Logically Constrained Pruning Towards Robust Neural Networks

Kirsty Duncan, Robert Stewart, and Ekaterina Komendantskaya

Herot-Watt University, Edinburgh, UK {krd1,R.Stewart,ek19}@hw.ac.uk

Abstract. Pruning algorithms aim to remove the redundancy in neural networks and can result in networks which require significantly less memory and computation than full-precision networks, with little or no difference in accuracy. Unfortunately, pruned networks are more sensitive to adversarial attacks. To address this, we propose Logically Constrained Pruning (LCP), a pruning algorithm that employs a semantic loss function containing information on both a network's accuracy and adherence to some robustness property. Our new pruning algorithm produces models that are no less accurate than a state-of-the-art pruning algorithm, but has the advantage of producing pruned models that are more robust to adversarial attack. We demonstrate that, using LCP, pruning 75% of parameters from a ResNet50 model on the CIFAR-100 dataset incurs no loss in accuracy, and is 29.6% more robust to FGSM adversarial attack.

Keywords: Neural network pruning · Adversarial robustness

1 Introduction

Neural network models are typically over-parameterized and contain a great deal of redundancy in the number of connections and filters [5]. Pruning methods aim to remove this redundancy by removing parameters from some or all layers of a network. Pruned networks require significantly less memory and computation than full-precision networks and have comparable accuracy [18]. The sheer size of modern deep learning models is a major obstacle in the feasibility of formal verification of such models. Pruning methods which preserve the properties of the original model have been proposed to generate equivalent networks which are more amenable to formal analysis [9, 12].

With the substantial threat of adversarial attacks [22] comes the question of how pruning a neural network impacts its robustness against such attacks. Networks compressed using quantization are more robust than full-precision models [2, 6, 8]. In contrast, there is limited research on the robustness of pruned networks, summarised as follows. Pruning of all layers of a convolutional neural network (CNN), or the convolutional filters only, substantially decreases a network's robustness to attack [13]. Sparsifying the input [11] or the fully-connected layers only [23] of a CNN both result in a small increase in robustness. Modern CNNs are made up mostly of convolutional layers making robust pruning of the full network crucial.

A recent trend in the literature on adversarial defense is robust training methods [1, 7, 24]. They train models using a semantic loss function resulting in a slight loss in prediction accuracy and significant increase in some other network property, such as robustness, when compared to using the standard loss.

This paper presents LCP, a novel pruning algorithm which aims to produce compressed and robust neural network implementations. LCP is similar to [18] that, like LCP, uses a loss-based pruning criterion. The key difference is that LCP factors in a semantic *logical constraint* into that loss function. LCP uses FGSM sampling as introduced in [3], when testing the logical constraint.

The contributions of this paper are:

– LCP, a novel pruning algorithm that produces neural networks robust against adversarial attack. LCP employs a semantic loss for both the selection of parameters to prune and retraining steps. We are able to preserve the robustness of pruned models without sacrificing accuracy. In some cases, the pruned model is more robust than the original (Section 3).
– An investigation into the deterioration of networks as they are over-pruned. Over-pruning results in models being unable to reach one or more output classes for a test dataset (Section 4).
– An evaluation of our LCP algorithm. It produces networks that are more robust than networks pruned with a state-of-the-art algorithm [18]. After pruning 75% of parameters from ResNet-50 [14] trained on the CIFAR-100 dataset, with our method the model is 29.6% more robust with 0% loss in accuracy. After pruning between between 9% and 75% of parameters of VGG [21], the models pruned with LCP at weight 0.1 have higher robustness than the original unpruned model. (Section 5).

2 Background

Neural networks are machine learning systems capable of learning to classify previously unseen inputs, typically after training on a set of human-labeled inputs.

Definition 1 (Neural Network). *Let $X \subseteq \mathbb{R}^{n_0}$ for some $n_0 \in \mathbb{N}$ and $Y \subseteq \mathbb{R}^{n_k}$ for some $n_k \in \mathbb{N}$. A* neural network *N is a function $N : X \to Y$ which transforms an* input *$x \in X$ to an* output *$y_x \in Y$. The* parameters *of a neural network are the set $\mathbf{W} = \{w_0, \ldots, w_m\}$ where $m \in \mathbb{N}$ is the total number of parameters.*

The output y_x of a **classifier network** represents a probability distribution over the possible classes $c \in C$. The highest probability in this distribution is used to determine the classification $c_N(x) \in C$ of x by N. Networks can be trained by finding values for parameters $\mathbf{W}$, which may represent connections between layers or elements of complex structures such as convolutional filters. Training involves presenting networks with input-output pairs from a dataset.

Definition 2 (Dataset). *Given a network N with an input space X,* dataset *$D \subseteq X$ is made up of inputs $d \in D$ each with a corresponding ground-truth label, $l_d \in L_D$ where $L_D = \{1, ..., n\}$ for some $n \subset \mathbb{N}$. When N is* trained on D*, it has output space $Y = [0, 1]^{|L_D|}$.*

The process of **training** a neural network involves finding values for **W** which minimize the network's error or **loss** $\mathcal{L}$ on some dataset D.

$$\min_{W} \mathcal{L}(D, \mathbf{W}) \tag{1}$$

Training is usually performed via gradient descent. Parameters are iteratively modified to lower the loss.This is repeated until the user is satisfied with the network's performance. The principal measure of a network's performance is *prediction accuracy*; the percentage of correctly classified inputs in the dataset.

Definition 3 (Prediction Accuracy). *Given a network N and dataset D, the prediction accuracy P of N on D is the percentage of points in D for which N correctly classifies.*

$$P = \sum_{\forall d \in D} \frac{pred(N,d)}{|D|} \quad where \quad pred(N,d) = \begin{cases} 1 & if\ c_N(d) = l_d \\ 0 & otherwise \end{cases} \tag{2}$$

The process of **pruning** a network N aims to minimize the number of parameters in the model while preserving accuracy. This is achieved by finding values for **W** which minimize a loss function with a sparsification term included:

$$\min_{\mathbf{W}} \mathcal{L}(D, \mathbf{W}) + \lambda ||\mathbf{W}||_0 \tag{3}$$

λ is a scaling coefficient and $||\mathbf{W}||_0$ represents the l_0 norm, the number of non-zero elements in **W**, i.e. the number of non-zero parameters in N.

In practice, pruning is achieved by incrementally removing parameters $w \in \mathbf{W}$ from a network based on each parameter's *saliency*; a measure of the contribution of each parameter to the network's overall performance. Starting with a trained network N, the following steps are performed iteratively:

1. Parameters with low saliency are removed from the network.
2. The remaining parameters are updated through several iterations of training.

Definition 4 (Saliency). *Given a network $N : X \to Y$ with parameters* **W***, the* saliency *of each parameter $w_i \in \mathbf{W}$ is the change in loss $\mathcal{L}$ caused by setting that parameter to zero:*

$$I_{w_i} = \Big(\mathcal{L}(D, \mathbf{W}) - \mathcal{L}(D, \mathbf{W}|w_i = 0)\Big)^2 \tag{4}$$

Taking a first order Taylor expansion gives:

$$I_{w_i}(\mathbf{W}) = \Big(\frac{\partial \mathcal{L}}{\partial w_i} w_i\Big)^2 \tag{5}$$

Inputs from a dataset modified using **adversarial perturbations** are statistically likely to cause misclassifications across different networks trained with different architectures and even trained on different data [22].

Definition 5 (Adversarial Perturbation). *Let N be a neural network trained on $D \subset X$, $d \in D$ be an input with output classification $c_N(d)$. An* adversarial perturbation *is a vector $\epsilon \in X$ which generates a* perturbed input $x' = d + \epsilon$.

If $c_N(x') \neq c_N(d)$, the adversarial perturbation is effective.

If $c_N(x') = c_N(d)$, the adversarial perturbation is ineffective.

One attempt to train neural networks to resist adversarial attack involves encoding a constraint into the loss such that during training, a network's goal will be to maximise both prediction accuracy and adherence to this constraint.

Logical constraints consist of of boolean combinations of comparisons between terms. A term t is defined over inputs $x \in X$ and parameters $\mathbf{W}$. A term can be any real-valued function that might appear as a subexpression of a loss function, such as a constant or an output y_x. Terms must be differentiable almost everywhere, in both x and $\mathbf{W}$. Two terms t and t' can be combined to form comparison constraints $t = t'$, $t \leq t'$, $t < t'$ $t \neq t'$. A constraint ϕ is either a comparison constraint; a conjunction $\phi' \wedge \phi''$, disjunction $\phi' \vee \phi''$ or implication $\phi' \implies \phi''$ of constraints ϕ' and ϕ''; or a negation $\neg\phi$ of constraint ϕ.

Definition 6 (Logically Constrained Training). *Given a neural network N with parameters $\mathbf{W}$, dataset D and constraint ϕ, the process of* logically constrained training *involves finding values for $\mathbf{W}$ which minimise the network's* constraint-aware loss*:*

$$\min_{W} \mathcal{L}_C(D, \mathbf{W}, \phi) \tag{6}$$

The constraint-aware loss $\mathcal{L}_C$ *is defined as:*

$$\mathcal{L}_C(D, \mathbf{W}, \phi) = \mathcal{L}(D, \mathbf{W}) + w_C \cdot \mathcal{L}_\phi(D, \mathbf{W}) \tag{7}$$

where w_C is the weight of the constrained learning, $\mathcal{L}(D, \mathbf{W})$ is the standard loss and $\mathcal{L}_\phi(D, \mathbf{W})$ is the constraint loss term $\mathcal{L}_\phi$, given by:

$$\mathcal{L}_\phi(D, \mathbf{W}) = \underset{d \in D}{avg} \mathcal{L}_\phi(d, \mathbf{W}) \quad where \quad \mathcal{L}_\phi = \begin{cases} 0 & if \ \ \phi \ is \ satisfied \\ 1 & otherwise. \end{cases} \tag{8}$$

3 Logically Constrained Pruning Algorithm (LCP)

We propose a logically constrained pruning algorithm which applies techniques from the logically constrained training approach to produce robust pruned models. The saliency of parameters is measured by taking gradients of the constraint-aware loss $\mathcal{L}_C$ introduced in Definition 6. Parameters with low saliency are iteratively set to zero, followed by retraining using the same semantic loss. All accompanying code for this paper is available online[1]

[1] https://github.com/KirstyRD/Logically_Constrained_Pruning

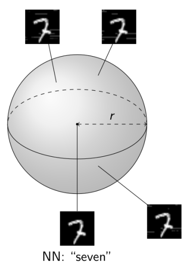


Fig. 1: Sampling for Neighbourhood Robustness

In this paper we explore the encoding of one constraint into the loss function, the **neighbourhood robustness constraint** [7]. This constraint evaluates the behaviour of a network's output y_d while sampling perturbed inputs within the neighbourhood of an input d from the dataset. Figure 1 depicts the sampling of perturbed inputs within a radius r around an input with label 7 from the MNIST dataset. The constraint states that inputs $x' \in X$ generated in the neighbourhood around an input $d \in D$ from the dataset $D \in X$ should have a high probability of being classified the same as d, $cN(d) = c_N(x')$.

Definition 7 (Neighbourhood Robustness Constraint). *Given a network N, input space X and dataset $D \subset X$, for any input $d \in D$ with classification $c_N(d)$, inputs $x' \in X$ in d's ϵ neighborhood which are valid images (pixels are between 0 and 1), have a high probability for $c_N(x') = c_N(d)$. i.e. The log-probability is larger than a given threshold δ:*

$$\forall x' \in B_\epsilon(d) \cap [0,1]^n.logp\big(x'\big)_{c_N(d)}\big) > \delta \tag{9}$$

where the epsilon ball $B_\epsilon(d)$ around d is an L_∞-ball around d with radius ϵ.

In practice, a finite number of points $x' \in B_\epsilon(d)$ are sampled during calculation of the loss. In [3] models were trained using this constraint with points sampled using FGSM [10] resulting in robust models. We use this method in the results presented in this paper. For some network N trained on $D \in X$ we generate a perturbed $x' = d + \epsilon_{FGSM}$ for each $d \in D$ using FGSM and compare the output classifictions $c_N(d)$ and $c_N(x')$.

3.1 The Logically Constrained Pruning Algorithm

The LCP algorithm takes as input a trained network N and the dataset D that it was trained on. Parameters with low saliency are iteratively removed from the network followed by retraining of the remaining parameters through gradient descent. Both the saliency calculations and the retraining steps are evaluated on a constraint-aware loss $\mathcal{L}_C$. The impotant parameters are as follows:

1. N_{train} Number of training iterations between pruning steps.
2. N_{prune} Percentage of parameters pruned per pruning step.
3. ϕ Constraint used in constraint-aware loss function (Eq 7) for pruning and retraining steps.
4. w_C Weight of the constraint term in the loss function (Eq 7.)

Pruning is performed iteratively starting from a trained model and producing n robust pruned models at different levels of pruning. As outlined in Algorithm 1, the following steps are repeated:

Algorithm 1 Logically Constrained Pruning (LCP) Algorithm

```
1: Input $N$: Pre-trained model
2: for $i = 1, ..., n$ do                                          ▷ Repeat for n pruning steps
3:     for $j = 1, ..., N_{train}$ do                               ▷ Repeat for $N_{train}$ retraining iterations
4:         $\mathbf{W} = \text{Train}(\mathbf{W}, \mathcal{L}_C)$              ▷ Logically constrained training step (Eq 7)
5:         for $k = 0, ..., m$ do                                   ▷ For each parameter $w_k \in \mathbf{W}$
6:             $I^j_{w_k}(\mathbf{W}) = \left(\frac{\partial \mathcal{L}}{\partial w_k} w_k\right)^2$          ▷ Calculate saliency for $j^{th}$ iteration (Eq 5)
7:         end for
8:         $I(\mathbf{W})^j = \{I^j_{w_0}(\mathbf{W}), ..., I^j_{w_m}(\mathbf{W})\}$          ▷ Record $j^{th}$saliency vector
9:     end for
10:    $I(\mathbf{W}) = \sum_{j=1,...,N_{train}} \frac{I(\mathbf{W})^j}{N_{train}}$          ▷ Calculate average saliency vector for all $j$
11:    $\mathbf{W} = \text{Prune}(\mathbf{W}, I(\mathbf{W}), N_{prune})$   ▷ Prune the $N_{prune}$ parameters w with lowest
       value saliency $I_w(\mathbf{W})$
12: end for
```

1. Gradients of the constraint-aware loss $\mathcal{L}_C(D, \mathbf{W}, \phi)$ (Eq 7) are calculated for each parameter $w \in \mathbf{W}$ (weights and filters). The saliency I_w (Eq 4) of each of these parameters is recorded. The parameter values are updated via gradient descent on the constraint-aware loss $\mathcal{L}_C$.
2. After a predefined number of iterations N_{train}, the average of all saliencies in previous retraining steps is recorded for each parameter. The N_{step} parameters with lowest saliency are removed.
3. After each pruning step, the pruned model is saved and prediction accuracy (Eq 2), deterioration (Eq 12) and robustness (Eq 11) are recorded.

3.2 Efficacy of Robust Pruning

We use three metrics to measure the efficacy of the LCP algorithm: (1) the network's percentage accuracy predicting the correct label of inputs from the dataset D (Definition 3), (2) the percentage of the dataset for which the constraint ϕ holds and (3) the network's deterioration (Section 4).

Definition 8 (Constraint Accuracy). *Given a network N, dataset D and constraint ϕ, the constraint accuracy A_c of N on D is the percentage of points in D for which ϕ is satisfied.*

$$A_C = \frac{\sum_{\forall d \in D} sat(\phi(N, d))}{|D|} \quad where \quad sat(\phi) = \begin{cases} 1 & if\ \phi is\ satisfied \\ 0 & otherwise \end{cases} \tag{10}$$

For the neighbourhood robustness constraint explored in this paper this measure is equivalent to a network's **robustness**.

Definition 9 (Robustness). *Given a network N, dataset D and adversarial attack algorithm F, the robustness - i.e. the accuracy against adversarial attack*

- R_F of N on D against F is the percentage of points in D around which the adversarial attack algorithm failed to generate a effective adversarial perturbation.

$$R_F = \sum_{\forall d \in D} \frac{r_F(N,d)}{|D|} \qquad \textit{where } r_F(N,d) = \begin{cases} 0 & \textit{if } F \textit{ fools } N \textit{ around } d \\ 1 & \textit{otherwise} \end{cases} \tag{11}$$

4 Deterioration of Pruned Neural Networks

5882	1	26	10	10	17	44	15	21	19
1	6639	27	1	8	3	7	14	24	6
2	26	5681	12	4	2	1	31	6	0
5	27	86	6024	1	78	0	75	60	52
1	7	12	0	5592	2	16	14	6	20
3	0	2	19	0	5237	31	2	17	18
8	4	5	1	14	15	5799	0	13	1
2	9	25	12	6	0	0	5950	2	18
10	20	83	25	11	35	20	21	5632	17
9	9	11	27	196	32	0	143	70	5798

(a) unpruned network

5817	0	21	7	7	51	49	24	32	42
5	6615	23	6	40	6	15	27	35	19
11	66	5699	91	13	6	3	57	34	2
3	2	27	5826	0	64	0	26	28	59
3	3	30	0	5621	6	46	23	9	44
13	1	2	67	0	5126	22	1	35	43
21	4	15	1	29	29	5758	0	28	2
1	9	45	26	5	1	0	6006	5	37
47	39	91	85	17	73	25	29	5602	42
2	3	5	22	110	59	0	72	43	5659

(b) 75% of parameters pruned

342	2	5	0	0	1	0	0	3	1
2782	6225	373	3	1738	66	5449	38	679	20
9	0	14	2	0	5	1	6	2	8
136	16	977	5486	746	4572	13	579	738	4238
0	0	0	0	0	0	0	0	0	0
5	0	34	10	230	126	17	6	39	69
0	0	0	0	0	0	0	0	0	0
2441	36	3605	484	8	87	0	5415	517	283
208	463	950	146	3120	564	438	221	3873	1330
0	0	0	0	0	0	0	0	0	0

(c) 95% of parameters pruned

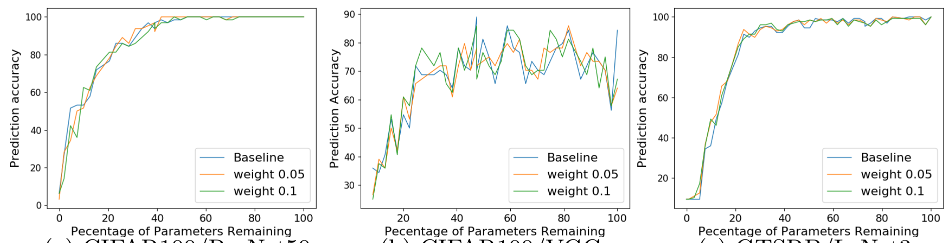
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
5923	6742	5958	6131	5842	5421	5918	6265	5851	5949
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

(d) 99% of parameters pruned

Fig. 2: Confusion matrices for Lenet3 on MNIST at various levels of pruning.

Pruning a high proportion of parameters from a neural network not only impacts prediction accuracy (Section 5.1). It may also result in a collapse of the network's range of possible outputs, resulting in some classes being unreachable for a given dataset comprising inputs for all classes. The behaviour of neural networks at different levels of pruning can be understood from confusion matrices of a network at various levels of pruning. Confusion matrices explain the misclassifications of a network by depicting confusion between classes. A network with high accuracy's martrix has high values along the diagonal. High values outside of the diagonal show that many inputs from the class associated with that column are classified as the class associated with that row.

Tables 2a and 2b depict confusion matrices for an unpruned LeNet [17] neural network, trained on MNIST[4] and a pruned network with 25% of original parameters remaining. Accuracies are 98.43% and 93.75% respectively. In both matrices, few values lie outside the diagonal, which implies that the vast majority of network outputs on the dataset are correctly classified. At 25% size, the network has mostly preserved prediction accuracy. Tables 2c and 2d represent pruned networks with 5% and 1% of parameters remaining respectively. These matrices show how certain classes become unreachable from the model when



(a) CIFAR100/ResNet50 (b) CIFAR100/VGG (c) GTSRB/LeNet3

Fig. 4: Prediction Accuracy

they are highly pruned. At 5% of parameters and 26.5% accuracy, classes 4,6 and 9 are unreachable by the network for the full test dataset. At 1% of parameters and 6.25% accuracy, all inputs are classed as character 5, i.e. 90% of classes are unreachable after pruning 99% of the parameters.

We introduce the notion of **network deterioration** to measure the percentage of unreachable classes at each pruning step.

Definition 10 (Network Deterioration). *Given a network N trained on dataset D and set of possible labels L_D, N has a* deterioration κ *for dataset D if there exist κ labels $l_1, ..., l_\kappa \in L_D$ such that no input $d \in D$ exists which is classified by N as any of $l_1, ..., l_\kappa \in L_D$.*

$$\exists l_1, ..., l_\kappa \in L_D | \forall l_i \in \{l_1, ..., l_\kappa\} \nexists d \in D | c_N(d) = l_i \tag{12}$$

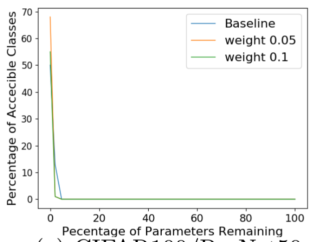
In Section 5.2, deterioration is presented as the percentage of labels that are unreachable $\frac{\kappa}{|L_D|}$.

5 Evaluation

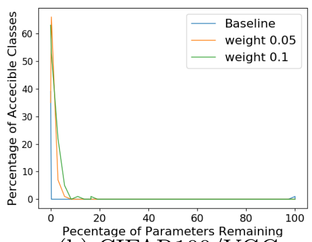
We present prediction accuracy and constraint accuracy/robustness for networks pruned from three full-precision networks ResNet50, VGG and LeNet3, on two data CIFAR-100 [16] and GTSRB [15] using our LCP algorithm and a state-of-the-art camparison pruning algorithm [18].

5.1 Prediction Accuracy of Pruned Networks

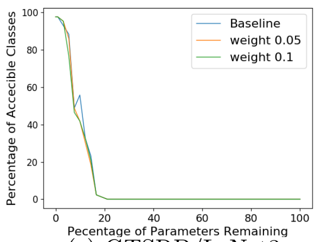
Figure 4 presents the prediction accuracy of networks at various levels of pruning. The baseline model is pruned using the methodology from [19]. Models are pruned with LCP using different values of the weight w_C for the constraint term $\mathcal{L}_\phi$ in the loss $\mathcal{L}_C$, as in Eq 7. In all cases, the greater the weight the greater the drop in prediction accuracy compared to the baseline. This difference in accuracy is very small and in most cases, negligible. It can be seen from these results that for each model, a level of pruning is reached past which the model's accuracy



(a) CIFAR100/ResNet50

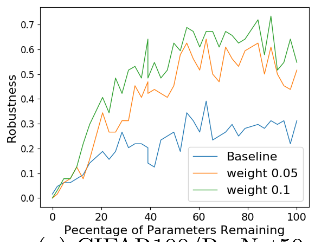


(b) CIFAR100/VGG

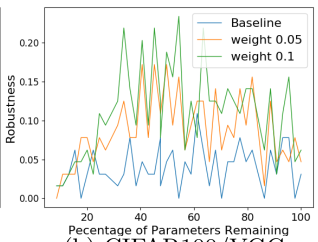


(c) GTSRB/LeNet3

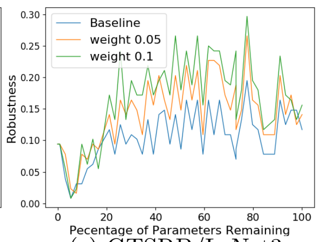
Fig. 5: Deterioration



(a) CIFAR100/ResNet50



(b) CIFAR100/VGG



(c) GTSRB/LeNet3

Fig. 6: Robustness Accuracy

can be seen to rapidly decrease, possibly due to deterioration (Section 5.2). This happens clearly in Figure 4a at around 20%. Models pruned with LCP are not significantly more or less accurate than the baseline method.

5.2 Deterioration of Pruned Networks

Figure 5 presents the deterioration of networks at various levels of pruning. For ResNet (Figures 5a) and VGG (Figure 5b), deterioration to unreachable output classes starts at around 95% of parameters pruned. For LeNet3 it begins at 80% parameters pruned. Deterioration occurs rapidly after these thresholds. Deterioration for LeNet3 (Figure 5c) occurs at the same rate as the fall in prediction accuracy (Figure 4c). Models pruned with LCP do not deteriorate more rapidly than the baseline method.

5.3 Robustness of Pruned Networks

Figure 6 presents the robustness of networks at various levels of pruning. FGSM epsilon is 0.3 for ResNet50 on CIFAR100 and LeNet3 on GTSRB and, 0.1 for VGG on CIFAR100.

Models pruned with our pruning method are more robust than models pruned to the same extent with the baseline method. Figure 6a shows that after pruning 75% of parameters from ResNet-50 on the CIFAR-100 dataset, the network pruned using the LCP algorithm at weight 0.1 is 29.6% more robust than the benchmark pruning algorithm with 0% loss in accuracy. Figure 6b shows that after pruning 59.4% of parameters the model pruned with LCP at weight 0.1 has the same accuracy but 15.6% higher robustness. Figure 5c shows that for 75% of parameters pruned, the model pruned using LCP with weight 0.1 has 11.7% higher robustness compared with the baseline and 1.4% higher accuracy.

There is an initial increase in robustness for lightly pruned networks when compared to the original models which were trained without any adversarial defense mechanisms and as such, the initial retraining using semantic loss improves on the robustness of the original model. This can be seen in Figure 6a where for all but one model between pruning levels of 11% and 48%, the models pruned with LCP at a weight of 0.05 have higher robustness than the original unpruned model. In Figure 6b, between 9% and 75% of parameters pruned, the models pruned with LCP at weight 0.1 have higher robustness than the original unpruned model. In Figure 5c, between 27% and 75% of parameters pruned, the models pruned with LCP have higher robustness than the original model.

For three models (Figure 6) our LCP algorithm outperforms the robustness of a baseline pruning algorithm [18], with little to no difference in prediction accuracy (Section 5.1) and deterioration occurs at the same rate (Section 5.2).

6 Conclusion

Pruning neural networks can produce faster and more resource efficient implementations for embedded hardware devices [20]. Unfortunately however, pruned networks are more sensitive to adversarial attacks [13]. This paper presents a pruning algorithm, LCP, to address this problem. LCP incorporates a logical constraint into the loss function to identify the least salient parameters for pruning. We evaluate LCP with a neighbourhood robustness constraint with FGSM sampling, although any logical constraint can be used with LCP.

LCP produces models that are, in some cases, more robust against adversarial attacks and with very little or no loss in accuracy. For example, pruning 75% of parameters from a ResNet50 model on the CIFAR-100 dataset with LCP incurs no loss in accuracy, and is 29.6% more robust to FGSM adversarial attack.

Adversarial training is a widely-used and successful defense. As future work, using an adversarial dataset during pruning and retraining steps could result in robust pruned models. An exploration of this and comparison to the robust pruning algorithm would be an interesting extension to this work. The network deterioration property could be used as a constraint and fed into the semantic loss. This may result in being able to prune models even more aggressively without deteriorating to non-reachable output classes.

Bibliography

[1] Ayers, E.W., Eiras, F., Hawasly, M., Whiteside, I.: Parot: A practical framework for robust deep neural network training. In: Lee, R., Jha, S., Mavridou, A. (eds.) NASA Formal Methods - 12th International Symposium, NFM 2020, Moffett Field, CA, USA, May 11-15, 2020, Proceedings. Lecture Notes in Computer Science, vol. 12229, pp. 63–84. Springer (2020). https://doi.org/10.1007/978-3-030-55754-6_4, https://doi.org/10.1007/978-3-030-55754-6_4

[2] Bernhard, R., Moëllic, P., Dutertre, J.: Impact of low-bitwidth quantization on the adversarial robustness for embedded neural networks. In: Int. Conf. on Cyberworlds. pp. 308–315. IEEE (2019)

[3] Casadio, M., Daggitt, M., Komendantskaya, E., Kokke, W., Kienitz, D., Stewart, R.: Property-driven training: All you (n)ever wanted to know about. CoRR **abs/2104.01396** (2021), https://arxiv.org/abs/2104.01396

[4] Deng, L.: The mnist database of handwritten digit images for machine learning research [best of the web]. IEEE Signal Processing Magazine **29**, 141–142 (2012)

[5] Denil, M., Shakibi, B., Dinh, L., De Freitas, N., et al.: Predicting parameters in deep learning. In: Advances in neural information processing systems. pp. 2148–2156 (2013)

[6] Duncan, K., Komendantskaya, E., Stewart, R., Lones, M.: Relative robustness of quantized neural networks against adversarial attacks. In: 2020 International Joint Conference on Neural Networks (IJCNN). pp. 1–8. IEEE (2020)

[7] Fischer, M., Balunovic, M., Drachsler-Cohen, D., Gehr, T., Zhang, C., Vechev, M.T.: DL2: training and querying neural networks with logic. In: Chaudhuri, K., Salakhutdinov, R. (eds.) Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA. Proceedings of Machine Learning Research, vol. 97, pp. 1931–1941. PMLR (2019), http://proceedings.mlr.press/v97/fischer19a.html

[8] Galloway, A., Taylor, G.W., Moussa, M.: Attacking binarized neural networks. In: 6th Int. Conf. on Learning Representations, Conf. Track Proc. OpenReview.net (2018)

[9] Gokulanathan, S., Feldsher, A., Malca, A., Barrett, C.W., Katz, G.: Simplifying neural networks using formal verification. In: Lee, R., Jha, S., Mavridou, A. (eds.) NASA Formal Methods - 12th International Symposium, NFM 2020, Moffett Field, CA, USA, May 11-15, 2020, Proceedings. Lecture Notes in Computer Science, vol. 12229, pp. 85–93. Springer (2020). https://doi.org/10.1007/978-3-030-55754-6_5, https://doi.org/10.1007/978-3-030-55754-6_5

[10] Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. In: Bengio, Y., LeCun, Y. (eds.) 3rd International Confer-

ence on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings (2015), http://arxiv.org/abs/1412.6572

[11] Gopalakrishnan, S., Marzi, Z., Madhow, U., Pedarsani, R.: Combating adversarial attacks using sparse representations. In: 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Workshop Track Proceedings. OpenReview.net (2018), https://openreview.net/forum?id=S10qYwywf

[12] Guidotti, D., Leofante, F., Pulina, L., Tacchella, A.: Verification of neural networks: Enhancing scalability through pruning. In: Giacomo, G.D., Catalá, A., Dilkina, B., Milano, M., Barro, S., Bugarín, A., Lang, J. (eds.) ECAI 2020 - 24th European Conference on Artificial Intelligence, 29 August-8 September 2020, Santiago de Compostela, Spain, August 29 - September 8, 2020 - Including 10th Conference on Prestigious Applications of Artificial Intelligence (PAIS 2020). Frontiers in Artificial Intelligence and Applications, vol. 325, pp. 2505–2512. IOS Press (2020). https://doi.org/10.3233/FAIA200384, https://doi.org/10.3233/FAIA200384

[13] Guo, Y., Zhang, C., Zhang, C., Chen, Y.: Sparse dnns with improved adversarial robustness. In: Bengio, S., Wallach, H.M., Larochelle, H., Grauman, K., Cesa-Bianchi, N., Garnett, R. (eds.) Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada. pp. 240–249 (2018), https://proceedings.neurips.cc/paper/2018/hash/4c5bde74a8f110656874902f07378009-Abstract.html

[14] He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016. pp. 770–778. IEEE Computer Society (2016). https://doi.org/10.1109/CVPR.2016.90, https://doi.org/10.1109/CVPR.2016.90

[15] Houben, S., Stallkamp, J., Salmen, J., Schlipsing, M., Igel, C.: Detection of traffic signs in real-world images: The German Traffic Sign Detection Benchmark. In: International Joint Conference on Neural Networks (2013)

[16] Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images. https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf (2009)

[17] LeCun, Y., Boser, B.E., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W.E., Jackel, L.D.: Handwritten digit recognition with a back-propagation network. In: Advances in neural information processing systems. pp. 396–404 (1990)

[18] Molchanov, P., Mallya, A., Tyree, S., Frosio, I., Kautz, J.: Importance estimation for neural network pruning. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019. pp. 11264–11272. Computer Vision Foundation / IEEE (2019). https://doi.org/10.1109/CVPR.2019.01152, http://openaccess.thecvf.com/

content_CVPR_2019/html/Molchanov_Importance_Estimation_for_Neural_Network_Pruning_CVPR_2019_paper.html

[19] Molchanov, P., Tyree, S., Karras, T., Aila, T., Kautz, J.: Pruning convolutional neural networks for resource efficient inference. arXiv preprint arXiv:1611.06440 (2016)

[20] Radu, V., Kaszyk, K., Wen, Y., Turner, J., Cano, J., Crowley, E.J., Franke, B., Storkey, A., O'Boyle, M.: Performance Aware Convolutional Neural Network Channel Pruning for Embedded GPUs. In: IISWC 2019. IEEE (October 2019)

[21] Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: Bengio, Y., LeCun, Y. (eds.) 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings (2015), http://arxiv.org/abs/1409.1556

[22] Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I.J., Fergus, R.: Intriguing properties of neural networks. In: 2nd Int. Conf. on Learning Representations, Conf. Track Proc. (2014)

[23] Wang, L., Ding, G.W., Huang, R., Cao, Y., Lui, Y.C.: Adversarial robustness of pruned neural networks (2018)

[24] Wong, E., Kolter, J.Z.: Provable defenses against adversarial examples via the convex outer adversarial polytope. In: Dy, J.G., Krause, A. (eds.) Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018. Proceedings of Machine Learning Research, vol. 80, pp. 5283–5292. PMLR (2018), http://proceedings.mlr.press/v80/wong18a.html