Team 42

Aerial Cactus Identification

Determine whether an image contains a columnar cactus

AVALLE Dario
EURECOM
avalled@eurecom.fr

FONTANA Umberto
EURECOM
fontana@eurecom.fr

SORBI Marco
EURECOM
sorbi@eurecom.fr

SPINA Gabriele
EURECOM
spina@eurecom.fr

***Abstract*—The VIGIA (Vigilancia Autónoma de Reservas de la Biósfera) project[1] in Mexico seeks to quantify the impact of human activities on protected natural areas and develop an autonomous surveillance system, beginning with the identification of a specific type of cactus in aerial imagery as part of a competition. We propose a solution to this challenge based on the Bayesian framework and compared the results with deterministic models. In the results section of this report, it will be shown how these probabilistic models better adapt to this challenge thanks to their ability to encode weight uncertainty and adapt to low cardinality datasets.**

I. Introduction

The goal of the challenge is to create a classifier able to predict whether a given aerial photo of a desertic environment contains or not a columnar cactus (Neobuxbaumia tetetzo). A standard approach to the task is represented by the usage of Convolutional Neural Networks (CNNs), for which computer vision has experienced remarkable progress. One key advantage of CNNs is the hierarchical learning of visual features, which is done automatically by extracting in the first moment the low-level features and then gradually building up to high-level concepts. Our idea was to start from well-established image classification models, such as ResNet [2], AlexNet [3] and ConvNeXt [4], then replicate them in a Bayesian framework to be compared with their classical version. We also designed a small toy network and tested the two approaches on it.

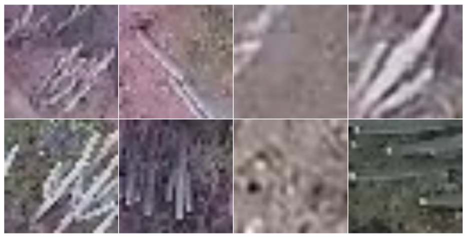


Fig. 1: Sample of 8 images from the original dataset

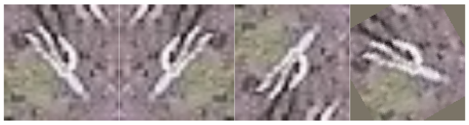


Fig. 2: From left to right: original image, horizontally flipped, vertically flipped, and rotated by 30°

II. Dataset and Preprocessing

The dataset consists of 17500 RGB images of size 32×32, of which 13136 have label 1 (cactus present) and 4364 have label 0 (cactus not present). From the dataset we created a train and a validation set, using a standard 80-20 split. Fig. 1 shows some examples of images. In order to increase the size of the dataset and the generalization ability of the model, we randomly applied several data augmentation techniques to the train set, such as horizontal and vertical flips and rotations of 30° at most, which can be seen in Fig. 2.

Finally, we computed the mean and the standard deviation of each channel, after decreasing the range of the values between 0 and 1. These values have been used to normalize the images prior to being fed into the models, except for ResNet50 which has its own normalization parameters computed on ImageNet.

III. Models

We began with the definition of our own baseline model consisting of a simple Convolutional Network (which we called DNet) structured with three convolutional and two fully connected layers, with a ReLU activation function inbetweens of each layer and a MaxPool module at the end of each of the last two convolutions. We then replicated the architecture of the AlexNet [3] and implemented the ConvNeXt block [4] in order to use it for building a new architecture. We additionally transposed these abovementioned architectures into a Bayesian framework and estimated the parameters of the models as distribution functions. As a term of comparison, a fine-tuned ResNet [2] architecture was also used.

A. ConvNeXt

Based on the work of Liu et al. [4], we created a network of 5 ConvNeXt blocks (depicted in Fig. 3) and a final fully-connected layer. The first convolutional layer has the special feature of being depthwise, that is, the convolution operation is applied separately on the channels. The last two layers, on the other hand, are pointwise, that is, they are 1x1 convolutions but on all channels; the first one increases the number of channels, while the second one reduces it to the original value. This technique is called Inverted Bottleneck [5]. By using only depthwise and pointwise convolutions, a property similar to vision Transformers is achieved, i.e., each operation either mixes information across spatial or channel dimensions, but not both. In addition, the block uses a skip connection, a layer normalization, and a GeLu activation function [6].

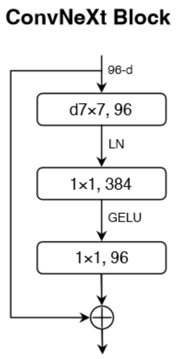


Fig. 3: ConvNeXt block

B. Convolutional Neural Networks with Variational Inference

Following the work of Shridhar et al. [7], we implemented a Bayesian Convolutional Neural Network. The advantage of this architecture is its power to estimate parameters not in a deterministic way, but as a form of probability distribution. This method of estimation is particularly effective as a form of regularization to the network that prevents overfitting, acting similarly to an ensemble of an infinite number of neural networks. The main idea is to use variational inference to learn the posterior distribution on the weights w distributed with variational probability distribution $q(w|D)$, which approximates the true posterior distribution $P(w|D)$. The definition of $q(w|D)$ should aim to be as similar as possible to the posterior $P(w|D)$ and this measure is computed using the Kullback-Leibler (KL) divergence that measures the difference between probability distributions. The optimal parameters are given by:

$$\begin{aligned}\theta^* = &\underset{\theta}{\operatorname{argmin}}\, KL[q(w|\theta)||P(w|D)] = \\ &\underset{\theta}{\operatorname{argmin}}\, \mathbb{E}_{q(w|\theta)}[\ln q(w|\theta)] - \mathbb{E}_{q(w|\theta)}[\ln P(w|D)] = \\ &\underset{\theta}{\operatorname{argmin}}\, \mathbb{E}_{q(w|\theta)}[\ln q(w|\theta)] - \mathbb{E}_{q(w|\theta)}[\ln P(D|w)] \\ &\quad - \mathbb{E}_{q(w|\theta)}[\ln P(w)] + \ln P(D)\end{aligned}$$

Rearranging the terms in the above equation we obtain

$$\theta^* = \underset{\theta}{\operatorname{argmin}}\, KL[q(w|\theta)||P(w)] - \mathbb{E}_{q(w|\theta)}[\ln P(D|w)]$$

The equation is mainly known as variational free energy or the expected lower bound (ELBO) on the marginal likelihood composed by the KL term and $\mathbb{E}_{q(w|\theta)}[\ln P(D|w)]$ called the likelihood cost. The term $\ln P(D)$ can be omitted in the optimization since it is constant. The likelihood cost can work as a regularization term that sometimes can be too dominant in the equation, leading to a very strong regularization on the model. Further research on this problem can be performed in the future. Minimizing this cost in an exact way is computationally prohibitive, also because of the KL-divergence term. In [8], they reformulate the exact cost as

$$F(D,\theta) \approx \sum_{i=1}^{n} (\ln q_\theta(w^{(i)}|D) - \ln P(w^{(i)}) - \ln P(D|w^{(i)}))$$

Where w_i is the i-th sample drawn from the variational posterior $q(w|D)$. Once defined this loss, it is possible to use the Backpropagation algorithm to estimate the optimal parameters θ^* of the posterior distribution of the weights. We applied this algorithm to recreate the proposed architectures in a Bayesian way.

TABLE I: AUC scores on the competition leaderboard

Model	AUC
ResNet	**0.9943**
DNet	0.9754
AlexNet	0.9430
ConvNeXt	0.9793
Bayesian DNet	**0.9904**
Bayesian AlexNet	0.9633
Bayesian ConvNeXt	0.9786

IV. Experiments and Results

A. Training and Validation

ResNet50 is the only pre-trained network, using a set of weights trained on ImageNet, and it has been fine-tuned by substituting only the last fully-connected layer. We trained all the other networks from scratch to be able to compare the classic version with the Bayesian one.

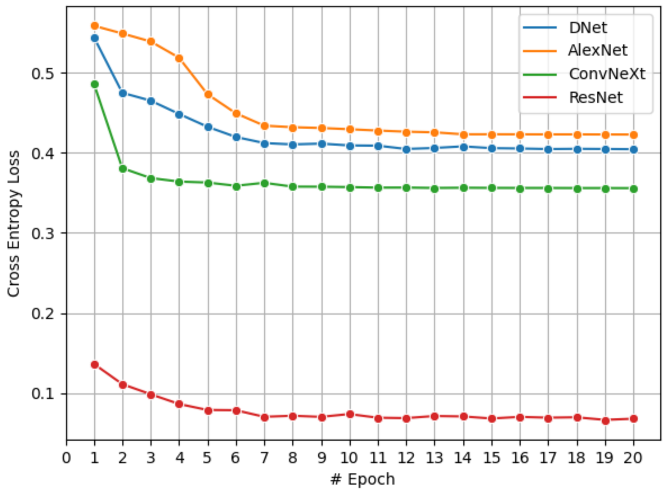
The models were trained using stochastic gradient descent with a learning rate of 0.001 and momentum of 0.9. Additionally, we used an exponential learning rate scheduler with a multiplicative factor of 0.1 and a period of decay of 7. The non-bayesian models use the Cross-Entropy Loss as loss function, while the Bayesian models rely on the ELBO loss defined as in the previous section. Each model has been trained for 20 epochs and with a batch size of 32: after each epoch, we evaluated the loss in the validation set and saved the set of model's parameters achieving the lowest loss. We finally used these best parameters for the evaluation on the Kaggle platform. Fig. 4 shows the validation losses of the models during the training procedure for the 20 epochs.

B. Results

Table I summarizes the score obtained by submitting the output of the softmax of the models on the competition leaderboard. The score measure is the Area Under the Receiver Operating Characteristic Curve (ROC AUC) computed from prediction scores.

ResNet50 is the model achieving the best score among all. This is not surprising since the model is pre-trained and because of its higher complexity. However, it is remarkable that the Bayesian version of DNet is able to achieve a score very close to the ResNet.

Moreover, we can see that the Bayesian versions of the models outperformed their deterministic versions, with the



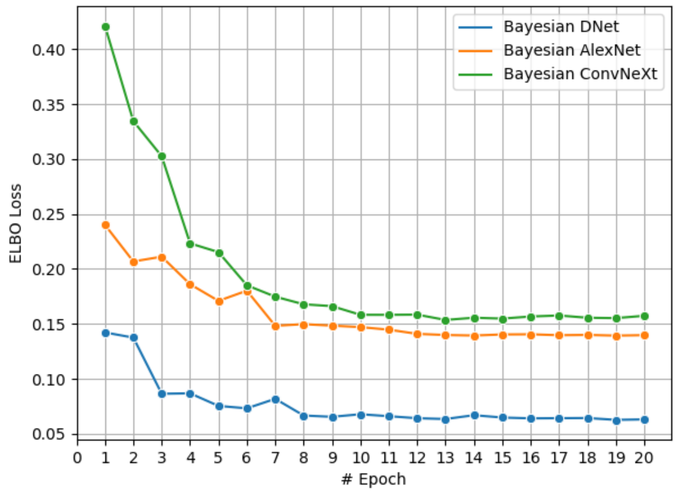


Fig. 4: Validation loss of the models during training. The models plotted on the left use the cross entropy loss, while the ones on the right use the ELBO loss

exception of ConvNeXt. This can be due to the robustness of the model to limited data. Indeed, learning weight distributions gives the possibility to encode weight uncertainty. This allows the network to make more reliable predictions even when data is scarce.

Fig. 5 shows the adjusting of the variational posterior of the Bayesian ConvNeXt weight during the training process. In these tasks, the guessing of the initial prior can be difficult, and could impact a lot the training procedure. Filippone et al. [9] proposed a method to estimate a good prior for the parameters using Gaussian Processes. In future work, it can be interesting to apply this method to the proposed architectures.

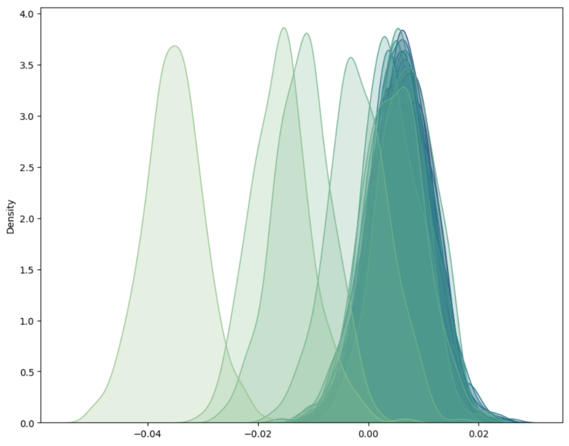


Fig. 5: Evolution of a density distribution of a given weight during the training: the leftmost graph is the prior, while the other distributions are estimated during a training epoch. The darkest one is the posterior after 20 epochs.

References

[1] A. S. A. Miguel, A. G. Eduardo, C. M. Rafael, M. C. L. Cresencio, E. L. Jiménez, and A. Uriarte, “Vigia: Vigilancia autónoma de reservas de la biósfera,”

[2] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” 2015.

[3] A. Krizhevsky, I. Sutskever, and E. G. Hinton, “Imagenet classification with deep convolutional neural networks,” 2012.

[4] Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell, and S. Xie, “A convnet for the 2020s,” 2022.

[5] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, “Mobilenetv2: Inverted residuals and linear bottlenecks,”

[6] D. Hendrycks and K. Gimpel, “Gaussiann error linear units (gelus), year=2016,”

[7] K. Shridhar, F. Laumann, and M. Liwicki, “A comprehensive guide to bayesian convolutional neural network with variational inference,” *arXiv preprint arXiv:1901.02731*, 2019.

[8] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, “Weight uncertainty in neural network,” 2015.

[9] B.-H. Tran, S. Rossi, D. Milios, and M. Filippone, “All You Need is a Good Functional Prior for Bayesian Deep Learning,” *Journal of Machine Learning Research*, vol. 23, pp. 1–56, 2022.