Bayesian Convolutional Neural Network with Variational Inference for Image Recognition

Yu Zhu

1. Introduction

The Simpsons is a beloved American TV series, which has been broadcasted in more than 90 countries all over the world. It is easy to notice that almost all of the characters in Simpsons have yellow skin, because Simpsons creator Matt Groening made the characters yellow to grab the attention of channel surfers. He also designed the Simpson family to have distinctive hairstyles and head-shapes. In this project, we propose Non-Bayesian Convolutional Neural Network (regularCNN) and Bayesian Convolutional Neural Network (BayesCNN), and compare the predictive performance between two models.

When dealing with image, we typically consider the feature as each the pixel, with or without transformations. However, the high dimensionality of the data can be quite challenging when we train the Artificial Neural Networks (ANNs) or some other machine learning methods. CNNs are exceptionally good at lowering the amount of parameters with convolutional layers without sacrificing model quality. And under different settings or combinations of the convolutional layers, CNNs can better extrapolate and identify the important patterns or edges in the data.

BayesCNN is a probablistic network that can learn from data with prior understandings. The weights in regular CNN are generally fixed, while in BayesCNN, we introduce probability distributions on weights in the layers such as the convolutional layers and fully-connected layers. With Variational Inference that incorporates a probability distribution over the weights, it is capable of measuring the model uncertainty. We follow the methodology of Bayes by Backprop by Blundell et al.[1].

2. Bayes by Backprop

Bayes by Backprop is a variational inference method that approximate the posterior distribution by a variational distribution $q_\theta(\boldsymbol{\omega})$ indexed by variational parameter $\theta \in \Theta$. We optimize the variational distribution via minimizing the closeness between $q_\theta(\boldsymbol{\omega})$ and $\mathrm{p}(\boldsymbol{\omega}|D)$ measured by KL divergence. Specifically, find the optimal parameter θ that

$$\theta^* = {}^{argmin}_{\theta} KL\{q_\theta(\boldsymbol{\omega}) \parallel p(\boldsymbol{\omega}|D)\}$$

$$\text{where the } KL\{q_\theta(\boldsymbol{\omega}) \parallel p(\boldsymbol{\omega}|D)\} \coloneqq \int q_\theta(\boldsymbol{\omega})\, log\, \frac{q_\theta(\boldsymbol{\omega})}{p(\boldsymbol{\omega}|D)}\, d\boldsymbol{\omega}$$

The resulting cost function is known as the variational free enery or the expected lower bound (ELBO):

$$\begin{aligned} F(D,\theta) &= KL\{q_\theta(\boldsymbol{\omega}) \parallel p(\boldsymbol{\omega})\} - \int q_\theta(\boldsymbol{\omega})\, logp(D|\boldsymbol{\omega}) d\boldsymbol{\omega} \\ &= KL\{q_\theta(\boldsymbol{\omega}) \parallel p(\boldsymbol{\omega})\} - \mathbb{E}_{q_\theta(\boldsymbol{\omega})}[logp(D|\boldsymbol{\omega})] \\ &= \mathbb{E}_{q_\theta(\boldsymbol{\omega})}[log\, \frac{q_\theta(\boldsymbol{\omega})}{p(\boldsymbol{\omega})}] - \mathbb{E}_{q_\theta(\boldsymbol{\omega})}[logp(D|\boldsymbol{\omega})] \\ &= \mathbb{E}_{q_\theta(\boldsymbol{\omega})}\left[log\, \frac{q_\theta(\boldsymbol{\omega})}{p(\boldsymbol{\omega})} - logp(D|\boldsymbol{\omega})\right] \\ &= \mathbb{E}_{q_\theta(\boldsymbol{\omega})}[logq_\theta(\boldsymbol{\omega}) - logp(\boldsymbol{\omega}) - logp(D|\boldsymbol{\omega})] \end{aligned}$$

We can observe that the cost function embodies a trade-off between satisfying the complexity of the data D and satisfying the simplicity prior $p(\boldsymbol{\omega})$. However, it's intractable to estimate this cost function, Blundell et al proposed to use an unbiased estimator of gradients of the cost to learn the distribution over the weights of a neural network under a reparameterization trick.

Proposition 1. Let ϵ be a random variable having a probability density given by $q(\epsilon)$ and let $\boldsymbol{\omega} = t(\theta, \epsilon)$ where $t(\theta, \epsilon)$ is a deterministic function. Suppose further that the marginal probability density of $\boldsymbol{\omega}$, $q_\theta(\boldsymbol{\omega})$, is such that $q(\epsilon)d\epsilon = q_\theta(\boldsymbol{\omega})d\boldsymbol{\omega}$. Then for a function f with derivatives in $\boldsymbol{\omega}$:

$$\frac{\partial}{\partial\theta}\mathbb{E}_{q_\theta(\boldsymbol{\omega})}[f(\boldsymbol{\omega},\theta)] = \mathbb{E}_{q(\epsilon)}[\frac{\partial f(\boldsymbol{\omega},\theta)}{\partial\boldsymbol{\omega}}\frac{\partial\boldsymbol{\omega}}{\partial\theta} + \frac{\partial f(\boldsymbol{\omega},\theta)}{\partial\theta}]$$

The deterministic function $t(\theta, \epsilon)$ transforms a sample of parameter-free noise ϵ and the variational posterior parameters θ into a sample from the variational posterior.Let $f(\boldsymbol{\omega},\theta)$=$logq\_\theta(\boldsymbol{\omega})$−log$p(\boldsymbol{\omega})$−$logp(D\mid\boldsymbol{\omega})$, we use the unbiased estimates of gradients of the cost $F(D, \theta)$, with $F(D, \theta)$ being approximated by $\sum_{i=1}^{n}\log q_\theta(\boldsymbol{\omega}^{(i)}) - logp(\boldsymbol{\omega}^{(i)}) - logp(D|\boldsymbol{\omega}^{(i)})$ using Monte Carlo sampling (here each $\boldsymbol{\omega}^{(i)}$ is drawn from the variational posterior $q_\theta(\boldsymbol{\omega}^{(i)})$).

Assume the variational posterior of weights follows the Gaussian distribution. To ensure the non-negativity of the standard deviation σ, we parameterize it as $\sigma = \log(1 + \exp(\rho))$. The variational posterior parameters are $\theta = (\mu, \rho)$, and $\boldsymbol{\omega} = t(\theta, \epsilon) = \mu + \log(1 + \exp(\rho)) \circ \epsilon$. The algorithm to update and optimize the variational parameters is as below:

Algorithm:

1. Sample $\epsilon \sim N(0, 1)$
2. Let $\boldsymbol{\omega} = \mu + \log(1 + \exp(\rho)) \circ \epsilon$
3. Let $\theta = (\mu, \rho)$
4. Let $f(\boldsymbol{\omega},\theta) = logq_\theta(\boldsymbol{\omega}) - \log p(\boldsymbol{\omega}) - logp(D|\boldsymbol{\omega})$
5. Calculate the gradient with respect to the mean

$$\Delta_\mu = \frac{\partial f(\boldsymbol{\omega},\theta)}{\partial\boldsymbol{\omega}} + \frac{\partial f(\boldsymbol{\omega},\theta)}{\partial\mu}$$

6. Calculate the gradient with respect to the standard deviation parameter

$$\Delta_\rho = \frac{\partial f(\boldsymbol{\omega},\theta)}{\partial\boldsymbol{\omega}}\frac{\epsilon}{1+\exp(-\rho)} + \frac{\partial f(\boldsymbol{\omega},\theta)}{\partial\rho}$$

7. Update the variational parameters:

$$\mu \leftarrow \mu - \alpha\,\Delta_\mu$$
$$\rho \leftarrow \rho - \alpha\,\Delta_\rho$$

We don't employ the closed version of the complexity cost (or entropic portion) since it enables for a greater number of prior and variational posterior families to be combined. Indeed, this approach is straightforward to implement and flexible enough to allow for the swapping of prior/posterior pairings. The spike-and-slab version of prior can also work as a regularization tool and speed up the training efficiency.

Although this method has conquered lots of challenges in Bayesian neural network, some typical disadvantages are still left unsolved. For instance, the number of parameters is doubled, and the performance in prediction is not better than a closed-form KL divergence cost function. There are some other methods that are newly proposed are worth discussion as well: A gradient-based optimization procedure on the dropout neural network, equivalent to a specific variational approximation on a Bayesian neural network (Gal 2016); and similar optimization procedure on batch normalization layers (Teye et al. 2018).

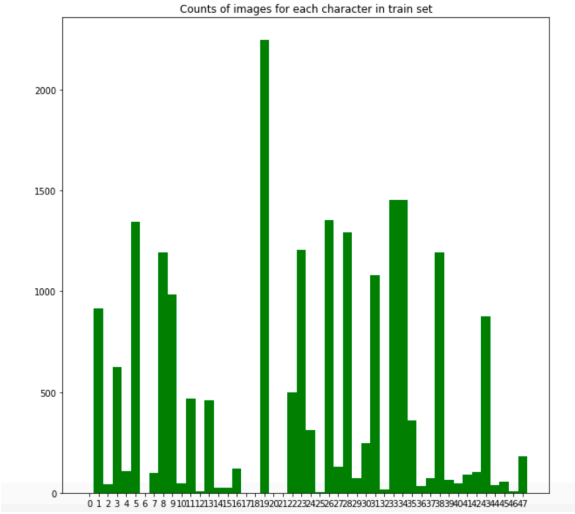
3. Uncertainty Quantification

We are interested in predictive distribution $pD(y^*|x^*) = \int p_{\boldsymbol{\omega}}(y^*|x^*)p(\boldsymbol{\omega}|D)d\boldsymbol{\omega}$. We obtain the posterior samples of $\boldsymbol{\omega}$, thus we can also get the posterior predictive samples of y^*. So the posterior predictive mean can be calculated as well as the 95% credible intervals to measure the uncertainty.

There are different methods for uncertainty quantification. Gal (2016) introduced a Shannon-entropy-based predictive uncertainty for classification given by $H\{p(y^*|x^*, D)\} \coloneqq -\sum_{k=1}^{K} p(y^* = e_k|x^*, D) log\{p(y^* = e_k|x^*, D)\}$ and suggested an easy-to-compute estimator given by $-\sum_{k=1}^{K} \hat{p}_{\hat{\theta}}(y^* = e_k|x^*, D) log\{\hat{p}_{\hat{\theta}}(y^* = e_k|x^*, D)\}$. Another method is popular in medical image data. Predictive variance can be decomposed into 2 parts: aleatoric and epistemic uncertainty. Aleatoric uncertainty is a measure for the variation of data (data quality), epistemic uncertainty is caused by the model (model performance). So the measurement of those two uncertainty can identify the source of the variation. Thus it can help decide whether collecting more data can improve the performance. However, there are still some limitations exist in some uncertainty quantification methods. For example, someone proposed to apply the diagonal modeling of the aleatoric uncertainty, which does not reflect negative correlations of class indicators due to mutual exclusiveness. In this project, we only consider to calculate the credible intervals of the $\hat{p}_{\hat{\theta}}(y^* = e_k|x^*, D)$ for the uncertainty estimation.

4. Case Study: Simpson Characters Recognition

In this case study, we perform the image classification of 13 different Simpson characters under non-Bayesian CNN (regularCNN) and BayesCNN. By examining the raw data, we found out that there are over 20,000 images in training set with 47 different Simpsons characters, while around 1000 images in testing set with over 20 characters, and each character has about 50 test data. We plot a histogram (figure 2) of the number of images for each character in training set, found out this number is very imbalance. Therefore, we decided to only use those characters with reasonably large images. After the cleaning of raw data, we now have 13 characters with the largest amount of training images in both train and test set.



Label	Characters	Counts(train set)
0	abraham grampa simpson	913
1	bart simpson	1342
2	charles montgomery burns	1193
3	chief wiggum	986
4	homer simpson	2246
5	krusty the clown	1206
6	lisa simpson	1354
7	marge simpson	1291
8	milhouse van houten	1079
9	moe szyslak	1452
10	ned flanders	1454
11	principal skinner	1194
12	sideshow bob	877

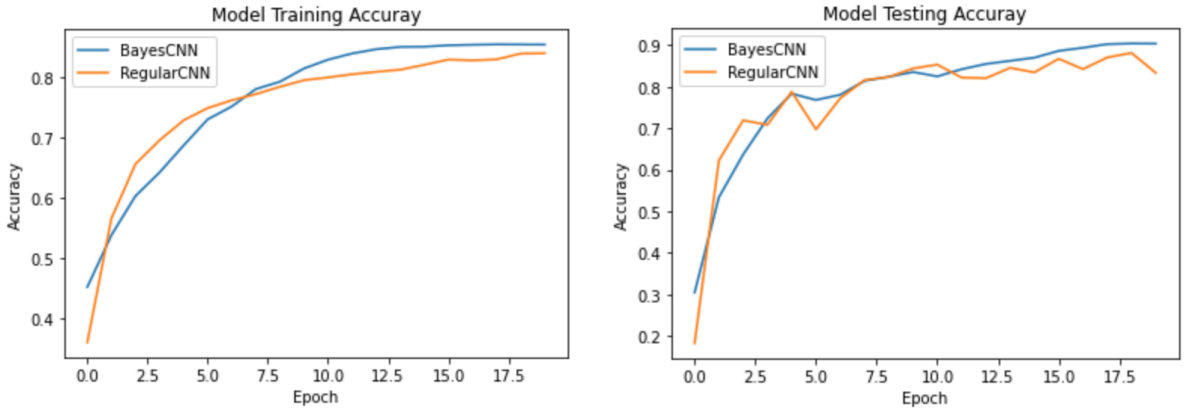
The maximum number of images we use to train our model for each character is 1000 (for example, 1354 images have been found for Lisa Simpson, we only use the first 1000 of them), to ensure our training data is balanced. So now the training data contains 12773 images, and the testing set contains 641 images. The following jobs are done to the data in this part:

- Cleaning: remove the image containing multiple characters
- Re-sizing: load the image as size of 64*64
- Normalization: normalize the data by dividing 255 (reduce the effect of illumination)
- One-hot Encoding: encode labels to hot vectors as response variable for CNNs

We follow the similar CNN structure as in MINST classifications here, we set up 2 convolutional layers with ReLU, following a max-pooling layer, and repeat this structure for three times, ending up with a fully-connected layer. The batch size is 128 and we run 20 epochs.

Layer Type	Filters	Padding
Convolutional (3*3)	32	0
Convolutional (3*3) Max-pooling (2*2)	32	0
Convolutional (3*3)	64	0
Convolutional (3*3) Max-pooling (2*2)	64	0
Convolutional (3*3)	128	0
Convolutional (3*3) Max-pooling (2*2)	128	0
Fully-connected		

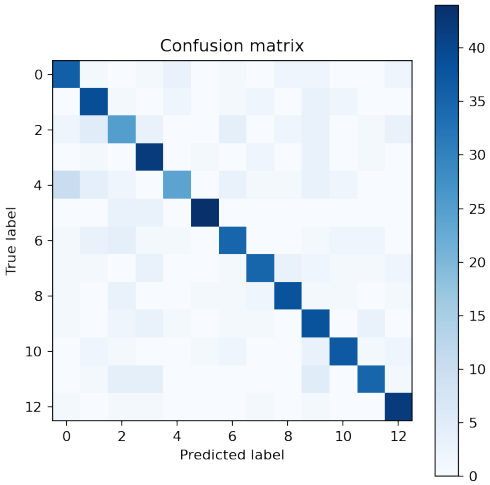
The BayesCNN is under the same structure as CNN but with random weights in each convolutional and fully-connected layer. We set the learning rate as 10^{-3}, π as 0.25, σ_1^2 as 1 and σ_2^2 as 0.005. After running 20 epochs, we obtain the training and testing accuracy for two methods as below:



The plots suggest that the predictive accuracy are quite similar under two methods, which is consistent with the results in some other paper. The training accuracy for BayesCNN seems to be a little bit better than RegularCNN when converges. However, as for the testing accuracy, the plot for RegularCNN is not as smooth as BayesCNN. At the end of the epoch, it even presents a tendency of over-fitting. So even though, the overall accuracy of two methods are not far from each other, the performance of BayesCNN is better in the aspect of overfitting. It makes sense since in this neural network structure, the dropout layers as well as the batch normalization layers are not contained. So it somehow restricted the chance for regularization of RegularCNN.

The confusion matrix for RegularCNN is ploted to further explore the classification performance.

It's obviou that Charles Montgomery Burns (label 2) and Homer Simpson (label 4) are more confusing with other characters. Specifically, Homer Simpson is easily mis-recognized as Grampa Simpson (label 0), which is reasonable since they have the father-and-son relationship.
Chief Wiggum (label 3) and Krusty the Clown (label 5) are easy to be recognized in the classifiers. It's also understandable since their hair and costumes are way more different than other characters.



In order to check the uncertainty estimation in BayesCNN, we also reported the 95% credible intervals of posterior predictive $\hat{p}_{\tilde{\theta}}(y^* = e_k|x^*, D)$ for some of the test images. We can clearly see the prediction performance in each bar plot. The wider the range of the 95% CI is, the more uncertainty is presented for each class. For example, the prediction of grampa Simpson, the CI for the posterior probability of predicting the correct label is the highest among all the other classes, so the prediction can be correct in most of the cases, but the width of this CI still indicates a huge uncertainty. If we see the plot for the clown, almost 100% it provides the correct classification. The width of the CI is pretty small and the mean is close to 1. Among the selected 8 images in the test set, Lisa Simpson is most likely to be mis-recognized, and specifically, to Bart Simpson. This image only shows a side face of Lisa, so it's difficult to separate it with Lisa's brother Bart. The BayesCNN does provide more interesting information comparing with Regular CNN.
.