Image Classification using Different Machine Learning Techniques

Image Classification using Different Machine Learning Techniques

Simegnew Yihunie Alaba, *Member, IEEE*
Mississippi State University
Department of Electrical and Computer Engineering

***Abstract*—Artificial Neural Networks and Convolutional Neural Networks are becoming common tools for classification and object detection tasks due to their power to learn features without prior knowledge. The networks learn the parameters, weights, and biases through training. This paper proposes a simple Neural Network and Convolutional Neural Network (CNN) to do a classification task. Additionally, the Bayesian neural network work is reproduced to compare the result to my proposed networks and used as a baseline for comparison. The MNIST dataset is used for all experiments.**

The simple neural networks and the convolutional networks adjust the parameters based on the cost function during training, whereas the Bayesian convolutional neural network updates the parameters based on the backdrop that drives a variational approximation to the true posterior. I trained the two proposed networks by varying the hyperparameters: optimizer, learning rate, regularizers, dropout, epochs, and others. My proposed work gives better classification accuracy, approximately 99 %, than the previously implemented Bayesian convolutional neural network, but it is difficult to predict how certain the prediction is, which is easy for Bayesian learning.

***Index Terms*—Neural Networks, CNN, Bayesian Neural Network, Activation function, Cost function, Accuracy, and Classification.**

I. Introduction

THE Classification task is a common task in machine learning and deep learning community. In the early days' classification, such as support vector machines (SVM), was done by using handcrafted features, but due to the growth of deep learning techniques recently, the network is used to extract the features, which makes training easy. There are different classification techniques. This work uses Bayesian Neural Network, Simple Neural network, and Convolutional Neural Network (CNN) to classify the MNIST handwritten digit classification. The Neural Network learns to perform tasks without prior knowledge by extracting features from the data. CNN is a type of neural network with higher-performing power than simple neural networks, surpassing the human-level classification accuracy [1]. Additionally, The Bayesian Neural Network, a type of neural network, classifies images by learning the uncertainty estimates of parameters.

In addition to the classification task, object detection works, such as [2], [3], [4], [5], [6], [7], [8], [9] used the convolutional network to get better performance over traditional machine learning. Various regularization and optimization techniques are used for the training. I compared the Adam Optimizer and Stochastic Gradient Descent (SGD) optimizer for the classification task. Cross Entropy loss is used as a loss function during all training. The Rectilinear linear unit (ReLu) is also applied as an activation function.

II. Related Work

Applying Neural networks, CNN, and Bayesian methods have been studied for different applications. Sham *et al.* [10] improve the MNIST dataset [11] classification error rate by using an ensemble method with heterogeneous deep network fusion based on the degree of certainty aggregation method. Agara [12] presents the classification task using CNN and support vector machine (SVM) as a classifier for the MNIST dataset and fashion-MNIST dataset [13]. Another work in [14] improves the MNIST classification by using different non-linear activation functions and comparing the results. The [15] used an adversarial neural network to do a classification task on the MNIST dataset. There are other works based on Bayesian learning. Buntine and Weigend [16] started to propose various maximum-aposteriori (MAP) schemes for neural networks. They were also the first who suggested second-order derivatives in the prior probability distribution p(w) to encourage smoothness of the resulting approximate posterior probability distribution. In subsequent work by Hinton and Van Camp [17], the first variational methods were proposed, which naturally served as a regularizer in neural networks. He also mentioned that the amount of information in weight could be controlled by adding Gaussian noise.

More recently, Graves [18] derived a variational inference scheme for neural networks, and Blundell *et al.* [19] extended this with an update for the variance that is unbiased and simpler to compute. Graves [20] derives a similar algorithm in the case of a mixture posterior probability distribution. A more scalable solution based on expectation propagation was proposed by Soudry [21] in 2014. While this method works for networks with binary weights, its extension to continuous weights is unsatisfying as it does not produce estimates of posterior variance.

The result of [1] work is reproduced for the same network setting using the MNIST dataset and developed a simple neural network and convolutional network to improve the classification accuracy further. As shown in the result and analysis section, the new network architecture results are better than the Bayesian network.

III. Background

A. Neural Network

Rosenblatt's perceptron/single neural network is a mathematical model based on how a biological neuron functions in our brain. A neuron takes a set of binary inputs, multiplies each input by the weight of a nearby neuron, and makes it 1 if the sum exceeds the threshold; otherwise, a 0 [22]. Based on the biological neuron system, an artificial neural network (ANN) was developed, as shown in Fig. 1.

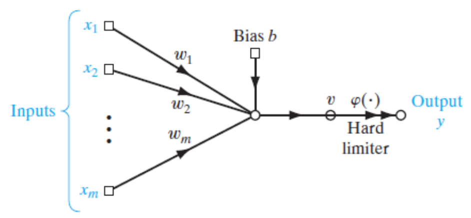

Fig. 1. A simple Neural Network that takes the weighted sum of the input x and weights w. [23]

$$y = 1\ if\ sign(\sum_{i=1}^{m} w_i x_i + b) \geq 0 \tag{1}$$

$$y = 0\ if\ sign(\sum_{i=1}^{m} w_i x_i + b) \geq 0, \tag{2}$$

where the sign acts as an activation function for perceptron.

A Neural Network consists of three layers: the input layer feeds the data to the model to learn representation, a hidden layer learns the representation, and the output layer outputs the results or predictions as shown in Fig. 2 and Fig. 3. The number of hidden layers may vary based on the application we are doing and the complexity of the task we will accomplish.

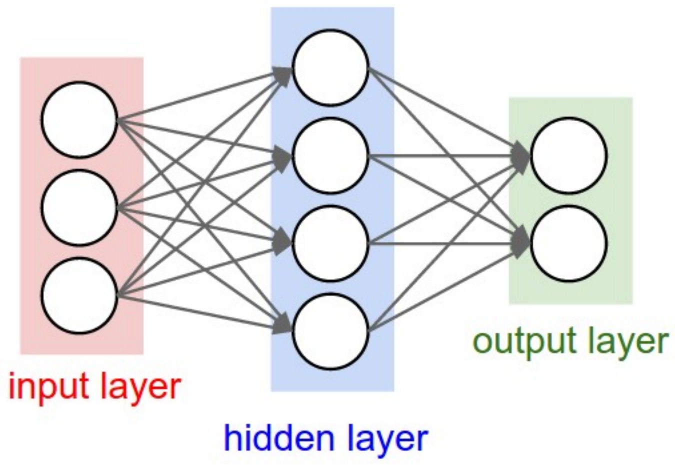

Fig. 2. Feed-forward neural network with 1 hidden layer [24]

B. Convolutional Neural Network (CNN)

Convolutional Neural Networks are similar to ordinary Neural Networks in that they are made up of neurons with

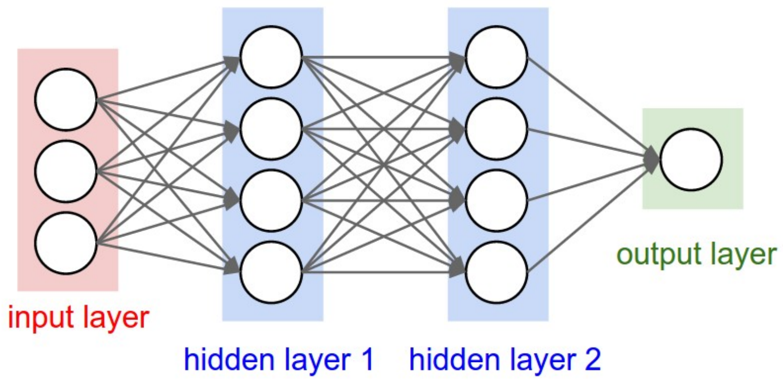

Fig. 3. Feed-forward neural network with 2 hidden layers [24]

learnable weights and biases. The ordinary neural network's hidden layer comprises a set of neurons, where each neuron is fully connected to all neurons in the previous layer, where neurons in a single layer function independently and do not share any connections. However, a CNN shares parameters: A single filter is applied across different parts of an input to produce a feature map and extract features using convolution operation. Generally, a convolutional neural network has different layers: Convolutional Layer, Pooling Layer, and Fully-Connected Layer.

The convolutional layer is important to compute a dot product between their weights and a small region connected to the input volume. Pooling layers perform a downsampling operation along the spatial dimensions. On the other hand, Fully-connected layers compute the class scores depending on the number of class categories. There is also an activation function to add non-linearity to the network.

C. Bayesian Neural Network

Bayesian learning measures the uncertainty of the parameters. Uncertainties in a network measure how certain the model is with its prediction. In Bayesian modeling, there are two main types of uncertainty one can model [1]: Aleatoric uncertainty and Epistemic uncertainty.

Aleatoric uncertainty measures the noise inherent in the observations. This type of uncertainty is present in the data collection method, like the sensor noise or motion noise which is uniform along the dataset. This cannot be reduced if more data is collected. Epistemic uncertainty, on the other hand, represents the uncertainty caused by the model. This uncertainty can be reduced given more data and is often referred to as model uncertainty.

Aleatoric uncertainty can further be categorized into homoscedastic uncertainty, which stays constant for different inputs, and heteroscedastic uncertainty. Heteroscedastic uncertainty depends on the inputs to the model, with some inputs potentially having more noisy outputs than others. Heteroscedastic uncertainty is particularly important so that model prevents outputting very confident decisions. Current work measures uncertainties by placing probability distributions over model parameters or outputs. Epistemic uncertainty is modeled by placing a prior distribution over a model's weights and then trying to capture how much these weights vary given some data. On the other hand, aleatoric uncertainty is modeled by placing a distribution over the model's output.

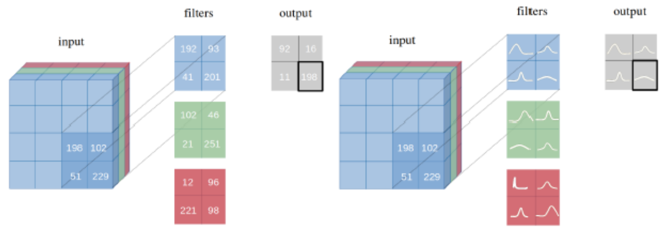

Fig. 4. Input image with exemplary pixel values, iters, and corresponding output with point-estimates (top) and probability distributions (bottom) over-weight. [25]

D. Activation Functions

Activation functions are important to fire neurons and add non-linearity to the network. There are different activation functions. The following are commonly used in the machine and deep learning communities. The simplest one is the step function (threshold-based activation) that decides the activation if it is greater than the threshold value, as shown in Fig. 5. Its output is 1 (activated) when value $>$ threshold value Or

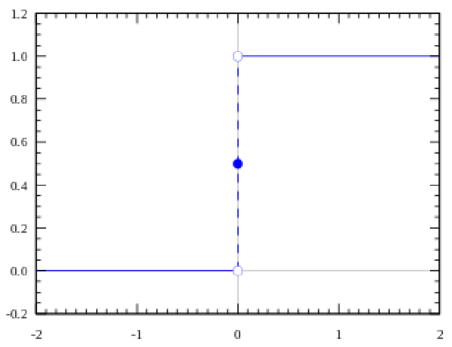

Fig. 5. Step Function [22]

outputs a 0 (not activated) otherwise. This activation is not important if we have two neurons that are activated at a time. Sigmoid activation can solve this problem, which is nonlinear activation too.

The sigmoid activation function can be expressed mathematically as:

$$y = \frac{1}{1 + e^{-z}} \tag{3}$$

, Where Z is the sum of the product of weight and input feature and bias.
The output of the sigmoid activation function ranges between 0 and 1, as shown in Fig. 6. The problem with this activation function is the vanishing gradient (the gradient becomes very small when there is a deep network), so there will not be a significant change during the training. The Tanh activation function solves this problem. It bounds the output value between -1 and 1, as shown in Fig. 7. Tanh also has a vanishing gradient problem. The other activation used

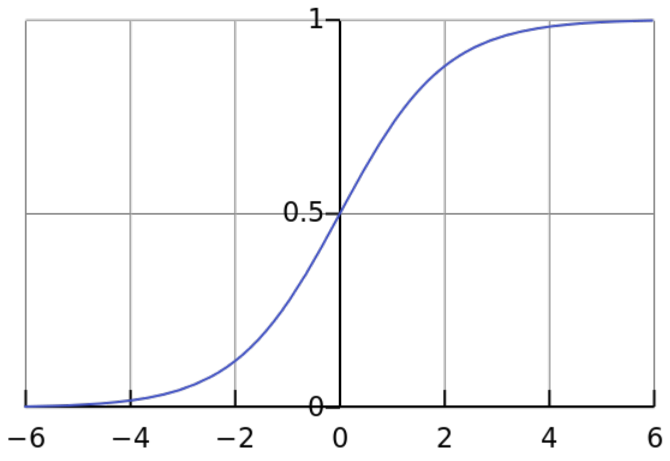

Fig. 6. Sigmoid Activation Function [22]

mostly is Rectified linear unit (ReLU). It is represented as y(x) = max(0, x) as shown in Fig. 8. Negative values of the activation output bound to 0.

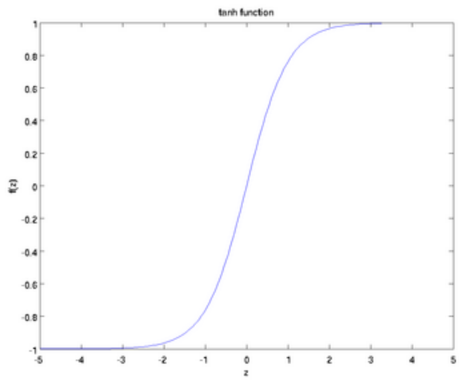

Fig. 7. Tanh activation function [22]

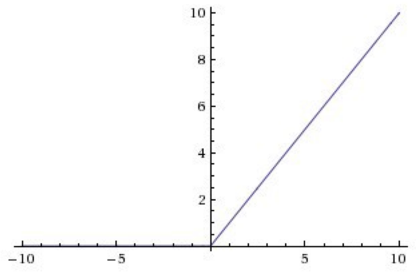

Fig. 8. ReLU activation function [22]

IV. Experimental setup and Methods

The MNIST dataset is used for all training and testing. The MNIST database [40] of handwritten digits has a training set of 60,000 examples and a test set of 10,000 examples. The digits have images of size 28 by 28 pixels. Each image is grayscaled and labeled with its corresponding class, which ranges from zero to nine. Example is shown in Fig. 9

Data preprocessing is done before the dataset load into the

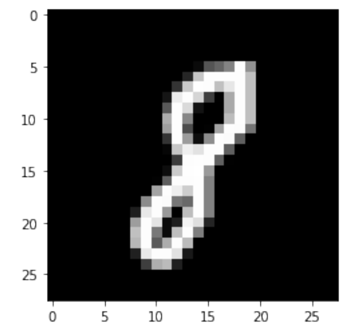

Fig. 9. Mnist dataset example

network. I normalized the dataset using mean and standard deviation. Randomly shuffling the training and test data for each iteration is done, which helps to create an independent change in the model.

For all architectures, the ReLU activation function is used. Maximum pooling is used for the CNN model as a pooling method. Batch sizes 32 and 64 are used for the neural network and CNN architectures. As in the original paper, the Bayesian network is trained with 256 batches. For a neural network, three fully connected layers are used. The input image is flattened before the first fully connected layer. Adam and SGD are both used as an optimization tools; the result is given in Table I and Table II. Trained for 20 and 50 epochs for all network architectures with a learning rate of 0.001, a momentum of 0.9, and a Cross Entropy Loss loss function. BatchNormalization avoids the vanishing gradient problem for neural networks and CNN by normalizing the inputs to a layer. I applied adaptive average pooling before the linear/classifier layer. Finally, dropout is done for CNN as an additional regularization mechanism to avoid over-fitting problems.

Tools and Hardware used for the experiment: I trained the models on Nvidia Geforce RTX 2080 super. Python programming language, Matplotlib plotting library, Jupyter Notebook as an IDE, and Pytorch deep learning framework are used for the experiment.

V. Results and Analysis

The MNIST dataset classification uses a Neural network, Convolutional neural network, and Bayesian neural network. The Bayesian Neural neural network is trained as provided by the original paper[1]. I trained the Bayesian network using LeNet and AlexNet as backbone networks. I did two experiments in different settings.

Experiment I: Adam optimizer, CrossEntropy loss, Epoch of 20, and batch size of 32 are used for this experiment. Note that the batch size for the Bayesian network is always 256, as proposed in the original work. Other parameters of the Bayesian neural networks are the same as other networks. The Bayesian and the simple neural networks have comparatively the same accuracy for this experiment. CNN gives the best accuracy of 97.5 %, whereas the Neural network and the Bayesian network, Lenet and AlexNet, give an accuracy of 96.90 %,96.16%, and 96.60% respectively, as shown in Table I.

Experiment II: For the second set, the Adam optimizer is replaced by Stochastic gradient descent (SGD), trained for 50 epochs, and batch size of 64. CNN gives the best accuracy of 99.2 % compared to Neural network accuracy of 98.0% and the Bayesian accuracy of 96.69 % and 97.4 % for LeNet and AlexNet backbone networks, respectively as shown in Table II. I trained the Bayesian network for 200 epochs as the original work, and its accuracy improved to 98.78 % using LeNet as a feature extractor. The authors reported an accuracy is 99 % , which is closest to my accuracy of 98.78%

The loss of the simple neural network and CNN shows that the cost decreases as the number of epochs increases, as shown in Fig.10. But, the neural network reduced faster than CNN. When I trained for more epochs, 50, the loss function of CNN reduced faster than the neural network, as shown in Fig.11. On the other hand, the accuracy increases as the number of epochs increases, as shown in Fig.12 and Fig.13. The Bayesian accuracy increases smoothly compared to the other two architectures for the first case; however, all increases smoothly for the second case of training. Further increasing the epochs does not increase accuracy significantly unless other hyper Parameter values are changed.

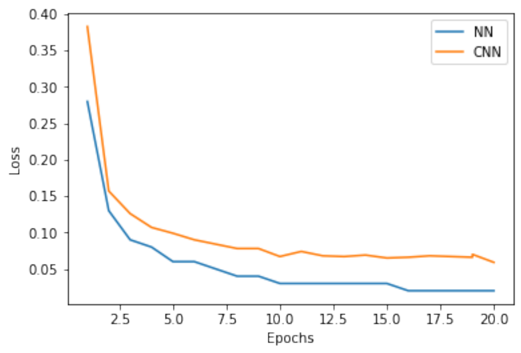

Fig. 10. Loss comparison of NN and CNN for the first experiment for each epoch

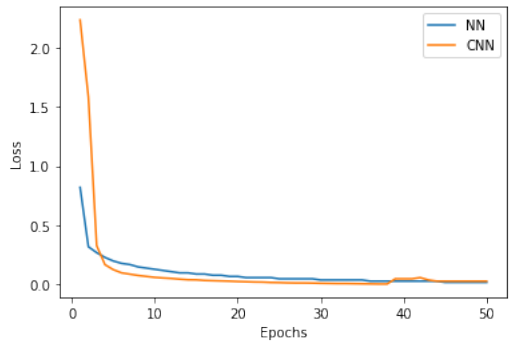

Fig. 11. Loss comparison of NN and CNN for the second experiment for each epoch

VI. Conclusions

In summary, this paper addresses the classification accuracy of different machine learning techniques for the MNIST

TABLE I
ACCURACY OF THE FIRST EXPERIMENT

Architecture	Accuracy (%)
Neural Network	96.9
CNN	97.5
Bayesian LeNet	96.16
AlexNet	96.60

TABLE II
ACCURACY OF THE SECOND EXPERIMENT

Architecture	Accuracy (%)
Neural Network	98.0
CNN	99.2
Bayesian LeNet	96.69
AlexNet	97.40

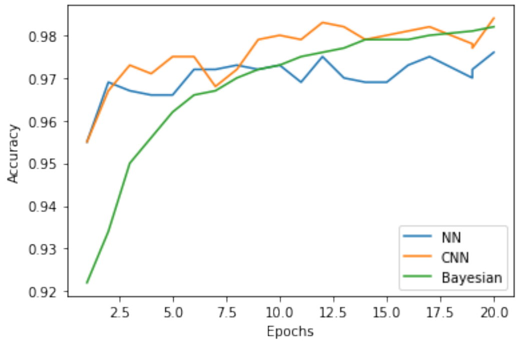

Fig. 12. Accuracy of the first experiment for each epoch

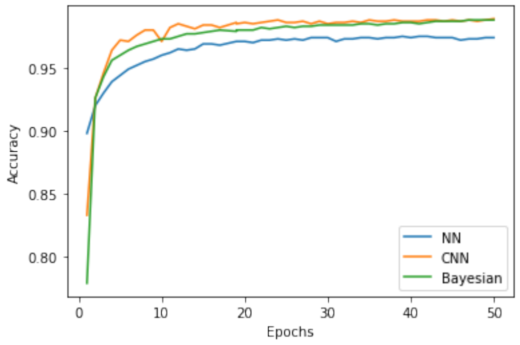

Fig. 13. Accuracy of the second experiment for each epoch

handwritten digit dataset, which can be applicable to other datasets too. All networks gave relatively good accuracy, but the convolutional neural network gave the best accuracy for the classification task, which is why it is used for many classifications and object detection works. The Bayesian neural network is important if we are looking for uncertainty estimation and can give a good result by running for more epochs.

REFERENCES

[1] K. Shridhar, F. Laumann, and M. Liwicki, "Uncertainty estimations by softplus normalization in bayesian convolutional neural networks with variational inference," *Computer Vision and Pattern Recognition*, 2019. 1, 2, 4

[2] S. Y. Alaba, M. Nabi, C. Shah, J. Prior, M. D. Campbell, F. Wallace, J. E. Ball, and R. Moorhead, "Class-aware fish species recognition using deep learning for an imbalanced dataset," *Sensors*, vol. 22, no. 21, p. 8268, 2022. 1

[3] S. Y. Alaba and F. A. Andargie, "Accelaration of preprocessors of the snort network intrusion detection system using general purpose graphics processing unit," *KICS Korea and Ethiopia Internation ICT Conference*, 2016. 1

[4] S. Alaba, A. Gurbuz, and J. Ball, "A comprehensive survey of deep learning multisensor fusion-based 3d object detection for autonomous driving: Methods, challenges, open issues, and future directions," *IEEE Preprint*, 2022. 1

[5] S. Alaba and J. Ball, "Deep learning-based image 3d object detection for autonomous driving: review," *IEEE Preprint*, 2022. 1

[6] S. Y. Alaba and J. E. Ball, "Wcnn3d: Wavelet convolutional neural network-based 3d object detection for autonomous driving," *Sensors*, vol. 22, no. 18, p. 7010, 2022. 1

[7] O. Boulais, S. Y. Alaba, J. E. Ball, M. Campbell, A. T. Iftekhar, R. Moorehead, J. Primrose, J. Prior, F. Wallace, H. Yu, *et al.*, "Seamapd21: a large-scale reef fish dataset for fine-grained categorization," *The Eighth Workshop on Fine-Grained Visual Categorization – CVPR21*, 2021. 1

[8] V. H. Oquino, T. H. Ayane, and T. B. Workie, "Design and development of automated electronic switching system for energy regulation," *Global Journals of Research in Engineering*, vol. 16, no. F7, pp. 79–86, 2016. 1

[9] S. Y. Alaba and J. E. Ball, "A survey on deep-learning-based lidar 3d object detection for autonomous driving," *Sensors*, vol. 22, no. 24, 2022. [Online]. Available: https://www.mdpi.com/1424-8220/22/24/9577 1

[10] S. Tabik, R. F. Alvear-Sandoval, M. M. Ruiz, J.-L. Sancho-Gómez, A. R. Figueiras-Vidal, and F. Herrera, "Mnist-net10: A heterogeneous deep networks fusion based on the degree of certainty to reach 0.1% error rate. ensembles overview and proposal," *Information Fusion*, 2020. 1

[11] Y. LeCun, C. Cortes, and C. J. Burges., "Mnist handwritten digit database." 2010., aT and T Labs. [Online]. Available: http://yann.lecun.com/exdb/ 1

[12] A. F. M. Agara, "An architecture combining convolutional neural network (cnn) and support vector machine (svm) for image classification," *arXiv:1712.03541v2*, 2019. 1

[13] H. Xiao, K. Rasul, and R. Vollgraf., "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms." *arXiv preprint arXiv:1708.07747*, 2017. 1

[14] D. Pedamonti, "Comparison of non-linear activation functions for deep neural networks on mnist classification task," 2018. 1

[15] L. Schott, J. Rauber, M. Bethge, and W. Brendel, "Towards the first adversarially robust neural network model on mnist," 2018. 1

[16] W. L. Buntine and A. S. Weigend, "Bayesian back-propagation. complex systems," 1991. 1

[17] G. E. Hinton and D. V. Camp, "Keeping the neural networks simple by minimizing the description length of the weights," 1993, pp. 5 –13. 1

[18] A. Graves, "Practical variational inference for neural networks," *In Advances in Neural Information Processing Systems*, pp. 2348 – 2356, 2011. 1

[19] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, "Weight uncertainty in neural networks," *arXiv preprint arXiv:1505.05424*, 2015. 1

[20] A. Graves, "Stochastic backpropagation through mixture density distributions," *arXiv preprint arXiv:1607.05690*, 2016. 1

[21] D. Soudry, I. Hubara, and R. Meir, "Expectation backpropagation: Parameter-free training of multilayer neural networks with continuous or discrete weights (with appendix)," *Advances in Neural Information Processing Systems*, pp. 963 –971, 2014. 1

[22] A. Rosebrock, *Deep Learning for Computer Vision: Starter Bundler*. PYIMAGESEARCH, 2017. 2, 3

[23] S. Haykin, *Neural Networks and Learning Machines*. Pearson Education, Inc., Upper Saddle River, New Jersey 07458, 2009. 2

[24] F.-F. Li, R. Krishna, and D. Xu, "Cs231n, convolutional neural networks for visual recognition," Apr. 2021, online, accessed 4/25/21. [Online]. Available: http://cs231n.stanford.edu/ 2

[25] K. Shridhar, F. Laumann, A. L. Maurin, M. Olsen, and M. Liwicki., "Bayesian convolutional neural networks with variational inference." *Computer Vision and Pattern Recognition*, 2018. 3