Special Topics: Deep Learning

Sheet 1 — COMP 691

Assignment 2

Submission: Submit a pdf of your written answers and code and output from programming portions. You can also submit a runnable ipynb (single file) including all answers to programming portions of the questions.

Note: The assignments are to be done individually. You may discuss ideas regarding questions with others but should not share answers. List the names of anyone you have extensively discussed a question with at the top of your final submission.

Eugene Belilovsky: eugene.belilovsky@concordia.ca

1. (a) **(5 points)** Consider the AlexNet Model designed for an input image size of 227x227x3. Find the receptive field (with respect to the input) at the center position of layer 1. At layer 1 the height and width is 55, the center spatial location is at 27 (round down). Your answer should provide 4 numbers for each layer (1 & 3) to describe the rectangle that gives the receptive field Height 1, Height 2, Width 1, Width 2. Here Height1, Width1 are the lower left corner of the input and Height2,Width2 are the upper right corner. Assume indexing starts at 1 - the bottom left corner of the image is (1,1). You must show your work for this question. **(extra credit 4 points)** What is the receptive field at the center position of Layer 2 and Layer 3

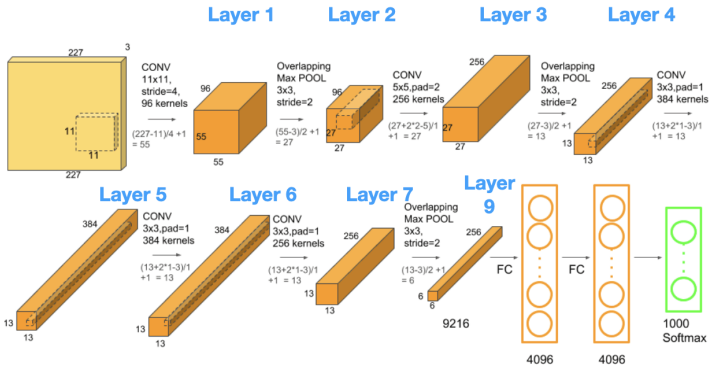

Figure 1: Labeled Alexnet

(b) **(15 points)** Download and load the pretrained alexnet network from the pytorch model repository. Write a helper function that will take a single input image and output the top prediction from alexnet using the actual class label names (e.g. "triceratops", not the label 51). This can be for example added as a label or text above the image. You can find the class labels here:
`https://gist.github.com/yrevar/942d3a0ac09ec9e5eb3a`
and some sample images here
`https://github.com/ajschumacher/imagen/tree/master/imagen`
Note you will need to resize the images to 224x224x3 to feed them into the model. Show your output for 3-5 images of your choosing.

(c) **(20 points)** We will create an adversarial example for each of 2 randomly selected images. Take 2 randomly selected images from different classes (from the link above). Now for each image select 3 classes that are not the true class and create 3 adversarial examples. Do this by optimizing the following objective: $\min_r \alpha\|r\|_1 + l(f_\theta(x+r), y_p)$, where y_p is the alternative class, f is the alexnet model, l is cross entropy, and x is the original image. Write a function that per-

forms this optimization. It is suggested that you use ADAM optimizer and a learning rate of 0.01. To stop the optimization you can use a criteria that checks if the sample has switched to a new class. You will need to tune α a bit and possibly other hyperparameters to get more reasonable looking adversarial images for the samples you select. In general the scale of α inversely correlates to the scale of the original r (r at the beginning of the optimization), the smaller are the values in r, the bigger α you would need. For example, for $r \in [0, 0.01]$, a suitable α could be in $[10, 100]$, and for $r \in [0, 0.1]$, a suitable α could be in $[0.1, 10]$. Your final image should be hard to distinguish from the original. You may adjust this optimization settings if you find ones which converge faster on the images you consider. Save your $2 \times 3 = 6$ adversarial examples and show that they indeed are misclassified by alexnet using your visualization tool from (b) to compare original and adversarial example and labels found by alexnet.

2. (a) (**25 points**) In this question we will train some basic RNN language model. The rest of this question can be found in `assignment2_starter_code.ipynb` which comes with the assignment.

Eugene Belilovsky: eugene.belilovsky@concordia.ca

3. (a) Consider the self-attention operation S($\boldsymbol{X}$) defined as follows

$$\boldsymbol{A} = \frac{1}{\alpha}\boldsymbol{X}\boldsymbol{W}_q\boldsymbol{W}_k^T\boldsymbol{X}^T$$
$$S(\boldsymbol{X})_{t,:} = softmax(\boldsymbol{A}_{t,:})\boldsymbol{X}\boldsymbol{W}_v$$

Where $\boldsymbol{X} \in R^{T\times D_{in}}$, $\boldsymbol{W}_v \in R^{D_{in}\times D_k}$, $\boldsymbol{W}_k \in R^{D_{in}\times D_k}$, $\boldsymbol{W}_q \in R^{D_{in}\times D_k}$. D_{in} being the input dimensionality and T a sequence (or set) length

(b) (**15 points**) Show that for any permutation matrix $\boldsymbol{P}$, $\boldsymbol{P}S(\boldsymbol{X}) = S(\boldsymbol{P}\boldsymbol{X})$

(c) (**5 points**) We would like to use $S(\boldsymbol{X})$ and a linear operation to construct a permutation invariant function $G(\boldsymbol{X})$ that outputs the same feature representation for any ordering of the input sequence (hence, it is invariant to the input's arrangement). What linear operation can we use? Specifically find an example of a vector $\boldsymbol{w}$ such that $G(\boldsymbol{X}) = \boldsymbol{w}^T S(\boldsymbol{X})$ satisfies the condition $G(\boldsymbol{X}) = G(\boldsymbol{P}\boldsymbol{X})$ for all permutation matrix P.

(d) (**15 points**) Consider a batched $\boldsymbol{Z} \in \mathcal{R}^{B\times T\times D}$ that is for example the output of a self-attention layer. Implement a `nn.module` or python function using functionalize that takes $\boldsymbol{Z}$ and applies a "Position wise feedforward network" layer, $H(\boldsymbol{x}) = relu(\boldsymbol{W_1}\boldsymbol{x} + \boldsymbol{b})$ which outputs a tensor of size $\mathcal{R}^{B\times T\times D}$ as commonly used in transformer models. Implement this in two ways (1) using `nn.Linear` and (2) using `nn.Conv1d` with padding 0, stride 1 and kernel size 1. Validate your implementations match with an assert statement. You may select a random Z with the size B, T, D of your choosing for validating your implementation.

Eugene Belilovsky: eugene.belilovsky@concordia.ca