

RECITATION 7: DEEP LEARNING

10-301/10-601 Introduction to Machine Learning (Summer 2024)

<http://www.cs.cmu.edu/~hchai2/courses/10601>

1 PyTorch Basics

1.1 Colab Notebook

See the following Colab Notebook:

<https://colab.research.google.com/drive/1gprL1pZ90GDftoNp5qHtu4AAc6boLcwY?usp=sharing>

2 Attention Parameters

Here is the illustration of a self-attention block from lecture:

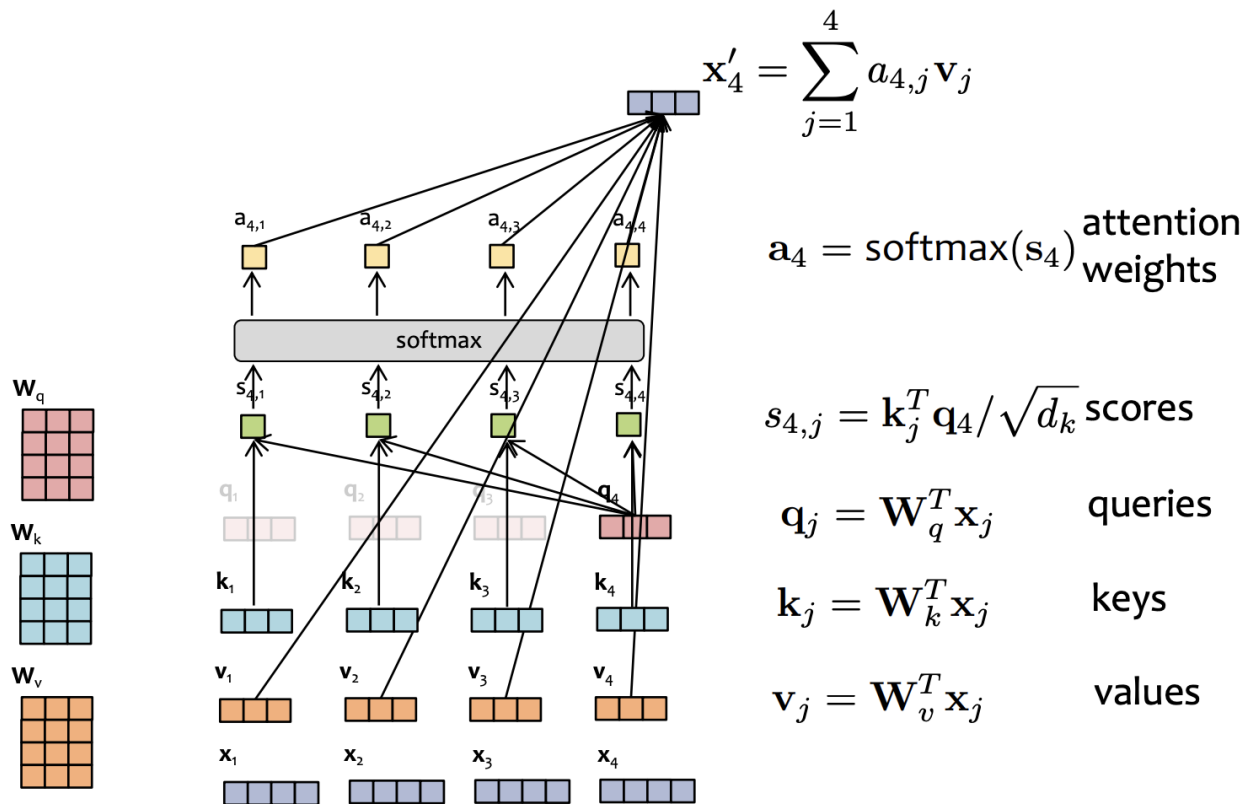


Figure 1: Self-attention block, illustrated

Let's define some terms:

- d_{model} is the dimension of each input vector \mathbf{x}_i
- d_k is the dimension of the query, key and value vectors.
- h is the number of attention heads

1. What is the shape of the W_q, W_k, W_v matrices (without biases)?
2. How many parameters are there in each of the W_q, W_k, W_v matrices (including biases)?
3. How many parameters are there in total for a multiheaded attention block, as illustrated in lecture?
4. Where else in the transformer are there additional parameters?
5. What is the big-O relationship between the number of parameters for a multiheaded attention block and the length of each input vector?

3 Convolutional Neural Networks

3.1 Concepts

1. What are filters?

- Filters (also called kernels) are feature extractors in the form of a small matrix used in convolutional neural layers. They usually have a width, height, depth, stride, padding, channels (output) associated with them.

2. What are convolutions?

- We sweep the filter around the input tensor and take element-wise product sums based on factors such as filter size, stride, padding. These output product-sums form a new tensor, which is the output of a convolutional layer.

3. How do we calculate the output shape of a convolution?

- Given input width W_{in} , kernel width K_w , padding P , and stride S , the output width W_{out} can be calculated as:

$$W_{out} = \lfloor \frac{W_{in} - K_w + 2 \times P}{S} \rfloor + 1$$

- Output height can be calculated similarly.

4. What are some benefits of CNNs over fully connected (also called dense) layers?

- Good for image-related machine learning (learns the kernels that do feature engineering)
- Pseudo translational invariance
- Parameter efficient

5. How does the number of channels vary through convolutional networks?

- Each convolution filter will have as many channels as the input, and there will be as many filters as there are output channels.
- Pooling and activations often maintain the number of channels.

3.2 Dance Dance Convolution

Consider the following 4 x 4 image and 2x2 filter below.

| | | | |
|---|----|----|---|
| 1 | 3 | -2 | 4 |
| 0 | 8 | 6 | 5 |
| 2 | 1 | -9 | 0 |
| 4 | -1 | 3 | 7 |

| | |
|----|----|
| 1 | 2 |
| -2 | -1 |

1. Assume that there is no padding and stride = 1. What are the dimensions of the output, and what is the value in the bottom right corner of the output image?
2. Now assume that we having padding = 1. Given that, what are the new dimensions of the output, and the new value in the bottom right corner?

3.3 Parameters

Suppose that we want to classify images that belong to one of ten possible classes (i.e. [cat, dog, bird, turtle, ..., horse]). The images come in RGB format (one channel for each color), and are downsampled to dimension 128x128.

Figure 2 illustrates one such image from the MS-COCO dataset¹.



Figure 2: Image of a horse from the MS-COCO dataset, downsampled to 128x128

We construct a Convolutional Neural Network that has the following structure: the input is first max-pooled with a 2x2 filter with stride 2 and 3 output channels. The results are then sent to a convolutional layer that uses a 17x17 filter of stride 1 and 12 output channels. Those values are then passed through a max-pool with a 3x3 filter with stride 3 and also 12 output channels. The result is then flattened and passed through a fully connected layer (ReLU activation) with 128

¹<https://cocodataset.org/>

hidden units followed by a fully connected layer (softmax activation) with 10 hidden units. We say that the final 10 hidden units thus represent the categorical probability for each of the ten classes. With enough labeled data, we can simply use some optimizer like SGD to train this model through backpropagation.

Note: By default, please assume we have bias terms in all neural network layers unless explicitly stated otherwise.

1. Fill the table below with channels and dimensions of the tensors before and after every neural net operation.

| Layer / Operation | Shape |
|-------------------|--------------------|
| Input | $3@128 \times 128$ |
| maxpool-1 | (a) |
| conv | (b) |
| maxpool-2 | (c) |
| flatten | (d) |
| fully-connected-1 | (e) |
| ReLU | (f) |
| fully-connected-2 | (g) |
| softmax | (h) |

2. Draw a diagram that illustrates the above table.

3. How many parameters are in this network for the convolutional components?

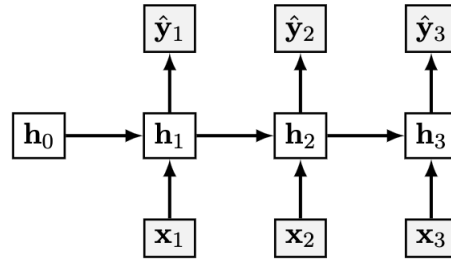
4. How many parameters are in this network for the fully connected (also called dense) components?

5. From these parameter calculations, what can you say about convolutional layers and fully connected layers in terms of parameter efficiency²? Why do you think this is the case?

²the ratio between the number of parameters from some layer type and the total number of parameters.

4 Recurrent Neural Networks

4.1 Sample RNN



Where the layers and their corresponding weights are given below:

$$\begin{array}{ll} \mathbf{x}_t \in \mathbb{R}^3 & \mathbf{W}_{hx} \in \mathbb{R}^{4 \times 3} \\ \mathbf{h}_t \in \mathbb{R}^4 & \mathbf{W}_{yh} \in \mathbb{R}^{2 \times 4} \\ \mathbf{y}_t, \hat{\mathbf{y}}_t \in \mathbb{R}^2 & \mathbf{W}_{hh} \in \mathbb{R}^{4 \times 4} \end{array}$$

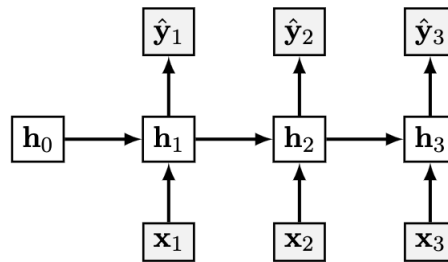
$$\begin{aligned} \hat{\mathbf{y}}_t &= \sigma(\mathbf{o}_t) \\ \mathbf{o}_t &= \mathbf{W}_{yh} \mathbf{h}_t \\ \mathbf{h}_t &= \psi(\mathbf{z}_t) \\ \mathbf{z}_t &= \mathbf{W}_{hh} \mathbf{h}_{t-1} + \mathbf{W}_{hx} \mathbf{x}_t \end{aligned}$$

Where σ and ψ are activations.

1. Redraw the above diagram in a compact form such that we don't need to unroll it across several timesteps.

4.2 Concepts

1. What are recurrent neural networks?
 - A recurrent neural network (RNN) can be characterized by connections between nodes creating a cycle³. Outputs from some nodes can affect subsequent computations. This allows it to exhibit temporal dynamic behavior.
 - the recurrent nature makes them useful when the input is sequential (or temporal).
2. How do they use both inputs and previous outputs?
 - Hidden nodes have two sets of weights, one to process input from the current timestep, and one to process their own outputs from the previous timestep.
3. How do we optimize RNNs?
 - Applying chain rule to the 'unrolled' RNN (as above) is no different than a regular feed forward neural network aside from the fact that the same parameters are repeated throughout the network at each timestep.
 - Called as backpropagation through time (BPTT).



³[Article linked here.](#)