Classification of Pipe Weld Images with Deep Neural Networks
— Literature Survey —

Dalyac Alexandre
ad6813@ic.ac.uk

Supervisors: Prof Murray Shanahan and Mr Jack Kelly
Course: CO541, Imperial College London

June 5, 2014

Abstract

Automatic image classification experienced a breakthrough in 2012 with the advent of GPU implementations of deep neural networks. Since then, state-of-the-art has centred around improving these deep neural networks. The following is a literature survey of papers relevant to the task of learning to automatically multi-tag images of pipe welds, from a restrictive number of training cases, and with high-level knowledge of some abstract features. It is therefore divided into 5 sections: foundations of machine learning with neural networks, deep convolutional neural networks (including deep belief networks), multi-tag learning, learning with few training examples, and incorporating knowledge of the structure of the data into the network architecture to optimise learning.

In terms of progress, a deep convolutional neural network, pretrained on 10 million images from ImageNet, has been trained on 150,000 pipe weld images for the simplest possible task of discriminating 'good' pipe weld from 'bad' pipe welds. A classification error rate of ??% has been attained. Although this figure is encouraging, it corresponds to a much simpler formulation of the task: the objective of this project is to achieve multi-tagging for 23 characteristics, some of which require considerable human training. Include a separate section on progress that describes: the activities and accomplishments of the project to date; any problems or obstacles that might have cropped up and how those problems or obstacles are being dealt with; and plans for the next phases of the project.

Contents

1 Background

This project aims to automate the classification of pipe weld images with deep neural networks. Classification is a machine learning task and deep neural networks are a class of machine learning tools, so we will start with fundamental concepts in machine learning and go through major refinements in neural network architectures to finish with the state-of-the-art machine learning model for image classification: convolutional deep belief nets. The last two sections focus on two challenges specific to the pipe weld image classification task: multi-tagging and learning features from a restricted training set.

1.1 Defining the problem: Single-Instance Multi-Label Supervised Learning

Given an instance space X and a set of class labels Y, learn a function f : X -¿ P(Y) from a given dataset (x1,y1),(x2,y2), ..., (xn,yn), where xi X is an instance and yi

The problem of learning to classify pipe weld images from a labelled dataset is a single-instance, multi-label, supervised learning classification problem. Indeed, the final classification of whether a pipe weld is accepted or rejected depends on the presence or absence of a number of characteristics (note that the term 'characteristic' is to be distinguished from feature and class, and could be defined as a component of a multidimensional class called the pipe weld). Formally, let X denote the instance space and Y = Y1 x Y2 x ... x Yp = 0,1p̂ the set of p characteristics. Then the task is to learn a function f : X -¿ Y from a given data set (x1, Y1), (x2, Y2), , (xm, Ym), where xi X is an instance and yi Y the known characteristics of xi.

This differs from the traditional supervised learning classification task of learning a function f : X Y where Y = e1, ... , ep i.e. the standard normal vectors of Rp̂. In other words, a traditional supervised learning classification task has one tag per instance, whereas in this case there can be a number of present characteristics.

This also differs from the Multi-Instance Multi-Label supervised learning task of learning a function f : X2̂ Y2̂, i.e. where any number of class instances can appear in one case-to-classify.

1.1.1 Supervised Learning

1.1.2 Unsupervised Learning

1.1.3 Approximation vs Generalisation

1.2 Existing Approaches

This section is divided into the most successful known approaches to tackling the problem, starting with the most successful, and finishing with an oft-times superior contender (which is inferior in this case): discriminative deep neural networks, hidden markov models, gaussian mixture models, and generative deep neural networks.

Or maybe deep belief nets have never been applied to multi-instance multi-label learning before and this project is a chance to explore their performance.

1.3 Deep Neural Networks

1.3.1 Architecture of Deep Neural Networks

Feed-Forward Architecture

Multilayer Peceptron

1.3.2 Training

Gradient Descent

Backpropagation

1.4 Deep Convolutional Neural Networks

CITATION BELOW! *"Convolutional neural network is a specific artificial neural network topology that is inspired by biological visual cortex and tailored for computer vision tasks by Yann LeCun in early 1990s. See http://deeplearning.net/tutorial/lenet.html for introduction. The convolutional neural network implemented in ccv is based on Alex Krizhevskys ground-breaking work presented in: ImageNet Classification with Deep Convolutional Neural Networks, Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton, NIPS 2012 The parameters are modified based on Matthew D. Zeilers work presented in: Visualizing and Understanding Convolutional Networks, Matthew D. Zeiler, and Rob Fergus, Arxiv 1311.2901 (Nov 2013)."*

1.4.1 Translation Invariance

1.4.2 Recent Architecture Improvements

1.5 Deep Belief Nets: a generative model

1.6 Multi-Instance Multi-Label Learning

1.7 Transfer Learning: learning with a restricted training set

2 Progress

2.1 Data: pipe weld images

There are 227,730 640x480 'RedBox' images. There are 1280x960 'BlueBox' images. They all have xx tags.

2.1.1 Visual Inspection

2.1.2 Analysis

ANOVA

t-SNE

2.2 Cuda-Convnet

Cuda-Convnet is a GPU implementation of a deep convolutional neural network implemented in CUDA/C++. It was written by Alex Krizhevsky to train the net that set ground-breaking records at the ILSVRC 2012 competition, and subsequently open-sourced. However, parts of his work are missing and his open source repository is not sufficient to reproduce his network (see test run for how this was dealt with).

2.2.1 Test Run

Use Alex Krizhevsky's GPU implementation. Use Daniel Nouri's noccn scripts. Download the file containing serialized code for Yangqing's decaf [1] network parameters, reverse engineer it using Daniel Nouri's skynet configuration files. Write batching script for decaf-net's image dimensions. Write data augmentation scripts to reduce overfit. Write c++ code to import the decaf parameters to initialise the weights of a new network, to re-initialise the weights of the fully connected layers, to train only the fully connected layers, keeping the lower layers 'frozen'.

888 batches of 128 images each. Used 10% of the data for testing - so trained on 102,400 images.

DeCAF *By Yangqinq Jia and University of California, Berkeley. "As the underlying architecture for our feature we adopt the deep convolutional neural network architecture proposed by Krizhevsky et al. (2012), which won the ImageNet Large Scale Visual Recognition Challenge 2012 (Berg et al., 2012) with a top-1 validation error rate of 40.7%. 3 We chose this model due to its performance on a difficult 1000-way classification task, hypothesizing that the activa- tions of the neurons in its late hidden layers might serve as very strong features for a variety of object recognition tasks. Its inputs are the mean-centered raw RGB pixel intensity values of a 224224 image. These values are forward propagated through 5 convolutional layers (with pooling and ReLU non-linearities applied along the way) and 3 fully-connected layers to determine its final neuron activ- ities: a distribution over the tasks 1000 object categories. Our instance of the model attains an error rate of 42.9% on the ILSVRC-2012 validation set 2.2% shy of the 40.7% achieved by (Krizhevsky et al., 2012). We refer to Krizhevsky et al. (2012) for a detailed discus- sion of the architecture and training protocol, which we closely followed with the exception of two small differences in the input data. First, we ignore the images orig- inal aspect ratio and warp it to 256 256, rather than re- sizing and cropping to preserve the proportions. Secondly, we did not perform the data augmentation trick of adding random multiples of the principle components of the RGB pixel values throughout the dataset, proposed as a way of capturing invariance to changes in illumination and color.*

Network Architecture *Don't want to saturate neurons. Need a standard deviation of weight initialisations proportional to the square root of the fan-in. Convolutional layers have a much smaller fan-in than, say, fully connected layers, so the standard deviation needs to be much lower.*

Use dropout of 0.5 in each fully connected layer, so add twice as many outputs. Go from 4096 to 8192. Actually no, keep it at 4096. Furthermore, reduce the number of neurons in each fully connected layer, because don't have as many classes to classify. Just need a few abstract features. The last layer has number of neurons fixed to number of classes. So actually, maybe reduce the number of neurons only in the 2nd fully connected one, but not the first: this is like having a few very abstract features, which are combinations of a very high number of less abstract features.

2.3 Transfer Learning

2.4 Further Work

Tweak Architecture *Change proportion of data used as training data. Change fan-in for the fully connected layers. Change dropout rate (probably don't need to). Does number of classes to classify influences network hyper-parameters, other than number of neurons in top layer? Maybe reduce the number of neurons in each fully connected layer even more so, because don't have as many classes to classify. Just need a few abstract features. The last layer has number of neurons fixed to number of classes. So actually, maybe reduce the number of neurons only in the 2nd fully connected one, but not the first: this is like having a few very abstract features, which are combinations of a very high number of less abstract features.*

Tweak Training *I'm training a CNN to recognise 3 classes, but 91% of my data is of the same class. Is there anything I should alter in the way I train?*

For example, should I make the proportions of each class in my validation batches even (1/3, 1/3, 1/3)? Because the thing is, weights are only saved when new validation error is lower than lowest validation error until now. But if this validation error is computed over hardly any examples of the other 2 classes, how can it be a good indicator that the network is on its way to learning good features?

Significantly Alter Architecture *Hidden Markov for pipe type. Whether the joint is a T-joint or not alters the way the image has to be assessed for each of the flags. In other words, whether or not the joint is a T-joint modifies the visual features that have to be picked up by the neural network for classification. Unless I'm mistaken, we do not have data for each image about whether or not it is a T-joint. At first, I had found that a bit disappointing, But I've just been thinking, this might be a very exciting research opportunity: there's a (powerful and super cool) type of machine learning model called a Hidden Markov model, where unknown states can be learned. In this case, the unknown state would "whether or not joint is a T-joint". Knowing this state should in theory help the network to improve classification, because it could indicate to it that a slightly different set of visual features need to be used for classification. I don't think much work has been done on combining neural networks with hidden markov models - and I don't know whether it's possible. But I'd like to mention it in my first report as a potential research path (unless Jack thinks it's a bad idea!). Hence why I wanted to be sure about the terminology for these different joints!*

2.4.1 Caffe

2.4.2 Theano

2.4.3 Hidden Markov Model

It might be interesting to explore hidden markov models because the data has obvious hidden states which humans benefit from learning: pipe welds can be T welds and standard welds, and this alters where scratch marks need to be seen.

References

[1] *Donahue, Jeff and Jia, Yangqing and Vinyals, Oriol and Hoffman, Judy and Zhang, Ning and Tzeng, Eric and Darrell, Trevor,* DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition
arXiv preprint arXiv:1310.1531, 2013