Assignment #2

Sima Shafaei

March 19

1. (25 Points) How many dimensions must the inputs of an RNN layer have? What does each dimension represent? What about its outputs?

- *For each recurrent cell the input is $X_{(t)}$ which is a $m \times n_{inputs}$ matrix where m is the number of instances in mini-batch and n_{input} is the number of input features.*

- *The output is $Y_{(t)}$ which is a $m \times n_{neurons}$ and contains the layers' outputs at time step t for each instance in mini-batch. M is the number of instances in mini-batch and $n_{neurons}$ is the number of neurons*

2. (25 Points) Consider a CNN composed of three convolutional layers, each with 3 × 3 kernels, a stride of 2, and "same" padding. The lowest layer outputs 100 feature maps, the middle one outputs 200, and the top one outputs 400. The input images are RGB images of 200 × 300 pixels. What is the total number of parameters in the CNN? If we are using 32-bit floats, at least how much RAM will this network require when making a prediction for a single instance? What about when training on a mini-batch of 50 images?

Number of parameters for convolutional layer:

$(k \times k \times number\ of\ channel\ \times n_{in} + 1) \times n_{out}$

$where:$

$k = size\ of\ kernel$

$n_{in} = number\ of\ input\ feature\ maps$

$n_{out} = number\ of\ output\ feature\ maps$

We add 1 for term bias in each feature map.

Layer 1: $(3 \times 3 \times 3 + 1) \times 100 = 2800$

*(memory=4*2800=0.112Mg)*

Layer 2: $(3 \times 3 \times 3 \times 100 + 1) \times 200 = 540200$

$(memory = 4 * 540200\ = 2.1608Mg)$

Layer 3: $(3 \times 3 \times 3 \times 200 + 1) \times 400 = 2160400$

$(memory = 4 * 2160400 = 8.641Mg)$

Total number of parameters: $720400 + 180200 + 1000 = \mathbf{901600}$

For mini-batch =50:

Layer 1: $50 \times (3 \times 3 \times 3 + 1) \times 100 = 140000$

Layer 2: $50 \times (3 \times 3 \times 3 \times 100 + 1) \times 200 = 27010000$

Layer 3: $50 \times (3 \times 3 \times 3 \times 200 + 1) \times 400 = 108020000$

$$n_{out} = \left\lfloor \frac{n_{in} + 2p - k}{s} \right\rfloor + 1$$

n_{in}: number of input features
n_{out}: number of output features
k: convolution kernel size
p: convolution padding size
s: convolution stride size

For one image:

*Required memory for input image: 3*200*300*4=720K =0.72 Mg*

First Layer:

*k=3*3* *p=same* *s=2*2* n_{in} *= 3* 200 * 300 =*	n_{out} *= 3* 200/2 * 300/2*
Output feature map= 100 *Number of bytes: 4*	*Memory=3*100*150*100*4=18000000 = 18 Mg*

second Layer:

*k=3*3* *p=same* *s=2*2* n_{in}= $3 * 100 * 150$	n_{out} *=3*100/2*150/2*
Output feature map= 200 *Number of bytes: 4*	*Memory=3*50*75*200*4=9000000 = 9 Mg*

third Layer:

*k=3*3* *p=same* *s=2*2* $n_{in}= 3 * 50 * 75$	n_{out} *=3*50/2*75/2*
Output feature map= 400 *Number of bytes: 4*	*Memory=3*25*38*400*4=4560000 = 4.56Mg*

We can consider two different case:

1) *After computing one layer the memory will release, so we can consider largest memory as required ram. In this case required RAM memory would be: Required memory for input + Required memory for parameters + Required memory for the output= 0.72 +0.112+18 =18.83 Mg*

2) *All the layers' computations will remain in the ram memory. In this case required ram memory would be:*

Required memory= size of image +

memory for parameter in first layer+ memory for result of first layer+

memory for parameter in second layer+ memory for result of second layer+

memory for parameter in third layer+ memory for result of third layer

0.72+0.112+18+2.16+9+8.64+4.56=43.192 Mg

For Mini-Bach:

To compute the total required memory for mini batch we need to compute memory required for a single instance and multiply it by 50 therefore:

*In first case that after computing one layer the memory will release required memory would be: 18.83 * 50 = 941 Mg*

And in second case that we suppose all computation will remain in the RAM, required memory would be:

*43.192*50*$\cong 2160\ Mg\ \cong 2.2\ Gig$

3. (50 Points) Use transfer learning for large image classification, going through these steps:
    a. Create a training set containing at least 100 images per class. For example, you could classify your own pictures based on the location (beach, mountain, city, etc.), or alternatively you can use an existing dataset (e.g., https://www.kaggle.com/puneet6060/intel-image-classification).
    b. Split it into a training set, a validation set, and a test set.
    c. Build the input pipeline, including the appropriate preprocessing operations, and optionally add data augmentation.
    d. Fine-tune a pretrained model on this dataset.

Sima_Shafaei.ipynb