Bayesian Networks for Image Classification

Made by: Diogo Araújo - 60997

Introduction

For this project, I aim to understand why Bayesian Networks are not as utilized in Image Classification as they are in other fields of study, such as information retrieval and filtering. Current implementations of deep learning for the field of Image Classification stray away from probabilistic models such as Bayesian Networks. This raises the questions as to why that is, which is what I attempt to figure out with this project.

In order to achieve this, I will be conducting / creating a Bayesian Network model with the purpose of image classification on three different data sets. I will then compare its results with a more popular model currently being used for Image Classification, which will be the Convolutional Neural Network (CNN) ([2]). This model is commonly used for image recognition and processing due to its ability to recognise patterns in images.

With this comparison, I aim to better understand why the Bayesian model is less popular than the CNN model and, by default, other Image Classification models. This will be approached from various angles, such as the results obtained, their real world application, the effect of the data quality and time that they took, as a way of analysing their differences in detail.

Approach

When starting the project, I decided on acquiring all the required data sets which would be used with our models. These seemed as an extremely important start, as the correct data would be crucial in obtaining the highest accuracy possible with the models and would also allow me to conduct an early evaluation of their prototypes. After running several applications of these models, I created the final version of both the Bayesian and CNN model which would be used. After their creation, I ran both of the models with the associated data sets and analysed their values for results.

This process began with deciding which data sets to use. For this, I took into consideration the context of the project and their best possible values for its objective. Because of this, the acquired data sets would have to contain images, due to being for an Image Classification model, would have to vary in both size and quality, in order to obtain more varied results from the models, and would have to come with labelled images as they would be used for training and testing the models. With these factors in mind, I went through a website which contained open data sets ([6]) and, even with these considerations, ended up obtaining a large amount. Due to this, I applied a few further constraints in order to limit them down to three separate and varied data sets. These included having a varied number of labels, varied amount of images, images of different sizes and a balanced amount of images per label. ([5])

Through this process, I ended up obtaining three data sets:

- Low Quality Data set - MNIST;
- Medium Quality Data set - Fashion-MNIST;
- High Quality Data set - CIFAR-10;

The low quality set, or the MNIST set ([7]), is a medium size data set which contains 70k images, where 60k are used for training and 10k are used for testing. Its task is to classify a given image of a handwritten digit into one of 10 labels which represent integer values from 0 to 9, inclusively. This is the "Low Quality" set is due to it only containing images which have the same "theme" and which are high quality. Because of this, it is an entirely theoretical data set with no practical real world applications.

The medium quality set, or the Fashion-MNIST set ([9]), is a medium size data set containing the same amount of images and labels as the MNIST data set. It is extremely similar to the MNIST data set due to all of its images sharing the same format but it is related to a different topic. This set was chosen as the "Medium Quality" set due to its average size as well as large similarities to the low quality data, but it has a more practical purpose and its images are more complicated.

The high quality set, or the CIFAR-10 (Canadian Institute for Advanced Research) set ([8]), is the smallest data set with 60k colored images, where 50k are used for training and the rest are used for testing. There are 10 labels with 6k images per label. It is the "High Quality" set due to having a large and balanced amount of data for each label as well as a large variation of images. It also has a worse image quality and more complex images, which can replicate real cases. It is, however, not perfect due to its lower data set size.

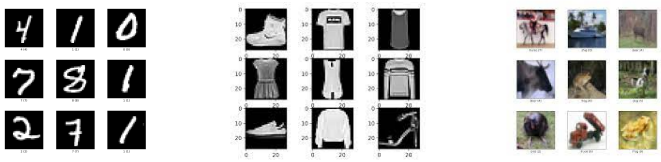

Figure 1: Data sets chosen (MNIST, Fashion-MNIST and CIFAR-10)

Implementation

After acquiring all the data sets, I started creating the models which were used for this project. This process involved the usage of Python and of its TensorFlow library. With these, I created two separate models, one for the Bayesian Network and the other for the Convolutional Network, which were applied to each of the data sets previously chosen. The models are similar feed-forward networks but with major differences.

I started by creating the Convolutional Neural Network (CNN) ([2]) model. These are a popular Image classification model which are used due to requiring relatively little pre-processing when compared to other image classification algorithms and due to excelling at assigning importance to various objects and features within the images through the use of convolutional layers. These Convolutional Layers are the key building block of the network, where most of the computations are carried out. They work by applying a filter to the input data to identify features. They are, however, prone to overfit on smaller data sets. For this project, I created a simple model based on a tutorial created by TensorFlow ([4]).

This model starts with a convolutional base. This is a common pattern in CNN models and is composed of a stack of "Conv2D", which are its convolutional layers, and "MaxPooling2D", which are its pooling layers. Its initial input shape is decided by which data set it is examining, which in the case of the CIFAR-10 data set is 32 by 32. I finished the model by adding a "Flatten" layer followed by two final "Dense" layers. These take the earlier output of the layers and convert them into a 1D array which is then used to make the final classification for the given labels. This model is then compiled, using an adam optimizer and categorical cross-entropy loss, and summarised so that we can observe its architecture:

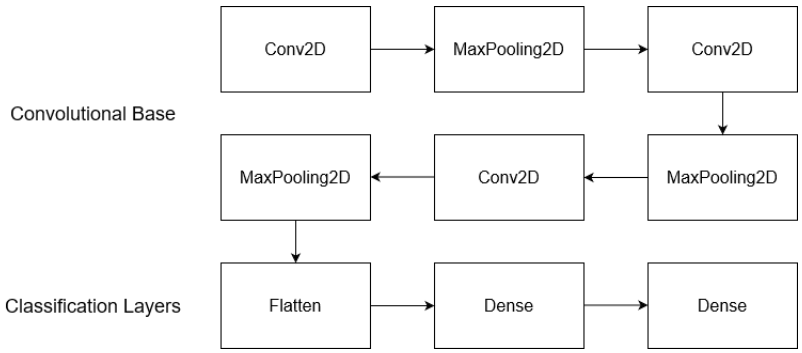

Figure 2: Convolutional Neural Network (CNN) Architecture

The Bayesian network model used is a variant of CNN that can reduce the chances of overfitting while training on small-size data. This architecture is known as Bayesian Convolutional Neural Network (BCNN) ([1]). These are probabilistic models which use statistical methods to cover a probability distribution that is attached to the network with weights and biases. This is unlike other methods (such as CNN), which find optimal points for its weights.

The variation applies Bayesian on the CNN in order to reduce overfitting, therefore the name Bayesian CNN. This can be done in a variety of ways, although the manner which it was done in the project was by inferring the model posterior via variational inference. This method aims to fit the parameter of distribution closer to the true posterior in order to express an uncertainty estimation of the models' parameters. The model created for this project closely followed the associated tutorial ([3]) which allowed me to do just that.

The model starts with a "Convolution2DReparameterization" layer. This layer creates a convolution kernel that produces a tensor of outputs and adds epistemic uncertainty over the weights. It does this based on the divergence function that is given. For this model, I used the Kullback-Leibler divergence ([10]) which is a way of comparing two probability distributions in order to measure just how much information we lose when we choose an approximation. After this, it goes through a pooling layer which is then flattened and followed by a "DenseVariational" layer. This layer uses variational inference to fit a "surrogate" posterior to the distribution.

For this, we create a custom prior called the "spike and slab", also known as a scale mixture prior, distribution which is the weighted sum of two normally distributed distributions, one with a standard deviation of 1 and the other of 10. The reason for using such a prior is that it is like a standard unit normal, but makes values far away from 0 more likely which allows the model to explore a larger weight space. We then finish the model with a "OneHotCategorical" layer which will allow us to check the prediction made as well as analyze the estimated probabilities of that prediction. The model is then compiled, using an RMSprop optimizer and negative log-likelihood loss (maximizes the likelihood estimate of the mean and the standard deviation of the weights), and summarised so that we can observe its architecture:

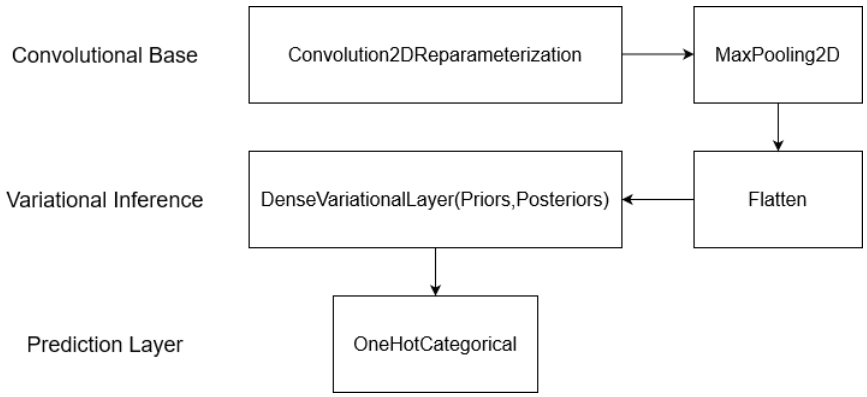

Figure 3: Bayesian Convolutional Neural Network (BCNN) Architecture

Results

Having created and compiled the models, I then had to train them. For this, I decided on training them using the same hyper parameters for each. I let the models run for around 50 epochs with a possible early stoppage when, within 5 epochs, there was no improvement in their accuracy. They were trained on the training data with the testing data serving as validation. After their training was finished, all that was left to do was to evaluate them based on their accuracy and loss and analyse their results.

Low Quality Analysis (MNIST)

For the low quality data set, both of the models ended up showing extremely high values for accuracy over their training process for both the training and validation sets. This was expected due to the simplicity of the data set and its images. However, their loss values are worse in comparison. The BCNN model showed normal loss values, starting at a high 2 and gradually declining down to 0.12, while the CNN model has the same pattern for its training set but is worse and more erratic in its validation set. It still has good values, but they do show a concerning pattern which could indicate the model is overfit.

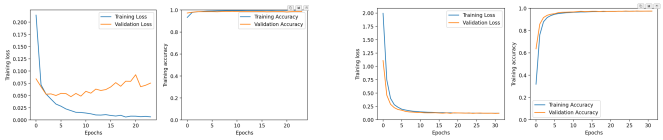

Figure 4: Training Process for Low Quality Data set (CNN and BCNN)

I also analyzed a few of the predictions made by the BCNN model. In these, we can analyze how the model calculates and assigns probabilities to each of the labels as well as analyze the epistemic uncertainty in the predictions. For the first prediction, the model is confident in the probability which can be seen by it assigning a probability of nearly one to the 7 label. It also has a low epistemic uncertainty which can be seen by the size of the bar (which indicates a narrow prediction interval). For the second prediction, the model is not as certain as to what it is as the uncertainty is a tiny bit higher and the probabilities are more scattered.

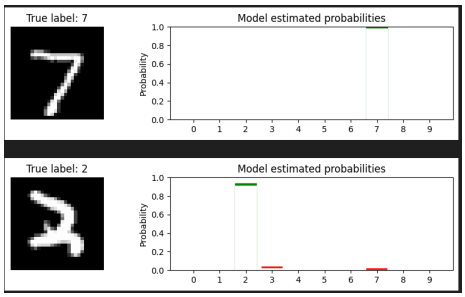

Figure 5: Two Image Predictions on Low Quality Data set (BCNN)

Medium Quality Analysis (Fashion-MNIST)

For the medium quality data set, both of the models still ended up showing high accuracy values, with the CNN model showing slightly higher values than the BCNN. These were, in comparison with the low quality data set, smaller values but still reliable. This was most likely due to the images now being more complex and detailed, even though their sizes remained the same. The CNN model also showed the same signs of overfitting as it did in the low quality data set which weren't found in the BCNN model.

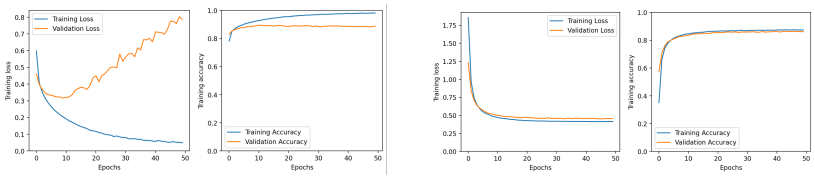

Figure 6: Training Process for Medium Quality Data set (CNN and BCNN)

In terms of predictions, the BCNN model ended up classifying more images as the wrong classification due to its lower accuracy values. This can be seen in the first prediction, where the model correctly classifies the label as a dress (3) but still has a slight probability towards the label 4. The second prediction is less confident and incorrectly classifies the jacket (2) as a t-shirt (6) but still has some probability towards other labels.

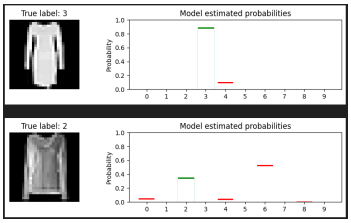

Figure 7: Two Image Predictions on Medium Quality Data set (BCNN)

High Quality Analysis (CIFAR-10)

For the high quality data set, we can see that the CNN model continues to overfit with a low training loss and high validation loss. However, it still continues to have high levels of accuracy (around 0.9). This is not the case for the BCNN model which shows high loss (around 2) and low accuracy (around 0.4). This is most likely due to the increase in image complexity and the lowering of image quality which makes them harder to accurately predict.

In terms of the predictions, we can see that the model has a much harder time classifying the images correctly due to their added complexity. These predictions now contain a high amount of epistemic uncertainty. For the first prediction, the model correctly classifies the image as a car (1) but has a high amount of probability on other labels. For the second prediction, the model incorrectly classifies the label with a low amount of probability to each.

(The plots and predictions were removed from this section due to requiring too much space but can still be found at the end of the Annex section)

Conclusion

From the results obtained, we can see that both of the models are capable of predicting on all of the data sets with varying degrees of success. This success seems to be based on the quality of the images of the data set which greatly influences the model accuracy and loss. When comparing the two, we can see that the CNN model overfits on all of the data, meaning it requires more data to be trained on, but manages to retain a high amount of accuracy independent of image quality. On the other hand, the BCNN model starts off strong with high quality images, but its accuracy drops off on the more complex / lower quality images as it has a harder time understanding each of them.

Due to this, I believe that BCNN models are not as utilized in the area of image classification, when compared to the CNN model, due to not only their added model complexity as well as their difficulties when predicting on complicated images (which will be the case for most real cases).

Bibliography

[1] A beginner's guide to Bayesian CNN, https://analyticsindiamag.com/a-beginners-guide-to-bayesian-cnn/

[2] Convolutional Neural Network: Benefits, Types, and Applications, https://datagen.tech/guides/computer-vision/cnn-convolutional-neural-network/

[3] Bayesian Convolutional Neural Network (Tutorial), https://goodboychan.github.io/python/coursera/tensorflow_probability/icl/2021/08/26/01-Bayesian-Convolutional-Neural-Network.html

[4] Convolutional Neural Network (CNN) (Tutorial), https://www.tensorflow.org/tutorials/images/cnn

[5] SelectStar (2020) Creating the Best Quality Image Dataset, https://selectstar-ai.medium.com/creating-the-best-quality-image-dataset-720f612944ed

[6] Papers with Code (Data set Website), https://paperswithcode.com/datasets

[7] MNIST Data Set, https://www.tensorflow.org/datasets/catalog/mnist

[8] CIFAR-10 Data Set, https://paperswithcode.com/dataset/cifar-10

[9] Fashion-MNIST Data Set, https://paperswithcode.com/dataset/fashion-mnist

[10] Kullback-Leibler Divergence Explained (Count Bayesie Blog), https://www.countbayesie.com/blog/2017/5/9/kullback-leibler-divergence-explained

Annexes

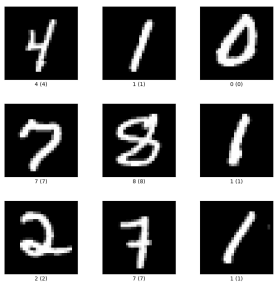

Figure 8: MNIST Data set

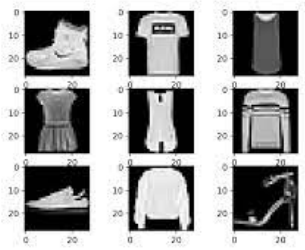

Figure 9: Fashion-MNIST Data set

Figure 10: CIFAR-10 Data set

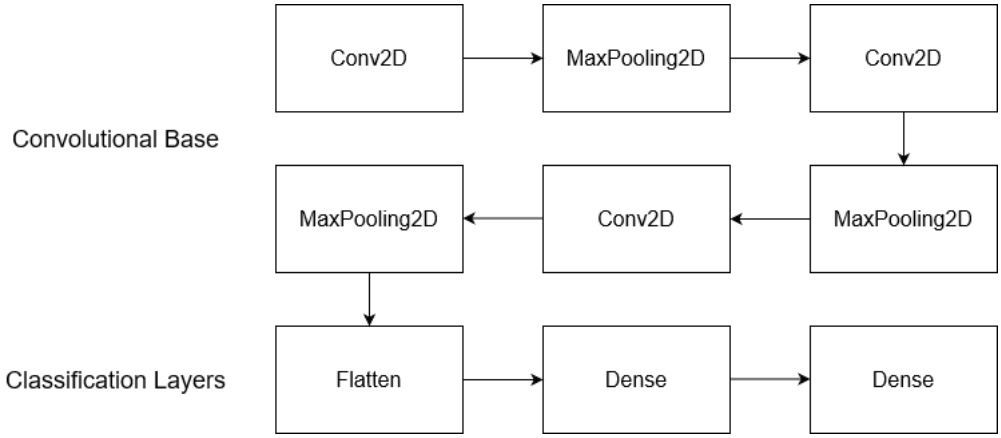

Figure 11: Convolutional Neural Network (CNN) Architecture

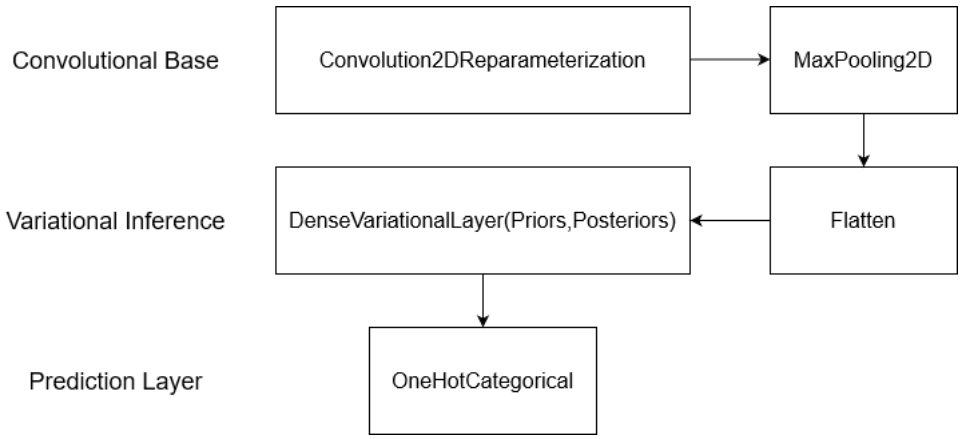

Figure 12: Bayesian Convolutional Neural Network (BCNN) Architecture

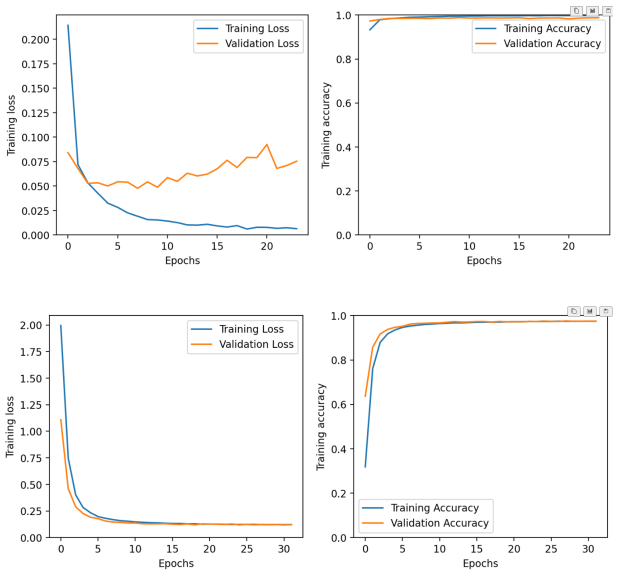

Figure 13: Training Process for Low Quality Data set (CNN and BCNN)

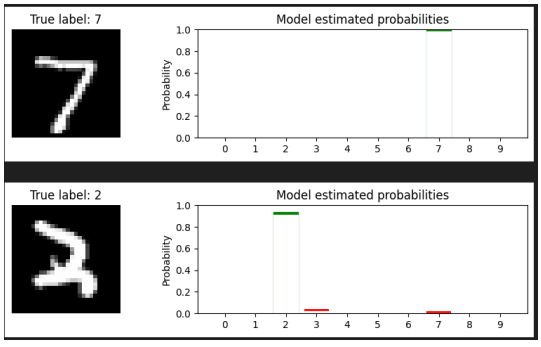

Figure 14: Two Image Predictions on Low Quality Data set (BCNN)

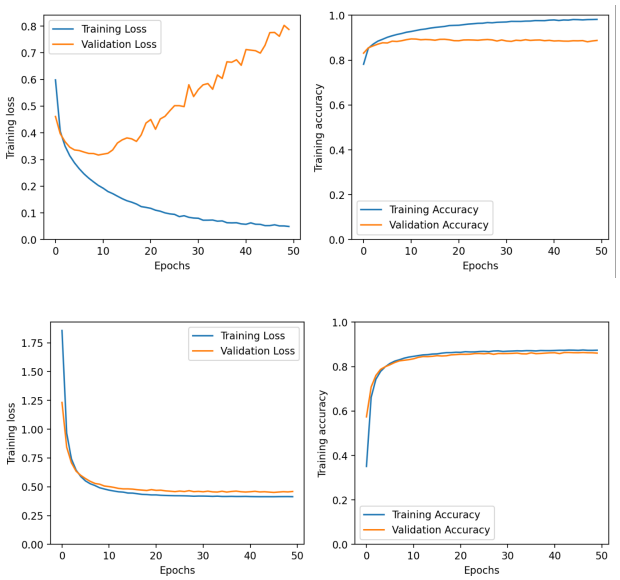

Figure 15: Training Process for Medium Quality Data set (CNN and BCNN)

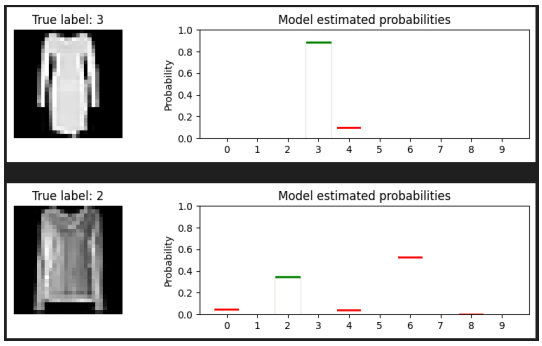

Figure 16: Two Image Predictions on Medium Quality Data set (BCNN)

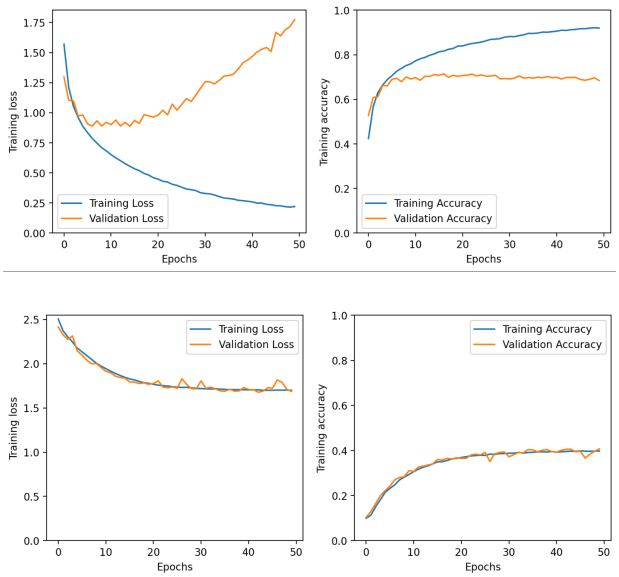

Figure 17: Training Process for High Quality Data set (CNN and BCNN)

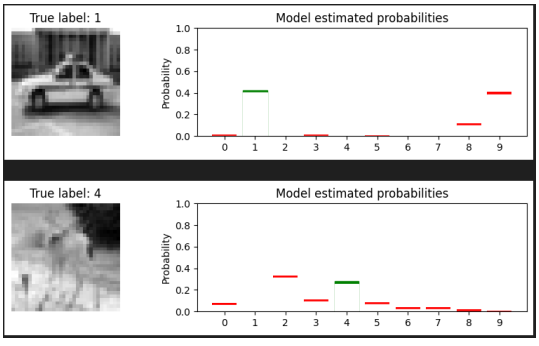

Figure 18: Two Image Predictions on High Quality Data set (BCNN)