EXPLAINABLE CLASSIFICATION OF X-RAY IMAGES USING BAYESIAN DEEP LEARNING

Sophie Kristine Vilter (s212403), Angelos Daglaroglou (s202915)

Technical University of Denmark
Department of Applied Mathematics and Computer Science

ABSTRACT

In this project, we use a Bayesian Neural Network to classify the body part present in an X-ray image, and use different methods to explain how our network made this classification. We will analyze our network's decision making process by obtaining posterior distributions for the network's last layer of weights, identifying the regions of the X-ray our network considers relevant in making this classification, and evaluating the certainty of the model weights. A convolutional neural network will be built and Class Activation Maps will be used to attribute relevance to features of the X-ray images. Uncertainty analysis will be incorporated via Bayesian Neural Network translation using Laplace Approximation. We find that the Class Activation Maps successfully aid in the interpretation of the networks decision making process. We also find how evaluating the uncertainty of a network's weights is significant in increase its explainability.

1. INTRODUCTION

The goal of of our project is to explore methods that improve the explainability of a neural network regarding its decision making process. Much of our methodology is influenced by the network explanation methods outlined in "Explaining Bayesian Neural Networks" in the context of classifying X-ray images [1].

A neural network is a computational learning system consisting of layers of nodes structurally designed to reflect that of the human brain in order to detect patterns for means of clustering and classification. The method by which a trained neural network generates classifications is considered a "black box" method. Due to their complex structure, there is a lack of transparency regarding which variables influence a given classification. Predictions can be considered unreliable, thus motivating the need to increase the interpretability of a neural network's decision making process. For our project, we will build a convolutional neural network. Using explainability methods discussed in "Explaining Bayesian Neural Networks", we will apply relevance attributions to model weights to interpret how the model makes classifications, and investigate how incorporating uncertainty analysis into our network impacts these interpretations.

2. DATA DESCRIPTION

The data used in this project is a set of musculoskeletal radiographs consisting of 14.863 studies of 12.173 patients provided by the Stanford ML Group. The MURA dataset is divided into 36.808 training images and 3.197 test images, contained in two separate folders. The images are further divided in their corresponding subfolders named after seven main upper body extremity types: elbow, finger, forearm, hand, humerus, shoulder, and wrist. Finally, the data is spilt in patients and then studies (multiple studies might exist for each patient). Each image's path is stored in separate text files (one for the training image paths and one for test), which are read by the Dataloader utility that comes with PyTorch, in order to import the images to our models.

3. METHODS

3.1. Convolution Neural Network

In this project, Pytorch is used to build a convolutional neural network, which is a class of neural networks commonly used for the analysis of visual imagery. The convolutional neural network is trained via maximum a-posteriori (MAP) inference. Assuming that we want to estimate an unobserved population parameter θ on the basis of observations x, the estimated value of weight θ is equal to the value of θ that maximises its posterior distribution. In other words, maximise the likelihood

$$\theta_{\text{MAP}} \quad = \quad \arg\max_{\theta} f(x \mid \theta) g(\theta)$$

where f is the sampling distribution of x so that $f(x \mid \theta)$ is the probability of x when the underlying population parameter is θ and $g(\theta)$ is the density function of θ.

3.2. Class Activation Map (CAM)

Class Activation Mapping is a technique used to identify the important regions a network uses to classify an image. The CAM functions utilizes Global Average Pooling to help attribute relevance to the network weights. The obtained features are fed into the fully connected neural network layer, and thus output the required probabilities. The importance of the weights with respect to a category can be found out by projecting back the weights onto the last convolution layer's feature map. An average is taken across all the activation maps which help us to find all the possible discriminative regions present in them.

The CAM utilizes K feature maps (essentially the number of the last hidden layer), Global Average Pooling turns a feature map into a single number by taking the average of the numbers in that feature map, thus:

$$\frac{1}{Z}\sum_i\sum_j A_{ij}^k$$

After performing Global Average Pooling, these feature maps are linearly transformed to produce a score y_c for each class c:

$$y_c = \sum_k w_k^c \frac{1}{Z}\sum_i\sum_j A_{ij}^k$$

Finally, to produce a localization map for class c, CAM computes the linear combination of the final feature maps using the learned weights of the final layer. This is normalized to lie between 0 and 1 for visualization purposes [2].

3.3. Bayesian Neural Network

A Bayesian neural network is a network that applies Bayes' theorem to each weight's prior distribution and likelihood in order to estimate their corresponding posterior distributions. The posterior distribution of weight θ is computed as follows:

$$\begin{aligned} p(\theta \mid x) &= \frac{p(x \mid \theta)p(\theta)}{p(x)} \\ &= \frac{p(x \mid \theta)p(\theta)}{\int p(x \mid \theta = t)p(\theta = t)dt} \end{aligned}$$

The integral in the denominator can not be computed directly, thus inducing the need for Laplace Approximation.

3.4. Laplace Approximation

In the context of neural networks, the Laplace method uses a Gaussian distribution centered at θ_{MAP} to estimate the posterior distribution of weight θ.

$$p(\theta|\mathcal{D}) \approx \mathcal{N}(\theta; \theta_{\text{MAP}}, \Sigma)$$

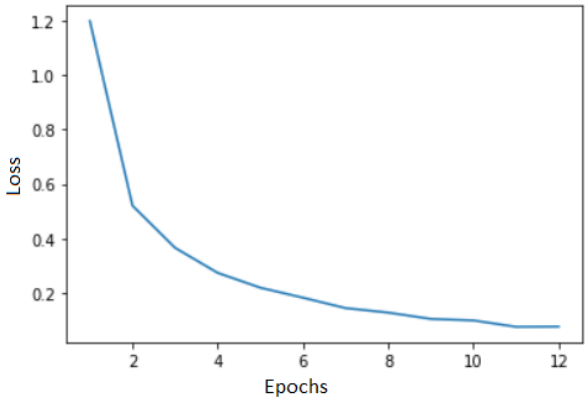

Fig. 1: Epoch loss of the CNN

where D is the original data set [3].

The covariance matrix Σ of this distribution is the negative inverse of the Hessian Matrix corresponding to the loss function evaluated at θ_{MAP} [4] . The Hessian will consist of the second order partial derivatives of the loss function, thus capturing the rate at which the loss is changing in different directions [5]. The Laplace library used in our project assumes a Gaussian prior $p(\theta) = \mathcal{N}(\theta; 0, \gamma^2 I)$ where γ is a parameter of the weight-decay regulariser induced by the neural network. The reported formula for the Hessian $H = \nabla_\theta^2 \mathcal{L}(\mathcal{D}; \theta) \mid_{\theta_{\text{MAP}}}$ is

$$H = -\gamma^{-2} I - \sum_{n=1}^{N} \nabla_\theta^2 \log p(y_n | f_\theta(x_n)) | \theta_{\text{MAP}}$$

Due to the volume of weights included in the network, It is not feasible to compute the Hessian directly using this formula. The Laplace library uses the positive semi-definite Fisher information matrix as the default choice to help approximate the Hessian. Formally, the Fisher information matrix is the covariance matrix of the score function $s(\theta) = \nabla_\theta \log p(x|\theta)$ [6]. The diagonal method approximates the Hessian as a diagonal matrix, where its entries are the diagonal entries of the Fisher matrix. The Kronecker method assumes the Hessian can be expressed as the Kronecker product of the final layer's covariance matrix and the Fisher matrix. The Hessian is then approximated finding the expected values of these matrices and taking their Kronecker product [7].

4. MODELS

4.1. Convolutional Neural Network Baseline

As a baseline model, a simple 2d Convolutional Neural Network that takes in a 224x224 image as input was used. It consists of 4 layers with a REctified Linear Unit (ReLU) activaton between them and a dropout rate of 0.2. Finally, the

Fig. 2: Sample images from the original MURA dataset

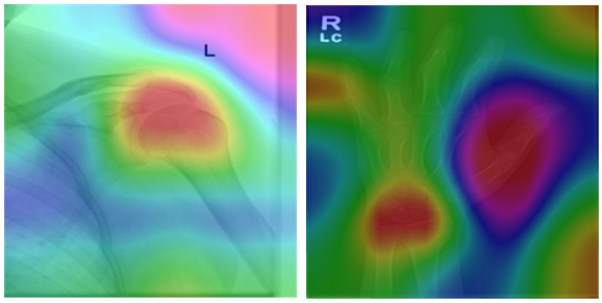

Fig. 3: Sample CAM images of a shoulder's (left) and a wrist's (right) x-ray.

cross entropy loss criterion was used and the Adam optimizer with a learning rate of 0.001 to tie the loss function with the parameters.

The activation function adds non-linearity to the model, while it is simple to calculate (compared to tanh or sigmoid activation function) and, most importantly, it has a derivative of 0 or 1 depending on if the input is negative or not. The latter has a big impact in backpropagation during training since it is computationally inexpensive.

The Dropout layer is a mask that nullifies the contribution of some neurons towards the next layer and leaves unmodified all others. Using dropout is important, since it prevents overfitting the data. Using dropout rate of 0.2 it simply means that during training 20% of the neurons are randomly ignored. This way, the training process becomes more noisy, reducing the weights that each neuron producing, therefore forcing the layers adapt to correct "mistakes" that previous layers made.

As said, the optimizer is the algorithm that ties the loss function with the parameters by updating the model weights in response to the output of the loss function. The Adam optimizer works by utilizing the concept of momentum, which is using fractions of the past gradients to the current. The logic is that the gradient should not be changed too much, in order to allow a careful search instead of going to the local optimum.

4.2. Laplace approximation models

After the convolutional neural network is trained, it is translated to Bayesian neural network using the Laplace approximation from the Laplace library. The convolutional neural network uses maximum a-posteriori estimate to calculate the weights of the model, which results in a single point estimate for each weight. We therefore lack information to conduct uncertainty analysis on these point estimates. Translating the convoultional neural network into a Bayesian neural network will insert the uncertainty inherent in the parameters of the network by inferring the full posterior disbritution $p(\theta|x)$. For the purposes of this study, three Laplace approximations were used; diagonal, full and Kronecker. The models inferred the weights of the last layer of the CNN model and fitted to the data. The weights are optimized by tuning hyper parameters post-hoc. The hyper parameters are the parameters required by the prior distribution and likelihood function used when training the weights via maximum a-posteriori estimation. They are tuned using cross-validation.

5. RESULTS

We utilised the class activation map technique to attribute relevance to the final layer of weights of our CNN. Once generated, the maps are combined with the original images, in order to understand which aspects of the image activate the neural network the most. In figure 3, we see two sample images from the dataset, one of a shoulder and one of a hand. In the hand's heatmap, we can see that the most instructive areas are the ones around the wrist and the thumb. The latter is also what discriminates it from the 'wrist' class. One thing worth mentioning is the fact that after seeing a few images, that the model is trained to focus more in the center of the image. For example, a wrist image might contain a finger close to the edge of the picture. However, this will not have a big impact in the model's decision and this is what provides the capability to the model to achieve high scores in those classes. These images tell us how the model identifies particular bones and joints in the X-ray images in order to identify the extremity present.

Next, we translated our CNN into a BNN using the diagonal, full, and kron methods to approximate the hessian. The accuracy, estimated calibration error, and negative log likelihood of each model are reported in Table 1.

Accuracy refers to the proportion of X-ray image correctly classified by the model. The accuracy, as seen in Table 1, remains virtually constant for each model. This outcome could be expected, since this classification problem is rather simple for our baseline model, which achieved a score of 87,5%.

The negative log likelihood is a loss function used in classification problems, given by a function similar to the Log Likelihood. The reason behind calculating the negative value

Model	Hessian Approximation	Accuracy	ECE	NLL	Fitting Runtime (hrs)	Optimization Time (hrs)
CNN	n/a	0.87519	0.07406	0.6465	n/a	n/a
BNN	Diagonal	0.87550	0.07269	0.5997	0.072	0.546
BNN	Full	0.87519	0.03033	0.4667	0.063	0.351
BNN	Kron	0.87519	0.02774	0.4751	0.074	0.709

Table 1: Comparison of results before applying Laplace approximation to the CNN and after.

is that it represents loss and therefore a minimization problem:

$$l(\theta) = -\sum_{i=1}^{n} \left(y_i \log \hat{y}_{\theta,i} + (1 - y_i) \log \left(1 - \hat{y}_{\theta,i}\right)\right)$$

We can see a substantial decrease in the negative log likelihood values of the Baysian neural networks, compared to the CNN. A decrease from approximately $0,65$ to $0,6$ is seen when applying the diagonal Hessian matrix. Furthermore, it is reduced lower to $0,467$ and $0,475$ for the full and Kronecker approximation retrospectively.

The runtime results of the training seem unexpected. Since the the Hessian matrix with the diagonal structure is simpler to calculate, it is expected that fitting and optimizing the corresponding BNN would be much faster with this method. However, the BNN with the diagonal approximation needed the almost the same amount of time to fit as the one with the Kronecker (and a little more than the one with the full Hessian) while the optimization time it needed was between them (Table 1).

The most interesting result of the Bayesian translation is the observed decrease in estimated calibration error. Estimated calibration error (ECE) is a statistic that evaluates the calibration capability of a classifier. In order to compute this statistic, the predictions are partitioned into K groups to calculate

$$\text{ECE} \quad = \quad \sum_{i=1}^{K} P(i) \cdot |o_i - e_i|$$

where $P(i)$ is the proportion of predictions in bin i, o_i is the average accuracy of the bin, and e_i is the average confidence [8]. In this paper, we use $K = 15$.

The confidence of a prediction reflects the likelihood that the generated prediction is correct.

Applying Laplace to our CNN using a diagonal representation of the Hessian successfully reduces the ECE from $0,074$ to $0,073$. The Full and Kronecker approximations, the more complex representation of the Hessian, notably further this reduction with ECE values of $0,03$ and $0,028$ respectively. As discussed, these Bayesian Neural Networks have learned a posterior distribution for the last layer of network weights.This provides a deeper understanding regarding the level of uncertainty surrounding the weight's point estimates. Convolutional neural network lacks this uncertainty analysis and therefore tend to be over-confident. The confidence the Bayesian neural networks have in their ability to make a correct classification, however, more accurately matches how often they make a correct classification in practice. The closer the average confidence in bin i is to its corresponding average accuracy, the smaller the term $|o_i - e_i|$ becomes. As this term decreases, the calculated expected calibration error decreases.

6. CONCLUSION

Using the class activation maps, we were successfully able to add transparency to our network's decision making process. The significant elements of the X-rays were the unique joints corresponding to the extremity in the image. For example, the CAM image of a shoulder in figure 3 identified acromion joint as a significant classification predictor. It is expected that the model would use the joints that differentiate extremities in order to generate classifications. As a result, we are more confident that our model is using appropriate features to classify these X-rays. We have therefore shown how understanding a neural network's classification method makes its corresponding predictions more reliable.

In addition, incorporating uncertainty analysis into our network via Bayesian translation showed how its explanation is dependent not only on how it makes predictions, but also on how confidently it manages to do so. The reduction in estimated calibration error is evidence of gap in the CNN's accuracy compared to its confidence. While we have identified the significant regions of the X-rays for classification using CAMs, the next step of this project would be to evaluate how certain our model is about these regions being more significant compared to others.

7. REFERENCES

[1] Kirill Bykov, Marina M.-C. Höhne, Adelaida Creosteanu, Klaus-Robert Müller, Frederick Klauschen, Shinichi Nakajima, and Marius Kloft, "Explaining bayesian neural networks," *CoRR*, vol. abs/2108.10346, 2021.

[2] Ramprasaath R Selvaraju, Abhishek Das, Ramakrishna Vedantam, Michael Cogswell, Devi Parikh, and Dhruv Batra, "Grad-cam: Why did you say that?," 2016.

[3] Erik Daxberger, Agustinus Kristiadi, Alexander Immer, Runa Eschenhagen, Matthias Bauer, and Philipp Hennig, "Laplace redux - effortless bayesian deep learning," in *Advances in Neural Information Processing Systems*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, Eds. 2021, vol. 34, pp. 20089–20103, Curran Associates, Inc.

[4] Runa Eschenhagen, Erik Daxberger, Philipp Hennig, and Agustinus Kristiadi, "Mixtures of laplace approximations for improved post-hoc uncertainty in deep learning," *arXiv preprint arXiv:2111.03577*, 2021.

[5] Hippolyt Ritter, Aleksandar Botev, and David Barber, "A scalable laplace approximation for neural networks," in *6th International Conference on Learning Representations, ICLR 2018-Conference Track Proceedings*. International Conference on Representation Learning, 2018, vol. 6.

[6] Agustinue Kristiadi, "Fisher information matrix," 2022.

[7] Mohammad Hadi Salari, *Kronecker-factored Hessian Approximation for Continual Learning*, Ph.D. thesis, 2019.

[8] Mahdi Pakdaman Naeini, Gregory Cooper, and Milos Hauskrecht, "Obtaining well calibrated probabilities using bayesian binning," in *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.