Reproduction of SNOW paper

Deep Learning with Multiple Tasks 2020/2021

Wiktor Daniec, Paweł Goliszewski, Konrad Wójtowicz

1 Introduction

Creating models which will be able to handle multiple tasks is current object of research in machine learning field. People usually do not need thousands of examples (like many ML model do) to learn how to classify objects into some number of classes of object - they usually use information learned through entire life and adapt quickly to new tasks. Idea is to use 'knowledge' from model trained on the other task to learn faster how to manage other task. Authors of paper [1] tackle the problem of image classification. They use pretrained model with smaller model connected to it to perform classification task on other classes. Our goal was to try to reproduce their results.

2 SNOW model architecture

This model uses 2 models connected via operation called Channel Pooling which will be described later. One model (called source model) is ResNet50 pretrained on ImageNet dataset with frozen weights. It is connected after each convolutional layer with second model (called delta model) which is smaller version of Resnet (in our experiments channels reduced by 8) with randomly initialized and trainable weights. We put image tensor to both model inputs, and we use only the output of the delta model into consideration.

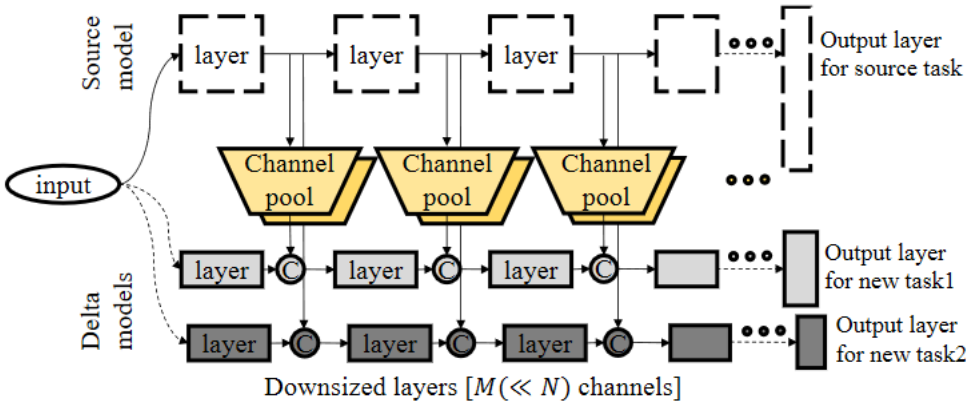

Figure 1: Illustration of SNOW model

2.1 Channel pooling operation

Channel pooling (CP) operation behaves differently during training stage and the evaluation stage.
CP layer contains weight vector w of size N where N is also number of input tensor layers. If it is training time we sample vector size N from normal distribution of mean 0 and standard deviation σ, and add it to the weights vector obtaining $rand_v$ vector. We take indexes of the L biggest values of $rand_v$. We take L channels of the input given by those indexes multiplied by w element-wisely. We concatenate the result to the corresponding input int the delta model.

3 Transfer learning

We use some standard transfer learning techniques for comparison. Those techniques are:

- full fine-tuning (FT) - We use pretrained ResNet50, and we modify last layer to adjust number of classes to a new task. Then we train network with all trainable parameters on a new task.
- feature extraction (FE) - Similarly as above, but weights of first 3 bottlenecks are frozen.

Algorithm 1: Channel Pooling pseudocode

w: channel pooling weight ($N \times 1$ vector);
x: input tensor (with N channels);
Function `ChannelPoolingFroward(`x`):`
  if *training* **then**
    $rand_v = w + \mathcal{N}(0, \sigma)$;
    $\_, idx = \text{topK}(rand_v, L)$;
    $selected\_weights = w[idx]$
  else
    $selected\_weights, idx = \text{topK}(rand_v, L)$;
  end
  $x = x[idx]$;
  $x = x \bigotimes selected\_weights$;
return x

- final layer only (FO) - Similarly as above, but only last linear layer is not frozen.

4 Experiments

4.1 Datasets

We used 5 datasets: Action, Cars, DTD, Birds, Food-101.
Action dataset contains 9532 images (4000 in training dataset and 5532 in the test dataset) divided into 40 classes. The images are photos of people performing an action, for instance: jumping, fixing a car, playing guitar etc.
Cars dataset contains 196 categories of car of 16185 images in total (8144 in training dataset and 8041 in the test dataset) They are photos of different models of cars like: 2012 Tesla Model S, 2012 BMW M3 coupe, etc.
There are 47 categories in DTD (Describable Textures Dataset) and 5640 images (3760 in training dataset and 1880 in the test dataset). Images contains different textures, for example: grid pattern, stripped pattern etc.
Birds dataset is made of 8891 photos (5994 in training set and 2897 in test dataset) of 200 species of birds.
Food-101 dataset contains 11000 photographs of 101 categories of meals like: hot dog, risotto etc. There are 75750 images in train dataset and 25250 ones

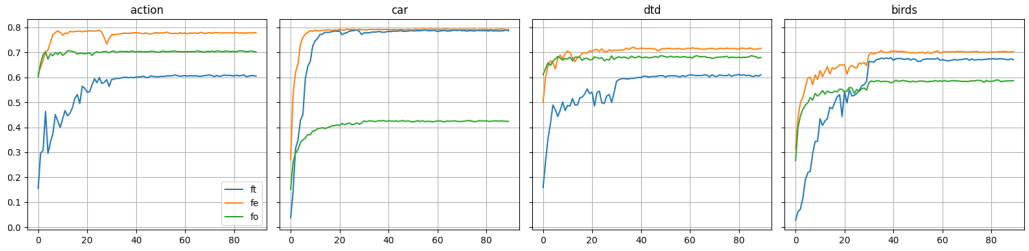

Figure 2: Accuracy at test set of Transfer learning methods

Parameter	K	M	σ^2	batch size	initial learning rate	momentum	weight decay	training epochs
Value	8	8	0.001	64	1.0	0.9	0.0001	50

Table 1: SNOW model hyperparameters

in the test dataset. Unfortunately we do not have results of training on this dataset because of lack of time and computational resources.
For each dataset we perform normalization and resize to 224×224.

4.2 Transfer learning

On 2 we present validation accuracy during training of three TL methods mentioned earlier.

4.3 SNOW

We performed experiments on all datasets using SNOW model. Hyperparameters' values are in table 1. K is parameter which means how many times delta model is smaller (by number of channels). We take $\frac{1}{M}$ of input channels as a parameter L of channel pooling layer. We use stochastic gradient descent algorithm with momentum and weight decay. We also use step learning rate scheduler which multiplies learning rate by 0.1 every 10 epochs. As we see from the plots: 3 and 4 during different runs training stuck in different local minima. This differs from what was shown in [1]. Models, we have implemented, were not learning so quickly and could not reach accuracies those obtained by transfer learning techniques. The reason of that can

Dataset	Action	Cars	DTD	Birds
Max accuracy on test set	10.59%	12.70%	30.53%	1.38%

Table 2: SNOW results (last running)

Figure 3: Plot of accuracy by epoch (last running)

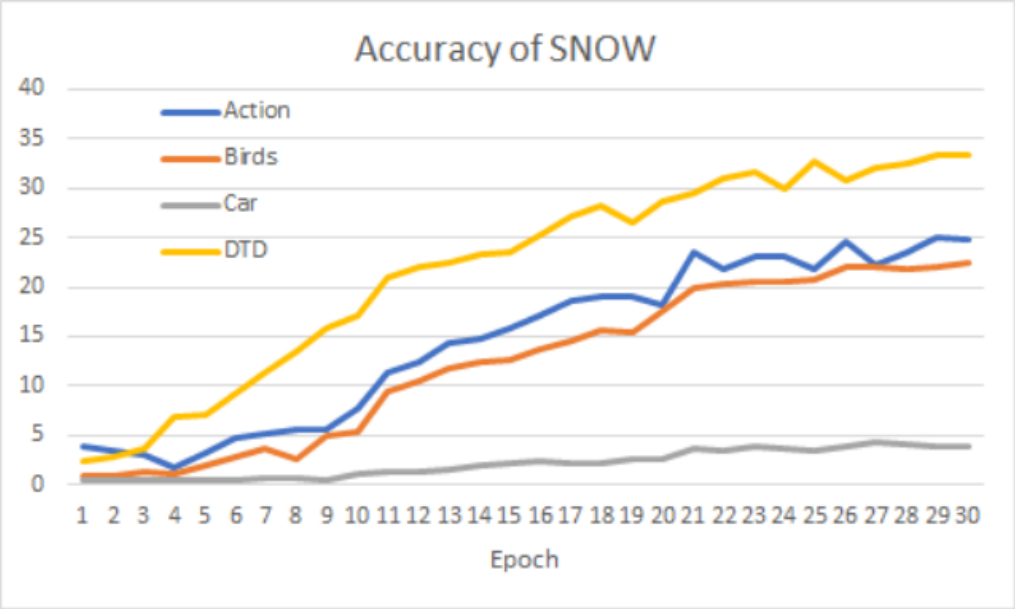

Figure 4: Plot of accuracy by epoch (from former run)

Dataset	Action	Cars	DTD	Birds
SNOW	217.09	62.86	214.04	218.89

Table 3: Throughput (images per second)

be differences between our implementation and what authors have coded. Maybe Channel Pooling operation is placed too densely in the model - we suppose that injection of results of frozen weights of source model may perform poor if it is used in every convolutional layer.

4.4 Evaluation and Throughput

We see that result from cars dataset is much lower than others. It can be caused by bigger size of those images. We can not compare results with those from paper because we use different computation server and graphics cards. We used Nvidia RTX2080Ti and RTX1080 GPUs. Each has memory of 11GB RAM what forced us to decrease batch size to 64.

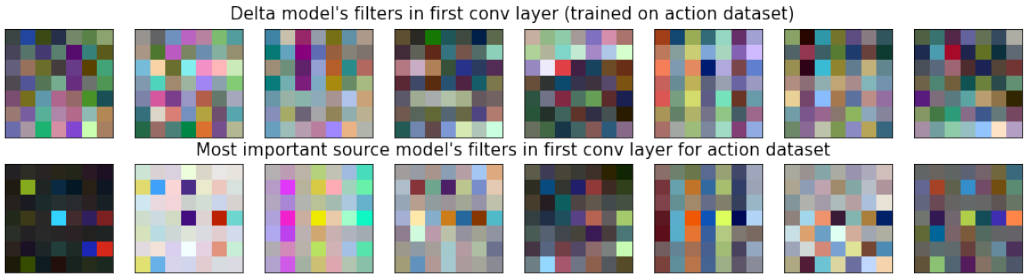

Figure 5: Visualisation of filters trained on Action dataset

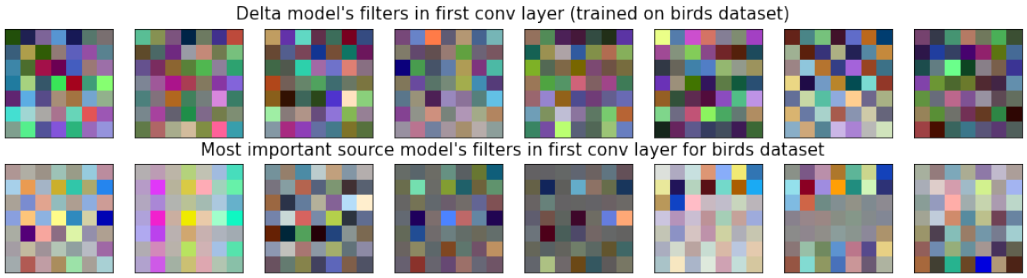

Figure 6: Visualisation of filters trained on Birds dataset

4.5 Visualization of filters

We have done visualization of $3x7x7$ filters of models trained on each of the datasets. Plots (5, 6, 7 and 8) contain delta model's filters of initial convolutional layer in the first row and source model's ones (eight of highest weigght in channel pooling layer).
We see that the source model's filters vary more than those from the delta models. This could mean filters in well-trained model are more specialized.

5 Conslusions

Unfortunately we did not manage to reproduce results of authors of [1]. Probably it was a hard task because of no code published by authors and lack of the details in paper. It was hard to decide how densely channel pooling operation should be placed. We placed it after each convolutional layer in the model. We have sent emails to the authors to get more details, but we have not got any responses. The SNOW approach did not learn as quickly as trans-

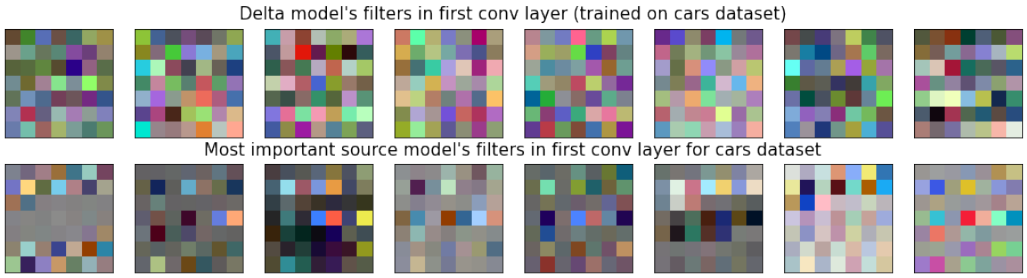

Figure 7: Visualisation of filters trained on Action dataset

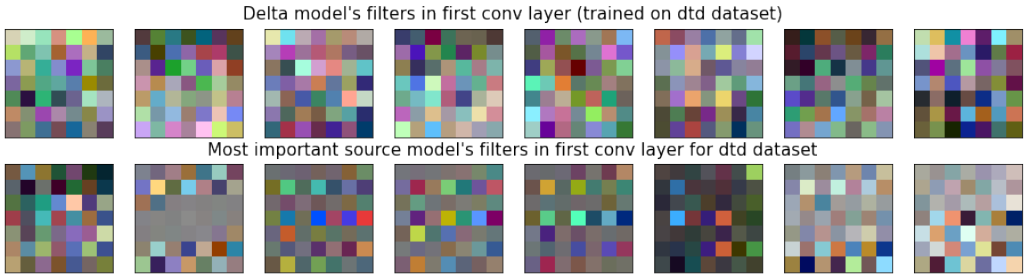

Figure 8: Visualisation of filters trained on DTD dataset

fer learning approaches, but it is still faster than learning from scratch. We did not manage to perform training on Food-101 dataset because of technical issues (we were unable to parallelize dataloader because of weird exception) and lack of time. The computational machine which belong to Robotics and Artificial Intelligence Students' Association of Jagiellonian University was not so fast as one used by authors of the paper.

```
    result = sel_weight.view(1,-1,1,1) * x[:, indices, :, :]
  File "/home/pgoliszewski/Desktop/venv/lib/python3.6/site-packages/torch/utils/data/_utils/signal_handling.py", line 66, in handler
    _error_if_any_worker_fails()
RuntimeError: DataLoader worker (pid 8233) is killed by signal: Killed.

Process finished with exit code 1
```

Figure 9: Exception which was showing up randomly after several epochs of training with paraller data loaders

References

[1] https://openreview.net/pdf?id=rJxtgJBKDr
Chungkuk Yoo, Bumsoo Kang, Minsik Cho, SNOW: Subscribing to Knowledge via Channel Pooling for Transfer & Lifelong Learning of Convolutional Neural Networks, ICLR 2020