Report on homework tasks "Deep Learning"

Ievgeniia Dobryden

https://github.com/zheden/MLCVhomeworks/tree/master/Task2_DCNN

THEORETICAL PART

Backpropagation with a single modified neuron: Suppose we modify neuron p on layer k so that the output from it is given by $f(w^T x + b)$. Modified backward propagation algorithm is following:

1. Input x: Set the corresponding activation a^1 for the input layer
2. Feedforward: For each $l = 2, 3, , L$ compute weighted input to neurons in layer l

$$z^l = w^l a^{l-1} + b^l, \quad a = \begin{cases} \sigma(z^l) & , \quad l \neq k \\ \begin{pmatrix} \sigma(z^l_{1..p}) \\ f(z^l_p) \\ \sigma(z^l_{p..n}) \end{pmatrix}, & \quad l = k \end{cases}$$

3. Output error δ^L: Compute the vector $\delta^L = \nabla aC \odot \sigma'(z^L)$
4. Backpropagate the error: For each $l = L-1, L-2, ..., 2$ compute

$$\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot G(z^l), \quad G(z^l) = \begin{cases} \sigma'(z^l) & , \quad l \neq k \\ \begin{pmatrix} \sigma'(z^l_{1..p}) \\ f'(z^l_p) \\ \sigma'(z^l_{p..n}) \end{pmatrix}, & \quad l = k \end{cases}$$

5. Output: The gradient of the cost function is given by $\frac{\partial C}{\partial w^l_{jk}} = a^{l-1}\delta^l_j$, $\frac{\partial C}{\partial b^l_j} = \delta^l_j$

PRACTICAL PART

The objective in practical task was to classify images with help of pretrained DCNN model, given in one of examples of Caffe framework. This network model was trained for ImageNet dataset which consists of 1000 classes. For classification following changes should be done for images:

1. Test image should be scaled to input data size that was defined before training size. In pretrained ImageNet model this size is 227x227 pixels. It is important to make couple samples for each testing image, since object that we want classify not always located in center of image averaging output for oversampled input will increase accuracy of classification (see experiments).
2. Mean value should be subtracted from test image. The reason for this is following: Suppose there is image I in our training set. After training we get network with neurons that are responsible for convolutions. Lets take neuron k whose response is calculated with formula

$f(W_k * I + b_k)$ where W_k is convolutional kernel, b_k is bias, f if response function such as sigmoid. Lets consider test image $J = I + b11^T$, where b is scalar value, $1^T = (1, ..1)^T$. Response for same neuron k will be now different, since $W * J = b(W * I)$. This means that even for same image with added intensity neuron k may not fire, that influence whole classification results. Therefore, mean intensity should be subtracted for images both for training and testing.

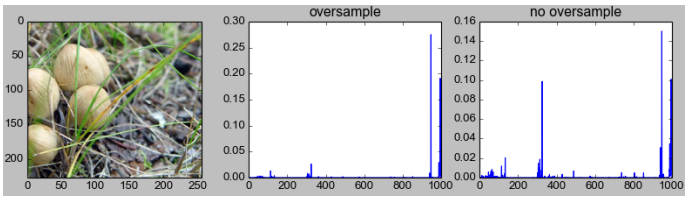
Above-mentioned main two image prepossessing is done by default in Caffe's "predict()" method.

Expertiments

Oversampling. For object classification image of size 227x227 may be input to network. There is also option to crate multiple samples: take 256x256 image and cut random patches inside of size 227x227, test them and take average probability distribution as final result. Oversampling produces better results:

1. In case object is not in our database (Chipmunk image 1) after oversampling we can see that entropy of our probability distribution increases dramatically from 2.4 to 4.5

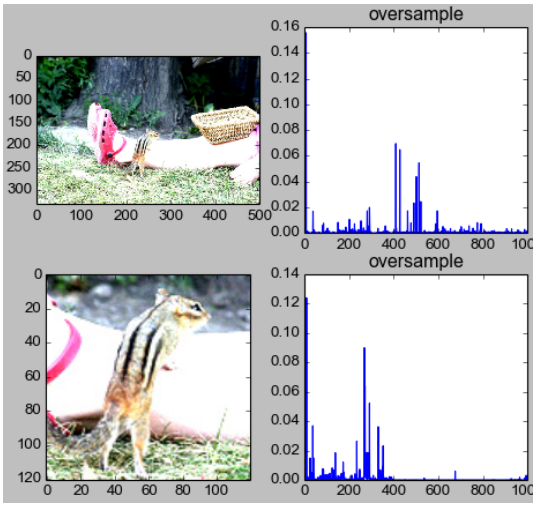
2. In case object is in our database (Mushroom image 6), but it is not in center of image, oversampling produces better result: probability 0.27/entropy 2.65 with oversampling versus 0.15/3.6, without oversampling:



Cropping image.

1. For training of ImageNet model all images were scaled to 256x256 size. Therefore it is important to ensure that testing images are about square size. Classification of image of size 200x800, for example, will not produce nice result, even if on training set were some rectangular images. I think test image proportions should not much deviate from average image proportions in training set. Therefore, I cropped rectangular images in our testing set to square ones first.

2. For better classification for image where the object was very small (chipmunk image 1) I cropped it also so that object became larger. Since we dont have chipmunk in ImageNet set of classed, classification was incorrect, but at least such classes as wolf and coyote became more probable:



Results

Image	Exits in DB	Predicted class	Probability	Entropy
1 chipmunk	no	brambling	0.123	4.216
2 beans	no	bell pepper	0.188	3.361
3 slides	no	bannister	0.100	4.367
4 cow	yes	maillot	0.047	4.916
5 ice scraper	no	mouse	0.259	3.072
6 mushroom	yes	mushroom	0.499	1.539
7 person	no	suit	0.642	1.107
8 fire	no	volcano	0.172	3.904

As we can see, from tested image set only two classes exist in ImageNet data set: mushroom and buffalo. Only mushroom was classified correctly. From results we can see that entropy of our probability distribution generally can indicate wrong matching. Only for incorrect prediction on image 7 (person) entropy was small, however classification of that image is contradictory. Human will classify that image as 'person', but suit is also present on that image.

Worth mentioning the fact that we have different image styles in out testing set. On image 4 we see cow, but it's style differs from other images:

To classify this image correctly will be interesting to train two networks for different image styles: one simple photos, second special style (cartoon style for example). Then determine style of image (as in Caffe 'Flikr style' classification example) and classify object in corresponding network. With this may classify cow both on simple images and images with special style.