CS231n: CNNs for Visual Recognition **Spring 2017**

Lecture 5

Course Coordinator: Prof. Fei-Fei Li *Scribes: Akash Gupta*

Convolutional NNs: They explicitly try to maintain spatial structure of an image.

A bit of history....

- Mark I perceptron (developed by Frank Rosenblatt in 1957) - first implementation of the perceptron algorithm. Same idea as getting the score functions but the outputs are going to be 1 or 0. Also, has an update rule for weights but no principled backpropagation technique.
- Adaline/Madaline (developed by Widrow and Hoff in 1960) - stacked up linear perceptron layers to form multi-layer perceptron. Starting to look kinda like NN. But no backpropagation technique.
- First time backpropagation became popular (developed by Rumelhart in 1986) - Starting to see chain rule equations and have a principled way to train these neural network architectures.
- Reinvigorating research in deep learning (developed by Hinton and Salakhutdinov in 2006) - Since there were not a lot of new things happening before 2000, in 2006 this paper showed that we could train a DNN but not a modern way of doing training.
- Papers on Acoustic Modeling, Speech Recognition, ImageNet Classification (developed by Hinton in 2010/12) - The usage of these kind of NNs became popular and we had strongest results till then for speech recognition and in 2012 for ImageNet Classification where they introduced the first CNN architecture.

Rise of ConvNNs

- Understanding how neurons in the visual cortex work (developed by Hubel & Wiesel in 1950) - Cat experiment and response to stimulus. Topographical mapping in the cortex of the visual representation. Hierarchical organization of neurons.
- Neurocognitron (developed by Fukushima 1980) - Introduced the first architecture of simple (modifiable params) and complex cells (pooling layers).
- Gradient based learning in document recognition (developed by Yann LeCun in 1998) - Gave the first example of applying backpropagation and gradient based learning for training CNN for zip code recognition. Not able to scale to complex problems.
- ImageNet Classification with DeepCNN (developed by Krizhevsky and Hinton in 2012) - AlexNet for ImageNet Classification and take advantage of parallel comoputing in GPUs.

Uses: Image classification, Image similarity, Object detection, Segmentation, Face recognition, Classifying Videos, Pose recognition, Atari games, Image captioning, Neural Styling,.....

CNNs (without the brain stuff):

Fully connected layer - 32x32x3 image $->$ stretch to 3072x1 $->$ Wx $->$ activation

Convolution layer - 32x32x3 image $->$ preserve spatial structure $->$ weights are going to be spatial filters of smaller dimensions but whole depth will be covered

Performing convolution - the result of taking a dot product between the filter and a small chunk 5x5x3 of the image $->$ receive a single number.

Q) *When we do the dot product, do we turn the 5x5x3 vector in 75x1?*
A) Yeah in essence that is what is happening.

Q) *Should we rotate the kernel by* $180°$ *to better match the definition of convolution?*
A) So, we don't worry about this since we are already convolving with the flipped version of the convolution layer.

Sliding operation: Convolve (slide) over all spatial locations and generate a smaller siza activation map. Choice of sliding (1-pixel, 2-pixel, etc...)

Multiple filters: Take multiple filters to ensure each template is recorded. This will create multiple activation maps. For e.g. - 32X32X3 with 6 5x5 filters generates 6 activation maps or a new image of 28x28x6.

Preview (how to use this):

- ConvNet is a sequence of Convolutional layers, interspersed with activation functions. Image $->$ multiple sequences of (CONV $->$ RELU $->$ POOLING)$->$.....Fully-connected layer (FC)
- Low-level features $->$ Mid-level features $->$ High-level features $->$ Linearly separable classifier.

Q) *What are the visualizations in each element of the grid in a CONV layer?*
A) It basically signifies what in the input would look like when it maximizes the activation of the neuron. So in a sense what is the neuron looking for.

Example: On visualizing one filter, a template is seen as an edge and convolving this filter produces an activation map with white lines on the edges (indicating high values).

A closer look:

- Stride is the amount by which the filter is moved.
- Output size: $(N - F)/stride + 1$ where N is the dimension of the image and F is dimension of the filter.
- Filters that are not able to cover the image (or fit) don't really work out.
- In practice, it is common to zero pad the images in order to make the size workout. So the output size is now - $(N + 2 * P - F)/stride + 1$ where P is no. of layers padded. Or in general case output - Output_dim x Output_dim x no. of filters.
- For maintaining the size, zero pad with $(F - 1)/2$ layers.
- Zero padding is important because it prevents the activation maps from reducing in size which could lead to loss of information.
- Common values:
  - K, Number of filters in powers of 2 - 32, 64, 128, 512,...
  - F = 3,5,7
  - S = 1,2

  - P = 1,2

Q) *If the image is rectangular, do we change the value of stride that is different horizontally and vertically?*
A) In practice, we prefer to operate over square images and don't prefer this type of convolution.

Q) *Intuition behind choosing stride?*
A) Helps in downsampling the images so has kinda like pooling effect. This also helps in reducing the number of params to deal with for fully connected layer

Pooling Layer:

- Makes the representations smaller and more manageable.
- operates over each activation map independently
- MAX POOLING: Operation is performed exactly like Convolution layer but instead of the dot product we take the maximum value of the localised region of the image.
- For pooling no need to use zero padding sine we are already downsampling.

Q) *If stride and pooling both produce the same effect of downsampling, can we just use stride?*
A) Yes. Also, in recent architectures people have started using stride instead of max pooling. (*Personal intuition*: Will help in reducing the no. of params if we just use stride)

Fully Connected layer (FC): Receives output of Convolution and outputs scores based on aggregating the input.