Machine Learning Project Report

Han Gao 40053734

Table of Contents

1 Introduction and motivation 3
2 Convolutional Neural Networks 4
Convolutional Layer 4
Pooling layer 5
Fully-connected layer 5
SoftMax 6
Cross-Entropy 6
3 The Dataset 7
4 The Architecture 7
5 Results 8
6 References 9

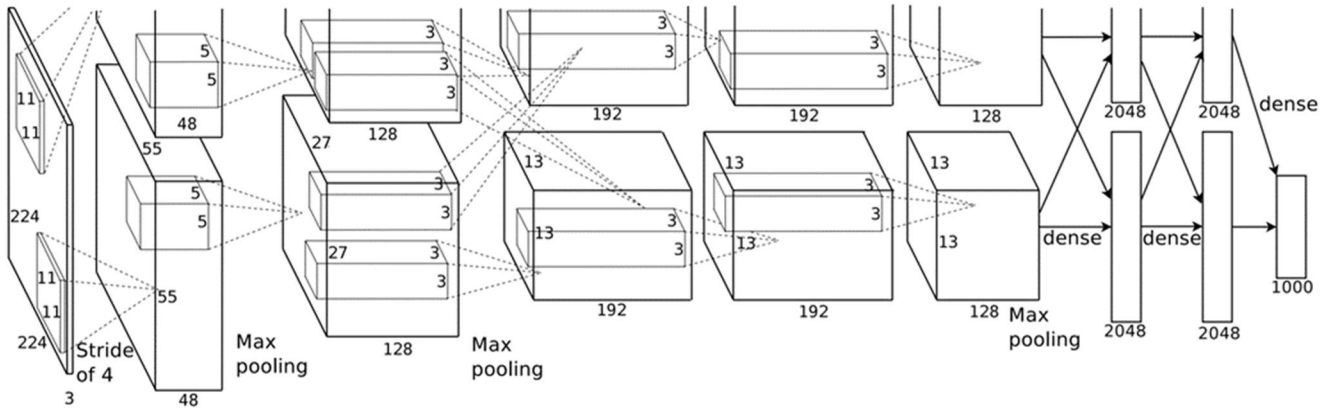
1 Introduction and motivation

We are constantly analysing the world around us. Without conscious effort, we make predictions about everything we see, and act upon them. When we see something, we label every object based on what we have learned in the past. Similar to how a child learns to recognise objects, we need to show an algorithm millions of pictures before it is be able to generalize the input and make predictions for images it has never seen before. Computers 'see' in a different way than we do. Their world consists of only numbers. Every image can be represented as 2-dimensional arrays of numbers, known as pixels.

My Project Mainly Base on:

Alex Krizhevsky, Ilya Sutskever and Geoffrey E. Hinton (2012). ImageNet Classification with Deep Convolutional Neural Networks, Advances in Neural Information Processing Systems 25, NIPS Proceedings, pp. 1097–1105.

This paper introduce that they trained a large, deep convolutional neural network to classify the 1.2 million high-resolution images in the ImageNet LSVRC-2010 contest into the 1000 different classes. The neural network, which has 60 million parameters and 650,000 neurons, consists of five convolutional layers, some of which are followed by max-pooling layers, and three fully-connected layers with a final 1000-way soft max.



Despite the attractive qualities of CNNs, and despite the relative efficiency of their local architecture, they have still been prohibitively expensive to apply in large scale to high-resolution images. Luckily, current GPUs, paired with a highly-optimized implementation of 2D convolution, are powerful enough to facilitate the training of interestingly-large CNNs, and recent datasets such as ImageNet
contain enough labeled examples to train such models without severe overfitting.

In the end, the network's size is limited mainly by the amount of memory available on current GPUs and by the amount of training time that we are willing to tolerate. The network takes between five and six days to train on two GTX 580 3GB GPUs.

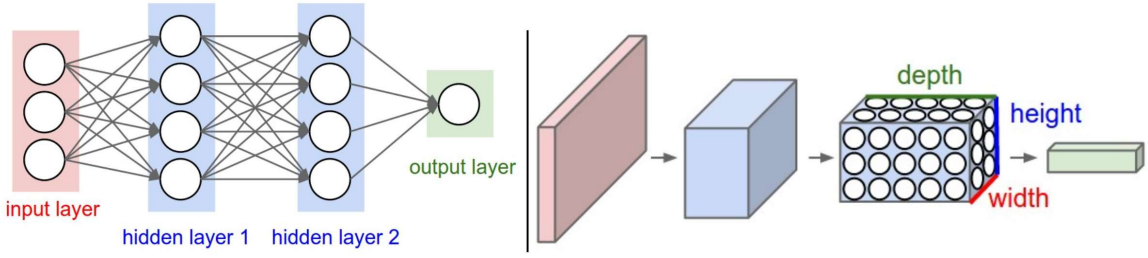
I don't have the GPU so I downsized the networks with two convolutional layers and two fully-connected layers to classify images into 2 classes, Which is Cat and Dog.

2 Convolutional Neural Networks

Convolutional Neural Networks (CNN) is a class of deep, feed-forward artificial neural networks, most commonly applied to analyzing visual imagery. CNNs use a variation of multilayer perceptrons designed to require minimal preprocessing.

Convolutional Neural Networks have a different architecture than regular Neural Networks. Regular Neural Networks transform an input by putting it through a series of hidden layers. Every layer is made up of a set of neurons, where each layer is fully connected to all neurons in the layer before. Finally, there is a last fully-connected layer—the output layer—that represent the predictions.

Convolutional Neural Networks take advantage of the fact that the input consists of images and they constrain the architecture in a more sensible way. In particular, unlike a regular Neural Network, the layers of a CNN have neurons arranged in 3 dimensions**:** width, height, depth**.**

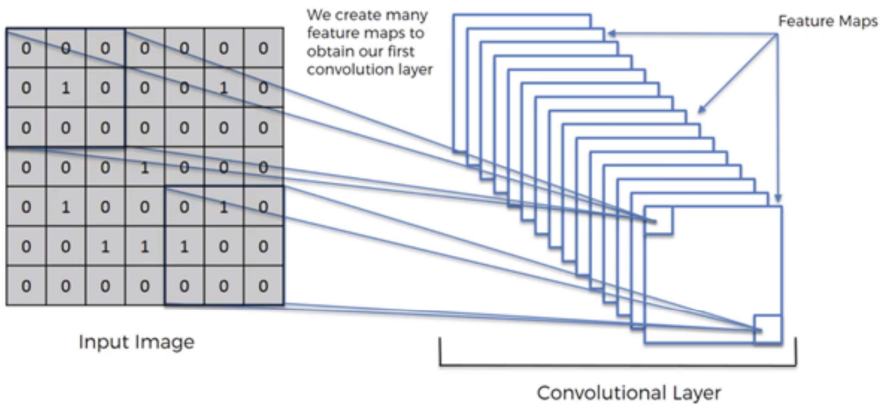


The convolution function is:

$$(f * g)(t) \stackrel{\text{def}}{=} \int_{-\infty}^{\infty} f(\tau)\, g(t-\tau)\, d\tau$$

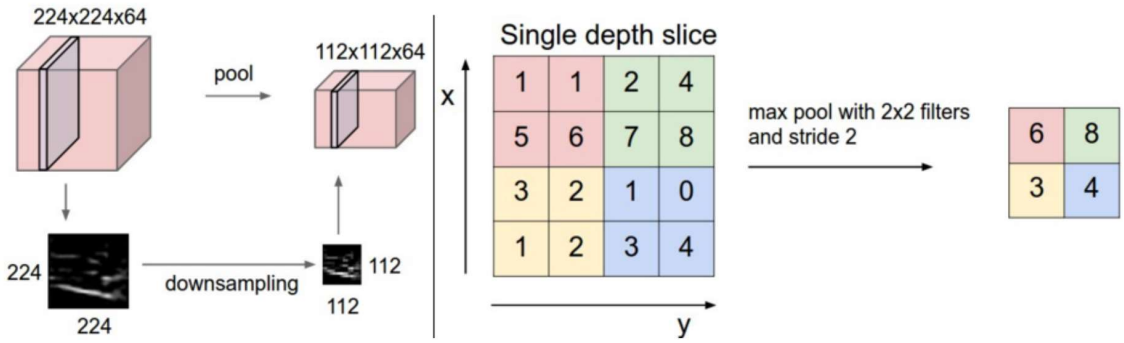
Convolutional Layer

A convolutional layer consists of multiple feature maps using different features of the image. Through the neural network training it decides which features are important.



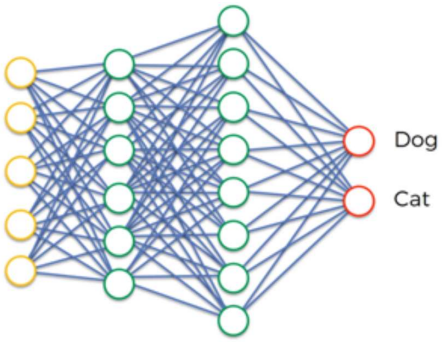
Pooling layer

CNN often also use pooling layers to reduce the size of the representation, to speed the computation, as well as make some of the features that detects a bit more robust.



Fully-connected layer

Neurons in a fully connected layer have full connections to all activations in the previous layer, as seen in regular Neural Networks. Their activations can hence be computed with a matrix multiplication followed by a bias offset.



SoftMax

The SoftMax function is used to provide output values with a probability between each of them. For example: there is an image of a dog that has been passed through a CNN, the machine decides that the image is 0.95 likely to be a dog and 0.05 likely to be a cat.

SoftMax Equation

$$f_j(z) = \frac{e^{z_j}}{\sum_k e^{z_k}}$$

The original value is squashed into much smaller values that both add up to 1 to allow the machine to provide a suitable probability of each image.

Cross-Entropy

Cross-Entropy is a function that comes hand-in-hand with SoftMax.

$$H(p, q) = -\sum_x p(x) \log q(x)$$

A Cross-Entropy function is used to calculate the Loss Function and helps to get a neural network to an optimal state. This method is the preferred method for classification. If using regression, you would use Mean Squared Error.

3 The Dataset

The training dataset is from Kaggle. The training archive contains 25,000 images of dogs and cats and testing dataset 10,000 images. Train your algorithm on these files and predict the labels for test1.zip (1 = dog, 0 = cat).

Web services are often protected with a challenge that's supposed to be easy for people to solve, but difficult for computers. Such a challenge is often called a CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart) or HIP (Human Interactive Proof). HIPs are used for many purposes, such as to reduce email and blog spam and prevent brute-force attacks on web site passwords.

Asirra (Animal Species Image Recognition for Restricting Access) is a HIP that works by asking users to identify photographs of cats and dogs. This task is difficult for computers, but studies have shown that people can accomplish it quickly and accurately. Many even think it's fun! Here is an example of the Asirra interface:

Asirra is unique because of its partnership with Petfinder.com, the world's largest site devoted to finding homes for homeless pets. They've provided Microsoft Research with over three million images of cats and dogs, manually classified by people at thousands of animal shelters across the United States. Kaggle is fortunate to offer a subset of this data for fun and research.

4 The Architecture

The project implements in python by using matplotlib, tensorflow, numpy libraries.

1. Processing training dataset (transfer images to tensor), save labels and image information

The name format of each image is “label. image number”, split the label and number by ‘.’, then save the label and number of each image as the input list.

Decoding image, resize the image into same size and standardization image.

2. Building Convolutional Neural Networks

The networks consisted of a two-level cascaded CNN (two sets of each convolution layer followed by a max-pooling layer) followed by two fully connected hidden layers and the final output layer. The details of each layer show in the source code.

Define the cross entropy loss function.

3. Training the model and testing.

Using the training dataset to train the model and testing by the model.

5 Results

The CNN cost 6 hours to training by using gradient decent(max steps 10000) on my laptop.

The average Training accuracy is 91% and average Testing accuracy is 88.6%

Dog with possibility 0.913

Dog with possibility 0.589

This is a cat with possibility 0.986623

This is a cat with possibility 0.991113

6 References

[1] Alex Krizhevsky, Ilya Sutskever and Geoffrey E. Hinton (2012). ImageNet Classification with Deep Convolutional Neural Networks, Advances in Neural Information Processing Systems 25, NIPS Proceedings, pp. 1097–1105.

[2] Kumar Chellapilla and David B Fogel. Evolution, neural networks, games,

and intelligence. Proceedings of the IEEE, 87(9):1471{1496, 1999.

[3] Siang Y. Chong, Mei K. Tan, and Jonathon D. White. Observing the evo

lution of neural networks learning to play the game of othello. IEEE Trans

actions on Evolutionary Computation, 9:240{251, 2005.

[4] Tambet Matiisen. Demystifying deep reinforcement learning, 2015.

[5] David Moriarty and Risto Miikkulainen. Evolving complex othello strategies

using marker-based genetic encoding of neural networks. Technical report,

1993.

[6] Andrew Ng. Lecture 16 - applications of reinforcement learning, April 2009.

[7] Brian Tanner and Richard S. Sutton. Td(Lambda) networks: Temporal

difference networks with eligibility traces. In Proceedings of the 22Nd Inter

national Conference on Machine Learning, ICML '05, pages 888{895, New

York, NY, USA, 2005. ACM.