Pruning Filters In Convolution Neural Network

Vincent Martineau

Department of Computer Science and Software Engineering, Université Laval

Introduction

We explore how reducing network expressivity can affect performance in Convolution Neural Network (CNN). We implemented a pruner that can remove filters from convolution layer and explore the effect on transfer learning tasks.

Motivations :

- **Reduce** network size.
- **Improve** execution speed.

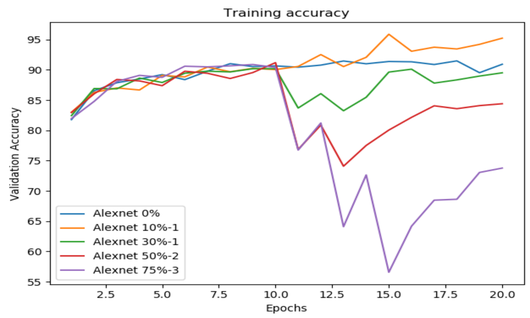
Related work :

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

Goals :

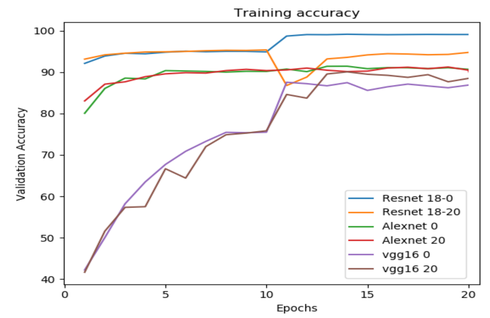
- Evaluate the impact of reducing network expression on performance.
- Compare training time on various model.
- Compare various strategies to prune.
- Provide a module that could.

Comparing Various Level of Pruning



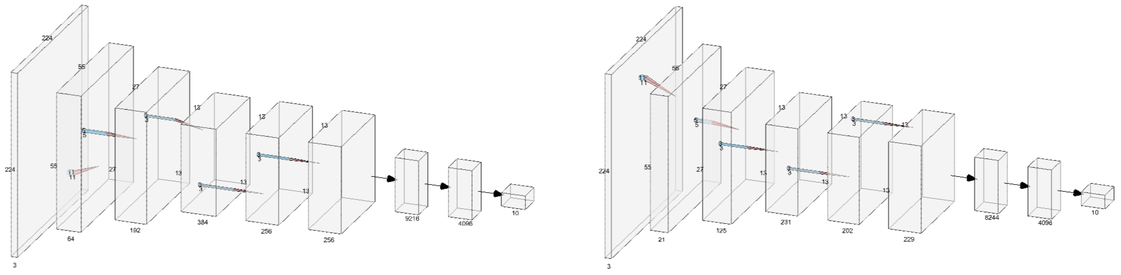
This graph compare various level of pruning. Each level of pruning is made on two iteration made after 10 epochs of training and 3 epochs or retraining. Pretrained weight were used to see the impact on transfer learning.

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. In this case transfer learning is not applied.

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 fully connected layer. Their 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
- Reset optimizer
- Finalize training

Settings

Labeling task net

Two nets working together : the first predicts OOV embeddings (see OOV handling net section) and the second one predicts tags.
The simple architecture of the labeling net is used to emphasize the usefulness of our module, and to minimize the influence of other factors.

Observation

- Not all convolutional layer can be pruned. Pruning layer 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. So the next layer have the right input size. This apply to convolution, linear and batchnorm layers.
- The algorithm used tend prefer removing filters that are deeper in the model and it is not uncommon to try to prune all filter in a layer.
- When pruning it is important to reset optimizer.

Performance gain

Task	Metric	Random Emb.	Our module	Gain
NER	F1	77.56	**80.62**	3.9%
POS	acc.	91.41	**92.58**	1.2%

The impact of our model on two NLP downstream tasks. We compare our OOV embeddings prediction scheme against random embeddings.

Conclusion

Discussion :

- **Morphology** and **context** help predict useful embeddings.
- **The attention mechanism works** : depending on the task, the network will use either more the context or the morphology to generate an embedding.