ImageNet Classification with Deep Convolutional Neural Networks

AlexNet

Luke McEachern and Maximilian Polaczuk

05/08/2016

Introduction

AlexNet[1] covers a lot of the topics we've done this week, so its a good way to show some real life applications of them. Previous labeled data sets for image recognition consisted of far less images than the ImageNet, however provided decent results in image recognition.
It has only recently been possible to collect data sets with millions of images. Having larger data sets allows us to train nets that are able to recognise images with more variation, , as to emulate human ability closer(in terms of accuracy and diversity of classifications).

Two measures of success that are used in AlexNet are top 5% and top 1%, these correspond to the % of images that were identified as the most likely or within the top 5 most likely, respectively.

[1]For ease of reference, we refer to this project/ neural net as AlexNet. This is of course referring to one of the primary authors of the paper, Alex Krizhevsky

Introduction continued...

The main problem we face in scaling image recognition problems is complexity. The nature in which we would model image recognition in a feed-forward fully connected net would lead to a net so complex, with so many (millions) of parameters, it would be impossible to train. We therefore rely on convolutional layers which under (mostly true) assumptions about the data, give us a much more efficient net with fewer connections and parameters, allowing us to train our net.
Previously training a net on a large scale, high resolution labeled data-set proved much too expensive to train. WIth the recent developments in GPU technology and optimization of 2D convolutional nets, using CNNs on high resolution labeled data sets like ImageNet has become possible.

Data set

The data set that this paper uses, ImageNet, is the largest of its kind. It is a labeled image dataset with over 15 million high resolution images with around 22,000 categories.
In particular, AlexNet was entered into the competition ImageNet Large-Scale Visual Recognition Challenge(ILSVRC); a subset of ImageNet containing 1000 images in each of 1000 categories was used for this competition. Overall AlexNet uses 1.2 million training images, 50,000 validation images[2] and 150,000 testing images.

Data as input: In nets with convolulutional layers, we require a set dimensionality of out input in order for us to be able to use it. ImageNet, however contains variable resolution images and we therefore need to normalise our images to a set resolution- 256x256. Asides from date augmentation (later on), input images are: RGB, resolution normalised and mean-centered on each pixel .

[2]A validiation set is a subset of the overall training set, used to tune our hyperparameters while avoiding overfitting

ReLU Nonlinearity

In the paper they opt for using rectified linear units ReLU as an activation function instead of the usual hyperbolic tangent or sigmoid function. ReLUs are non-saturating nonlinearities, tanh and logistic sigmoid are saturating nonlinear functions. Using ReLU lead to much faster training time.
Non-saturating means that the input is not squeezed into a certain interval.
ReLU is summarized by the equation: $f(x) = \max(0, x)$
For comparison, recall:
Hyperbolic tangent is: $f(x) = \tanh(x)$
Sigmoid is $f(x) = \frac{1}{(1+e^{-x})}$

Training on Multiple GPUs

Computationally, a neural net's size is constrained by the amount of memory (and efficiency) of GPU used in training. The GPU's used in this paper were GTX 580 3GB GPUs- at the time of the contest (2010) this was 'the best money could buy'.
As it happens, the neural net that was made for the ILSVRC data set was unable to be processed by a single GPU, therefore two were used in parallel and this duality was built specifically into the architecture. Using the parallel GPU architecture increased top-1 and top-5 error rates by 1.7% and 1.2% respectively.

Later when we look at architecture, we can see the net as being 'split' horizontally, such that each GPU holds half of the responsibility of the net. Under this architecture, our GPU's communicate only at certain levels, while otherwise being independent of each other. An interesting feature of this limited connectivity is that the GPU's specialize (see figure 3 in paper).

Local Response Normalization

Although ReLUs do not require inputs to be normalized (since they are non-saturating), AlexNet authors found that including local normalization improved model generalisation.
Let $a^i_{x,y}$ denote the activity of a neuron computed by applying kernel i at position (x, y)

The response-normalized activity is shown as:
$b^i_{x,y} = a^i_{x,y} / (k + \alpha \sum (a^j_{x,y})^2)^\beta$

Where k, n, α and β are constant hyperparameters - selected via the validation set.
And we sum over n "adjacent" kernel maps at the same spatial position.

Including the normalization led to a lower top-1 and top-5 error rates by 1.4 and 1.2 percent.

It also reduced the test error rate on the CIFAR-10 dataset by 2%.

Overlapping Pooling

In class we have seen example of max pooling where a zxz sized grid is spaced $s = z$ apart.
This means that the pooling does not overlap.

AlexNet sets $s = 2$ and $z = 3$ which means our max pooling overlaps (rather than just dividing up the image and pooling from that), because $s < z$

This improved their error rates in top 1 and top 5 prediction by 0.4% and 0.3%, respectively, when compared to the non-overlapping pooling of $s = 2$ and $z = 2$.

In general overlapping pooling reduces the degree of overfitting.

Reducing Overfitting

Overfitting is a major problem in image recognition. The nature of the data we are dealing with means that the number of parameters that our net has increases exponentially as we increase the resolution of our images. With the size of the training set (and number of categories) fixed for the competition, altering them is not possible and alternative methods are employed.

Data Augmentation:
Data augmentation is essentially a way of increasing our dataset, without really increasing it. This is done in one of two ways:

- 1: Generating image translations and horizontal reflections. This was done by treating the input images as 224x224 'sub-images'.That is for every original 256x256 image, we iterate 1024 224x224 images, with horizontal reflections for each iteration, this effectively increases our training set by a factor of 2048.

Reducing Overfitting continued...

- 2: Altering intensities of RBG channels in training images, is a bit less intuitive than 1). A description, in plain English is creating iterations of our input image with different levels of brightness.

  From the paper: "consists of altering the intensities of the RGB channels in training images. Specifically, we perform PCA on the set of RGB pixel values throughout the ImageNet training set. To each training image, we add multiples of the found principal components, with magnitudes proportional to the corresponding eigenvalues times a random variable drawn from a Gaussian with mean zero and standard deviation 0.1"

  Doing this also captures the important 'human-like' quality that different levels of luminosity should not affect our ability to classify an image.

Reducing Overfitting continued...

Dropout: In a perfect world, we would like to consider the outputs of many different models as a way to have a 'distribution' of outputs and to reduce test errors. As the name suggests, however, this is a 'perfect world', in reality doing this in a neural net this large would be much too expensive to do. Dropout is a technique that allows us to compare models but in a much more efficient way. Basically:

Reducing Overfitting continued...

Dropout continued...

- in our first two fully connected layers the probability of a neuron being active is 0.5, neurons that are not active do not participate in forward pass or back prop and so their weights do not change from priors (in the first iteration).
- every time an image is read, we re-apply this probability of those neurons being active, neurons that are active have their weights updated, those who are not, stay static.
- we can consider each iteration a new model architecture, and the weights shared (remembered) between each model architecture (iteration).
- with probability of any neuron being active for any image read in training, during test we need to multiply the neurons output by 0.5 to reflect this.
- Dropout roughly doubles the amount of iterations required to converge. It also gives some robustness in the result.

Architecture

The network has 5 convolutional layers and a following 3 fully connected layers.

The last FC layer is given to a 1000-way softmax.
This maximises the average (across training cases) of the log-probability of the correct label under the prediction distribution.

ReLU nonlinearity is applied to each output of every CONV and FC layer.

Architecture continued...

Start with 224x224x3 (image)

Move to 96 kernels of size 11x11x3 with stride of 4

Response-normalized and pooled data becomes input for next layer

Filter with 256 kernels of size 5x5x48

The third, fourth and fifth CONV layers are connected without any downsampling or normalization

Details of Learning

The model used stochastic gradient descent with a batch size of 128 examples, momentum of 0.9, weight decay of 0.0005 and an equal learning rate for all layers.
The weight decay was found to lower the model's training error.

To update weight w they used:
$v_{i+1} := 0.9v_i - 0.0005\epsilon w_i - \epsilon\langle \frac{\partial L}{\partial w}|_{w_i}\rangle_{D_i}$
$w_{i+1} := w_i + v_{i+1}$

The weights were initialized by $N(0, 0.01)$

To speed up learning in the early stages they initialized the biases for the 2nd,4th and 5th convolutional layers and the fully connected layers, as ReLU learns well with positive inputs.
The remaining biases were initialized at 0.

Results

AlexNet scored winning results across the board at ILSVRC-2010 with top-1 rates of 37.5% and top-5 rates of 17.5% with the next closest contender at 45% and 25.7%, respectively. Results on different versions of the model from different years are given in the paper.

Improvements

In 2013, Matthew Zeiler and Rob Fergus improved the AlexNet by adjusting the hyperparameters in the model. They reduced the size of filter and stride on the first layer and extended the size of the middle convolutional layers.
In 2014 Szegedy et al. from Google developed the "GoogLeNet" which improved the AlexNet by reducing the number of parameters down to 4M. The main innovation is opting for average pooling instead of the fully connected layers at the top of the network, severely reducing the number of parameters.