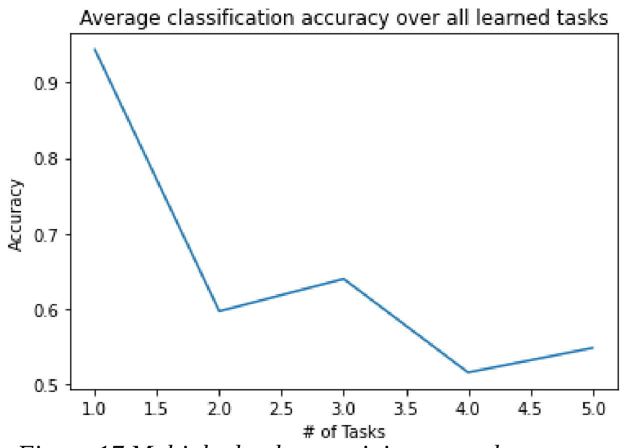

Figure 17 Multiple database training example. Bayesian MLP trained for 10 epochs with QAT.

Chapter 4 Discussion

4.1 Effects of quantization over learning

Starting our analysis with two common simple single task databases: Fashion MNIST and SVHN. From our results, the quantized stochastic models did not suffer any appreciable accuracy degradation, the quantized ordinary models' accuracy varied greatly across databases: for the Fashion MNIST database the model experienced a decrease in the accuracy, whilst for the SVHN the opposite happens, the quantized model performed better than the full precision one. We repeated the training of the ordinary model for both datasets using 100 epochs to validate our results. The full precision model scored an accuracy of 85.35% and 68.05%, on the Fashion MNIST and SVHN respectively, and the quantized model accuracy of 78.54 and 73.68%, thus confirming the previous results. The symmetric behaviours could be caused by differences in the databases, each task is composed of subgroups of classes that could present different degrees of similarity and thus react differently to the quantization learning process. The analysis continued by testing the models on three continual learning databases of increasing difficulty: 5-Split MNST, Permuted MNIST and a mix of 5 different databases. For all three tests, the stochastic model scored the highest accuracy and the lowest degradation, both models experienced a small increase in the accuracy when quantized. We can conclude that the stochastic model is better suited for continual learning and it presented a lower BWT degradation and a higher, or equal, accuracy.

Comparing our results to previous work, (Sayna Ebrahimi, 2020) presented two versions of their uncertainty-guided training, the first one used a full Bayesian Neural Network without any compression technique applied, while the second presented an additional pruning phase; network pruning is a popular technique used to obtain compact networks that maintain the accuracy of the original network with orders of magnitude of fewer parameters (Lucas Liebenwein, 2021).

Differently from Quantization-Aware Training, UCB-P prevented forgetting by saving a task-specific binary mask of important vs. unimportant parameters, similar to (Packnet, 2018). The results of the pruned networks are reported in Table 7.

Table 7 Accuracy and BWT comparison between UCB and UCB with pruning. (Sayna Ebrahimi, 2020)

	5-Split MNIST	Permuted MNIST	Alternating CIFAR10/100	Sequence of 8 tasks
UCB	99.63%	97.42%	79.44%	84.04%
UCB-P	99.32%	97.24%	77.32%	80.38%
BWT degradation	-0.72%	-0.95%	-1.17%	-1.70%

It is important to note that the aim of the compression was not to reduce the size of the network to a much lower size, like in our case, but to compress the network without dropping the accuracy below 1%, 2% and 3% of their baseline accuracy. The pruning technique was able to reduce the size of the network, but there is a considerable increase in the backward knowledge transfer, this negative effect was not present when using a quantization technique. This is possibly the result of the missing parameters which, in continual learning, holds information on the subsequent tasks, conversely in quantized models, the number of parameters remains constant, but the precision of the network is lowered. It is possible that networks with a greater number of parameters, even low precision ones, are better suited for continual learning than pruned networks, thus indicating that the number of parameters is more important than their precision.

4.2 Limitations

This project presented some known limitations. The aim of the project was never to train high accuracy models, that is why there was little fine-tuning of the networks and a low number of epochs, further fine-tuning could increase or decrease the difference between pointwise and stochastic models, but we are confident that the overall behaviour would remain the same when the training method is the same. Moreover, the experiments were performed with simple networks and the obtained results could differ when using more complex models, such as CNNs and RNNs. Our experiments only measured the training time, the size reduction, the overall accuracy and the backward knowledge transfer, but more metrics could be tracked. A further problem, we did not take into account is the effects of Non-I.I.D. (Independent and Identically Distributed) samples. Machine learning models' training is based on the hypothesis that the training and testing data should consist of samples I.I.D. However, this ideal hypothesis is fragile in real cases where we can hardly impose constraints on the testing data distribution, especially in the case of continual learning (Yue He, 2019). Finally, we point out that we were not able to compress the stochastic model since the compression of this kind of model is not yet supported by the PyTorch Quantization Aware Training library, this did not affect the analysis since the compression phase takes part at the end of the training phase when the layers between the Quant Stub and the DeQuant Stub are transformed into 8-bit integers. Looking at the standard model, we can see that the compression works and the resulting models are four times smaller than the original ones, as expected.

4.3 Further work

Further work could address the limitation presented in section 4.2 by using networks specifically designed for continual learning and quantization, this could mean using more sophisticated architecture and training framework specifically designed for edge computing, similar to (Naigang Wang, 2018) chunk-based accumulation and floating-point stochastic rounding, that enabled a reduction of bit-precision for additions down to 16 bits and 8-bit weights with any accuracy degradation. It could be possible to repeat the same type of experiments using CNNs and RNNs to study the effects of quantization on these types of networks. The use of additional databases could support our findings, in particular, the use of the NICO (Yue He, 2019) databases is a suitable choice to study Non-I.I.D distributions. This project focused on one type of model compression technique, quantization, but there exist several different techniques and each of them can affect continual learning differently, further work could aim to identify the most suitable technique in the contest of continual learning.

4.4 Conclusion

In conclusion, this work aimed to study the effects of quantization over continual learning and which model performed better between stochastic and pointwise. We compared a standard MLP to its Bayesian version, the latest was trained using the UCB's ability to exploit the uncertainty predictions to perform continual learning. Each model was tested at full precision and 8-bit precision. We tested the networks on both single-task and multi-task learning and we were able to confirm that Bayesian Neural Networks perform better than their pointwise version on continual learning tasks and demonstrate that they could be better suited for quantization aware training.

Bibliography

TensorFlow Model Optimization team. (2021, 9 27). *Optimization Toolkit - Performance with Accuracy*. Tratto da TensorFlow: https://blog.tensorflow.org/2020/04/quantization-aware-training-with-tensorflow-model-optimization-toolkit.html

A. S. Razavian, H. A. (2014). CNN Features Off-the-Shelf: An Astounding Baseline for Recognition. *IEEE Conference on Computer Vision and Pattern Recognition Workshops*.

Armbrust, M. a. (2010). A view of cloud computing. *Commun. ACM*, vol. 53, no. 4, pp. 50–58.

Benoit Jacob, S. K. (2017). Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference. arXiv:1712.05877.

Cuong V. Nguyen, Y. L. (2018). Variational continual learning. International Conference on Learning Representations.

Eric J. Ma. (2021, 7 10). Tratto da GitHub: https://ericmjl.github.io/bayesian-deep-learning-demystified/#/IntroductionSlide

Floyer, D. (2019, 8 4). *The Vital Role of Edge Computing in the Internet of Things*. Tratto da Wikibon: https://wikibon.com/the-vital-role-of-edge-computing-in-theinternet-of-things

French, R. M. (1999). Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences*, 128–135.

German I. Parisi, R. K. (2019). Continual lifelong learning with neural networks: A review. *Neural Network*, Volume 113, Pages 54-71.

Han S, P. J. (2015). Learning both weights and connections for efficient neural network. *Advances in neural information processing systems*, 1135–1143.

Han Xiao, K. R. (2017). Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. arXiv:1708.07747.

Hanna Tseran, M. E. (2018). Natural variational continual learning. Continual Learning Workshop NeurIPS.

Hensch, T. K. (1998). Local gaba circuit control of experience-dependent plasticity in developing visual cortex. *Science 282*, 1504–1508.

James Kirkpatrick, R. P.-B. (2017). Overcoming catastrophic forgetting in neural networks. *National Academy of Sciences*.

Jason Weston, S. C. (2014). Memory networks. arXiv:1410.3916.

Jason Yosinski, J. C. (2014). How transferable are features in deep neural networks? arXiv:1411.1792.

Java Point. (2021, 7 12). *Single Layer Perceptron in TensorFlow*. Tratto da Java Point: https://www.javatpoint.com/single-layer-perceptron-in-tensorflow

Krishnamoorthi, R. (2021, 7 19). *Introduction to Quantization on PyTorch*. Tratto da PyTorch: https://pytorch.org/blog/introduction-to-quantization-on-pytorch/

Lopez-Paz, D. &. (2017). Gradient episodic memory for continual learning. Long Beach.

Lucas Liebenwein, C. B. (2021). Lost in Pruning: The Effects of Pruning Neural Networks beyond Test Accuracy. *MLSys*.

Naigang Wang, J. C.-Y. (2018). Training Deep Neural Networks with 8-bit Floating. *NeurIPS*.

Packnet, A. M. (2018). Adding multiple tasks to a single network by iterative. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*

Patrizio, A. (2021, 07 01). *IDC: Expect 175 zettabytes of data worldwide by 2025*. Tratto da NetworkWorld: https://www.networkworld.com/article/3325397/idc-expect-175-zettabytes-of-data-worldwide-by-2025.html

Quadrato, G. (2014). Adult neurogenesis in brain repair: Cellular plasticity vs. cellular replacement. *Frontiers in Neuroscience*.

Robins, A. V. (1995). Catastrophic forgetting, rehearsal and pseudorehearsal. *Connection Science*, 123–146.

Rupesh K Srivastava, J. M. (2013). Compete to compute. *In Advances in neural information processing systems*, 2310–2318.

Rusu, A. A. (2016). Progressive Neural Networks. arXiv:1606.04671.

Sayna Ebrahimi, M. E. (2020). Uncertainty-guided Continual Learning with Bayesian Neural Networks. ICLR.

Sebastian Farquhar, Y. G. (2018). A Unifying Bayesian View of Continual Learning. Bayesian Deep Learning Workshop at Neural Information Processing Systems.

TensorFlow Model Optimization team. (2021, 9 27). *Quantization Aware Training with TensorFlow Model Optimization Toolkit - Performance with Accuracy*. Tratto da TensorFlow Blog: https://blog.tensorflow.org/2020/04/quantization-aware-training-with-tensorflow-model-optimization-toolkit.html

W. Shi, J. C. (2016). Edge computing: Vision and challenges. *IEEE Internet Things J.*, vol. 3, no. 5, pp. 637–646.

Y. Mao, C. Y. (2017). A survey on mobile edge computing: The communication perspective. *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2322–2358, 4th Quart.

Yue He, Z. S. (2019). Towards Non-I.I.D. Image Classification: A Dataset and Baselines. *Computer Vision and Pattern Recognition*. arXiv:1906.02899.

Yue Wu, Y. C. (2019). Large scale incremental learning. *IEEE Conference on Computer Vision and Pattern Recognition*, 374–382.

Yuval Netzer, T. W. (2011). Reading digits in natural images with unsupervised feature learning. NIPS workshop on deep learning and unsupervised feature learning.