Exploring Pruning Filters In Convolution Neural Network

Vincent Martineau

Department of Computer Science and Software Engineering, Université Laval

UNIVERSITÉ LAVAL

Introduction

We explore how it would be possible to use Convolution Pruning to reduce size of popular model and see how it affects performance of these network based on pruning. This should allow to run more complexe solution on smaller hardware.

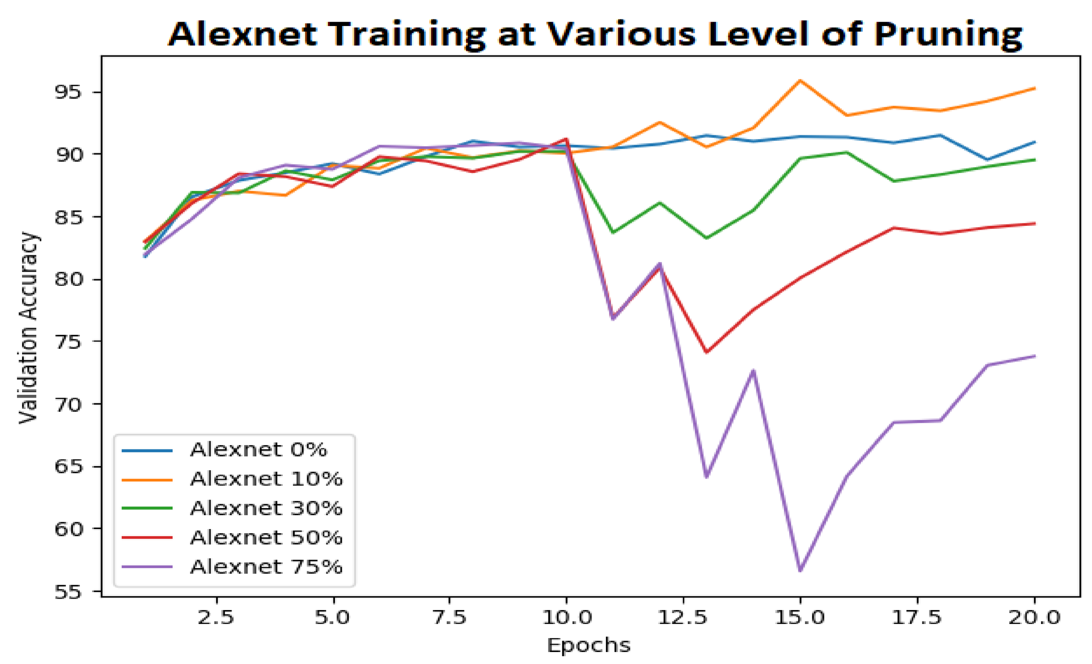
Related work :

- P.Molchanov et al. (2017) : Pruning Convolutional Neural Networks for Resource Efficient Inference.

Goals :

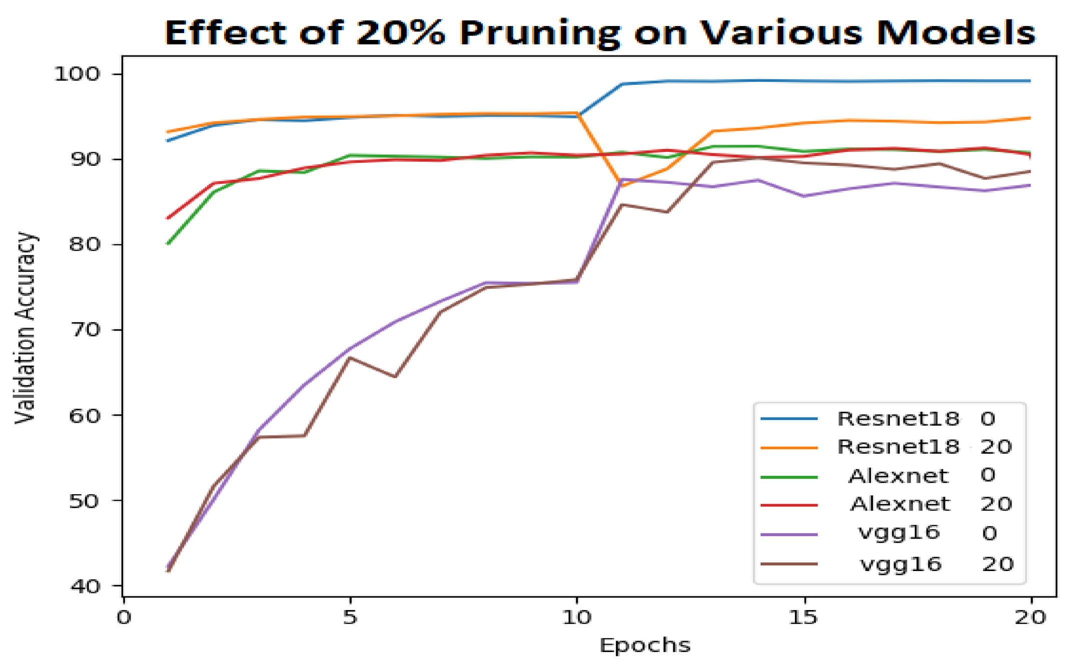
- Reduce training time to produce sufficient network.
- Provide a module that could handle multiple models.
- Explore the effect of pruning for speed and size.

Comparing Various Level of Pruning



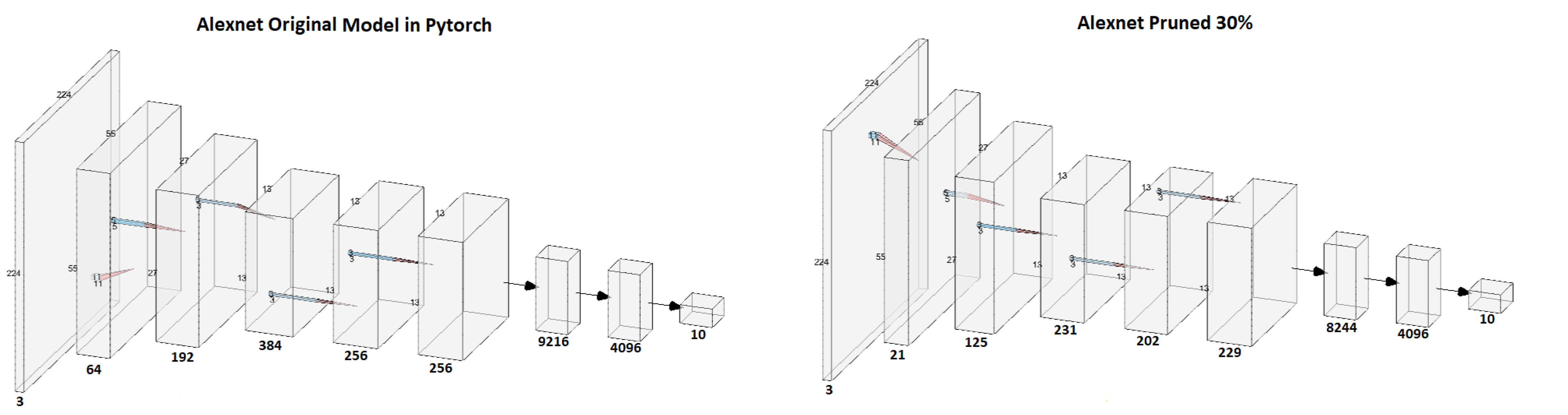
This graph compare various level of pruning. Each level of pruning is made on two iterations made after 10 epochs of training and 3 epochs or retraining per iteration. Pretrained weight were used to see the impact on transfer learning.

Comparing Pruning on Multiple Models



This graph explore the effect of pruning on different models. Each model shows the effect of pruning 20% of the convolution filters in one step after 10 epochs of training.

Example of Network Reduction



Comparing the effect of pruning on Alexnet. Left is the original model provided by pytorch. On the right is Alex net pruned 30%. In the case of network like Alexnet there is an important reduction of parameters based on the reduction of the first fully connected layer. There is also an important reduction in the first layers.

Algorithm

- Pretrain network with full paramters
- **Prepare Pruning**
  - Convert model to ONNX
  - Extract execution graph
  - Determine which layer can be pruned
- **Prune network**
  - Find number of filter to prune on iteration
  - Wort filter based on activation mean
  - Remove filters
  - Apply pruning effect to next layers
  - Reset optimizer
- Finalize training

Settings

- **Dataset** : Cifar10
- **Optimizer** : Stochastic Gradient Descent
- **Learning Rate** : 0.01
- **Momentum** : 0.0
- **Nesterov** : False
- **Batch Size** : 64
- **Use GPU** : Yes

Pruning :

- **Pretrain Epoch** : 10
- **Retrain Epoch** : 3
- **Total nb. Epoch** : 20

Nb Iteration on Alexnet : 2

Nb Iteration on Model Compare : 1

Performances

Comparing Alexnet Attributes After Pruning

	0%	10%		30%		50%		75%	
FLOPs(G)	0.815	0.665	(-18.4%)	0.445	(-45.3%)	0.283	(-65.3%)	0.133	(-83.7%)
Params(M)	57.0	54.8	(-3.86%)	47.3	(-17.0%)	37.6	(-34.0%)	27.0	(-52.6%)

Comparing Pruning On Various Models

	VGG			Alexnet			Resnet18		
	0%	20%	Diff	0%	20%	Diff	0%	20%	Diff
FLOPs(G)	17.31	12.23	-29.3%	0.815	0.592	-27.3%	1.83	1.13	-38.2%

Observations

- Not all convolutional layer can be pruned. Pruning layers before a residual connection is dangerous because both side of the residual connection must have the same side.
- When pruning in a convolution layer it is important to propagate to the following layers so the next layers have the right input size. This apply to convolution, linear and batchnorm layers..
- It is been seen that algorith leave only one filter on a layer. When this

Conclusion

Discussion :

- It is a **possible** to support multiple model type using the same module.
- The reduction on some model is **impressive** and could be run on lower trier hardware.

Future Works :

- Pruning proved to be a valid form of regulation.
- Would be possible to use different criteria to sort filters.