Supervised Deep Learning for Multi-Class Image Classification

Xiaodong Zhou (xdzhou@stanford.edu)

Abstract

Multi-Class Image Classification is a big research topic with broad application prospects in Artificial Intelligence field nowadays. This course project describes the supervised machine learning methods, Convolutional Neural Networks (a.k.a. CNNs) along with Softmax logistic regression, to perform Multi-Class image classification and implement these deep learning algorithms on a large-scale Multi-Class Image Classification dataset from ImageNet annual competition task [1]. The implementation categorizes various images by visual features and shows illustrative examples of the training performance.

1 Introduction

Multi-Class Image Classification refers to an image classification task with more than two classes. In other words, one image needs to be labeled by one category from a set of categories based on analyzing the numerical properties of various image features and organization information. For example, to classify a set of animal images of cat, dog, fox, and pet, the Multi-Class Image Classification makes assumption that each image only could be labeled by one and only one class label (ground truth). That means the animal could be assigned to label "cat" or "pet", but never be both. This technique is widely used in multiple areas, such as producing thematic maps, textured image recognition, etc.

Moreover, the image is a complex context and it contains a complex arrangement of pixels. Some sub-regions of pixels also could be considered as objects which have special meanings to human being in the image[2]. The Deep Learning is the effective learning method for analyzing image which generates features by multiple sub-sampling layers.

As stated on the ImageNet website, the ImageNet is an ongoing research effort to provide researchers around the world an easily accessible large-scale image database. This research community also keeps posting AI related challenges, such as image classification, image localization and object detection, on its website through 2010 till now [1]. In order to master the deep learning models, this project chooses the classification task and images from the ImageNet since it is a typical Multi-Class Image Classification problem.

2 Dataset

The Dataset 2 from ILSVRC2014[1] is chosen as the dataset for this project. The training data is the subset of ImageNet which contains 1000 categories and more than 1.2 million images. The number of images in each category (synset) ranges from 732 to 1300. The validation data is a random subset of 50,000 images which doesn't contains any training data, 50 images per synset, with ground truth labels. All images are in JPEG format.

However, considering the restriction of hardware and software resources, we randomly pick up 20 out of 1,000 categories from the training set and validation dataset (1,000 images) to implement the deep learning algorithms. We modify the labels of these training images and validation images from 1 to 20 respectively based on their original category information and relationship. We call the dataset as ILSVRC2014_R20.

3 Methods

Intuitively, the convolution of two functions (input function $f(x)$ and the kernel function $k(x)$) represents the amount of overlap between them. Convolution is often used for filtering images with a kernel. It detects gradients in the values of pixels (sudden changes in brightness), which correspond to edges [3].

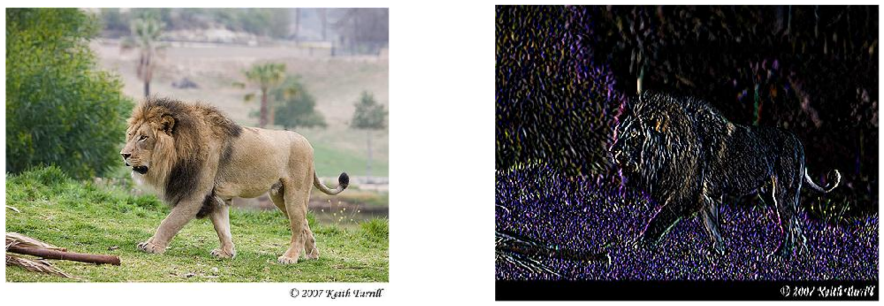

Figure1 *Edge detector: n-dimensional convolution function (convn) with input image (386×500×3) and randomly initialized kernel function (3×3 matrix filter)*

3.1 Convolutional Neural Networks (CNNs)

The CNNs is a biologically-inspired, locally connected, typical MLP (Multilayer Perceptron, a.k.a. Artificial Neural Network - ANN) and is sensitive of sub-regions of the visual field [3].

In this project, the pipeline of CNNs model consists of 5 layers: 2 convolutional layers, 2 max pooling layers, and 1 fully-connected MLP layer with Softmax regression. The output of each layer is the input of next layer.

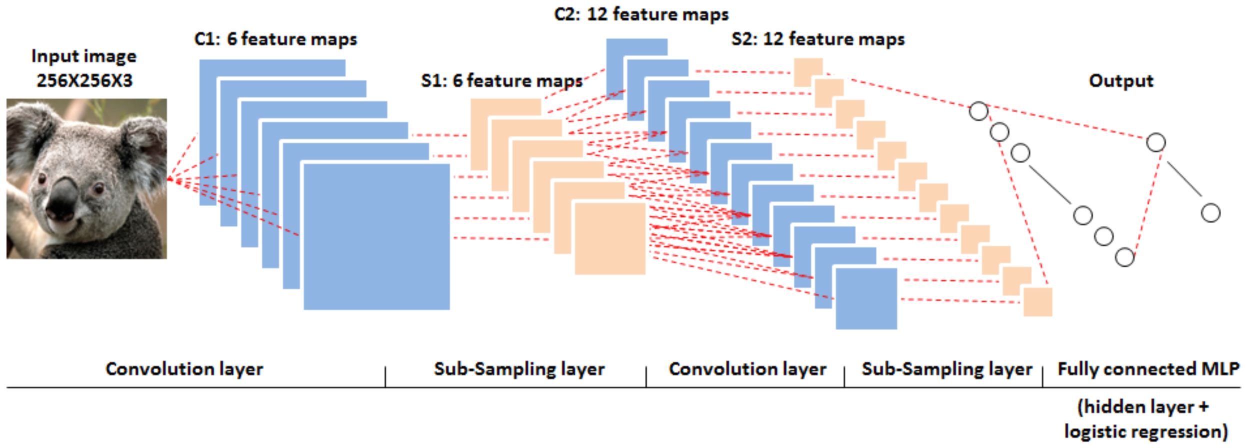

Figure 2 *Pipeline of CNNs model – 5 layers. Input: RGB image with 256×256 resolution; output: probabilities of classes.*

Sparse Connectivity

The CNNs is locally connected MLP. The inputs of hidden units (neurons) in layer i are from a subset of units in adjacent lower layer $i-1$, the neurons which have spatially contiguous receptive fields. We use convolutional filters with 9×9 receptive fields. The number of hidden units is dataset driven and depends on the complexity of the input distribution. We construct a large number of hidden units since the input is high resolution color images.

Let D denote the size of input x and L denote the size of output $f(x)$. Formally, with weight matrices $W^{(1)}, W^{(2)}$, bias vectors $b^{(1)}, b^{(2)}$, and activation functions G and g, for input x and output $f(x)$ of each layer MLP, the function $f: R^D \to R^L$ is:

$$f(x) = G(b^{(2)} + W^{(2)}(g(b^{(1)} + W^{(1)}x)))$$

The hidden layer is constituted as:

$$h(x) = g(b^{(1)} + W^{(1)}x).$$

The output is obtained as:

$$o(x) = G(b^{(2)} + W^{(2)}h(x)).$$

Shared Weights

Each convolutional filter is replicated across the entire visual field. These replicated units share the same parameterization (weight matrix W and bias vector b) and form a feature map. We initialize the weights W randomly from a uniform distribution in the range $\left[-\sqrt{\frac{6}{n_v + n_l}}, \sqrt{\frac{6}{n_v + n_l}}\right]$, where n_l is the number of labels, and n_v is the number of output neurons at the last layer. The size of matrix W is (n_l, n_v).

Feature map

A feature map is obtained by repeated application of a function $h(x) = g(b^{(1)} + W^{(1)}x)$ across sub-regions of the entire image. In order to yield faster training and better local minima, we use function $\tanh$ as activation functions g,

where $\tanh(z) = \frac{\exp(z) - \exp(-z)}{\exp(z) + \exp(-z)}$.

We randomly initialize the $\tanh$ activation function results interval to be $\left[-\sqrt{\frac{6}{fan_{in} + fan_{out}}}, \sqrt{\frac{6}{fan_{in} + fan_{out}}}\right]$ for each layer, where fan_{in} is the number of input features (input maps × filter height × filter width) and fan_{out} is the number of output features (output maps × filter height × filter width × pooling size).

Max-Pooling

Max-Pooling is a form of non-linear down-sampling and reducing the dimensionality of intermediate representations. We use a 2x2 region for Max-Pooling (scale = 2).

Stochastic gradient descent (SGD)

We perform stochastic gradient descent with fixed batch size (i.e. 50) for training input images. SGD estimates the gradient per batch at a time from the training examples instead of the entire training set. It reduces variance in the estimate of the gradient and proceeds more quickly.

We test different batch sizes to tweak the other parameters and discover the better training performance.

Loss Function & back-propagation algorithm

In the case of multi-class logistic regression, we apply the back-propagation algorithm to obtain the gradients $\partial\ell/\partial\theta$ for learning optimal model parameters, $\theta = \{W^{(2)}, b^{(2)}, W^{(1)}, b^{(1)}\}$, minimize negative log-likelihood (loss function).

To learn optimal model parameters smoothly, we try different modifications of the loss function.

Learning rate

We setup constant learning rate (i.e. 0.95) and test different learning rates for each layer since gradients at the lower layers are smaller and less reliable.

3.2 Softmax Regression

We construct the last layer as fully-connected MLP with hidden layer and logistic regression, and the set of all features maps at the layer as input. We use Softmax Regression (20-way softmax) as a classifier since the verification method includes 20 categories (classes).

Softmax logistic regression is a linear probabilistic classifier. It is parameterized by a weight matrix W and a bias vector b [3]. The probability that an input x belongs to class i is as:

$$p(y=i \mid x;W,b)=\frac{\exp(W_i x+b_i)}{\sum_j \exp(W_j x+b_j)}$$

$$y=\arg\max_i p(y=i \mid x;W,b)$$

4 Result

4.1 Preprocessing the data

In the implementation program, 20 out of 1,000 categories images are randomly chosen from the training set and validation dataset (1,000 images), read into Octave, rescaled with down-sampled to a fixed resolution of 256X256X3 (*imresize*), and saved as input with .mat format. We modify the labels of these training and validation images from 1 to 20 respectively based on their relationship of original category information. The summary information about training data set, ILSVRC2014_R20, is listed in Table 1.

ImageNet Categories	Images	Assigned Classes	Descriptions (words)
781	1,300	1	mask
997	1,300	2	rubber eraser
388	1,300	3	house finch, linnet
255	1,300	4	mountain bike
914	1,300	5	cellular telephone
655	1,300	6	sea slug, nudibranch
165	1,300	7	bison
615	1,300	8	fiddler crab
948	1,300	9	pizza, pizza pie
624	1,300	10	long-horned beetle
974	1,300	11	ice cream
442	1,300	12	great white shark
864	1,300	13	dough
571	1,253	14	radiator
386	1,300	15	brambling, montifringilla
17	1,300	16	Great Dane
126	1,300	17	Irish terrier
470	1,300	18	Gila monster
908	1,300	19	basketball
992	1,300	20	bubble
	25,953		

Table 1 *Training dataset: ILSVRC2014_R20*

4.2 Training error

The training error result of the algorithm is showed in the figure below.

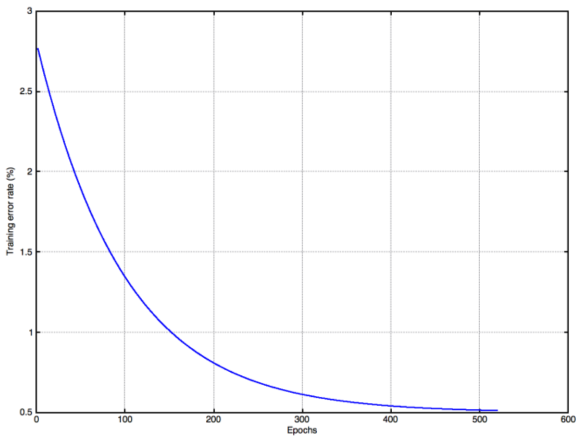

Figure 3 *training errors of the CNNs model*

The valuation result is list in Table 2.

Model	**Generational Error**
CNNs	58.3%
CNN ILSVRC-2012[4]	*37.5%*

Table 2 *Error Rates Table*

4.3 Discussion

We received acceptable training error rates from applying the CNNs and Softmax model on the Multi-Class Image Classification problem. However, comparing with the results of the 1st place of ImageNet 2012 competition in the Table 2, the classifier has higher testing error than expected.

We developed the algorithm with open-source software Octave and tested it on a small Linux server with a dual-core CPU. Due to the array size limitation of Octave and the hardware memory availability, we are only be able to construct 5 layers Neural Networks to process 20 out of 1000 categories data set and downsize all the images to resolution 256X256X3. It takes long time to read (~20 hours) and train (~4 hours) the large data set. We believe the results should be improved if we can run the implementation on a higher performance CPU/GPU system with bigger training dataset.

Due to the time consuming of the large-scale image dataset, we do not have enough time to tune the algorithm variables of our CNNs and Softmax regression to lower validation errors and improve the running performance. We also found that the return of the loss function is not smooth for the high resolution images trained by deep neural networks. We would like to find a better modification to update the loss function and smooth the result.

We validated our algorithm based on top-1 error which compares the output label of highest probability with validation ground truth. Most of the images in the ImageNet dataset contain multiple objects. The complexity of large-scale high resolution images is a quite challenge to classify them accurately and consistently. It would be helpful to obtain better classification accuracy if we integrate object detection with image classification, but it will be more expensive for the large neural networks. In the future, we would like to apply some efficient version of model combination, such as drop out [4].

We also tested our Convolution Neural Networks on both RGB and grayscale (rgb2gray) with same dataset. The results do not have much difference.

5 Future Works

In the future, we will try to integrate object detection and dropout models to continue working on improving the performance of the Multi-Class Image Classification problem.

We would also like to apply the probability model, Latent Dirichlet Allocation, on the SIFT features (bag of word) released by ImageNet to provide suggestions to CNNs based on the semantics foundation and discover the relationship between features.

References

[1] Olga Russakovsky*, Jia Deng*, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg and Li Fei-Fei. (* = equal contribution) **ImageNet Large Scale Visual Recognition Challenge**. *arXiv:1409.0575,* 2014.

[2] David G. Lowe, **Distinctive Image Features from Scale-Invariant Keypoints.** *International Journal of Computer Vision, 2004.*

[3] Ng, Andrew, Jiquan Ngiam, Chuan Y. Foo, Yifan Mai, and Caroline Suen. **UFLDL Tutorial**. Ufldl at Stanford University. n.d. Web. 12 Dec. 2013.

[4] Alex Krizhevsky, Ilya Sutskever, Geoffrey E Hinton, **ImageNet Classification with Deep Convolutional Neural Networks,** *NIPS 2012.*