REVIEW OF DEEP LEARNING FRAMEWORKS FOR COMPUTER VISION

Shriarulmozhivarman Gobichettipalayam
université de bourgogne
shriarul643@gmail.com

Sofia Mehraj Asraf Ali
université de bourgogne
sofia.mehraj@gmail.com

ABSTRACT

XXXXXXXXXXXXXXXXXXXXXX

1 INTRODUCTION

Deep learning methods have been shown to be superior to the latest machine learning technologies in many fields in the past few years, and computer vision is one of the most prominent cases. This review article provides a brief overview of some of the most important deep learning schemes used in computer vision problems, namely convolutions neural networks,and deep belief networks, and stacked noise reduction auto encoders.Limitations, and their applications in various computer vision tasks, such as object detection, facial recognition, action and activity recognition, and human pose estimation are highly computation. The most prominent factors that contributed to the huge impetus for deep learning include the emergence of large, high-quality, publicly labeled data sets, and support for parallel GPU computing, which makes the transition from CPU-based to GPU-based transition possible. Training it can greatly accelerate the training of the deep model. Other factors may also play a smaller role, due to the separation of saturated activation functions (such as hyperbolic tangent and logistic functions), the disappearing gradient problem is alleviated, and new regularisation techniques (such as dropout, batch normalisation and Data augmentation), and the emergence of powerful frameworks that can speed up prototyping.We shall review in the main Frameworks for Deep learning in this paper.Deep learning usually involves using a series of layers to learn hierarchical representations. These layers process the input to generate a series of representations and then provide these representations to the next layer. By having a sufficiently high space, such methods can capture the range of relationships found in the raw data. This can be seen as separating multiple manifolds representing data through a series of transformations

2 DEEP LEARNING ARCHITECTURES

A deep learning model is a neural network that allows multiple hidden layers. There are various deep learning architectures, such as convolutional neural networks, deep belief networks, and recurrent neural networks. These have been applied to the fields of computer vision, automatic speech recognition, natural language processing, audio recognition and bio-informatics, where they have been shown to produce the latest results on various tasks. In those architectures, convolutional neural networks and recurrent neural networks are classified as supervised learning models. And, in recent years, those supervised learning models are more popular than unsupervised learning models such as deep belief networks, because supervised learning models have shown popular applications in the above-mentioned fields. Deep learning models can be trained using back-propagation algorithms.This method calculates the gradient of the error function with respect to all weights in the network. The gradient is fed to the optimisation method, which in turn uses it to update the weights in an attempt to minimise the error function. Convolutional neural networks use a special architecture that is particularly suitable for image classification. Using this architecture enables fast training of convolutional networks. In turn, this helps us train deep multi-layer networks that are very good at classifying images. Today, deep convolutional networks have been used for image recognition in most neural networks. Convolutional neural networks use three basic concepts: local receptive fields, shared weights, and pooling. The so-called local receptive field means that each neuron in the first (or any) hidden layer will be connected to a small part of the input (or upper layer) neuron. Weight sharing means that we will use the same weight and bias for each local acceptance domain.A few years ago, the training of deep learning networks took several weeks, but due to advances in GPU and algorithm enhancements, the training time has been reduced to several hours.This makes the learning of the early layers extremely slow, because the gradient not only propagates along the layer, but also propagates back over time. If the network runs for a long time, it may make the gradient extremely unstable, and it is difficult to learn from it.

2.1 Deep Belief Networks

DBN is a typical network architecture, but contains a novel training algorithm. DBN is a multi-layer network (usually deep, including many hidden layers), where each pair of connected layers is a restricted Boltzmann machine (RBM) ilustration can be seen in 1. In this way, DBN is represented as a stack of RBM. In DBN, the input layer represents the original sensory input, and each hidden layer learns an abstract representation of the input. The output layer, which is slightly different from the other layers, implements network classification. The training is divided into two steps: unsupervised pre-training and supervised fine-tuning.In unsupervised pre-training, each RBM is trained to reconstruct its input (for example, the first RBM reconstructs the input layer as the first hidden layer). Train the next RBM similarly, but treat the first hidden layer as the input (or visible) layer, and train the RBM by using the output of the first hidden layer as input. This process continues until each layer has been pre-trained. After pre-training is completed, fine-tuning begins. At this stage, labels will be applied to the output nodes to give them meaning (the meaning they represent in the network context). Then, complete network training is applied by using gradient descent learning or back propagation to complete the training process.

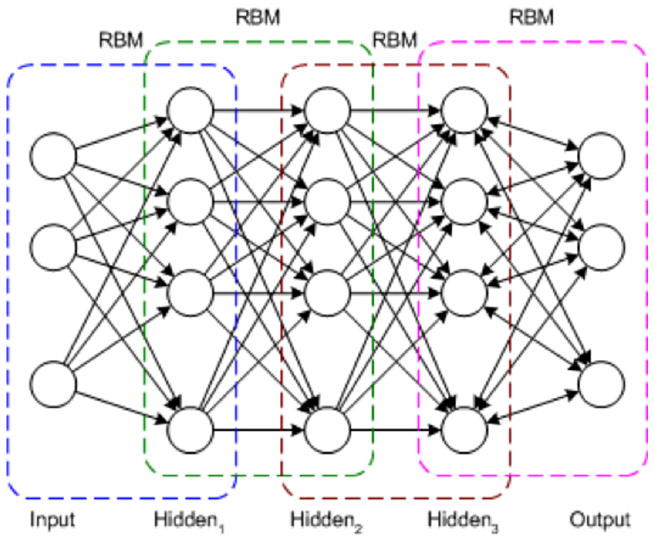

Figure 1: Deep belief networks architecture.

2.2 Convolutional Neural Network

Convolutional Neural Networks (CNNs) has establish as an adequate class of models for conception image content, segmentation, image recognition, detection and image retrieval. CNN perhaps competent for the assignment for motion indication in details of optical flow. CNN has highly concentrated on the recognition task in both videos and images. The extensive performance off CNN in video classification, here the network has approach to individual and static images, likewise their complex temporal evolution. Recent days CNN has access with the small image recognition issues in the data set namely CIFAR10/100, MNIST, Caltech 101/256 and NORB. The hardware has enable CNN to scale the networks for millions of parameters which led to significant development in segmentation, labeling, detection and image classification.But DL is still just a tool for CV, for example, the most commonly used neural network In CV it is CNN. But what is convolution is actually a widely used image Processing technology (for example, see Sobel Edge Detection). The advantages of DL are obvious, Reviewing the latest technology will be beyond the scope of this article. DL is Of course it is not a panacea for all problems, as we will see in the following This article has some problems and applications, among which the more conventional Algorithm is more suitable.Convolution is one of the basic components of CNN. The main purpose of the convolution operation is to extract features from the input image, such as edges, curves, corners, gradient directions, etc. We will learn about convolution operations through edge detection examples. Given an image, we want to extract all the horizontal and vertical edges in the image

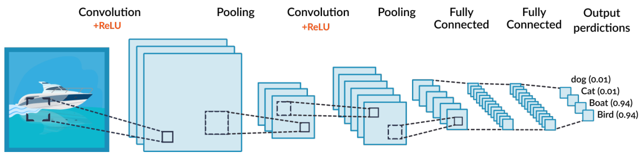

Figure 2: CNN Architecture

3 DEEP LEARNING FRAMEWORKS

The Deep learning framework provides building blocks for designing, training, and validating deep neural networks through high-level programming interfaces. Widely used deep learning frameworks (such as MXNet, PyTorch, TensorFlow, etc.) rely on GPU-accelerated libraries (such as cuDNN, NCCL and DALI) to provide high-performance multi-GPU accelerated training. Developers, researchers and data scientists can easily access NVIDIA's optimized deep learning framework container through deep learning examples, which have been adjusted and tested for performance on NVIDIA GPUs. This eliminates the need to manage packages and dependencies or build deep learning frameworks from source code.

3.1 Deep Learning Frameworks

- **PyTorch** is a Python package that provides two high-level features are Tensor computation (like numpy) with strong GPU acceleration and Deep Neural Networks built on a tape-based autograd system.PyTorch is a cousin of the Torch framework developed and used at Facebook. However, PyTorch is not a simple wrapper that supports popular languages, but has been rewritten and tailored to make it pleasant and original.Compared with competitors, PyTorch is relatively new, but it is developing rapidly. PyTorch also includes some easy-to-use implementations of popular computer vision architectures. In PyTorch, things have become more necessary and dynamic. It can define and change nodes without using special session interfaces or placeholders. Overall, the framework is tightly integrated with the Python language, and in most cases feels more native. When you write in TensorFlow, you sometimes feel that the model is behind a brick wall. In any case, it still sounds like a problem. PyTorch allows us to create complex deep learning architectures. There are a large number of ready-made modules in the torch.nn package. Please note how PyTorch uses an object-oriented approach to define the basic building blocks and allows us to move on, while providing the ability to extend functionality through subclasses.
- **TensorFlow** is an open source software library for numerical calculations using data flow graphs. The nodes in the graph represent mathematical operations, and the edges of the graph represent multidimensional data arrays (tensors) flowing between them. This flexible architecture allows you to deploy calculations to one or more CPUs or GPUs in desktops, servers or mobile devices without rewriting code. In order to visualize TensorFlow results, TensorFlow provides a TensorBoard visualization tool suite.For high-performance inference deployment of TensorFlow training models,Use

TensorFlow-TensorRT integration to optimize models in TensorFlow and deploy with TensorFlow.Export TensorFlow models and use the built-in TensorFlow model importer of NVIDIA TensorRT to import, optimize and deploy.TensorFlow is very powerful and mature deep learning library with strong visualization capabilities and several options to use for high-level model development. It has production-ready deployment options and support for mobile platforms.

- **MXNet** is a deep learning framework designed to improve efficiency and flexibility. It allows you to mix symbolic programming and imperative programming to maximize efficiency and productivity. The core of the dynamic dependency scheduler is to dynamically and automatically parallelize symbolic and imperative operations. The top graphics optimization layer makes symbol execution fast and memory efficient. The library is portable and lightweight, and can be extended to multiple GPUs and multiple machines.

3.2 Programming languages used for deep learning.

- **Python** - These days, with the development of Python in the field of deep learning and AI branches,developers around the world use Python. Python has a wide range of libraries and is widely used to implement algorithms. It also supports object-oriented and process-oriented programming. Most computer learning software can be run on Python.
- **C++** - It may be one of the oldest programming languages, but it is necessary to learn C++ for deep learning. The reason C++ can be used well for deep learning is that it is compatible with resource-intensive applications. Since C++ can be used for low-level and high-level programming, it provides developers with more control and efficiency.

4 DL ON MICRO COMPUTERS

5 CITATIONS

Some examples of references. A paginated journal article [2], an enumerated journal article [7], a reference to an entire issue [6], a monograph (whole book) [15], a monograph/whole book in a series (see 2a in spec. document) [13], a divisible-book such as an anthology or compilation [10] followed by the same example, however we only output the series if the volume number is given [9] (so Editor00a's series should NOT be present since it has no vol. no.), a chapter in a divisible book [17], a chapter in a divisible book in a series [8], a multi-volume work as book [14], an article in a proceedings (of a conference, symposium, workshop for example) (paginated proceedings article) [3], a proceedings article with all possible elements [16], an example of an enumerated proceedings article [11], an informally published work [12], a doctoral dissertation [5], a master's thesis [4], an finally two online documents or world wide web resources [1, 18].

REFERENCES

[1] Rafal Ablamowicz and Bertfried Fauser. 2007. *CLIFFORD: a Maple 11 Package for Clifford Algebra Computations, version 11.* Tennessee Technological University. Retrieved February 28, 2008 from http://math.tntech.edu/rafal/cliff11/index.html

[2] Patricia S. Abril and Robert Plant. 2007. The patent holder's dilemma: Buy, sell, or troll? *Commun. ACM* 50 (Jan. 2007), 36–44. https://doi.org/10.1145/1188913.1188915

[3] Sten Andler. 1979. Predicate Path expressions. In *Proceedings of the 6th. ACM SIGACT-SIGPLAN symposium on Principles of Programming Languages (POPL '79).* ACM Press, New York, NY, 226–236. https://doi.org/10.1145/567752.567774

[4] David A. Anisi. 2003. *Optimal Motion Control of a Ground Vehicle.* Master's thesis. Royal Institute of Technology (KTH), Stockholm, Sweden.

[5] Kenneth L. Clarkson. 1985. *Algorithms for Closest-Point Problems (Computational Geometry).* Ph.D. Dissertation. Stanford University, Palo Alto, CA. UMI Order Number: AAT 8506171.

[6] Jacques Cohen (Ed.). 1996. Special issue: Digital Libraries. *Commun. ACM* 39, 11 (Nov. 1996).

[7] Sarah Cohen, Werner Nutt, and Yehoshua Sagic. 2007. Deciding equivalances among conjunctive aggregate queries. *J. ACM* 54, 2, Article 5 (April 2007), 50 pages. https://doi.org/10.1145/1219092.1219093

[8] Bruce P. Douglass, David Harel, and Mark B. Trakhtenbrot. 1998. Statecarts in use: structured analysis and object-orientation. In *Lectures on Embedded Systems*, Grzegorz Rozenberg and Frits W. Vaandrager (Eds.). Lecture Notes in Computer Science, Vol. 1494. Springer-Verlag, London, 368–394. https://doi.org/10.1007/3-540-65193-4_29

[9] Ian Editor (Ed.). 2008. *The title of book two* (2nd. ed.). University of Chicago Press, Chicago, Chapter 100, 201–213. https://doi.org/10.1007/3-540-09237-4

[10] Peter Eston. 1993. *The title of the work* (3 ed.). 5, Vol. 4. The name of the publisher, The address of the publisher, Chapter 8, 201–213. https://doi.org/10.1007/3-540-09237-4 An optional note.

[11] Matthew Van Gundy, Davide Balzarotti, and Giovanni Vigna. 2007. Catch me, if you can: Evading network signatures with web-based polymorphic worms. In *Proceedings of the first USENIX workshop on Offensive Technologies (WOOT '07).* USENIX Association, Berkley, CA, Article 7, 9 pages.

[12] David Harel. 1978. *LOGICS of Programs: AXIOMATICS and DESCRIPTIVE POWER.* MIT Research Lab Technical Report TR-200. Massachusetts Institute of Technology, Cambridge, MA.

[13] David Harel. 1979. *First-Order Dynamic Logic.* Lecture Notes in Computer Science, Vol. 68. Springer-Verlag, New York, NY. https://doi.org/10.1007/3-540-09237-4

[14] Donald E. Knuth. 1997. *The Art of Computer Programming, Vol. 1: Fundamental Algorithms (3rd. ed.).* Addison Wesley Longman Publishing Co., Inc., USA.

[15] David Kosiur. 2001. *Understanding Policy-Based Networking* (2. ed.). Wiley, USA.

[16] Stan W. Smith. 2010. An experiment in bibliographic mark-up: Parsing metadata for XML export. In *Proceedings of the 3rd. annual workshop on Librarians and Computers (LAC '10)*, Reginald N. Smythe and Alexander Noble (Eds.), Vol. 3. Paparazzi Press, Milan Italy, 422–431. https://doi.org/10.1038/nphys1170

[17] Asad Z. Spector. 1990. Achieving application requirements. In *Distributed Systems* (2nd. ed.), Sape Mullender (Ed.). ACM Press, New York, NY, 19–33. https://doi.org/10.1145/90417.90738

[18] Harry Thornburg. 2001. *Introduction to Bayesian Statistics.* Stanford University. Retrieved March 2, 2005 from http://ccrma.stanford.edu/~jos/bayes/bayes.html