Recitation 7
Deep Learning

10-601: Introduction to Machine Learning

11/10/2023

1 Convolutional Neural Networks

1.1 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.

1.2 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? output is 3x3, and the bottom right value is $-9 + 0 - 6 - 7 = -22$.
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? output is now 5x5, and bottom right value is $7 + 0 + 0 + 0 = 7$.

1.3 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 1 illustrates one such image from the MS-COCO dataset[1].

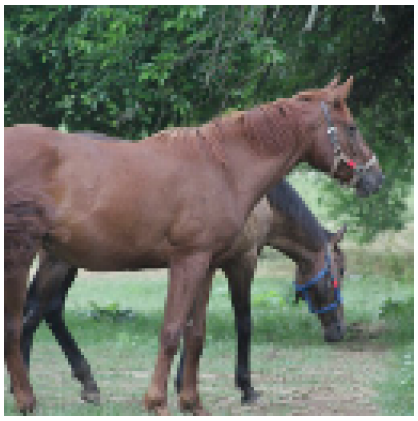


Figure 1: Image of a horse from the MS-COCO dataset, downsampled to 128x128

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

[1] https://cocodataset.org/

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 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 backprogation.

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 × 128
maxpool-1	(a)
conv	(b)
maxpool-2	(c)
flatten	(d)
fully-connected-1	(e)
ReLU	(f)
fully-connected-2	(g)
softmax	(h)

Layer / Operation	**Shape**
Input	3@128 × 128
maxpool-1	3@64 × 64
conv	12@48 × 48
maxpool-2	12@16 × 16
flatten	3072
fully-connected-1	128
ReLU	128
fully-connected-2	10
softmax	10

2. Draw a diagram that illustrates the above table.

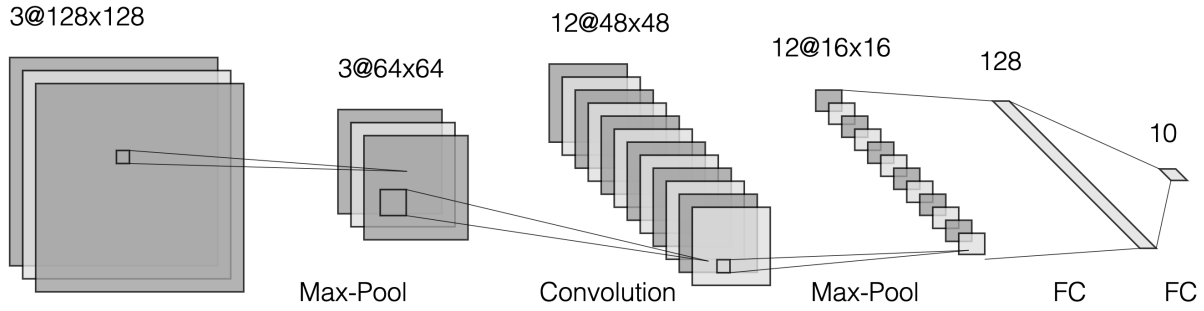


Figure 2: Full CNN structure, illustrated

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

$$\begin{aligned} N_{\text{conv}} &= (3 \times 12 \times 17 \times 17 + 12) \\ &= 10416 \end{aligned}$$

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

$$\begin{aligned} N_{\text{fc}} &= (3072 \times 128 + 128) + (128 * 10 + 10) \\ &= 393344 + 1290 \\ &= 394634 \end{aligned}$$

5. From these parameter calculations, what can you say about convolutional layers and fully connected layers in terms of parameter efficiency[2]? Why do you think this is the case?

$$\begin{aligned} N_{\text{total}} &= 10416 + 394634 \\ &= 405050 \end{aligned}$$

$$\begin{aligned} N_{\text{conv}} N_{\text{total}} &= 2.57\% \\ N_{\text{fc}} N_{\text{total}} &= 97.43\% \end{aligned}$$

Convolutional layers are much more parameter efficient, mainly because we are reusing the convolutional filter repeatedly for each convolutional layer (we only need to train one kernel per channel per layer). In comparison, the fully connected layer requires all nodes between two layers to be fully connected.

1.4 Links

Visualization of convolutional filter sweep steps https://github.com/vdumoulin/conv_arithmetic

Visualization of convolutional filter smooth sweep with outputs https://www.youtube.com/watch?v=f0t-OCG79-U

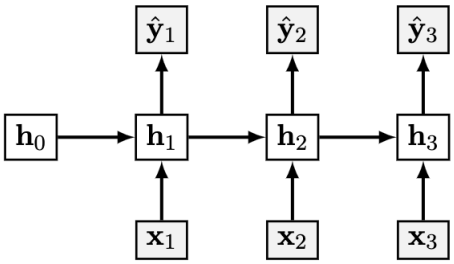
Visualization of neural network layer outputs http://cs231n.stanford.edu/

The architecture used there is (conv→ relu → conv → relu → pool) x3 → fc → softmax

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

2 Recurrent Neural Networks

2.1 Sample RNN



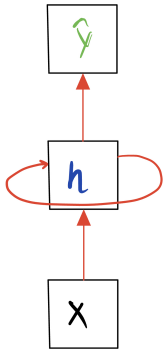
Where the layers and their corresponding weights are given below:

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

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

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.



2.2 Concepts

1. What are recurrent neural networks?
    - A recurrent neural network (RNN) can be characterized by connections between nodes creating a cycle [3]. 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.