Homework 5
601.482/682 Deep Learning
Spring 2019

March 29, 2019

Due Fri. 03/29 11:59pm.
Please submit a latex generated PDF
to Gradescope with entry code MYRR74

1. Computing Network Sizes

   (a) Given the multi-layer perceptron below in Figure 1, compute the number of free parameters for this model (Note that biases are explicitly disregarded in this case).
   Ans:
   1st layer: 3x3+2=11
   2nd layer: 4x3=12
   prediction: 3
   total: 11+12+3=26

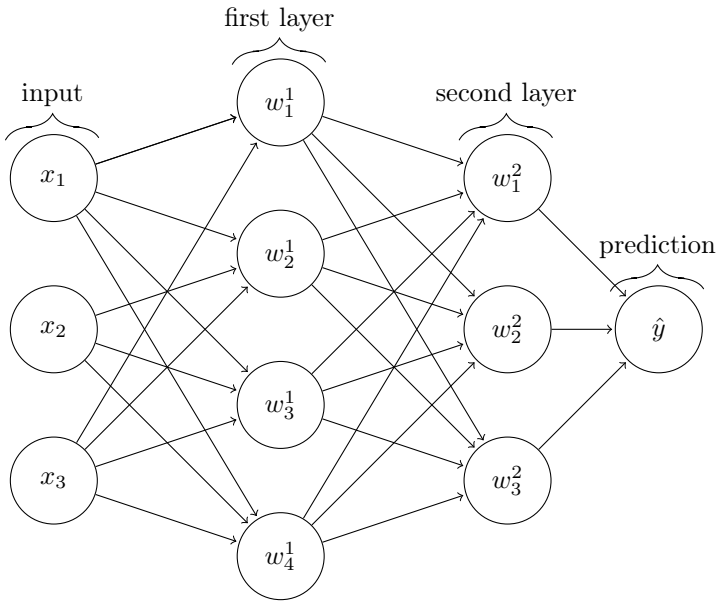


Figure 1: Multi-layer Perceptron

In this part, you will analyze the size of a state-of-the-art architecture not discussed in class. In the ImageNet ILSVRC 2014 contest, two fairly deep networks performed very well: The VGG network and the GoogLeNet. While VGG performed best in the localization task and ranked second in classification, GoogLeNet won the classification task achieving a top-5 error rate of 6.67%. Figure 2 presents the architecture of GoogLeNet, which is built up of 9 stacked "inception" modules displayed in Figure 3.

   (b) Consider the layer "inception (3a)" from Table 1(Figure 4) in the GoogLeNet paper. Notice how "3x3 reduce" and "5x5 reduce" are used between layers - from section 5, this "stands for the number of 1x1 filters in the reduction layer used before the 3×3 and

5×5 convolutions". Compute the number of free parameters for the "inception (3a)" layer.
Ans: number for 1x1: (1x1x192+1)*64 = 12,352
number for 3x3 reduce: (1x1x192+1)*96 = 18,528
number for 3x3: (3x3x96+1)*128 = 110,720
number for 5x5 reduce: (1x1x192+1)*16 = 3,088
number for 5x5: (5x5x16+1)*32 = 12,832
number for Pool Proj: (1x1x192+1)*32 = 6,176
Total = 163,696

(c) Now, consider that the reduction portion of "3x3 reduce" and "5x5 reduce" were omitted (i.e. no 1x1 filters were used before the 3x3 and 5x5 convolutions). Compute the number of free parameters for this updated "inception (3a)" layer. How does this compare to the original layer?
Ans: number for 1x1: (1x1x192+1)*64 = 12,352
number for 3x3: (3x3x192+1)*128 = 221,312
number for 5x5: (5x5x192+1)*32 = 153,632
number for Pool Proj: (1x1x192+1)*32 = 6,176
Total = 393,472

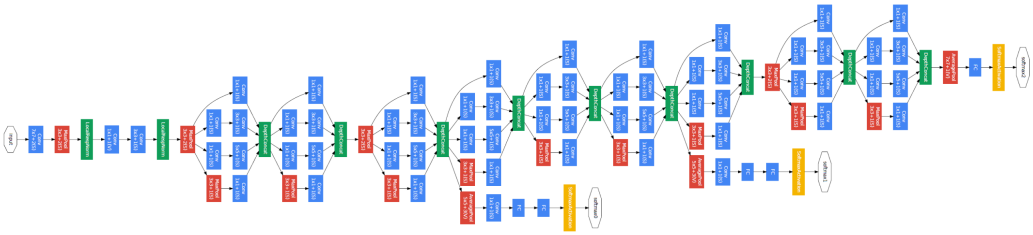
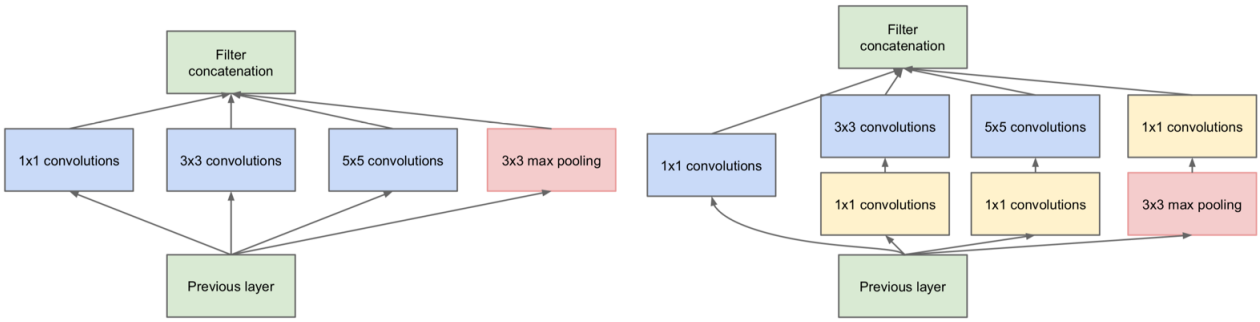


Figure 2: GoogleNet Architecture



(a) Inception module, naïve version

(b) Inception module with dimensionality reduction

Figure 3: Inception Module

type	patch size/ stride	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj	params	ops
convolution	7×7/2	112×112×64	1							2.7K	34M
max pool	3×3/2	56×56×64	0								
convolution	3×3/1	56×56×192	2		64	192				112K	360M
max pool	3×3/2	28×28×192	0								
inception (3a)		28×28×256	2	64	96	128	16	32	32	159K	128M
inception (3b)		28×28×480	2	128	128	192	32	96	64	380K	304M
max pool	3×3/2	14×14×480	0								
inception (4a)		14×14×512	2	192	96	208	16	48	64	364K	73M
inception (4b)		14×14×512	2	160	112	224	24	64	64	437K	88M
inception (4c)		14×14×512	2	128	128	256	24	64	64	463K	100M
inception (4d)		14×14×528	2	112	144	288	32	64	64	580K	119M
inception (4e)		14×14×832	2	256	160	320	32	128	128	840K	170M
max pool	3×3/2	7×7×832	0								
inception (5a)		7×7×832	2	256	160	320	32	128	128	1072K	54M
inception (5b)		7×7×1024	2	384	192	384	48	128	128	1388K	71M
avg pool	7×7/1	1×1×1024	0								
dropout (40%)		1×1×1024	0								
linear		1×1×1000	1							1000K	1M
softmax		1×1×1000	0								

Figure 4: Table 1: GoogLeNet incarnation of the Inception architecture.

2. Receptive Fields

(a) Consider the network in Figure 6, which is constructed by 2 5×5 convolution kernels. What is the receptive field of one pixel in layer 2? Please draw a 2D graph (excluding the channel dimension) to illustrate your calculation.
Ans: The receptive field of one pixel in layer 2 is 9 x 9. In the figure below, the right matrix has 25 fields from layer 1 numbered, each field has a corresponding 5 x 5 kernel in the input image. The left matrix demonstrates the receptive fields of kernel number 1, 5, 21, 25 in the input image, as demonstrated on the left.

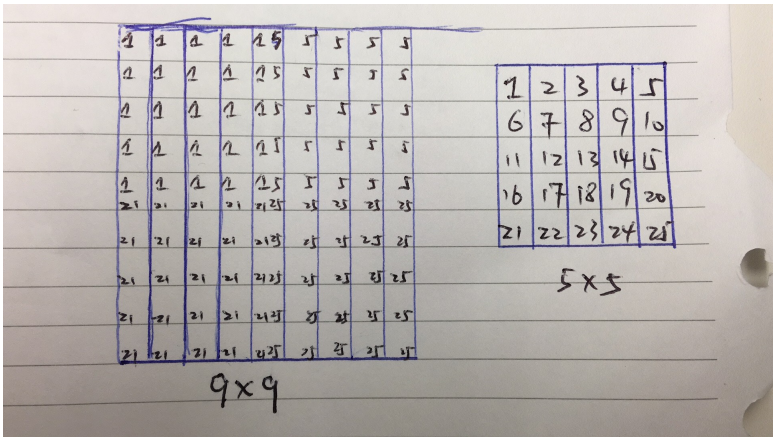


Figure 5: 2D graph illustrating receptive field of input image

(b) Now consider that we add a n-by-n max-pooling layer following layer 2. What will be the receptive field after the max-pooling layer?
Ans: one pixel from max-pooling layer's receptive field in:
layer 2: n x n
layer 1: (5+(n-1)*2) x (5+(n-1)*2)
= (2n+3) x (2n+3) let L1 = (5+(n-1)*2)
receptive field of input: (5+(L1-1)*1) x (5+(L1-1)*1)

$= \mathbf{(4 + L1)\ x\ (4 + L1)}$
$= \mathbf{(7 + 2n)\ x\ (7 + 2n)}$

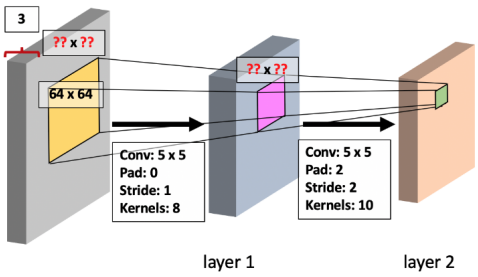


Figure 6: Convolutional Architecture

3. Gradient Descent Optimization

(a) Consider Figure 8, which depicts a cost function $L(x) : \mathbb{R}^2 \to \mathbb{R}$. The red dot represents the current estimate of $\mathbf{x}_t = [x_1, x_2]$ at time t. Please give a rough sketch of the direction of update steps that would be taken by vanilla SGD.
Ans: The sketch of the direction update is drawn in red.

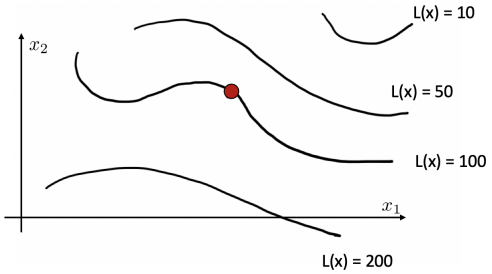


Figure 7: Contour lines of an arbitrary cost function with current estimate $\mathbf{x}_t$.

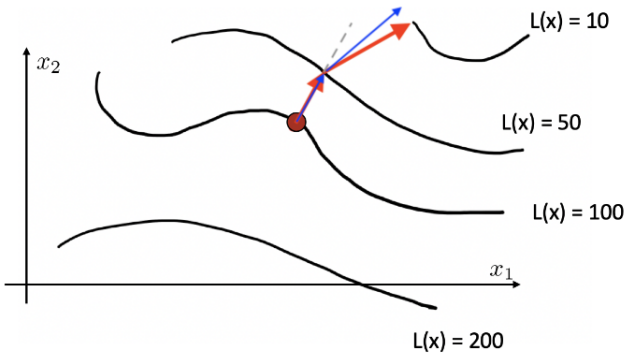


Figure 8: sketch of direction update for 3(a) drawn in red and (b) drawn in blue.

(b) It is worth mentioning that the contour lines shown in Fig. 8 will change during optimization since the loss is evaluated over a single batch rather than the whole dataset. As discussed in class, this observation suggests that for unfortunate updates we might get stuck in saddle points, where the vanilla SGD gradient is 0. One way to combat this problem is to use first and/or second order momentum. Please briefly explain why these moments are helpful and how they would change the update direction sketched in (a).
Ans:
The change of direction update is drawn in the same graph in blue.
The first momentum calculates the velocity of the learning rate, which helps to address local minima issues. The second momentum calculates the parameter-wise normalization of the learning rate, which helps for saddle points. These moments are helpful since it prevents the updates to either get stuck or zigzag, as illustrated in blue vectors.
They change the update direction by adding the moment vector accumulated from first update to the steepest descent direction.