Quantifying Uncertainty in Traffic Sign Recognition

A Preprint

Bjarke Hautop
Aarhus University
bjarke.hautop@gmail.com

Christina Grand
Aarhus University
christinagrand@hotmail.com

Laura Fur
Aarhus University
201909286@post.au.dk

December 4, 2024

Abstract

Autonomous driving technology relies heavily on accurate and robust traffic sign recognition to ensure road safety and facilitate real-time decision-making. While traditional convolutional neural networks (CNNs) achieve high accuracy, they often cannot handle real-world challenges such as image noise, lighting variations, and out-of-distribution (OOD) data. This paper addresses these limitations by incorporating uncertainty quantification techniques into traffic sign recognition using the German Traffic Sign Recognition Benchmark (GTSRB) dataset. We explore three methods to quantify uncertainty in CNNs: Monte Carlo (MC) dropout, learning mean and variance of weights, and Variational Inference (VI). Our results demonstrate that MC dropout achieves the best balance of performance and uncertainty estimation, with a test cross-entropy of 0.04 and effective differentiation between confident and uncertain predictions. However, challenges remain in distinguishing OOD data, where the variance, related to uncertainty, is less pronounced than expected. By showing prediction confidence and identify unreliable classifications, the proposed uncertainty-aware models enhance the robustness of traffic sign recognition systems.

Keywords Traffic Sign Recognition · Uncertainty Estimation · Bayesian Neural Networks · Autonomous Driving

1 Introduction

Traffic sign recognition is a critical component of autonomous driving technology. The ability to accurately and efficiently detect and classify traffic signs is essential for ensuring road safety and enabling vehicles to make informed decisions in real time. This project aims to develop a robust model for traffic sign recognition using the GTSRB dataset. The dataset covers a diverse range of traffic signs with varying shapes, colors, and lighting conditions, providing a realistic test set [Stallkamp et al., 2011].

Traditional CNNs have been successfully applied to traffic sign recognition tasks, achieving high levels of accuracy. However, these models may struggle with robustness, especially when confronted with noisy data, variations in lighting, weather conditions, or OOD data that are not well represented in the training set. Such limitations can be problematic in real-world scenarios where traffic signs may appear damaged or distorted.

To address the challenges of improving robustness in traffic sign recognition, this project explores the use of uncertainty estimation techniques within CNNs. This work incorporates uncertainty estimation techniques, such as Monte Carlo (MC) dropout and variational inference, to improve the model's robustness against noisy data, lighting variations, and out-of-distribution traffic signs. These approaches allow the network to not only make predictions but also quantify the uncertainty in those predictions, making the system more reliable in real-world applications.

Ultimately, the goal of this work is to develop a traffic sign recognition model that is not only accurate but also provides an understanding of the uncertainty associated with each prediction, making the system more reliable in real-world autonomous driving applications.

The remainder of this paper is structured as follows: Section 2 reviews related work in traffic sign recognition and uncertainty estimation methods. Section 3 describes the methodology, including dataset details, model architecture, and techniques used for uncertainty quantification. Section 4 presents the experimental setup, results, and evaluation. Finally, Section 5 discusses the implications of our findings.

2 Related work

The task of traffic sign recognition has been extensively studied in the context of autonomous driving. Several approaches have used deep learning techniques to achieve low loss (high accuracy) in benchmark datasets such as the GTSRB dataset. This section reviews key related works and discusses their relevance to the current project.

Sermanet and LeCun [2011] proposed one of the early applications of deep learning for traffic sign recognition using the GTSRB dataset. Their approach employed multi-scale CNNs to achieve state-of-the-art performance at the time, showcasing the strength of deep learning in extracting features from complex visual data. The method involved hierarchical feature extraction using multiple convolutional (conv) layers to capture traffic sign details at different scales. Although traditional CNNs achieved high accuracy, this work highlighted some limitations, such as susceptibility to variations in lighting and noise. These limitations motivate this paper and ways to quantify uncertainty.

To incorporate uncertainty estimation in deep learning models, Gal and Ghahramani [2016] proposed an approach that interprets dropout as an approximate Bayesian inference in deep Gaussian processes. By applying dropout at test time, their method enables deep neural networks, including CNNs, to estimate uncertainty in their predictions. This is particularly relevant for traffic sign recognition in autonomous driving, where understanding prediction confidence can help the system decide when human intervention or additional caution is needed. The present work extends this concept by integrating Bayesian methods into the traffic sign recognition pipeline, aiming to improve performance under uncertain scenarios.

Further exploration of Bayesian approaches, Shridhar et al. [2019] provided a comprehensive guide to Bayesian Convolutional Neural Networks (Bayes-CNN) using VI. Their method modifies backpropagation to learn a mean and variance of the weights. Applied to tasks like image classification and super-resolution, Bayes-CNN demonstrated performance comparable to point-estimate-based architectures while quantifying uncertainty and improving regularization.

The reviewed studies provide a foundation for the current project by demonstrating the efficacy of deep learning in traffic sign recognition and the value of using methods to quantify uncertainty. This work aims to combine these insights to develop a more resilient traffic sign recognition system that performs well even in challenging real-world conditions.

3 Methods

3.1 Dataset

We used the GTSRB dataset from Kaggle, where we used the training data as our full dataset since the test dataset has hidden labels. This dataset has 39000 images with 43 classes. The number of images in each class ranges from around 200 to 2000. We randomly split the dataset into 70% train, 15% validation, and 15% test. The images were preprocessed by resizing them to a 160×160 resolution and normalizing their pixel values. We show some example images and their associated class labels from the dataset in Figure 1.

3.2 CNN model

The encoder layer of the final neural network is constructed in a sequential architecture comprising five conv layers. Each conv layer is followed by batch normalization and a ReLU activation function to introduce non-linearity. The

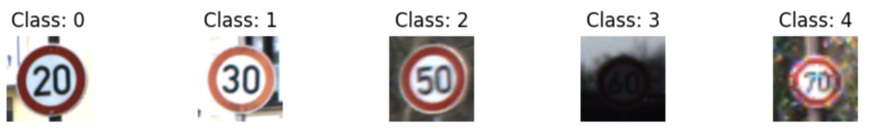

Figure 1: Example images in the dataset GTSRB.

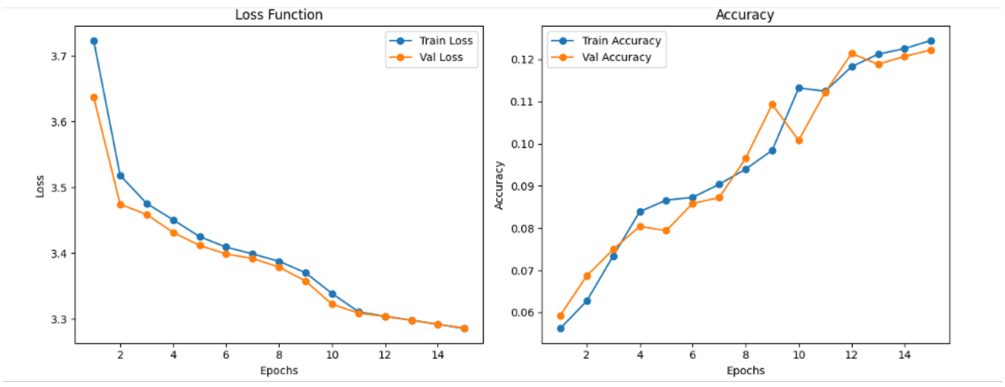

Figure 2: Loss curve (left) and accuracy curve (right) of the baseline model.

conv layers progressively increase their number of filters, with 16, 32, 64, 128, and 256 filters, respectively. All layers utilize a kernel size of $(3, 3)$, a stride of two, and padding equal to one.

First, a baseline model with three conv layers and a stochastic gradient descent optimizer was constructed. The training and validation loss and accuracy curves from this baseline model are shown in Figure 2. The loss and accuracy curves indicate slow model learning, with the loss and accuracy peaking at respectively 3.3 and 0.12 for both training and validation after 15 epochs. Several strategies were explored to address this issue, including tuning the optimizer and changing the learning rate. In an attempt to improve learning, momentum was added to the optimizer, which significantly improved performance, although convergence remained slow. Subsequently, the Adam optimizer was used instead of SGD, which significantly improved performance, lowered the validation loss to 1.4 and increased validation accuracy to 55%. The Adam optimizer combines benefits of both momentum and adaptive learning rates, enabling faster and more stable convergence.

The third baseline model was extended with more conv layers after observing that the model stopped learning effectively as seen in Figure 3. In the accuracy and loss curve, both the accuracy and loss of the training and validation reached a plateau after around 10 epochs, and showed no further improvement in model performance. Based on this, it was

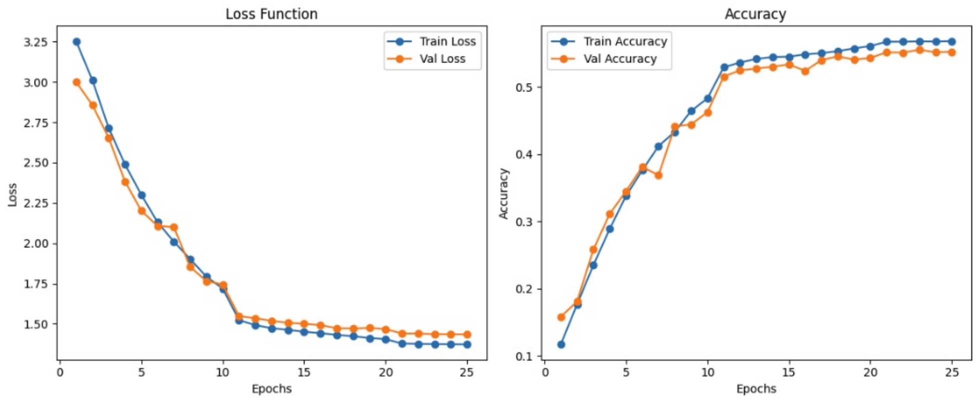

Figure 3: Loss curve (left) and accuracy curve (right) of model trained with Adam optimizer

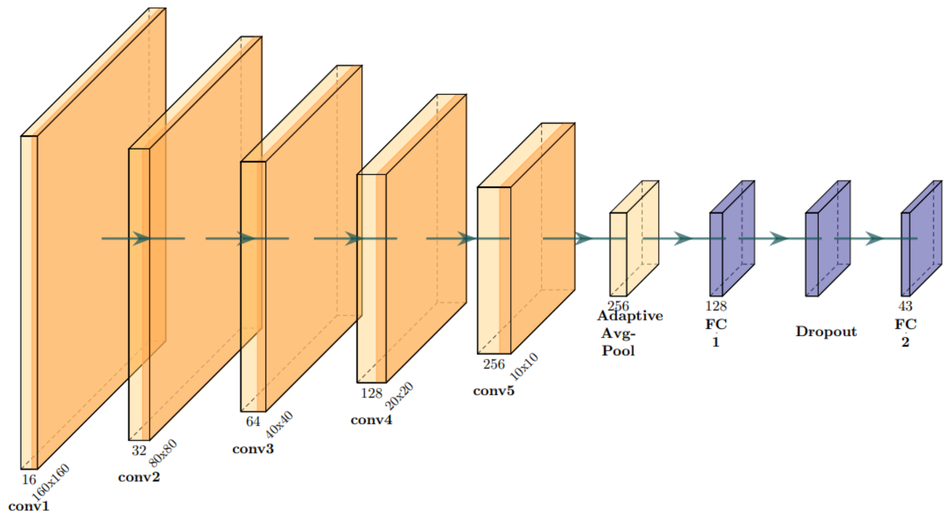

Figure 4: CNN model architecture. Conv layers show input shapes, while pooling and fully connected layers show output shapes.

concluded that this was likely due to the architecture of the model being to simplistic for the data, and therefore, the CNN was extended further. Additionally, batch normalization was introduced to prevent early convergence, which is likely observed due to vanishing gradients. This issue was evident in the loss curve, where both the training and validation accuracies and loss plateaued in the final epochs, forming a flat line. By normalizing the input to each layer, it appears that the batch normalization stabilized the gradient flow, leading to more consistent and efficient learning. This architecture was used as encoder for the final model.

Adaptive average pooling was then applied, and the feature map was flattened using a lambda function, reshaping it into a 1D tensor to feed into a fully connected layer.

The decoder section begins with a fully connected layer comprising 128 output units, followed by a ReLU activation and a dropout layer. Finally, a fully connected layer with an output size equal to the number of classes serves as the classification layer, producing class scores.

To avoid overfitting, dropout was applied with a rate of $p = 0.5$. Overfitting was observed in the loss and accuracy curves, where training accuracy significantly exceeded validation accuracy in later epochs. Adding dropout improved the CNN's ability to generalize to unseen data by randomly deactivating nodes during training, thereby reducing the model's reliance on specific neurons.

Lastly, learning rate decay was applied to further enhance model performance. This strategy reduced the learning rate by a factor of 10 every 10 epochs. The initial learning rate was set to 0.01. Applying this decay led to a slight improvement in accuracy and loss, helping the model to converge more effectively. The final CNN model is visualized in Figure 4.

The model was trained using the cross-entropy loss function. The CNN model was trained for 20 epochs, while the uncertainty models, which have nearly twice as many parameters, were trained for 25 epochs. All models were trained with PyTorch [Paszke et al., 2019], and the uncertainty models were trained with `Bayesian Torch` [Krishnan et al., 2022].

3.3 Using Grad-CAM for prediction insights and explainability

Grad-CAM (Gradient-weighted Class Activation Mapping) Selvaraju et al. [2019] was utilized to pinpoint the regions of an image that contributed the most to the CNN's prediction for a specific class in the final conv layer. It generates a heatmap that overlays the input image, where brighter or warmer areas represent regions with greater influence on the model's decision. By comparing these heatmaps with the input images, true labels, and predicted labels, Grad-CAM provides valuable insights and possible explanations to misclassifications. This analysis helps identify weaknesses in the model, such as sensitivity to specific image regions or environmental conditions.

3.4 Quantifying uncertainty in CNN

In this section, we will explore and give a brief overview of several methods to quantify uncertainty, referring the interested reader to Blundell et al. [2015] and Benatan et al. [2023] for a more comprehensive treatment.

One simple approach is MC dropout, introduced by Gal and Ghahramani [2016], where dropout is applied during both training and evaluation. This technique allows for sampling from the approximate posterior to quantify uncertainty. Gal and Ghahramani (2016) demonstrated that dropout, traditionally used as a regularization technique, can be interpreted as a variational approximation to a Bayesian model. This interpretation enables efficient uncertainty estimation without the need for more computationally expensive inference methods.

Beyond MC dropout, another approach involves treating the network weights as random variables drawn from a distribution. Rather than directly learning a fixed weight for each parameter, we learn the parameters of a distribution (commonly Gaussian), which defines the weights. This allows the model to learn both the mean and variance for each weight, leading to a probabilistic interpretation. To quantify uncertainty for a given image, we sample from the learned distribution for each weight and perform forward passes through the network using these sampled weights.

In order to enable backpropagation for sampling from these distributions, the reparameterization trick is often applied. This trick allows for the separation of the stochasticity of the weights from the deterministic part of the model. Specifically, the weights are sampled as $\boldsymbol{w} = \mu + \sigma \odot \epsilon$, where μ is the learned mean, σ is the learned standard deviation, and $\epsilon \sim \mathcal{N}(0, I)$ is a random variable. This enables the model to learn the variance alongside the mean and facilitates efficient gradient-based optimization.

This approach, where weights are treated as random variables, is often discussed within the framework of Bayesian inference in deep learning. The goal is to compute the posterior distribution of the model parameters given the data. In practice, computing the posterior distribution is intractable due to the complexity of deep neural networks. Traditional methods such as Markov Chain Monte Carlo (MCMC) are too computationally expensive, particularly for high-dimensional models like deep CNNs. To address this, Variational Inference (VI) is employed, approximating the posterior distribution by optimizing a simpler, parameterized distribution that is close to the true posterior.

The VI approach works by defining a family of distributions $q(\theta)$ (the variational distribution) and minimizing the Evidence Lower Bound (ELBO) to find the best approximation to the posterior. The ELBO is given by:

$$\text{ELBO} = \text{KL}[q(\boldsymbol{w} \mid \theta) \| P(\boldsymbol{w})] - \mathbb{E}_{q(\boldsymbol{w}|\theta)} \left[\log P(D \mid \boldsymbol{w})\right],$$

where the first term, the KL divergence, encourages the approximate posterior to remain close to the prior, promoting simplicity in the model, and the second term, the expected log-likelihood, encourages fitting the observed data.

Uncertainty in predictions is quantified by generating multiple samples of the network's weights, yielding different predictions for the same input. Regardless of the method used, the loss is computed using the mean of the probabilities for each class. When calculating uncertainty, the variance of the predicted probabilities for each class is first computed across all the Monte Carlo iterations. Then, we define the uncertainty as the average of these variances.

By assessing the variance in the probabilities from these samples, we can quantify the model's uncertainty.

3.4.1 Model architecture for models to quantify uncertainty

In this work, we explored three different approaches to modeling uncertainty in CNN's. For the Monte Carlo (MC) dropout approach, we used the trained CNN model, where we kept the dropout layer during evaluation to enable stochastic sampling for uncertainty estimation.

For the second approach, in which network weights are treated as random variables, we assumed that weights follow a Gaussian distribution. We slightly changed our CNN architecture by removing the dropout layer. We kept the loss function as cross-entropy.

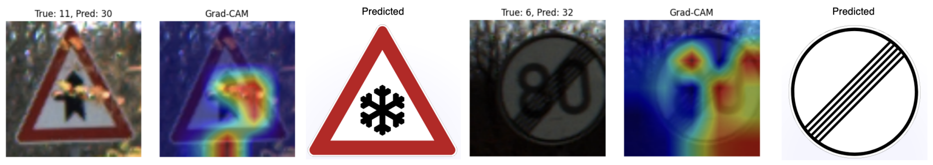

Figure 5: Grad-CAM example misclassified images and their predictions from the CNN model. Reference images for the predicted images are from https://routetogermany.com

Finally, for the Bayesian approach using Variational Inference (VI), we used the prior $N(0, I)$ on the weights, replaced the loss function from cross-entropy to ELBO, and used the same architecture as our second approach.

4 Results

Our CNN model achieved a cross-entropy of **0.04** (an accuracy of **0.99**) on the validation dataset. The MC dropout model, which keeps the dropout active during evaluation, produced comparable cross-entropy values, since it uses the same underlying model. In our second uncertainty estimation approach, where we learned a mean and variance for the weights, the cross-entropy was **0.05** (an accuracy of **0.98**). Finally, for the Bayesian approach using variational inference (VI), we observed a cross-entropy of **0.30** (an accuracy of **0.96**), which had a notably higher cross-entropy than the other methods. This difference, particularly when compared to the second approach that only differs in the loss function, suggests that the KL term in the ELBO loss, acting as a form of regularization, provided a worse performance, which may be due to the large dataset. We used our MC dropout model on the test dataset since it performed the best.

The MC dropout model on the test dataset gave a cross-entropy of **0.04** (an accuracy of **0.99**).

4.1 Grad-CAM results

Misclassified images are often of low resolution, blurred, poorly lit, very bright, or partially obscured by objects such as leaves, as illustrated in Figure 5. In these cases, the predicted labels are generally similar to the actual labels. For example, in the case of the 'Right-of-way at the next intersection' sign (label 11), the leaves and the pole of the sign strongly influence the decision-making in the final conv layer of the CNN. For the 'End of speed limit (80 km/h)' sign (label 6), the model's decision is influenced by the lower-right area of the image and parts of the sign. This is likely due to poor lighting conditions, which may make it harder for the model to distinguish the sign from its background.

4.2 Uncertainty

We can use the variance to quantify how certain our model is in its predictions. Figure 6 illustrates this: the left image, with high variance, represents a case where our MC dropout model (and even a human observer) may struggle to determine the true label, while the right image has low variance, indicating higher confidence in the prediction.

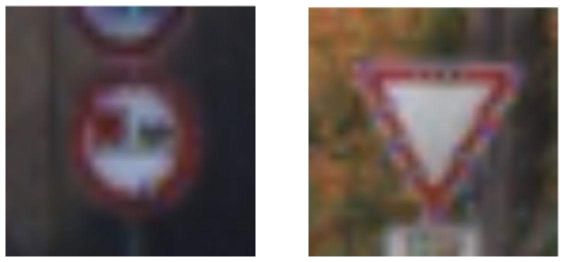

Figure 6: The left image has the highest variance from the MC dropout model in the validation dataset, indicating higher uncertainty. The right image has the lowest variance, indicating higher confidence in the prediction.

Figure 7: Example of an out-of-distribution image we tried.

4.3 Out-of-distribution data

To evaluate how our models handle out-of-distribution (OOD) data, we tested them on an image not present in the dataset. Figure 7 shows an example: a "no u-turn" traffic sign, a class absent from our training data. As expected, our standard CNN model confidently misclassified the image into a specific class.

We hypothesized that models incorporating uncertainty estimation, such as the MC dropout model, would exhibit higher predictive variance for OOD data. We show the results based on 15 MC draws from our MC dropout model in Figure 8, illustrating that the MC dropout model indeed shows some uncertainty. However, for this and other OOD images, the variance observed in the MC dropout model is only slightly higher than the average variance calculated across the validation dataset. This trend was consistent across all three uncertainty estimation models we developed.

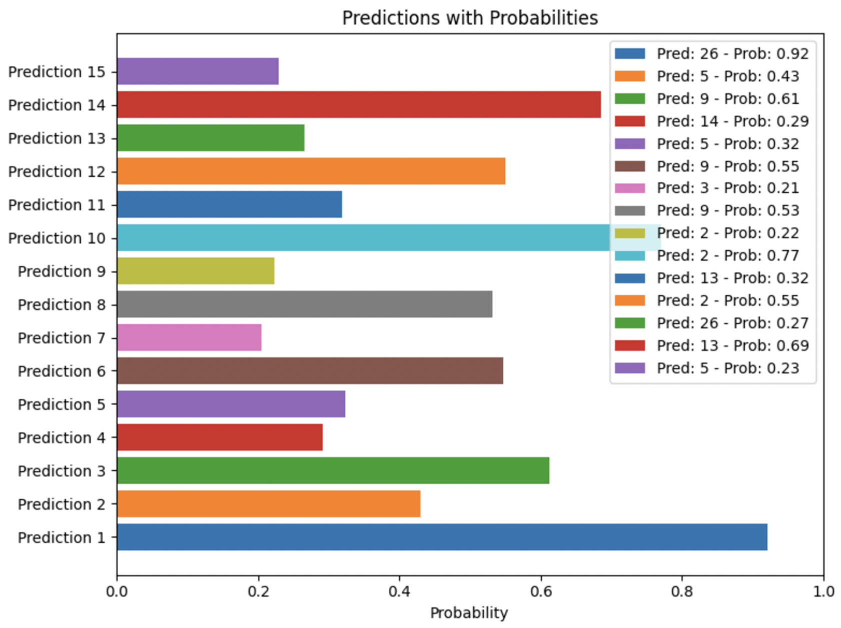

Figure 8: Predicted class probabilities using Monte Carlo (MC) dropout for the 'no-u-turn' sign in Figure 7. The bar chart represents the highest probability class for each of the 15 samples, along with their associated probabilities.

5 Discussion

We managed to make a model that achieved a cross-entropy loss of **0.06** (an accuracy of **0.99**) on the test data, and we explored several approaches to quantifying the uncertainty in deep neural networks.

In evaluating uncertainty on the in-distribution images, we observed that the models effectively differentiated between easy-to-classify and challenging images. The images with low variance predictions were consistently associated with high confidence, and correct classifications, while images with high variance indicated a greater likelihood of misclassification, which is also reflected in the Grad-CAM output. This outcome highlights the effectiveness of uncertainty quantification in identifying areas of lower model confidence, offering valuable insights for potential deployment scenarios where uncertainty estimates can help identify unreliable predictions.

However, the model's performance on the OOD data revealed some limitations. Ideally, OOD images would yield significantly higher variances than in-distribution images, clearly signaling the model's unfamiliarity with these examples. Our results showed that while variance was somewhat elevated for OOD data compared to the average image in the validation set, the difference was less pronounced than anticipated. This suggests that the uncertainty estimation mechanisms employed may not effectively distinguish between in- and out-of-distribution data.

In conclusion, our work demonstrates the feasibility of using uncertainty estimation in traffic sign recognition and highlights areas where these methods function as intended, as well as potential directions for improvement. The code to replicate our results can be found at `https://github.com/BjarkeHautop/project-deep-learning`.

References

Johannes Stallkamp, Marc Schlipsing, Jan Salmen, and Christian Igel. The german traffic sign recognition benchmark: A multi-class classification competition. In *The 2011 International Joint Conference on Neural Networks*, pages 1453–1460, 2011. doi:10.1109/IJCNN.2011.6033395.

Pierre Sermanet and Yann LeCun. Traffic sign recognition with multi-scale convolutional networks. In *The 2011 International Joint Conference on Neural Networks*, pages 2809–2813, 2011. doi:10.1109/IJCNN.2011.6033589.

Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1050–1059, New York, New York, USA, 20–22 Jun 2016. PMLR. URL `https://proceedings.mlr.press/v48/gal16.html`.

Kumar Shridhar, Felix Laumann, and Marcus Liwicki. A comprehensive guide to bayesian convolutional neural network with variational inference, 2019. URL `https://arxiv.org/abs/1901.02731`.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL `http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf`.

Ranganath Krishnan, Pi Esposito, and Mahesh Subedar. Bayesian-torch: Bayesian neural network layers for uncertainty estimation, January 2022. URL `https://doi.org/10.5281/zenodo.5908307`.

Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. *International Journal of Computer Vision*, 128(2):336–359, October 2019. ISSN 1573-1405. doi:10.1007/s11263-019-01228-7. URL `http://dx.doi.org/10.1007/s11263-019-01228-7`.

Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural networks, 2015. URL `https://arxiv.org/abs/1505.05424`.

M. Benatan, J. Gietema, and M. Schneider. *Enhancing Deep Learning with Bayesian Inference: Create more powerful, robust deep learning systems with Bayesian deep learning in Python*. Packt Publishing, 2023. ISBN 9781803237251. URL `https://books.google.dk/books?id=oELJEAAAQBAJ`.