IN5400

Alex

Week 07: Fine Tuning of neural networks

[The following notes are compiled from various sources such as textbooks, lecture materials, Web resources and are shared for academic purposes only, intended for use by students registered for a specific course. In the interest of brevity, every source is not cited. The compiler of these notes gratefully acknowledges all such sources.]

1 two exemplary ways of fine-tuning, setup with a separate validation set

- take the 102 class flowers dataset `https://www.dropbox.com/s/w9yfsr8xifkngs7/102flowersn.zip?dl=0` and write a dataset class which can work with a train/val/test split for it. The difference to the 102 flowers from oxford is that I provided in it for you train/val/test split for your convenience (bowing politely in front of the students).
- take any deep network you like which has pretrained weights (a smaller resnet, a smaller densenet are training fast)
- train a deep neural network in three different modes

  (A) once without loading weights and training all layers.

  (B) once with loading model weights before training and training all layers,

  (C) once with loading model weights before training and training only the last **one** trainable layer (note: for quite some problems, the approach B is better than C)

  For each of these 2 modes select the best epoch by the performance of the model on the validation set. Typically less than 20 epochs should suffice for training when using finetuning. You can run also optionally a selection over a few learning rates, if you use a GPU.

What do you need to do for steps when you start with a code like the MNIST training code?

- write a new dataloader for your training dataset, the flowers
- adjust paths for data (and if necessary for label paths/files or files determining splits into train/val/test)
- write the code so that it works with paths relative to the directory of your script
- as you did not have data augmentation so far (next lecture), you can simply use the following:

I am aware that I am using here the imagenet mean and the imagenet standard deviation.

- use some deep learning model from the model zoo, load its weights before training for settings B and C
- adjust the number of classes in the last linear/dense layer
- in the optimizer provide the proper parameters for training for settings A,B,C
- collect and plot curves of the training loss, the validation set loss and the validation set accuracy. also print the final test accuracy of the selected model
- A note: Calling a model constructor with `pretrained=True` does not tell you what really goes on when one. Check `https://github.com/pytorch/vision/blob/master/torchvision/models/resnet.py` to see what routine is used to load a model.

1.1 Bonus Task

Try out the effect of mixup training on your problem:

`https://arxiv.org/abs/1710.09412`. The paper is good to read and it is not too complicated to implement that one. I have no guess how much it can bring here because the flowers are a relatively simple problem. It is good to know this kind of trick anyway.

2 GPU resources (more exercises in the next section!)

you have two options: use your own GPU, or use the university provided resources

- ml6.hpc.uio.no
- ml7.hpc.uio.no

How to use them ?

- log in using ssh and your ifi username:

On windows PuTTY or MobaXterm may help you. On mac you can use ssh as is. I do use windows, but for games :D.

- each of these nodes has 8 GPUs, each with 11 Gbyte GPU Ram. The critical resource will be GPU RAM. If you go over the limit, your script will die with a mem allocation error.
- **keep the scripts at a training batchsize of 16 with using a resnet18 - in order to keep mem usage below 2Gbyte**. This does not apply if you use your own GPU, but then keep it below 5Gbyte (in case i got to check your code on my home GPU, alternatively i will reduce your batchsize manually).
- use `nvidia-smi` to see which on which GPUs scripts are running and how much memory is used on each GPU. Choose a GPU such which has still 2 Gbyte RAM unused.
- to start a script on a specific GPU with numerical number $x \in \{0, \ldots, 7\}$ use the following command below. However this will stop when you log out of ssh. Thus this makes sense only to debug your code.

- to start a script which does not hang up on logout (on a specific GPU with numerical number $x \in \{0, \ldots, 7\}$), please use

What does this do?

- nohup starts the command without hangup
- > out1.log redirects normal output onto out1.log
- 2 > error1.log redirects error messages onto error1.log
- & places the job in the background

- do not start a script when there are already 5 jobs running on it or when it is foreseaable that your 2Gbyte wont fit into this GPU RAM.
- how to kill your own process?

  ps -u pineapple

  ↑ shows only the processes of the user pineapple

  ps -u pineapple | grep -i python

  ↑ shows only the processes of user pineapple which are python. The -i makes a case sensitive grep search. If you see nothing, then you may have mistyped your command, or you are not using python, or your process has already finished.

  – both of these will show you process ids (PID)s

  kill -9 PID

  ↑ kills your process with pid PID

3 Some theory

- You are given a 2-dimensional convolution with feature map input size $(78, 84)$. When using a kernel of size $(5, 5)$ and stride 3 with padding of 2, what will be the spatial size of the feature map which is the output of the convolution? Note that the *spatial* size does not depend on the number of input or output channels.
- You are given a 1-dimensional convolution. When using a kernel of size 9 and stride 3 with padding 1, which spatial input size do you need to have, so that you have a spatial output size of 16?
- You are given a 2-dimensional convolution. When using a kernel of size $(3, 5)$ and stride 2 with padding of 0, what will be the spatial size of the feature map which is the output of the convolution, which spatial input size do you need to have, so that you have a spatial output size of $(128, 96)$?

Solution

Let O = output dimension, M = input dimension, r = padding, $ksize$ = kernel size, and S = stride. Then we have the formula

$$O = \text{floor}\left(\frac{M + 2r - ksize}{S}\right) + 1 \tag{1}$$

1. $O = \text{floor}\left(\frac{78+4-5}{3} + 1, \frac{84+4-5}{3} + 1\right) = (25, 28)$
2. $16 = \text{floor}\left(\frac{M+2-9}{3} + 1\right)$. One possible value of M can be calculated from $3 * 15 = M - 7 \Longrightarrow M = 52$.

3a Let (M_1, M_2) be the input size. Then $O = \text{floor}\left(\frac{M_1-3}{2}+1, \frac{M_2-5}{2}+1\right)$.

3b $(128, 96) = \text{floor}\left(\frac{M_1-3}{2}+1, \frac{M_2-5}{2}+1\right)$. Possible values of M_1, M_2 can be calculated by $M_1 = 127 * 2 + 3 = 257, M_2 = 95 * 2 + 5 = 195$.

4 Some more theory

How many trainable parameters are in

- a 2-D convolutional layer with input $(32, 19, 19)$, kernel size $(7, 7)$, stride 3, 64 output channels?
- a 2-D convolutional layer with input $(512, 25, 25)$, kernel size $(1, 1)$, stride 1, 128 output channels?
- how many multiplications and how many additions are performed in the first case above?

Solution

Note that the number of parameters for a convolutional network depends upon the kernel size, and number of input and output channels, but not on the input size or the stride.

1. The number of trainable parameters $= (32 * 7 * 7 + 1) * 64 = 1569 * 64 = 100416$. Here $32*7*7$ is the kernel size for a 7x7 window with 32 channels, 1 is for the bias, and there are 64 output channels.
2. $(512 + 1) * 128 = 65664$.

3a The number of multiplications is $32 * 7 * 7 = 1569$ each time an inner product is calculated. The output shape for a 7x7 window over a 19x19 image with stride 3 is $\text{floor}\left(\frac{19-7}{3}+1, \frac{19-7}{3}+1\right) = (5, 5)$ so there are 25 inner product evaluations per output channel. The total number of multiplications is then $1569 * 25 * 64 = 2510400$.

3b The number of additions = 1 each time an inner product is calculated. The total number of additions is therefore $1 * 25 * 64 = 1600$.

5 Logistic sigmoid saturation

Consider the logistic sigmoid output for two outputs given as

$$\sigma(y) = \frac{1}{1 + e^{-y}}$$

- Compute its derivative $\frac{\partial \sigma}{\partial y}(y)$.

Prove that

- for $y \longrightarrow -\infty$ we have $\frac{\partial \sigma}{\partial y}(y) \longrightarrow 0$, meaning that the derivative is vanishing. How ? For example you can obtain an upper bound on the derivative which holds if y is negative and sufficiently large in absolute value. Then take the limit of that upper bound.
- for $y \longrightarrow +\infty$ we have $\frac{\partial \sigma}{\partial y}(y) \longrightarrow 0$, meaning that the derivative is vanishing as well. How ? For example you can obtain an upper bound on the derivative which holds if y is positive and sufficiently large in absolute value. Then take the limit of that upper bound.

Note: since the softmax is the generalization of the logistic sigmoid to more than two outputs, the same argument holds for it as well. To see this, consider the following equivalence between a softmax of two outputs and the logistic sigmoid of the difference of these outputs:

$$\begin{aligned} S(y_1, y_2) &= \left(\frac{e^{y_1}}{e^{y_1} + e^{y_2}}, \frac{e^{y_2}}{e^{y_1} + e^{y_2}} \right) \\ &= \left(\frac{1}{1 + e^{y_2 - y_1}}, \frac{1}{e^{y_1 - y_2} + 1} \right) \\ &= (\sigma(-y_2 + y_1), \sigma(-y_1 + y_2)) \end{aligned}$$

5.1 solution

$$\sigma'(y) = \frac{\partial \sigma}{\partial y}(y) = \frac{e^{-y}}{(1 + e^{-y})^2}$$

plausibility check: the sigmoid is strictly monotonously growing and its derivative is always positive.

Proof of

$$\lim_{y \longrightarrow -\infty} \frac{\partial \sigma}{\partial y}(y) = 0$$

idea before we show the proof (I think it is important to have some intuition to guide oneself when trying to derive something...): $y \longrightarrow -\infty \Rightarrow \frac{1}{(1+e^{-y})^{\alpha}}$ will be the term which draws the derivative towards zero, while e^{-y} will diverge towards $+\infty$. so we must upper bound $\frac{e^{-y}}{(1+e^{-y})^{2-\alpha}}$. Its worth to try $\alpha = 1$.

$$\begin{aligned} \frac{e^{-y}}{(1 + e^{-y})^2} &= \frac{1}{1 + e^{-y}} \frac{e^{-y}}{1 + e^{-y}} \leq \frac{1}{1 + e^{-y}} \frac{1 + e^{-y}}{1 + e^{-y}} \\ &= \frac{1}{1 + e^{-y}} \end{aligned}$$

How to prove that this converges to zero for $y \longrightarrow -\infty$?

Formally convergence to a limit b means when the input goes to $-\infty$:

$$\lim_{y \longrightarrow -\infty} f(y) = 0 \Leftrightarrow \forall \epsilon > 0 \; \exists K(\epsilon) \text{ such that } \forall y < K : \; |f(y) - b| \leq \epsilon$$

Choose an $\epsilon > 0$:

$$\begin{aligned} &\left| \frac{1}{1+e^{-y}} - 0 \right| \leq \epsilon \Leftrightarrow \frac{1}{1+e^{-y}} \leq \epsilon \Leftrightarrow \frac{1}{\epsilon} \leq 1 + e^{-y} \\ \Leftrightarrow &\frac{1}{\epsilon} - 1 \leq e^{-y} \Leftrightarrow \ln\left(\frac{1}{\epsilon} - 1\right) \leq -y \\ \Leftrightarrow &-\ln\left(\frac{1}{\epsilon} - 1\right) \geq y \end{aligned}$$

Makes sense ? say $\epsilon = 0.001 = 10^{-3}$, then $\ln(10^3 - 1) = \ln(999)$ is positive, and $-\ln(10^3 - 1)$ is a negative number.

Remember here

$$f \leq g \Leftrightarrow -f \geq -g$$

Thus: $K(\epsilon) = -\ln\left(\frac{1}{\epsilon} - 1\right)$ satisfies and we have proven convergence.
Alternatively you can use the L'Hopital Rule:

$$\lim_{y \longrightarrow -\infty} \frac{\partial \sigma}{\partial y}(y) = \lim_{y \longrightarrow -\infty} \frac{\frac{d}{dy} e^{-y}}{\frac{d}{dy}(1+e^{-y})^2} = \lim_{y \longrightarrow -\infty} \frac{-e^{-y}}{2(1+e^{-y})(-e^{-y})} = \lim_{y \longrightarrow -\infty} \frac{1}{2(1+e^{-y})} = 0.$$

Proof of

$$\lim_{y \longrightarrow +\infty} \frac{\partial \sigma}{\partial y}(y) = 0$$

idea before we show the proof: $y \longrightarrow \infty \Rightarrow e^{-y}$ will be the term which draws the derivative towards zero. So we must upper bound $\frac{1}{(1+e^{-y})^2}$.

$$\frac{e^{-y}}{(1+e^{-y})^2} = e^{-y} \frac{1}{(1+e^{-y})^2}$$

We know: for any $y > 0$ it holds that $e^{-y} > 0$, thus

$$\frac{e^{-y}}{(1+e^{-y})^2} = e^{-y} \frac{1}{(1+e^{-y})^2} \leq \frac{e^{-y}}{(1+0)^2} \overset{y \to \infty}{\longrightarrow} 0$$

The proof to find a K such that $\forall y \geq K$ we have $e^{-y} < \epsilon$ is obvious. $K = -\ln \epsilon$