Proceedings of Machine Learning Research 1:1–21, 2022 Conformal and Probabilistic Prediction with Applications

Pruning neural networks for inductive conformal prediction

Xindi Zhao Xindi.Zhao@nottingham.edu.cn
Anthony Bellotti Anthony-graham.bellotti@nottingham.edu.cn
School of Computer Science, University of Nottingham Ningbo China

Abstract

Neural network pruning techniques are used to prune redundant parameters in overparameterized neural networks in order to compress the model size and reduce computational cost. The goal is to prune a neural network in such a way that it has the same, or nearly the same, predictive performance as the original. In this paper we study neural network pruning in the context of conformal prediction. In order to explore whether the neural network can be pruned while maintaining the predictive efficiency of conformal predictors, our work measures and compares the efficiency of the prediction sets provided by the inductive conformal predictor built with an underlying pruned neural network. We implement several existing pruning methods and propose a new pruning method based specifically on the conformal prediction framework. By evaluating with various neural network architectures and across several data sets, we find that the pruned network can maintain, or indeed improve, the efficiency of the conformal predictors up to a particular pruning ratio and this pruning ratio varies with different architectures and data sets. These results are instructive for deploying pruned neural network in real-work applications within the context of conformal predictors, where reliable predictions and reduced computational cost are relevant, e.g. in healthcare or safety-critical applications. This work is also relevant for further work applying continual learning techniques in the context of conformal predictors.

Keywords: Inductive conformal prediction; Neural networks; Pruning; Overparameterization

1. Introduction

Despite the success and advances of neural networks across various application domains, several questions about neural networks still remain poorly answered. One such open question is: what is the proper network size for a given task? A neural network with too few parameters may be insufficiently expressive to fit the data and to deal with a complex real-world problem; while a neural network with too many free parameters may tend to cause overfitting (Lawrence and Giles, 2000).

A promising approach to find an appropriate network size is network pruning. It selectively removes redundant parameters from a well-trained over-parameterized neural network until there is an intolerable loss in test accuracy. Studies show pruning can not only reduce both time and space requirements during predictive inference, and can also improve generalization (Urolagin et al., 2011). Due to these benefits, there is an increasing popularity to develop pruning algorithms. However, test accuracy is typically used as the only evaluation metric among these algorithms. In this regard, we cannot help casting a question: what is the *reliability* of these pruned neural networks?

Uncertainty measures are especially important in safety-critical decision-making systems powered by deep learning algorithms. For example, a disease diagnosis system which provides a confidence level for its predictions can help doctor's judgment when assessing its diagnoses. As another example, if a perception system in a self-driving car estimates the distance from the obstacle with uncertainty, a controller can take better control of the braking force so that it may safely and comfortably stop the car.

There are several approaches to measuring the uncertainty of the predictions of machine learning algorithms. The Bayesian neural network provides a natural way to measure the predictive uncertainty with posterior distribution (Kendall and Gal, 2017). However, both an incorrect assumption on the prior distribution over the model weights and the intractability of the true posterior can result in inaccurate uncertainty estimations. Moreover, it does not have general adaptabilities to all the network pruning methods, though several Bayesian learning-based pruning methods have been proposed. Probably Approximately Correct (PAC) learning does not have these drawbacks; however, in real world applications, it is difficult to produce tight error bounds.

Another approach to uncertainty estimation is conformal prediction (Vovk et al., 2005). Conformal predictors are constructed on top of any machine learning algorithm to provide each of its predictions with a statistically valid uncertainty measure. In particular, given a test object, conformal predictors produce a prediction set which is guaranteed to contain the true label with a user-predefined confidence level. To achieve the validity, it does not require any further assumption except for exchangeability on the data distribution, with the special case being the generally accepted independently and identically distributed (i.i.d.) assumption. Given a certain confidence level, the predictive efficiency, e.g. as may be measured by the size of the prediction set, is a crucial evaluation metric of conformal predictors. Generally speaking, at the same confidence level, the smaller the prediction set, the more efficient is the conformal predictor. The efficiency of a conformal predictor typically depends on the machine learning algorithm upon which it is built. The original conformal predictor, i.e., the Transductive Conformal Predictor (TCP), requires the underlying machine learning model to be retrained for each new test object, which leads to computation inefficiency and makes it unsuitable for the application of neural networks which typically require large amounts of data. Fortunately, its variant, the Inductive Conformal Predictor (ICP) (Papadopoulos et al., 2002) solves the problem. ICP requires the model to be trained only once while preserving the validity property. Therefore, we focus on ICP in this study.

This study measures the ICP predictive efficiency of the pruned neural network with several pruning methods on various data sets and network architectures. In addition, we propose a new pruning method based directly on ICP efficiency. The benefit of this work is to enable computationally efficient conformal prediction models, control of neural network complexity and additionally, relevance for conformal predictors to be used in continual learning settings which typically require a parameter tuning step. The arrangement of this paper is as follows. Further background on network pruning is provided in the following subsections. The ICP framework, the two existing pruning methods and our proposed puning method are introduced in Section 2. The details of data sets, experiment settings and results are given in Section 3. Finally, conclusions and limitations of our approach are discussed in Section 4.

1.1. Neural network pruning

Neural network pruning has been used to reduce redundancy and improve generalization for deep learning models (Han et al., 2015). An earliest work on pruning can be dated back to 1990's when LeCun et al. introduced this as Optimal Brain Damage (LeCun et al., 1989). Pruning can be applied on individual weights, hidden units, channels, filters and layers (Wen et al., 2016; Li et al., 2016; He et al., 2017). A typical pruning algorithm starts with a dense network and removes redundant parameters based on a certain criterion that can reflect the parameters' relative importance, either during or after training (Blalock et al., 2020). Fine-tuning the pruned network is required when pruning degrades the network performance (Blalock et al., 2020). Pruning and fine-tuning is sometimes iteratively repeated in order to further reduce network complexity (Han et al., 2015). Here, we focus on pruning a well-trained network and fine-tuning just once by retraining.

The pruning methods differ in their pruning criterion. Some methods prune parameters based on their gradient or Hessian information (LeCun et al., 1989; Hassibi et al., 1993; Wang et al., 2019). Others prune the parameters with relatively small magnitude (Li et al., 2016) and many of them use regularization such as Lasso (l_1), Ridge (l_2) or Group Lasso ($l_{2,1}$) weight regularization to force the network to learn smaller weights or shrink weights before pruning (Han et al., 2015; Wen et al., 2016; Lebedev and Lempitsky, 2016). Variational inference provides a Bayesian interpretation to regularization techniques (Williams, 1995). For example, Lasso is equivalent to specifying Laplace prior distribution on the individual weights and Ridge is equivalent to specifying Normal prior distribution on the individual weights. Therefore, some pruning methods (Molchanov et al., 2017; Louizos et al., 2017) are grounded in Bayesian statistics. They utilize Bayesian learning to learn sparsity in the neural network and prune the parameters that have the highest probabilities of being zero.

Considering that pruning imposes a trade-off between the model computational efficiency and predictive performance, it is preferable to characterize a pruning method with a computational efficiency-predictive performance curve (Wang et al., 2019) which shows the correlation between both properties on the same task. The computational efficiency is often quantified by pruning ratio or Floating-Point Operations (FLOPs). Pruning ratio refers to the ratio of the number of pruned parameters and the number of the parameters in the original model. FLOPs refers to the number of multiply-add operations that are used to perform predictive inference with the pruned network. The predictive performance of the model is quantified by test error rate or accuracy measured on point predictions given by the pruned network. However, none of these approaches measure the prediction uncertainty of the pruned model. Here, we propose to use prediction uncertainty, i.e., the size of label sets, as an evaluation metric.

1.2. Continual learning

Continual learning is a learning paradigm in which a single model is required to learn a sequence of tasks in an incremental way. The biggest challenge of continual learning is to overcome catastrophic forgetting. Catastrophic forgetting refers to a significant performance drop on the previous tasks after learning a new task and is caused by parameters being changed to meet a new objective (Kirkpatrick et al., 2017). A common approach to overcome catastrophic forgetting is to identify and isolate important parameters to a specific task and free those unimportant parameters for further learning. Hence, a pruning technique is often used as a crucial tool in continual learning algorithms (Mallya et al., 2018; Jung et al., 2020) to identify the important parameters, thus reduce catastrophic forgetting and improve computational efficiency in a neural network. An existing real application of continual learning is medical imaging (Hofmanninger et al., 2020). However, the current continual learning algorithms focus on point estimates. Conformal predictor based pruning techniques can be potentially useful in continual learning algorithms due to the demand for quantifying predictive uncertainty in medical image analysis (Ghesu et al., 2021), though it is not a focus of this study.

2. Methodology

The main purpose of this study is to evaluate the conformal predictor's performance after neural network pruning based on the existing pruning methods and to design a new pruning criterion based directly on conformal prediction objectives, in order to obtain a more robust or more efficient conformal predictor in the context of network pruning.

2.1. Inductive Conformal Predictor

The setting for inductive conformal predictor (ICP) is to define a problem where we wish to predict some outcome variable y from some given predictor variables $\boldsymbol{x}$ and split development data into training, calibration and test sets. The formalization is as follows.

- Let $\boldsymbol{z}_1, \cdots, \boldsymbol{z}_n$ be a sequence of examples $\boldsymbol{z}_i = (\boldsymbol{x}_i, y_i)$ with vector of m predictor variables $\boldsymbol{x}_i \in \mathbb{R}^m$ and response variable $y_i \in Y$ where Y is a set of labels.
- Without loss of generality, let 1 to k index training data, $k+1$ to l index calibration data and $l+1$ to n index test data, for $1 < k < l < n$.
- A conformity measure (CM) is some function $A(\boldsymbol{x}, y) = \mathcal{A}(\boldsymbol{z}_1, \cdots, \boldsymbol{z}_k, (\boldsymbol{x}, y))$ such that $\mathcal{A}$ is exchangeable: ie $\mathcal{A}(\boldsymbol{z}_1, \cdots, \boldsymbol{z}_k, \boldsymbol{z}_{k+1}) = \mathcal{A}(\boldsymbol{z}_{\pi(1)}, \cdots, \boldsymbol{z}_{\pi(k)}, \boldsymbol{z}_{k+1})$ for all permutations π of $1, \cdots, k$.
- Let $\alpha_i = A(\boldsymbol{x}_i, y_i)$ denote CM for observation i.

The ICP gives the prediction set for an input $\boldsymbol{x}$ at significance level ε as

$$\Gamma^{\varepsilon}(\boldsymbol{x}) = \left\{ y \in Y : \sum_{j=k+1}^{l} \mathbb{I}(A(\boldsymbol{x}, y) \geq \alpha_j) + 1 > \varepsilon(l-k+1) \right\} \tag{1}$$

where $\mathbb{I}$ is the indicator function. Assuming the sequence of examples are exchangeable, it has been shown that ICP predictions are valid (Vovk et al., 2005); ie for all test examples $i \in \{l+1, \cdots, n\}$,

$$\mathbb{P}(y_i \in \Gamma^{\varepsilon}(\boldsymbol{x}_i)) \geq 1 - \varepsilon. \tag{2}$$

If the learning problem is classification, i.e. so that $Y = \{1, \cdots, C\}$ is finite with C labels, then Equation (1) can be computed directly. However, if the learning problem is regression, i.e. $Y = \mathbb{R}$, then a normalized nonconformity measure is typically used (Papadopoulos et al., 2002). For this study, our experiments focus on classification problems. However, the pruning methods described in this study could also be applied to the problem of regression.

2.1.1. Performance Measures

Although Equation (2) guarantees validity, it is good practice to measure empirical validity on the test set, in case the exchangeability assumption is violated, e.g. due to selection bias or population drift, or in case of programming errors. If the ICP is behaving correctly, the errors on test data should be less than or approximately equal to ε:

$$\mathrm{Err} = \frac{1}{n-l} \sum_{i=l+1}^{n} \mathbb{I}\left(y_i \notin \Gamma^{\varepsilon}(\boldsymbol{x}_i)\right). \tag{3}$$

Given that this error measure is fixed, assuming the ICP is behaving correctly, the more important measure of performance for ICP is the *inefficiency* of predictions. In this paper we measure inefficiency based on the size of the prediction set, the larger the prediction set the less precision the predictor has, and hence has less usefulness as a predictor. However, other measures can be used (Vovk et al., 2016). In particular, on the test set,

$$\mathrm{Ineff} = \frac{1}{n-l} \sum_{i=l+1}^{n} |\Gamma^{\varepsilon}(\boldsymbol{x}_i)|. \tag{4}$$

For classification problems, the size is the cardinality of the prediction set, whereas for regression, using a normalized nonconformity measure, the prediction set is a prediction interval and then the size is the length of the prediction interval.

2.2. Neural network pruning methods

In this section, we will first introduce two existing pruning methods: weight magnitude-based pruning method and Bayesian-based pruning method. And then we will describe our proposed method: pruning based on conformal prediction efficiency. Firstly, we introduce some notation for neural networks and training data.

- Let f be a fully-connected neural network parameterized by a set of hidden units (neurons), $\mathcal{W} = \{\boldsymbol{w}_0, \cdots, \boldsymbol{w}_p\}$ with each

  $$\boldsymbol{w}_i = (w_{i,1}, \cdots, w_{i,\theta_i}) \in \mathbb{R}^{\theta_i}$$

  where θ_i is the feature dimension, P is the number of hidden units in the whole network and each $w_{i,j}$ is an input weight for unit i. We only consider pruning the hidden units in this work, as the number of output units is fixed by the task at hand. Therefore, we leave out the output units.

- Let $\mathcal{Z} = (\boldsymbol{z}_1, \cdots, \boldsymbol{z}_k)$ be the set of k i.i.d. training examples.

- Let $\mathcal{L}(\mathcal{Z}, \mathcal{W})$ be a loss function to be minimized with respect to neuron weights $\mathcal{W}$.

2.2.1. Pruning based on weight magnitude

Our first method is a variant of the method proposed by (Li et al., 2016). They prune neurons in the context of convolutional neural network by removing the neurons which have the smallest sum of the weight magnitude in each layer. Here, we focus on fully-connected neural network which is similar to a convolutional neural network. The neurons in both networks perform dot products and thus have the same functional form. Therefore, we adapted their method to fully-connected neural network. The only difference between our method and theirs is: they train a neural network without introducing any regularization; in our work, we add l_1 regularization in the objective function in order to penalize large weights. Therefore, $\mathcal{W}$ is learned by minimizing the following objective function,

$$\min_{\mathcal{W}} \mathcal{L}(\mathcal{Z}, \mathcal{W}) + \lambda \sum_{i=1}^{P} \sum_{j=1}^{\theta_i} |w_{i,j}|$$

where λ is the factor controlling the regularization strength and $|w_{i,j}|$ is the absolute value of $w_{i,j}$. After this objective function is minimized, the l_1 norm of each $\boldsymbol{w}_i$, denoted as s_i, is calculated as

$$s_i = \|\boldsymbol{w}_i\|_1 = \sum_{j=1}^{\theta_i} |w_{i,j}|$$

Following the method proposed by (Li et al., 2016), we sort the neurons within a single layer by s_i in ascending order. The neurons in the same layer have the same dimension θ_i so that we can have a fair comparison among their s_i values. Given a pruning ratio κ, i.e., the ratio of the number of pruned neurons and the number of total neurons P, we prune the top $\kappa \cdot P$ neurons in each layer.

2.2.2. Pruning based on Bayesian learning

In a Bayesian neural network, each weight is assumed to be taken from a known parametric distribution. Typically a Gaussian distribution is assumed and that is what we use in this study. Consequently, in a Bayesian neural network, the estimate for each weight $w_{i,j}$ is given by a mean $\mu_{i,j}$ and standard deviation $\sigma_{i,j}$.

To prune neuron weights in a Bayesian neural network, we use a standard normal prior distribution following (Nguyen et al., 2017),

$$p(\mathcal{W}) = \prod_{i=1}^{p} \prod_{j=1}^{\theta_i} \mathcal{N}(w_{i,j}|0,\, 1),$$

although alternatives have also been used such as hierarchical sparse priors (Louizos et al., 2017). The corresponding approximate posterior is chosen to be the most commonly used Gaussian mean-field approximate posterior,

$$q(\mathcal{W}|\phi) = \prod_{i=1}^{p} \prod_{j=1}^{\theta_i} \mathcal{N}(w_{i,j}|\mu_{i,j},\, \sigma_{i,j})$$

where $\mu_{i,j}$ and $\sigma_{i,j}$ is the mean and standard deviation of Gaussian distribution over weight $w_{i,j}$. The network is then trained on $\mathcal{Z}$ by minimizing objective function $\mathcal{L}(\mathcal{Z}, \phi)$ (also known as the Evidence Lower Bound (ELBO)) with respect to $\phi = \{\mu_{i,j}, \sigma_{i,j}\}_{i=1,\, j=1}^{p,\ \theta_i}$,

$$\mathcal{L}(\mathcal{Z}, \phi) = \mathbb{E}_{q(\mathcal{W}|\phi)}[-\log p(\mathcal{Z}|\mathcal{W})] + D_{kl}(q(\mathcal{W}|\phi) \mid\mid p(\mathcal{W}))$$

where $\mathbb{E}_{q(\mathcal{W}|\phi)}[\cdot]$ denotes the expected log-likelihood and $D_{kl}(\cdot)$ denotes KL-divergence.

Following Nguyen et al. (2017), we employ the BBB algorithm (Blundell et al., 2015) which combines Monte Carlo sampling and variational inference to enable backpropagation in the Bayesian neural network to train the network and use the reparameterization trick (Kingma and Welling, 2013) to compute the gradients. After training the network, we prune the weights by their signal-to-noise ratio given by $|\mu_{i,j}|/\sigma_{i,j}$ as suggested in (Graves, 2011; Blundell et al., 2015), since a lower value indicates that the true weight is closer to zero, and much of the estimate is "noise" given as variance in the posterior distribution. Notice that for this pruning method, the pruning is at weight level, rather than unit (neuron) level, since it is in the weight parameter that the posterior distribution is computed.

2.2.3. Pruning based on Conformal Prediction (CP) Efficiency

The intuition behind our method is to keep a hidden unit that results in much lower predictive efficiency when it is removed, and prune the ones that do not change predictive efficiency significantly or result in higher efficiency when removed.

Without loss of generality, we first divide $\mathcal{Z}$ into two disjoint subsets $\mathcal{Z}^t = (\boldsymbol{z}_1, \cdots, \boldsymbol{z}_\nu)$ and $\mathcal{Z}^v = (\boldsymbol{z}_{\nu+1}, \cdots, \boldsymbol{z}_k)$ to create a validation set $\mathcal{Z}^v$ for the pruning process, that is independent of the data $\mathcal{Z}^t$ used to train the neural network. In real-world problems, it could be that the training and validation set are the same. However, this may introduce overfitting in the pruning process. Additionally, consistent with the notation in Section 2.1, we have the proper calibration set $\mathcal{Z}^c = (\boldsymbol{z}_{k+1}, \cdots, \boldsymbol{z}_l)$.

1. We train a neural network f with just $\mathcal{Z}^t$.
2. Compute $\alpha_i = A(\boldsymbol{x}_i, y_i)$ for each $(\boldsymbol{x}_i, y_i) \in \mathcal{Z}^c$. The conformity measure A used in our method is,

$$A(\boldsymbol{x}_i, y_i) = \frac{o_u^i + \gamma}{\max\limits_{j=1,\dots C: j\neq u} o_j^i}$$

   where $y_i = u \in \{1,\ 2, \dots,\ C\}$ and o_u^i is the value given by the uth output neuron when feeding $\boldsymbol{x}_i$ to the trained network, the denominator is the maximum value amongst the values given by the other output neurons, as proposed by (Papadopoulos et al., 2007). Here, we set $\gamma = 0$ for simplicity. Using a softmax output unit, $o_i^j \in (0, 1)$ and this ensures that the denominator is non-zero.

3. For each input vector $\boldsymbol{x}_{\nu+i}$ in Z^v, we assume its true label is unknown and calculate its prediction set with Equation (1). As a result, we obtain a sequence of $k-\nu$ prediction sets, $\Gamma^\varepsilon(\boldsymbol{x}_{\nu+1}), \cdots, \Gamma^\varepsilon(\boldsymbol{x}_k)$. We measure the average width of the prediction set using Equation (4), i.e.,

$$a_0 = \frac{1}{k-\nu} \sum_{i=1}^{k-\nu} |\Gamma^\varepsilon(\boldsymbol{x}_{\nu+i})|$$

We refer to a_0 as the efficiency baseline. For classification, the optimal prediction set size is 1, since an empty prediction is non-informative. However, if the significance level ε is sufficiently low or the learning problem sufficiently hard, this is unlikely to happen, as can be seen from Equation (1). On the other hand, if empty sets are expected to occur, then the ICP can be set up to make *forced* predictions, which means the prediction set always includes the label with highest conformity score. Conservative validity Equation (2) still holds in this case.

In addition to efficiency, we compute the error rate to check validity using Equation (3).

4. For each $\boldsymbol{w}_k \in \mathcal{W}$, we consider removing $\boldsymbol{w}_k$ and keep the other hidden units unchanged:

   - Recompute the conformity scores for each example $(\boldsymbol{x}_i, y_i) \in \mathcal{Z}^c$ with the neural network without hidden unit k
   - Repeat step 3 to compute a_k

   As a consequence, we obtain a sequence $a_1, \cdots, a_p$.

5. We propose two pruning criteria based on the differences measure between the efficiency baseline and the efficiency after pruning one neuron:

   - absolute difference: $\delta_i^a = |a_i - a_0|$
   - sign difference: $\delta_i^s = a_i - a_0$

   This is because we observe that pruning some neurons does not change efficiency significantly while pruning some neurons causes lower efficiency in the experiment. We consider that the former does not make any contribution to efficiency and the latter is detrimental to efficiency. We are going to compare and discuss the effectiveness of these two criteria in Section 3.2.

6. The top κ % of hidden units with smallest values of δ_i are pruned, since these represent those neurons whose removal makes little impact or, indeed, improves predictive efficiency in the case of sign difference.

In the current setting for this study, we use the calibration set in step 3 above which means that validity does not strictly hold in the sense of Equation (2). However, our experimental results demonstrate that empirical validity does hold. The process could easily be made strictly valid by allowing the use of a separate calibration set in step 3, so that $\mathcal{Z}^c$ remains independent; indeed, the training or validation set could be used since validity is not required during the training process, only during the testing and operational stage.

3. Experiments and Results

We present experiments in two parts. First, in Section 3.1, we provide details on the experiment datasets and setup. Second, in Section 3.2, we exhibit how the inefficiency of the conformal predictor varies with the pruning ratio and discuss the results.

3.1. Experiments

We used Tensorflow and Tensorflow Probability packages to build neural networks and Bayesian neural networks separately. We used the Python implementation of the ICP framework provided by (Linusson, 2015).

3.1.1. Data sets

We demonstrate our method on three datasets:

Covtype The Covtype dataset (Blackard and Dean, 1999) is a classification dataset that describes 581,012 forest covers with 54 cartographic variables and their 7 species (Spruce/Fir, Lodgepole Pine, Ponderosa Pine, Cottonwood/Willow, Aspen, Douglas-fir, Krummholz) as outcome. The dataset is available from Scikit-learn.

MNIST Our second dataset is the well-known handwritten digits dataset. It consists of 70,000 28×28 greyscale images of handwritten digits 0-9 with pixel values in the range [0, 255]. Among them, 60,000 are used for training and 10,000 are used for testing. It is available from Tensorflow.

FashionMNIST The fashionMNIST dataset (Xiao et al., 2017) is an image classification dataset, containing 28×28 grayscale images of 10 fashion products: T-shirt/top, Trousers, Pullover, Dress, Coat, Sandal, Shirt, Sneaker, Bag, and Ankle boot. It includes 60,000 training examples and 10,000 test examples. It is also available from Tensorflow.

For the Covtype dataset, as a pre-processing step we normalized all the features to have mean 0 and standard deviation 1. We randomly permuted the original dataset and divided it into quarters for $\mathcal{Z}^t, \mathcal{Z}^c, \mathcal{Z}^v$ and $\mathcal{Z}^{te}$ respectively for the CP efficiency-based pruning method. In the experiments with the magnitude-based and Bayesian-based pruning method, we abandon $\mathcal{Z}^v$ and used $\mathcal{Z}^t$ as proper training set and $\mathcal{Z}^c$ as calibration set to implement ICP.

For MNIST and fashionMNIST dataset, we converted the 28×28 images to a 784-dimension feature vector in order to feed them into a fully-connected neural network. Also, as a pre-processing step, we normalized all the features to have mean 0 and standard deviation 1. We randomly permuted the original training set and divided it into three equal-sized sets for $\mathcal{Z}^t, \mathcal{Z}^c, \mathcal{Z}^v$ and used the original test set given by the data providers as our test set $\mathcal{Z}^{te}$. In the experiments with the magnitude-based and Bayesian-based pruning methods, we abandoned $\mathcal{Z}^v$ and used $\mathcal{Z}^t$ as proper training set and $\mathcal{Z}^c$ as calibration set to implement ICP.

Table 1 summarizes the information of these three datasets including the number of their features, classes and examples contained in four subsets. Only Covtype exhibits class-imbalance with two major classes representing just over 85% of examples.

3.1.2. Experimental setup

Considering the size of each dataset, we used two overparameterized (the number of parameters is much larger than the number of training examples) fully-connected neural network

Table 1: Dataset Information. # represents the number of observations in each set.

Dataset	**Features**	**Classes**	$\#\mathcal{Z}^t$	$\#\mathcal{Z}^c$	$\#\mathcal{Z}^v$	$\#\mathcal{Z}^{te}$
Covtype	54	7	145,253	145,253	145,253	145,253
MNIST	784	10	20,000	20,000	20,000	10,000
fashionMNIST	784	10	20,000	20,000	20,000	10,000

architectures with different numbers of hidden neurons and layers to assess the impact of pruning in the context of conformal prediction:

- two hidden layers with number of units in each layer: 500 - 300
- three hidden layers with number of units in each layer: 1024 - 1024 - 512

We selected the former architecture in order to be consistent with the previous study (Wen et al., 2016). Considering there is no sufficient study on pruning deeper fully connected neural network, we specify the latter architecture by ourselves taking the network architecture used in (Srivastava et al., 2014) as a reference.

We implemented these two architectures on the three datasets separately. Furthermore, we employed another 3-layer network architecture 128 - 128 - 128 for Covtype dataset since it represents an under-parameterized (the number of parameters is much less than the number of training examples) network for Covtype and so allows us to explore that scenario.

For each data set, pruning methods are applied with and without post-pruning fine-tuning. For this study, fine-tuning means retraining the neural network on the original training data, initializing with the original pre-pruning weights given in the network. If no fine-tuning (i.e. retraining) is done then this means the pruned network just uses the pre-pruning weights originally estimated.

All the models used rectified linear units (ReLU) in the hidden layers and a softmax output layer. The training epoch was empirically determined by observing the objective function converges, the fine-tune epoch was 1/8 of the original training epochs. We considered learning rates of $1 \times 10^{-2}, 1 \times 10^{-3}$ and 1×10^{-4} with minibatches of size 64 and used the one that yielded the most efficient conformal predictor. For the magnitude-based method and the CP-efficiency based method, we trained the models for 40 epochs with a constant learning rate of 1×10^{-3} and fine-tuned the pruned model for 5 epochs with the same learning rate. For magnitude-based pruning method, we only used regularization during training and did not use regularization during fine-tuning. We also ran a grid search for the l_1 regularization strength hyperparameter λ over $\{1 \times 10^{-6}, 5 \times 10^{-6}, 1 \times 10^{-5}, 5 \times 10^{-5}, 1 \times 10^{-4}, 5 \times 10^{-4}, 1 \times 10^{-3}, 5 \times 10^{-3}, 0.01, 0.05, 0.1, 0.5\}$ for magnitude-based method. For the Bayesian-based pruning method, we trained the Bayesian neural network for 100 epochs with a constant learning rate of 1×10^{-4} and fine-tuned the pruned model for 12 epochs with the same learning rate.

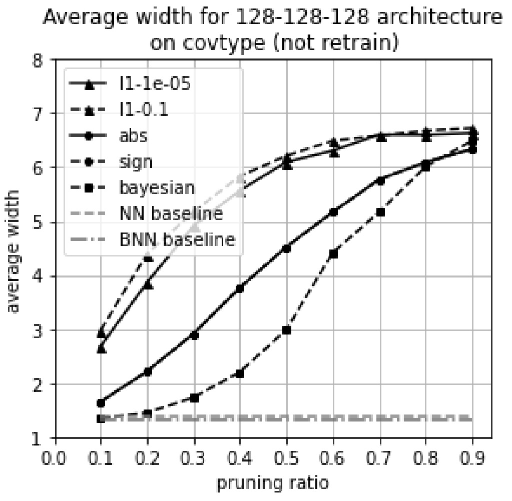
We employed the Adam (Kingma and Ba, 2014) optimizer for all the models during training. When implementing inductive conformal predictors on top of these neural networks, we predefined a $\varepsilon = 0.01$ significance level for all the experiments, considering that the classification on MNIST is a simple task, if the significance level higher than 0.01, the average width will be less than 1; if the significance level lower than 0.01, it will be a strict setting to another two data sets. We performed 8 runs for each experiment with each run using a different random permutation of the data set, to observe the robustness of the networks and pruning methods.

3.2. Results

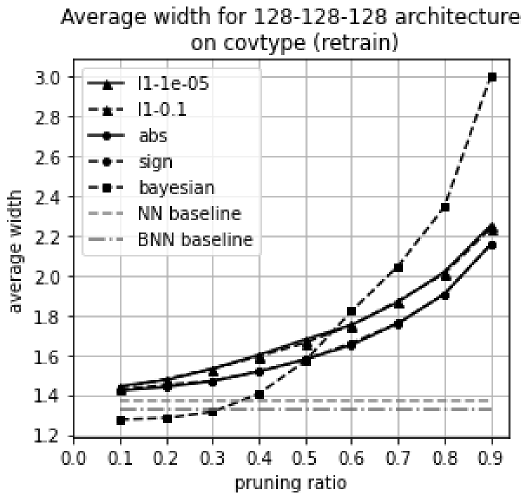
Firstly, we observed that the error rate in each experiment was close to the predefined significance level, hence demonstrating that validity is achieved. Hence, we will focus our discussion on predictive efficiency. The results of the ICP efficiency measured after pruning for all datasets with different network architectures are shown in Figures 1 to 7. Figures 1(a) to 7(a) show the results of pruning without retraining (no fine-tuning); Figures 1(b) to 7(b) show the results of pruning with retraining (fine-tuning). All the results are averaged over 8 runs. We choose two baselines for each figure: 'NN baseline' refers to the mean widths measured with a standard neural network trained from scratch prior to pruning; 'BNN baseline' refers to the mean widths measured with Bayesian neural network prior to pruning. The 'Bayesian' line represents the mean widths measured with Bayesian neural network after pruning. As mentioned above, we did grid search for l_1 regularization strength hyperparameter and we did not observe any significant difference in their results, therefore, we only show two of the most divergent lines in each figure; e.g. $\lambda = 10^{-5}$ and 0.1 for Covtype in Figure 1. The 'abs' line in each figure represents the CP-efficiency based pruning method with absolute differences and the 'sign' line in each figure represents the CP-efficiency based pruning method with sign differences for δ_i.

Comparison between retraining and not retraining Comparing Figures 1(a) to 7(a) and Figures 1(b) to 7(b), note that the scales of Figures 1(a) to 7(a) and those of Figures 1(b) to 7(b) are different. It is clear that pruning without retraining increases the mean widths of the prediction sets, while pruning with retraining can lead to decreasing the mean widths, suggesting that pruning is not just a step to compress the network but to improve predictive efficiency too.

Comparison among pruning methods The performance of the three methods varies with the different learning tasks. For Covtype dataset, whose results are shown in Figures 1, 2 and 3, Bayesian-based pruning method yields the smallest mean widths until the pruning ratio reaching a particular value, with and without retraining, as can be seen in the figure. Moreover, retraining decreases the ICP mean widths measured with Bayesian-based method below the BNN baseline until the pruning ratio reaches a particular value as shown in the graph. Nevertheless, there is a sharp rise in mean width afterwards. The particular pruning ratio value at which this happens increases with the size of network: 0.3 for 128-128-128, 0.5 for 500-300 and 0.8 for the 1024-1024-512 architecture. The CP-efficiency-based method cannot maintain baseline efficiency except for the large network size, 1024-1024-512. From Figure 3(a), 'abs' method can maintain the mean width when pruning ratio is less than 0.3; and from Figure 3(b), 'abs' and 'sign' methods achieve mean widths lower than NN baseline when pruning ratio is 0.6 to 0.8. The l_1-norm (magnitude) based method yields the largest mean widths, i.e., the lowest efficiency (highest mean width), and is the most sensitive to pruning.

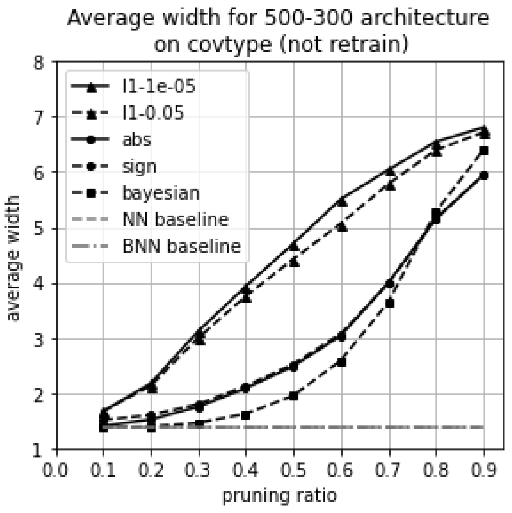


(*a*) pruning without retraining

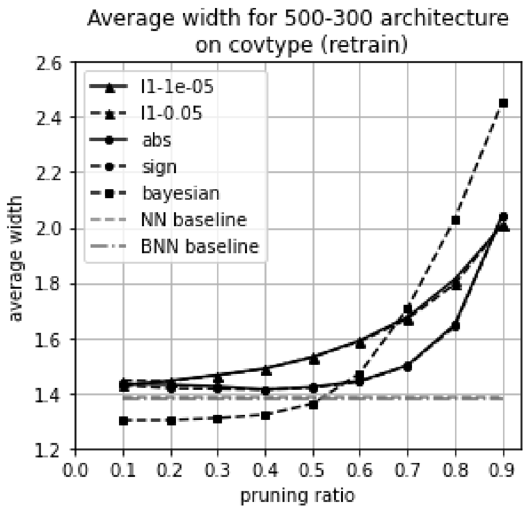


(*b*) pruning with retraining

Figure 1: Comparison results between methods for Covtype with 128-128-128 NN

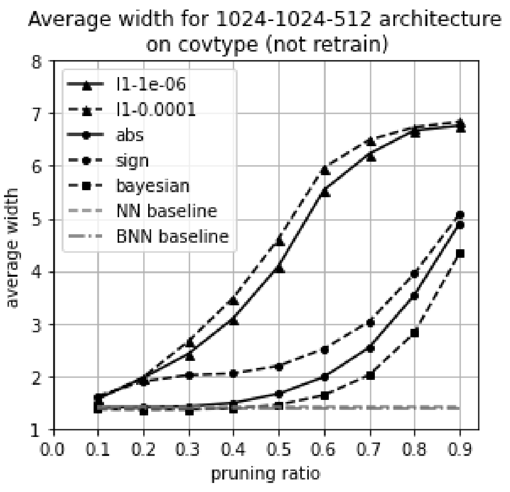


(*a*) pruning without retraining

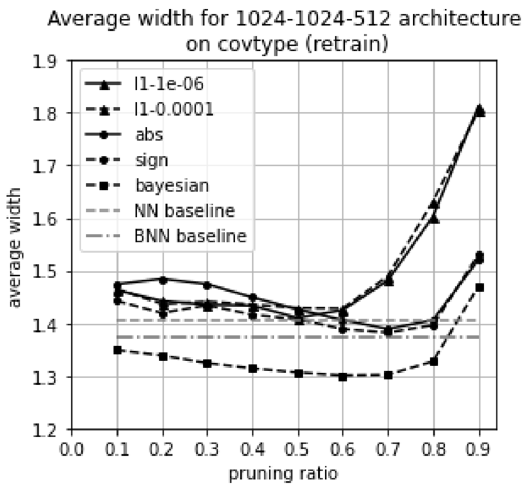


(*b*) pruning with retraining

Figure 2: Comparison results between methods for Covtype with 500-300 NN

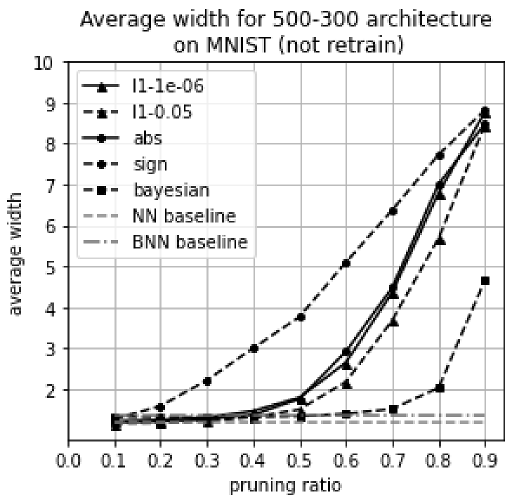


(*a*) pruning without retraining

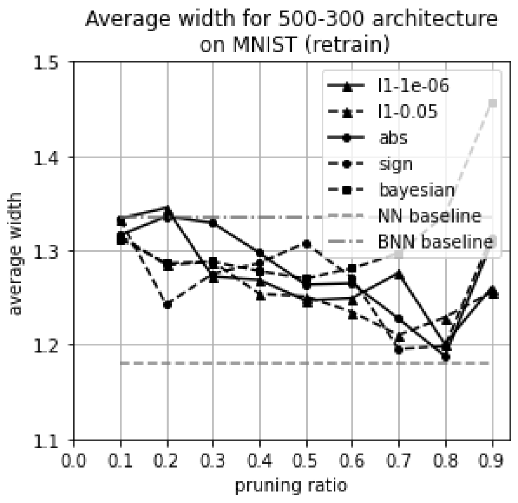


(*b*) pruning with retraining

Figure 3: Comparison results between methods for Covtype with 1024-1024-512 NN

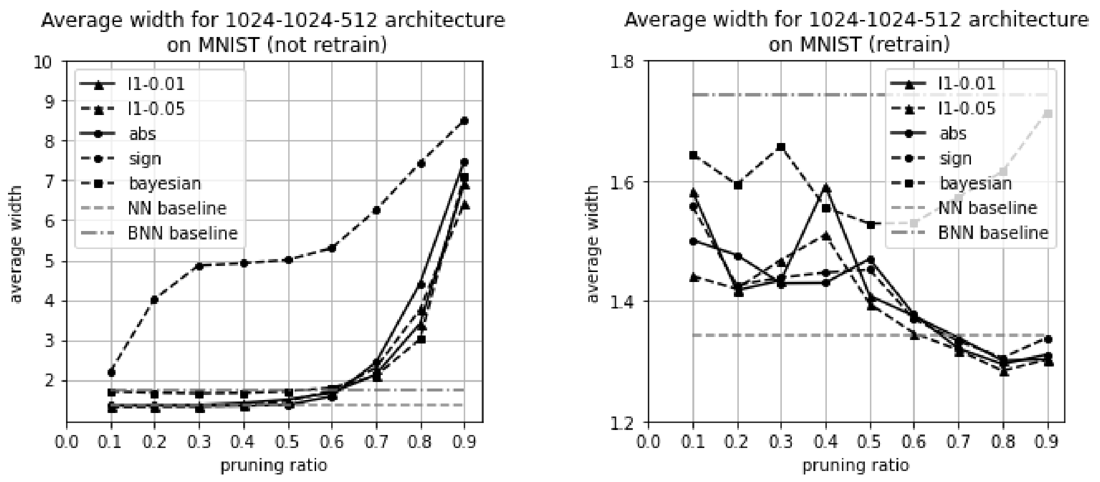


(*a*) pruning without retraining



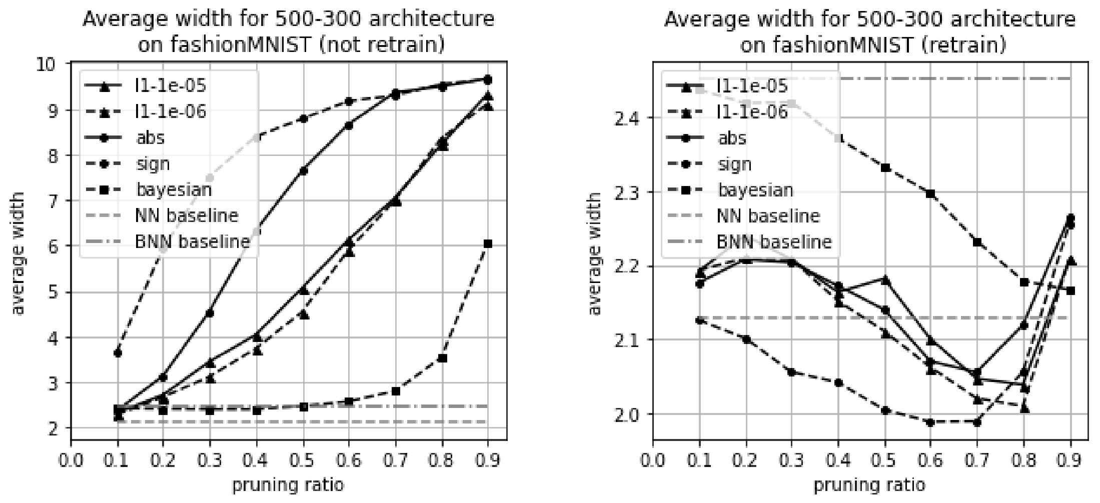
(*b*) pruning with retraining

Figure 4: Comparison results between methods for MNIST with 500-300 NN



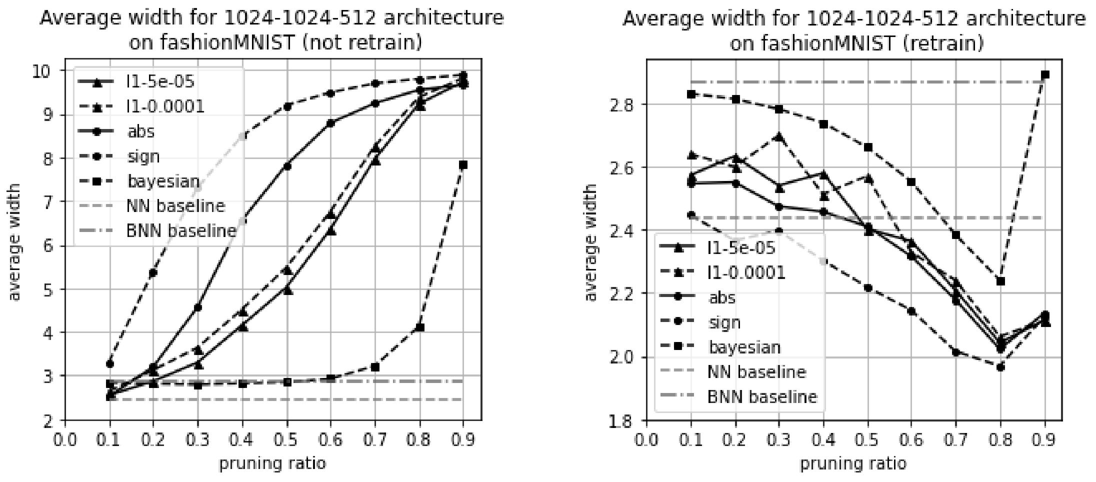
(a) pruning without retraining (b) pruning with retraining

Figure 5: Comparison results between methods for MNIST with 1024-1024-512 NN



(a) pruning without retraining (b) pruning with retraining

Figure 6: Comparison results between methods for fashionMNIST with 500-300 NN



(a) pruning without retraining (b) pruning with retraining

Figure 7: Comparison results between methods for fashionMNIST with 1024-1024-512 NN

For MNIST dataset whose results are shown in Figures 4 and 5, in pruning without retraining, all the methods except for the 'sign' CP-efficiency method can maintain almost the same mean widths as the baseline until the pruning ratio reaching a particular value. While when pruning followed by retraining, all the methods except for Bayesian-based pruning method fluctuate above the NN baseline until the pruning ratio reaches a particular value, i.e., 0.7 or 0.8. For the 500-300 network architecture, the ICP mean widths measured with pruned neural network are larger than but very close to the NN baseline. For the 1024-1024-512 network architecture, when the pruning ratio exceeds 0.7, the mean widths measured with pruned neural networks are smaller than the NN baseline. Retraining makes mean widths on Bayesian-based pruning method decrease below BNN baseline. Nevertheless, even the minimum of its mean widths is still higher than NN baseline.

Results on fashionMNIST dataset are shown in Figures 6 and 7. From Figures 6(a) and 7(a), the Bayesian-based pruning method maintains the same mean widths as the BNN baseline until the pruning ratio reaches 0.5. Other pruning methods cannot maintain the ICP efficiency when there is no retraining, especially the 'sign' CP-efficiency-based method which is the most sensitive to pruning. While it is totally contrary in the case of retraining: the 'sign' CP-efficiency-based method provides the smallest mean widths almost across all the pruning ratios; Bayesian-based pruning method provides the largest mean widths due to its relatively high baseline, nevertheless the mean widths are reduced following the retraining.

In addition, it is worth mentioning that in Figure 1, the 'abs' line and the 'sign' line overlap and the divergence between these two lines increases as network size increases, as in Figures 2 and 3. This is because a neural network with 128-128-128 architecture is an under-parameterized model for the Covtype dataset, and all the units are dedicated to the learning task, and no hidden unit leads to lower mean widths when removing it from the network, which means the sign difference equals the absolute difference, in this case.

Table 2: The minimum mean widths (with standard deviation) on Covtype

Covtype			
architecture	method	minimum mean width	pruning ratio at minimum mean width
128-128-128	NN baseline	1.37505 (0.01030)	n/a
	BNN baseline	1.32806 (0.02491)	n/a
	l1 ($\lambda = 10^{-5}$)	1.44397 (0.01110)	10%
	l1 (λ=0.1)	1.43552 (0.00964)	10%
	Bayesian	**1.27543** (0.00925)	10%
	cp-abs	1.42125 (0.01002)	10%
	cp-sign	1.42332 (0.02999)	10%
500-300	NN baseline	1.38539 (0.01170)	n/a
	BNN baseline	1.37702 (0.00753)	n/a
	l1 ($\lambda = 10^{-5}$)	1.42787 (0.00930)	10%
	l1 (λ=0.5)	1.43151 (0.00898)	10%
	Bayesian	1.30139 (0.00774)	10%
	cp-abs	1.41302 (0.01413)	10%
	cp-sign	1.41269 (0.02083)	40%
1024-1024-512	NN baseline	1.40547 (0.02163)	n/a
	BNN baseline	1.37447 (0.00282)	n/a
	l1 ($\lambda = 10^{-6}$)	1.41015 (0.01319)	50%
	l1 (λ=0.0001)	1.42803 (0.01200)	60%
	Bayesian	1.30066 (0.00486)	60%
	cp-abs	1.38913 (0.01156)	70%
	cp-sign	1.38211 (0.01265)	70%

Table 3: The minimum mean widths (with standard deviation) on MNIST

MNIST			
architecture	method	minimum mean width	pruning ratio at minimum mean width
500-300	NN baseline	**1.18032** (0.03135)	n/a
	BNN baseline	1.33574 (0.02256)	n/a
	l1 (λ=0.05)	1.21036 (0.03116)	70%
	l1 ($\lambda = 10^{-6}$)	1.20074 (0.05597)	80%
	Bayesian	1.26992 (0.02501)	50%
	cp-abs	1.18804 (0.04368)	80%
	cp-sign	1.19561 (0.07397)	70%
1024-1024-512	NN baseline	1.34383 (0.02986)	n/a
	BNN baseline	1.74208 (0.10599)	n/a
	l1 (λ=0.01)	1.30101 (0.03585)	80%
	l1 (λ=0.05)	1.28373 (0.07185)	80%
	Bayesian	1.52877 (0.05533)	50%
	cp-abs	1.29511 (0.08846)	80%
	cp-sign	1.30568 (0.07745)	80%

Table 4: The minimum mean widths (with standard deviation) on fashionMNIST

fashionMNIST			
architecture	method	minimum mean width	pruning ratio at minimum mean width
500-300	NN baseline	2.12835 (0.06237)	n/a
	BNN baseline	2.45074 (0.04785)	n/a
	l1 ($\lambda = 10^{-5}$)	2.03866 (0.04285)	80%
	l1 ($\lambda = 10^{-6}$)	2.00965 (0.02368)	80%
	Bayesian	2.16652 (0.02711)	90%
	cp-abs	2.05575 (0.05722)	70%
	cp-sign	1.98874 (0.03973)	60%
1024-1024-512	NN baseline	2.43918 (0.24328)	n/a
	BNN baseline	2.86497 (0.05235)	n/a
	l1 ($\lambda = 5 \times 10^{-5}$)	2.0426 (0.07892)	80%
	l1 (λ=0.0001)	2.06191 (0.03605)	80%
	Bayesian	2.24010 (0.03007)	80%
	cp-abs	2.02215 (0.05716)	80%
	cp-sign	**1.96863** (0.02983)	80%

Generalization in over-parameterized neural networks A general assumption behind the current pruning approaches is that the neural network is overparameterized. Classical thinking is that overparameterized neural network leads to bad generalization and thus pruning improves generalization by reducing model complexity (Urolagin et al., 2011). However, there is an argument in favour of overparameterized neural network: a recent idea is that overparameterized neural network generalize well because of the "denoising-like effect" of overparameterization itself (Chang et al., 2020).

In the context of conformal prediction, this study's empirical results support the former classical thought. Tables 2, 3 and 4 separately summarize the minimum mean width and the corresponding pruning ratio given by different pruning methods and the two baselines on Covtype, MNIST and fashionMNIST datasets in the case of pruning with retraining. In Table 2, we can observe that a large network size (1024-1024-512) leads to a larger improvement in mean width from the baseline (1.40547), compared with small networks (1.37505 for 500-300 and 1.38539 for 128-128-128). In Table 3, the best result (1.18032) is given by 500-300 architecture 'NN baseline' and the CP-efficiency pruning method with 'abs' gives a comparable result (1.18804). For 1024-1024-512 architecture, the mean widths given by pruning methods are smaller than corresponding baselines. However, even its best result (1.28373 given by $l_1(\lambda = 0.01)$) is larger than 500-300 NN baseline. In Table 4, both NN baseline and BNN baseline given by 1024-1024-512 network architecture are worse than 500-300's. However, the smallest mean width (1.96863) of 1024-1024-512 network is smaller than 500-300 architecture's (1.98874), which benefits from pruning.

While our results tend to support the former classical view in the context of conformal prediction, for a definitive conclusion, further research work is required.

4. Conclusion

In this paper, we measured the predictive efficiency of pruned neural networks as underlying machine learning algorithms for inductive conformal predictors. By performing an empirical study on various datasets and network architectures, we have shown that effective pruning is achievable without hurting the predictive efficiency of the model. Indeed, for some tasks we observe an improvement in predictive efficiency at a particular pruning ratio. We also introduced a new pruning method based on conformal prediction efficiency which is competitive among existing pruning methods in maintaining predictive efficiency, and may be the pruning method of choice if we consider reliable machine learning through conformal prediction as our learning goal.

This study is relevant for developers looking to select the correct size of neural networks, or compress neural networks for computational efficiency, in the context of conformal prediction. It is also an important stage in applying continual learning within a conformal prediction framework.

Our proposed pruning method is currently not sufficiently computationally efficient to be applied deep neural network, so our future work will focus on improving computational efficiency by exploring a reduced pruning search space, and extending our method to deep convolutional neural networks and also exploring in the context of continual learning.

5. Acknowledgements

This research was supported by Faculty of Science and Engineering, University of Nottingham Ningbo China: Doctoral Scholarship with reference number 19073FOSE.

References

Jock A Blackard and Denis J Dean. Comparative accuracies of artificial neural networks and discriminant analysis in predicting forest cover types from cartographic variables. *Computers and electronics in agriculture*, 24(3):131–151, 1999.

Davis Blalock, Jose Javier Gonzalez Ortiz, Jonathan Frankle, and John Guttag. What is the state of neural network pruning? *Proceedings of machine learning and systems*, 2: 129–146, 2020.

Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural network. In *International conference on machine learning*, pages 1613–1622. PMLR, 2015.

Xiangyu Chang, Yingcong Li, Samet Oymak, and Christos Thrampoulidis. Provable benefits of overparameterization in model compression: From double descent to pruning neural networks. *arXiv preprint arXiv:2012.08749*, 2020.

Florin C Ghesu, Bogdan Georgescu, Awais Mansoor, Youngjin Yoo, Eli Gibson, RS Vishwanath, Abishek Balachandran, James M Balter, Yue Cao, Ramandeep Singh, et al. Quantifying and leveraging predictive uncertainty for medical image assessment. *Medical Image Analysis*, 68:101855, 2021.

Alex Graves. Practical variational inference for neural networks. *Advances in neural information processing systems*, 24, 2011.

Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. *Advances in neural information processing systems*, 28, 2015.

Babak Hassibi, David G. Stork, and Gregory J. Wolff. Optimal brain surgeon and general network pruning. In *IEEE international conference on neural networks*, pages 293–299. 1993.

Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks. In *Proceedings of the IEEE international conference on computer vision*, pages 1389–1397, 2017.

Johannes Hofmanninger, Matthias Perkonigg, James A Brink, Oleg Pianykh, Christian Herold, and Georg Langs. Dynamic memory to alleviate catastrophic forgetting in continuous learning settings. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 359–368. Springer, 2020.

Sangwon Jung, Hongjoon Ahn, Sungmin Cha, and Taesup Moon. Continual learning with node-importance based adaptive group sparse regularization. *Advances in Neural Information Processing Systems*, 33:3647–3658, 2020.

Alex Kendall and Yarin Gal. What uncertainties do we need in Bayesian deep learning for computer vision? *Advances in neural information processing systems*, 30, 2017.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Diederik P Kingma and Max Welling. Auto-encoding variational Bayes. *arXiv preprint arXiv:1312.6114*, 2013.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

Steve Lawrence and C Lee Giles. Overfitting and neural networks: conjugate gradient and backpropagation. In *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium*, volume 1, pages 114–119. IEEE, 2000.

Vadim Lebedev and Victor Lempitsky. Fast convnets using group-wise brain damage. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2554–2564, 2016.

Yann LeCun, John Denker, and Sara Solla. Optimal brain damage. *Advances in neural information processing systems*, 2, 1989.

Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. *arXiv preprint arXiv:1608.08710*, 2016.

H Linusson. Nonconformist: Python implementation of the conformal prediction framework. *URL https://github. com/donlnz/nonconformist*, 2015.

Christos Louizos, Karen Ullrich, and Max Welling. Bayesian compression for deep learning. *Advances in neural information processing systems*, 30, 2017.

Arun Mallya, Dillon Davis, and Svetlana Lazebnik. Piggyback: Adapting a single network to multiple tasks by learning to mask weights. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 67–82, 2018.

Dmitry Molchanov, Arsenii Ashukha, and Dmitry Vetrov. Variational dropout sparsifies deep neural networks. In *International Conference on Machine Learning*, pages 2498–2507. PMLR, 2017.

Cuong V Nguyen, Yingzhen Li, Thang D Bui, and Richard E Turner. Variational continual learning. *arXiv preprint arXiv:1710.10628*, 2017.

Harris Papadopoulos, Kostas Proedrou, Volodya Vovk, and Alex Gammerman. Inductive Confidence Machines for Regression. In T. Elomaa, H. Mannila, and H. Toivonen, editors, *Machine Learning: ECML 2002. ECML 2002. Lecture Notes in Computer Science*, volume 2430. Springer, Berlin, Heidelberg, 2002.

Harris Papadopoulos, Volodya Vovk, and Alex Gammerman. Conformal prediction with neural networks. In *19th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2007)*, volume 2, pages 388–395. IEEE, 2007.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

Siddhaling Urolagin, Prema KV, and NV Reddy. Generalization capability of artificial neural network incorporated with pruning method. In *International Conference on Advanced Computing, Networking and Security*, pages 171–178. Springer, 2011.

V Vovk, V Fedorova, I Nouretdinov, and A Gammerman. Criteria of efficiency for conformal prediction. In *COPA 2016: Proceedings of the 5th International Symposium on Conformal and Probabilistic Prediction with Applications*, volume 9653, pages 23–29, April 2016.

Vladimir Vovk, Alexander Gammerman, and Glenn Shafer. *Algorithmic learning in a random world*. Springer US, 2005. ISBN 0387001522. doi: 10.1007/b106715.

Chaoqi Wang, Roger Grosse, Sanja Fidler, and Guodong Zhang. Eigendamage: Structured pruning in the Kronecker-factored eigenbasis. In *International Conference on Machine Learning*, pages 6566–6575. 2019.

Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Learning structured sparsity in deep neural networks. *Advances in neural information processing systems*, 29, 2016.

Peter M Williams. Bayesian regularization and pruning using a Laplace prior. *Neural computation*, 7(1):117–143, 1995.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.