Advanced Topics in Machine Learning

Yuri Paglierani

Challenge 2

University of Trieste

1 Introduction

This report presents the implementation and analysis of a Dynamic Pruning Convolutional Neural Network (CNN) for image classification on the CIFAR-10 dataset. The primary goal of this project is to develop a CNN architecture that can dynamically adjust its structure during training by pruning less important layers, potentially improving efficiency without significant loss in performance.

2 Source Code

The complete source code, including all implementations and experiments discussed in this report, is available in the following GitHub repository:

`https://github.com/YuriPaglierani/adv-ml-units`

This repository contains the code for the Advanced Topics in Machine Learning practica at the University of Trieste, Spring 2023. Readers are encouraged to explore the repository for more detailed implementations and additional resources related to this project.

3 Methodology

3.1 Network Architecture

The core of our implementation is the DynamicPruningCNN class, which consists of a variable number of convolutional layers. Each layer is followed by batch normalization and a custom non-linearity function. The custom non-linearity is defined as:

$$f(x) = x + \alpha \cdot ReLU(-x) \tag{1}$$

where α is a learned parameter computed using sigmoid functions:

$$\alpha = \sigma((\sigma(\gamma) - 0.5) \cdot \nu_t) \tag{2}$$

Here:

- γ is a learnable parameter
- σ is the sigmoid function
- $\nu_t = 0.01 \cdot \text{currentstep}$ is a value that increases over time during training

This formulation is inspired by method (2) from the paper "Unsupervised relational inference using masked reconstruction".

3.2 Dynamic Pruning Mechanism

The dynamic pruning mechanism is implemented in the `prune_and_rebuild` method of the `DynamicPruningCNN class`. The process works as follows:

1. After each epoch, the method checks the α values of each layer.
2. If an α value falls below a specified threshold (default 0.01), the corresponding layer is considered less important.
3. The identified layer is then pruned from the network.
4. A new model is created with one fewer layer.
5. The weights, and the past histories of gamma, and alpha from the remaining layers of the old model are transferred to the new model.

This process allows the network to adapt its structure during training, potentially removing redundant or less useful layers.

3.3 Training Process

The network is trained on the CIFAR-10 dataset, which consists of 60,000 32x32 color images in 10 classes. The dataset is prepared and used as follows:

1. The dataset is split into training (80%) and validation (20%) sets.
2. Pixel values are normalized in $[0, 1]$.
3. The training process includes the following steps for each epoch:
   (a) Forward pass through the network
   (b) Computation of loss using cross-entropy
   (c) Backpropagation and parameter update using Adam optimizer
   (d) Checking for potential layer pruning
   (e) Evaluation on the validation set
4. The training loop continues for a specified number of epochs (default 20), with the possibility of pruning occurring after each epoch.

4 Results and Analysis

4.1 Training and Validation Performance

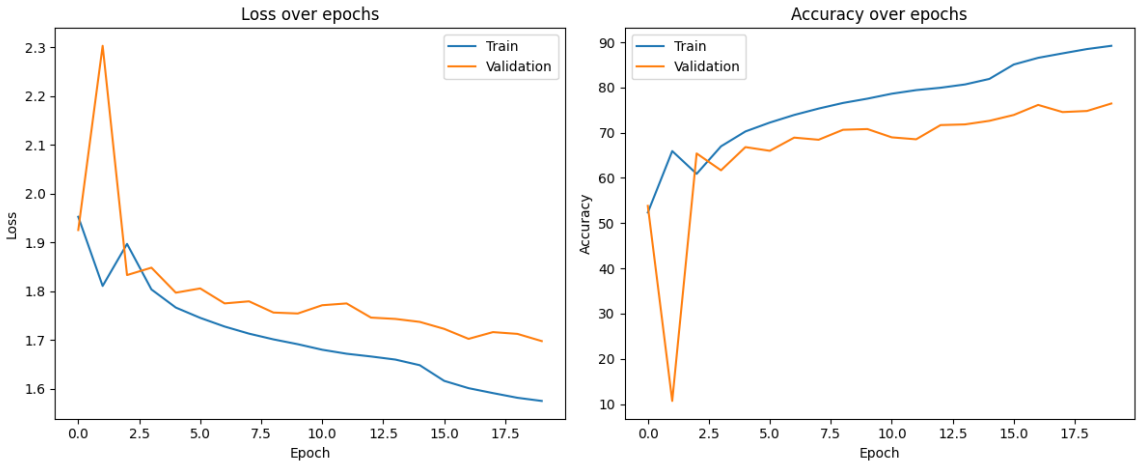

Figure 1: Training and Validation Loss/Accuracy

Figure 1 shows the training and validation loss and accuracy over the course of training, we see a bit of overfitting.

4.2 Alpha and Gamma Evolution

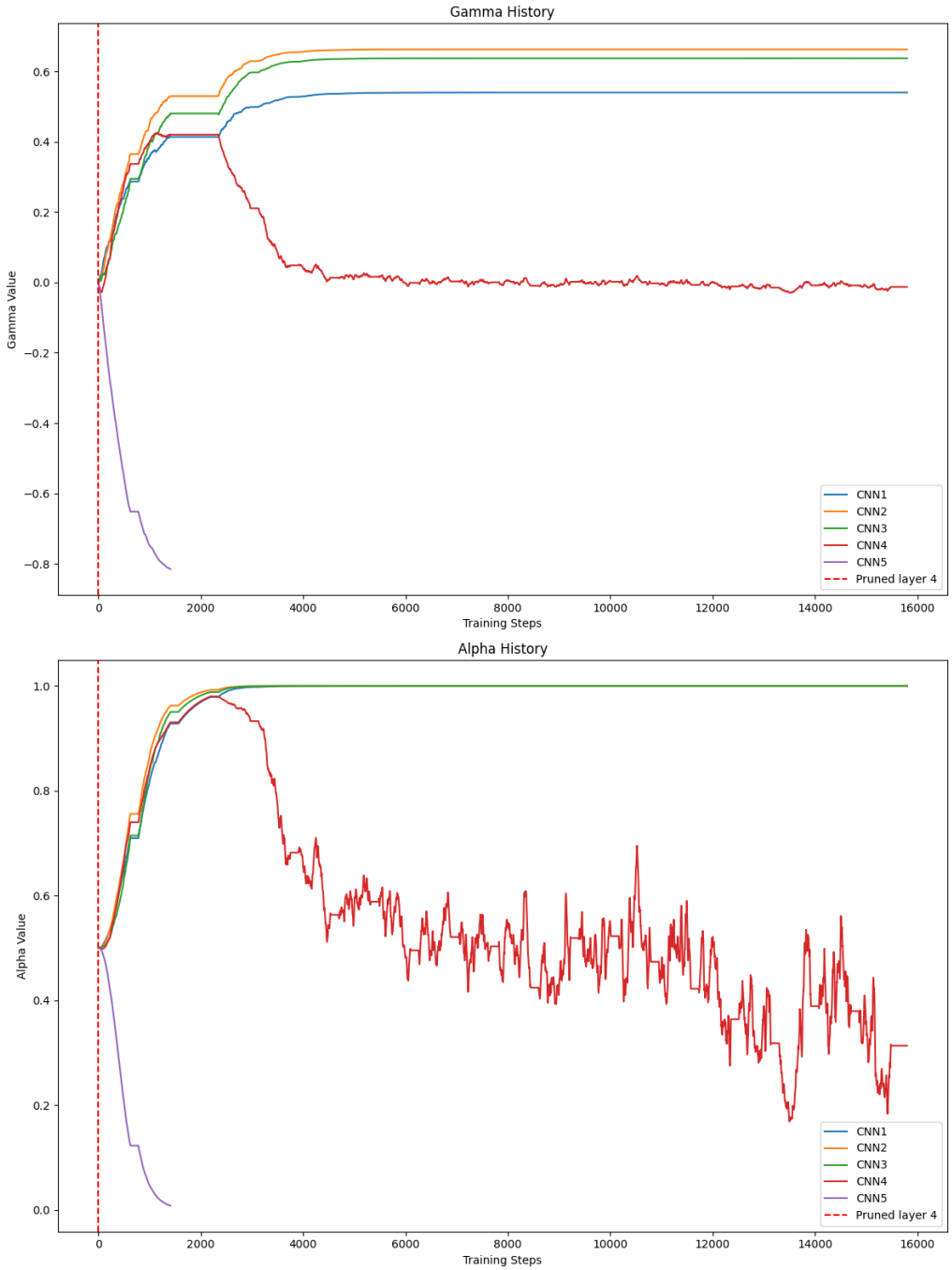

Figure 2: Evolution of Alpha and Gamma Values

Figure 3 illustrates the evolution of α and γ values for each layer throughout the training process, purple is stopped, the flat zone is due to parameter freezing, after that we see also the last CNN layer alive "in trouble".

4.3 Pruning Events

A typical workflow of my program is the following

```
Epoch 2/20
Model parameters:
layers.0.2.ga_compute.gamma: 0.2872
layers.1.2.ga_compute.gamma: 0.3652
layers.2.2.ga_compute.gamma: 0.2950
layers.3.2.ga_compute.gamma: 0.3371
layers.4.2.ga_compute.gamma: -0.6518
Training: 100%|██████████| 625/625 [00:43<00:00, 14.24it/s, loss=1.8104, accuracy=65.94%]
Pruned layer 4 at epoch 2
Epoch 2 summary:
Train Loss: 1.8104, Train Accuracy: 65.94%
Val Loss: 2.3028, Val Accuracy: 10.69%

Epoch 3/20
Model parameters:
layers.0.2.ga_compute.gamma: 0.4141
layers.1.2.ga_compute.gamma: 0.5302
layers.2.2.ga_compute.gamma: 0.4808
layers.3.2.ga_compute.gamma: 0.4204
Training: 100%|██████████| 625/625 [00:24<00:00, 25.32it/s, loss=1.8966, accuracy=60.91%]
Epoch 3 summary:
Train Loss: 1.8966, Train Accuracy: 60.91%
Val Loss: 1.8328, Val Accuracy: 65.42%
Unfreezing all layers
```

Figure 3: Pruning Workflow

As you can see the pruning happened successfully, moreover right after the pruning, we have a drop of performance on validation, but after an epoch the model has completely recovered.

The "flat zone" you observed is due to layer freezing, the only training there is performed on Fully Connected, after the un-freezing we observe a drop in performances of the last CNN layer, suggesting that maybe also this one is not needed.

4.4 Final Model Performance

The final model achieved a test accuracy of 76.32%. This performance is not bad for this naive model class.

5 Discussion

5.1 Effectiveness of Dynamic Pruning

We can see that even if our model has been pruned, both training, and validation accuracy increases during training, so that suggest that the idea might not be bad, and could be worth to develop some advanced tool on top on that idea.

5.2 Learnable Non-Linearity

The idea on custom non-linearity function with learnable γ parameter allow us to:

- Interpret better the model, in fact this is a Residual Network layer basically;
- Determine the impact on non-linearity, and model complexity;
- Give us a Pruning Criterion.

5.3 Challenges and Limitations

The limitations of this model is that if we are too much aggressive in pruning, we could end up in a too simple model; moreover there are several hyperparameters to fine-tune, so this is definetely the end of the story; Finally, a better freezing technique must be developed (supposing that this is a good choice, and by looking only at this data we don't know it).

6 Conclusion

This project implemented a Dynamic Pruning CNN with a custom, learnable non-linearity function for image classification on the CIFAR-10 dataset. The dynamic pruning mechanism shown in the work suggest some interesting results that can be furtherly improved. Overall, this approach is successfull, at least on the Cifar10 Dataset.

7 Future Work

Potential areas for future research and improvement include:

- Exploring different pruning criteria or thresholds
- Investigating the impact of initial network depth on pruning effectiveness
- Applying this approach to larger datasets or more complex tasks
- Combining dynamic pruning with other efficiency-focused techniques like quantization or knowledge distillation