Yelp Image Classification Report

Teng Peng

Summary

We have images and labels of many restaurants. Can we automatically tag new restaurant? Each restaurant has multiple images and multiple labels. For example,

Restaurant	Label	Image
A	0 1 0 1 1 1 0 0 0	Image1, Image2, Image3

We use a pre-trained neural network to extract feature vectors. Then compute mean feature vector for each restaurant. Next, perform SVM to do multilabel classification. The evaluation metric is f-1 score. We ranked 90 out of 355.

Challenges

1. Adding up train set and test set, there are half million images. 16 GB.
2. Multilabel classification. For multiclass classification, classes are mutually exclusive, while it is exact opposite for multilabel classification.
3. Multi-instance problem. For single-instance problem, one input is corresponding to one output, while multiple inputs are corresponding to one output in multi-instance problem. For our problem, multiple images (for a fixed restaurant) are corresponding to one set of labels. There are unknown relationships between restaurants and images.

Three Naive Approaches

An *image* is a matrix of pixels. A pixel has three color channels: Red,Green,Blue.

We first formulate the problem as a traditional machine learning problem. We view each image matrix as n-row vectors. The independence assumption is too strong and it is computational infeasible.

Second, we consider coerce the multilabel problem to a multiclass problem. The idea is to select the classes with top probability from softmax layer. The problem is that we do not know how to design a new objective function for multilabel and how to choose threshold for converting softmax probability to binary labels.

Third, we consider formulate multilabel problem as multiple binary classification problem. We use a convolution neural network LeNet, developed around 1990s. The LeNet is fast to train and does not demand very large GPU memory. This method works, but it lost the point that why we use LeNet, it does not address the specific challenges, and there is very large room to improve.

Convolutional Neural Network

Introduction and motivation

The basic difference between computer vision problem and traditional machine learning problem is that the inputs are images rather than vectors. What if we can find a good way to transform images to vectors? We are seeking a vector representation of images. Such representation must contain information as much as possible, and make trade off to make it computational feasible.

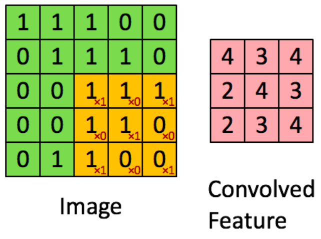
The solution is to use pre-trained convolution neural network. We first explain the motivation and structure of convolution neural network, then introduce how does pre-train works in practice.

Due to the fully connected nature of multilayer neural network, the number of parameters are soon exploded for images. Consider a image of 248x400. The first fc layer, connected to the input layer, has more than 297,600 weights. It again impose a very strong independence assumption. However, in reality, pixels are highly correlated.

The CNN is designed to address the problems for images. It reduces the number of parameters by several ways: 1. local connectivity rather than fully connectivity 2. parameter sharing 3. max pooling layer and so on.

What is Convolution?

The basic difference between CNN and MLN is that the former has convolution layer. To describe convolution layer, we first explain what is convolution. There are two way to explain convolution: one is pure math, the other by images. The following image goes with the second way. We have a 5x5 image on the left hand side, in which 0 means white, and 1 means black. On the right hand side, we have A 3x3 sliding(convolving) window(or kernel, filter). When the window sliding over the image, we do dot product and aggregate the result into a matrix.



The new matrix is a new representation of original image. Or more intuitively, it is a filtered image of the original one. The following four images are: original, sharpened, edge detection, and edge enhancement.

Features are gradually captured by convoluting multiple times. And the local connectivity originated from the usage of filters(or its receptive field).

What is a Convolutional layer?

In summary, we slide several filters over the input, and the results are called activation maps. The stack of activation map is a representation of the input.

We attempt to explain details by an example. Suppose we have an input 7x7x3. We first choose several hyperparameters: pad the input with 1 round of zeros; 2 filters, each 3x3x3; 2 stride(the filters move 2 cell each step). By padding, the input is 8x8x3(it is wrapped by a round of zeros). First filter sliding over the input with stride 2. The second filter sliding over the input with stride 2. We get 2 activation maps.

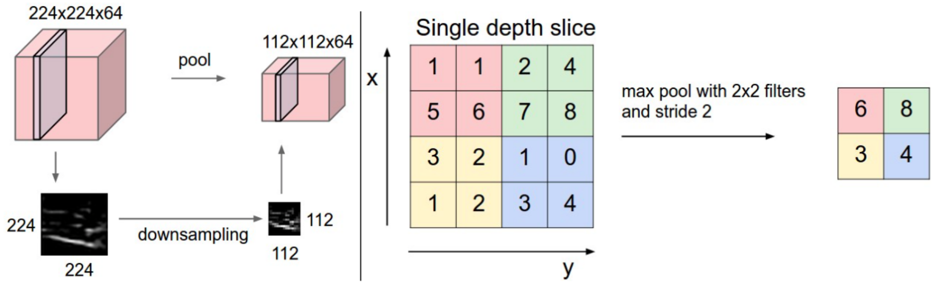
We use a real world example to calculate the number of parameters. Input: 227x227x3. Filter: 11x11. Stride = 4. Depth(not the depth of NN, no.of stacks of activation map) = 96. Total Neurons = 55x55x96 = 290,400. Parameters per neuron = 11x11x3 + 1(bias) = 364. Total parameters = 290400x364 = 105,705,600.

The number of parameter is still very large. We use parameter sharing scheme to reduce the number of parameters: for each slide, all neurons share 1 weight. Total number of parameters: 96x(11x11x3) + 1 = 34,848. As Prof.Pan commented, it is a regularization method.

The Max pooling layer

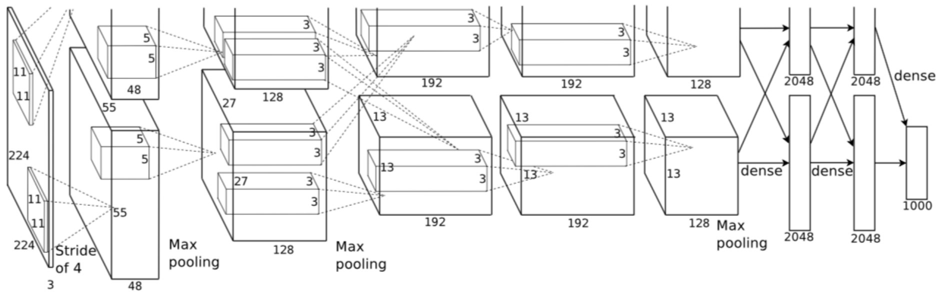
The architecture of a CNN is several pairs of convolution layer and max pooling layer. It looks like Conv -> Pool -> Conv -> Pool -> Conv -> Pool. A pair of layer would reduce the size of input by 50%. After many pairs, we end up with an acceptable output for fc layer.

Consider an input of 224x224x64. For the max pooling layer, we slide a 2x2 filter with stride 2, and take the max out. We end up with an output 112x112x64.



A real world example

Here is the architecture of AlexNet, state-of-the-art in 2012; the first modern and popular CNN.



Pre-trained neural network

The AlexNet takes 5~6 days to train on 2 GPU by the authors. We use pre-trained neural network to avoid training. It works as follows:

1. Someone trains a NN on a large image database ($>$ 1 million)
2. Pack the parameter information into a text file (usually $>$ 100MB)
3. Share with us through the internet
4. We download the file
5. Feed our image into the trained NN

We get a probability vector by softmax layer and many feature vectors by fc layer. The method works very well if our data is large and close enough to the original image database. It takes only 2 hours to extract features.

Any English interpretation is valid(but may not be complete) unless it is a contradiction to the mathematical description. However, a good interpretation may suggests new method implicitly. We can view pre-trained network as a stubborn yet experienced old guy, and training from scratch is like a young and inexperienced guy. This analogy suggests that we can seek a balanced way in the middle of the two extreme cases: we can get a pre-trained network and use our image to tune it by a very small learning rate.

The above process has already been written into python library `caffe`.

Replace the softmax layer with SVM

We use fc7 of AlexNet to extract features and feed feature vectors into SVM. We can also view it as replacing the softmax layer with SVM. (This paraphrase seems redundant. However, it suggests new methods like replace the softmax layer with another neural network.)

We have feature vectors of images at hand. Our challenge is that the labels are tagged on each restaurant, i.e. a set of images. We do not know how to uncover the relationship between images and each restraint. Our approach is to take mean vectors of all images for each restraint. This method is simple and works very well for our problem.

To deal with the multilabel problem, we use One-Vs-The-Rest or one-vs-all. The strategy consists in fitting one classifier per class. For each classifier, the class is fitted against all the other classes.

Next we do SVM as follows:

1. Conduct simple scaling on the data
2. Consider the RBF kernel
3. Use cross-validation to find the best parameter C and γ
4. Use the best parameter C and γ to train the whole training set
5. Predict on test set

Evaluation Metric

The evaluation metric is mean F score.

$$F1 = 2\frac{pr}{p+r} \quad \text{where} \quad p = \frac{tp}{tp+fp}, \quad r = \frac{tp}{tp+fn}$$

Precision(positive predictive value) is the ratio of true positives (tp) to all predicted positives (tp + fp). Recall(sensitivity, or true positive rate) is the ratio of true positives to all actual positives (tp + fn). The F1 metric weights recall and precision equally, and a good retrieval algorithm will maximize both precision and recall simultaneously. Thus moderately good performance on both will be favored over extremely good performance on one and poor performance on the other.

Winners' Solution

We ranked 90 out of 355. We believe if the full potential of the method is realized, like use newer CNN, do PCA and do averaging, we may rank around 50. No.1 and No.2 scored 0.3 better than us in terms of F-1 score. Several days ago, many top ranker shared their approaches. The No.2's approach is the most interesting one, because he is really thinking in deep learning, rather than throwing a bunch of statistical methods into the problem and stack them up.

The great part of his method is that he uses CNN to handle multilabel and multiclass problem simultaneously. He first extract features by a pre-trained network by Facebook. Next, he feed images of a restaurant into a new CNN. That is, a Nx2048 matrix where N is the number of images the restaurant has. He uses 9 sigmoid units(I think he means 10) for multilabel. The normal FC layer is modified: the activation is a softmax over images, weighting the importance of each image for a particular feature.

Credits

The encouragement and comments of Prof.Pan and Prof.Shen are priceless to me. Without help from Nina Chen, I would give up early. I learnt many ideas and methods from Stanford cs231n taught by Fei-Fei Li.

Reference

Chen, Nina. 2016. "Kaggle Yelp Image Classification." https://github.com/ncchen55414/Kaggle-Yelp.

Fei-Fei Li, Andrej Karpathy. 2015. "CS 224D: Deep Learning for NLP."

Hsu, Chih-Wei, Chih-Chung Chang, Chih-Jen Lin, and others. 2003. "A Practical Guide to Support Vector Classification."

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E Hinton. 2012. "ImageNet Classification with Deep Convolutional Neural Networks." In *Advances in Neural Information Processing Systems*, 1106–14.

Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, et al. 2011. "Scikit-Learn: Machine Learning in Python." *Journal of Machine Learning Research* 12: 2825–30.

Yelp. 2016. "Kaggle Yelp Image Classification." https://www.kaggle.com/c/yelp-restaurant-photo-classification.