1 Convolutional Nerual Networks

1.1 Different layers in CNN

1.1.1 Convolution Layer

Convolution layer: uses filters as parameters, convolve filters with the image by multiplying its values element-wise with the original matrix.

In forward pass, we take many filters and convolve them on the input, each convolution gives a 2D matrix output, then stack them into a 3D volume. Usually every convolution layer is followed by a relu layer.

In backward pass, we calculate the derivations of each variables based on the following parameters.

This is the formula for computing dA.

$$dA \mathrel{+}= \sum_{h=0}^{nH} \sum_{w=0}^{nW} W_c * dZ_{hw}$$

Where W_c is a filter and dZ_{hw} is a scalar corresponding to the gradient of the cost with respect to the output of the conv layer Z at the hth row and wth column (corresponding to the dot product taken at the ith stride left and jth stride down).

This is the formula for computing dW_c with respect to the loss:

$$dW_c \mathrel{+}= \sum_{h=0}^{nH} \sum_{w=0}^{nW} a_{slice} * dZ_{hw}$$

This is the formula for computing db with respect to the cost for a certain filter.

$$db = \sum_h \sum_w dZ_{hw}$$

1.1.2 Pooling Layer

The pooling layer reduces the height and width of input volume/tensor. It helps reduce the computation, as well as helps make feature detectors more irvariant to its position in the input. There are two types of pooling layer: max-pooling and average-pooling.

The pooling layer's implementation is some what similiar to convolution layer, it uses a filter slicing over the input volume. Usually a pooling layer doestn't contain zero-padding and it doesn't contain parameters. It will need two hyper-parameters: filter size and stride. If filter size is 2 and stride is 2, then after pooling the height and width of volume will be half of before.

1.2 Multi-class learning

For multi-class classification, we use Softmax function to output classification.

$$\sigma(\boldsymbol{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}$$

1.2.1 Derivation

If we use cross-entropy loss function, such as following:

$$L(\boldsymbol{W}) = -\frac{1}{N}\sum_{n=1}^{N}[y_n \log \hat{y} + (1 - y_n)\log(1 - \hat{y})]$$

Then the derivation should be:

$$\frac{\partial L}{\partial \boldsymbol{z}} = \hat{\boldsymbol{y}} - \boldsymbol{y}$$

1.2.2 Notes

When used in back propagation in practical programs, espicially in mini-batch, don't forget to divide the derivation by the batch size.

$$\frac{\partial L}{\partial \boldsymbol{z}} = (\hat{\boldsymbol{y}} - \boldsymbol{y})/batch - size$$

1.3 Computation Improvment

As it is mentioned before, if we compute convolution and pooling explicitly by single steps, it will be very time-consuming as it will need four nested four-loops to compute a 4D volume/tensor, and this type of computing contains a lot duplicates computation. In pracitical code, using explicit for-loop is usually avoided as it will take a lot time for computation.

Usually in most deep learning frameworks, unrolling convolution layers are used to speeding up the computation process of convolution and pooling layers. The central idea is an unfolding and duplication of the input and rearrangement of the kernel parameters that produces a CPU/GPU friendly ordering.

Using this approach each convolutional layer is converted to a matrix product during forward propagation. A nice byproduct of this is that we can simply write down the back-propagation step as another matrix product.

Simple unfolding of convolution is a well known technique. It is commonly implemented in signal processing and communications applications. For example, Matlab has two functions, convmtx and convmtx2 (signal processing toolbox) which create convolution matrices in order to transform convolution into a matrix multiplication.

2 CNN Case studies

2.1 Classic Networks

2.1.1 LeNet-5

Two Convolution layers followed by avg-pool layers, then two fully-connected layer, use softmax as the output layer. As the network goes deep, the height and width of the volume go down, and the number of filters (or the depth of the volumn) go up.

2.1.2 AlexNet

Same padding is used, $pad = (f-1)/2$. AlexNet is similar to LeNet, but much bigger. And uses Relu as activate function.

2.1.3 VGG-16

All convolution layers are 3*3 filters, stride = 1, same pooling.
All max-pooling layers are 2*2 filters, stride = 2.
VGG simplied neural network architectures. The architecture of VGG-16 is uniform. Use max-pooling with 2*2 filters, stride = 2 makes the height and width go half every time.

2.2 ResNet

2.2.1 Introduction

Use resudual blocks allow to build deep networks. Number of training error goes down with ResNet layer numbers goes up.

2.2.2 Intuition

$$a^{[l+2]} = g(z^{[l+2]} + a^{[l]})$$
$$a^{[l+2]} = g(w^{[l+2]} * a^{[l+1]} + b^{[l+2]} + a^{[l]})$$

Use same padding convolve to ensure the demisons of the matrix. Or using pooling layers to adjust the demisons.

2.3 1*1 convolution

Filters are 3D volume, and size is $1 * 1 * N_{c_{p}rev}$.Works as fully connected network with Relu. Network in Network. Useful to shrink the number of channels.

Inception layer uses 1*1 convolutions, other convolutions and pooling layers, and stack up all the outputs. Using 1*1 convolution reduces the computational cost without harming the performance.

2.4 Inception Network

Inception module concatenates convolution and maxpooling layers output.
Inception network puts inception modules together.
Inception module makes network go deeper.

2.5 Pratical Advices

- Use Open-Sourece Implementation
- Transfer Learning
- Data Augmentation

2.5.1 Data Augmentation

- Mirroring
- Random Cropping
- Rotation
- (PCA) Color shifting

2.5.2 Deep Learning for Computer Vision

Two sources of knowledge: Labeled data and hand engineering/network architecture.

Tips for doing well on benchmarks or competetions:

- Train several networks independently and average their outputs.
- Multi-crop at test time and average results.
- Use architectures of networks published in the literature.
- Use open sourece implementations if possible.
- Use pretrained models and fine-tune on your dataset.

3 Detection Algorithm

3.1 Object Location

For an image, the coordinate is set like this: the left up corner is set to $(0, 0)$, and the right bottom corner is set to $(1, 1)$. The center point of an bounding box is then (b_x, b_y), in the coordinate of image. The bounding box height and width is (b_h, b_w).

3.2 Convolutional Implementation of Sliding Windows

Running sliding windows explicitly slide over the image contains a lot dublicate computation. Instead, treating the whole computation process of a window sliding as a convolutional layer, and compute the image convolutional, each part of output volume corresponds to a sliding window output.

3.3 YOLO Algorithm

YOLO Algorithm sets grid on the image, and for each grid cell, it will feed the data into convolutional network, output the predict labels, and then use back propagation to train the convolutional network.

The center of a bounding box (b_x, b_y) is relative position in a grid cell, so b_x, b_y are between 0 and 1. So the output function of b_x, b_y can be sigmoid function The heigth and width of a bounding box (b_h, b_w) are relative length in a grid cell, but b_h, b_w can be larger than 1. The output function of b_h, b_w can be exponential function.

3.4 Intersection Over Union

Intersection Over Union is a way to evaluating object localization. It calculate the intersection part of output and ground truth over the union part of output and ground truth.
Usually the output is considerred as "correct", if $IoU \geq 0.5$.
More generally, IoU is a measure of the overlap between two bounding boxes.

3.5 Non-max Suppression

Non-max Suppression is a way to make sure the algorithm detects the object only once.

Basically, Non-max Suppression will discard all bounding boxes with $P_c \leq 0.6$. Then it will do the following steps iteratively:

1. Pick the bounding with highest P_c, output that as prediction.
2. Suppress those bounding boxes that $IoU \geq 0.5$ with previous bounding box.

3.6 Anchor Boxes

Anchor boxes are pre-difined to avoid object overlapping.
Each object in the training image is assigned to grid cell that contains objects midpoint and anchor box for the grid cell with the highest IoU.