Uses of Complex Wavelets in Deep Convolutional Neural Networks

Fergal Cotter

Supervisor: Prof. Nick Kingsbury
Prof. Joan Lasenby

Department of Engineering
University of Cambridge

This dissertation is submitted for the degree of
PhD

Trinity College April 2019

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements.

Fergal Cotter
April 2019

Acknowledgements

I would like to thank my supervisor Nick Kingsbury who has dedicated so much of his time to help my research. He has not only been instructing and knowledgeable, but very kind and supportive. I would also like to thank my advisor, Joan Lasenby for supporting me in my first term when Nick was away, and for always being helpful. I must also acknowledge YiChen Yang and Ben Chaudhri who have done fantastic work helping me develop ideas and code for my research.

I sincerely thank Trinity College for both being my alma mater and for sponsoring me to do my research. Without their generosity I would not be here.

And finally, I would like to thank my girlfriend Cordelia, and my parents Bill and Mary-Rose for their ongoing support.

Abstract

Image understanding has long been a goal for computer vision. It has proved to be an exceptionally difficult task due to the large amounts of variability that are inherent to objects in scene. Recent advances in supervised learning methods, particularly convolutional neural networks (CNNs), have pushed the frontier of what we have been able to train computers to do.

Despite their successes, the mechanics of how these networks are able to recognize objects are little understood. Worse still is that we do not yet have methods or procedures that allow us to train these networks. The father of CNNs, Yann LeCun, summed it up as:

> There are certain recipes (for building CNNs) that work and certain recipes that don't, and we don't know why.

We believe that if we can build a well understood and well-defined network that mimicks CNN (i.e., it is able to extract the same features from an image, and able to combine these features to discriminate between classes of objects), then we will gain a huge amount of invaluable insight into what is required in these networks as well as what is learned.

In this paper we explore our attempts so far at trying to achieve this. In particular, we start by examining the previous work on Scatternets by Stephané Mallat. These are deep networks that involve successive convolutions with wavelets, a well understood and well-defined topic. We draw parallels between the wavelets that make up the Scatternet and the learned features of a CNN and clarify their differences. We then go on to build a two stage network that replaces the early layers of a CNN with a Scatternet and examine the progresses we have made with it.

Finally, we lay out our plan for improving this hybrid network, and how we believe we can take the Scatternet to deeper layers.

Table of contents

List of figures

List of tables

Chapter 1

Introduction

This work is stimulated by the intuition that wavelet decompositions, in particular complex wavelet transforms, are good building blocks for doing image recognition tasks. Their well understood and well defined behaviour as well as the similarities seen in learned networks, implies that there is potential gain for thinking about CNN layers in a new light.

To explore and test this intuition, we begin by looking at one of the most popular current uses of wavelets in image recognition tasks, in particular the Scattering Transform.

1.1 Series Expansions of Signals

Look at the intro to Vetterli's book. Want to make a statement about expanding signals in some form or another.

1.2 Contributions

The contributions and layout of this thesis are:

- **Software for wavelets and** DT$\mathbb{C}$WT **based ScatterNet (chapter 3)**
- **ScatterNet analysis and visualizations (chapter 4)**. Presented at MLSP2017, this chapter
- **Invariant Layer/Learnable ScatterNet (chapter 5)** Presenting at ICIP2019.
- **Learning convolutions in the wavelet domain (chapter 6)**.

1.2.1 Desirable Properties

Unlike CNNs introduced earlier which have little prior constraints (apart from the commonly used L_2 regularization), the scattering operator may be thought of as an operator S that

imposes structural priors on learning by extracting features with manually chosen, desirable properties. The extracted features can be used In classical paradigms of image understanding, it makes sense to add these priors, but it remains yet to be shown that these help learning.

limit variability these properties areview on these properties are manually chosen with the ultimate goal of aiding image understanding.

Chapter 2

A Faster ScatterNet

The drive of this thesis is in exploring if wavelet theory, in particular the DT$\mathbb{C}$WT, has any place in deep learning and if it does, quantifying how beneficial it can be. The introduction of more powerful GPUs and fast and popular deep learning frameworks such as PyTorch, Tensorflow and Caffe in the past few years has helped the field of deep learning grow very rapidly. Never before has it been so possible and so accessible to test new designs and ideas for a machine learning algorithm than today. Despite this rapid growth, there has been little interest in building wavelet analysis software in modern frameworks.

This poses a challenge and an opportunity. To pave the way for more detailed research (both by myself in the rest of this thesis, and by other researchers who want to explore wavelets applied to deep learning), we must have the right foundation and tools to facilitate research.

A good example of this is the current implementation of the ScatterNet. While ScatterNets have been the most promising start in using wavelets in a deep learning system, they are orders of magnitude slower and significantly more difficult to run than a standard convolutional network.

Additionally, any researchers wanting to explore the DWT in a deep learning system have to rewrite the filter bank implementation themselves, ensuring they correctly handle boundary conditions and ensure correct filter tap alignment to achieve perfect reconstruction.

This chapter describes how we have built a fast ScatterNet implementation in PyTorch with the DT$\mathbb{C}$WT as its core. At the core of that is an efficient implementation of the DWT. The result is an open source library that provides all three, available on GitHub as *PyTorch Wavelets* **pytorch_wavelets**.

In parallel with our efforts, the original authors of the ScatterNet have improved their implementation, also building it on PyTorch. My proposed DT$\mathbb{C}$WT ScatterNet is $15-35$x faster than their improved implementation, depending on the padding style and wavelet length, while using less memory.

2.1 The Design Constraints

The original authors implemented their ScatterNet in matlab **oyallon_deep_2015** using a Fourier based Morlet wavelet transform. The standard procedure for using ScatterNets in a deep learning framework up until recently has been to:

1. Pre scatter a dataset and store the features to disk. This can take several hours to several days depending on the size of the dataset and the number of CPU cores available.
2. Build a network in another framework, usually Tensorflow **abadi_tensorflow:_2015** or Pytorch **paszke_automatic_2017**.
3. Load the scattered data from disk and train on it.

We saw that this approach was suboptimal for a number of reasons:

- It is slow and must run on CPUs.
- It is inflexible to any changes you wanted to investigate in the Scattering design; you would have to re-scatter all the data and save elsewhere on disk.
- You can not easily do preprocessing techniques like random shifts and flips, as each of these would change the scattered data.
- The scattered features are often larger than the original images, and require you to store entire datasets twice (or more) times.
- The features are fixed and can only be used as a front end to any deep learning system.

To address these shortcomings, all of the above limitations become design constraints. In particular, the new software should be:

- Able to run on GPUs (ideally on multiple GPUs in parallel).
- Flexible and fast so that it can run as part of the forward pass of a neural network (allowing preprocessing techniques like random shifts and flips).
- Able to pass gradients through, so that it could be part of a larger network and have learning stages before scattering.

To achieve all of these goals, we choose to build our software on PyTorch, a popular open source deep learning framework that can do many operations on GPUs with native support for automatic differentiation. PyTorch uses the CUDA and cuDNN libraries for its GPU-accelerated primitives. Its popularity is of key importance, as it means users can build complex networks involving ScatterNets without having to use or learn extra software.