Columbia University in the City of New York

Neural Networks and Deep Learning Project

Spectral Representation for Convolutional Networks

Maxime He, Victor Le Breton, Julien Michel, Michael Weil

May 8, 2016

Contents

1 Introduction

The article exposed by Rippel, Snoek & Adams is about a new prospect of using the discrete Fourier Transform (DFT) for Convolutional Neural Networks (CNNs). Rather than using the frequency domain for faster computations of convolutions, this paper exposed two new techniques : Spectral pooling and Spectral parametrization of convolutional filters.
Spectral pooling is a dimensionality reduction technique by cropping the frequency domain of the input. The idea of Spectral parametrization is to learn the filters of CNN in the frequency domain.
This project is a review of this article. First we will discuss about Fourier's theory to better understand the prospect of this paper. Then we will discuss about the two new techniques exposed before running our own simulation and considering eventual improvements.

2 Fourier

In this section, we set the theory about Fourier necessary for the understanding of the article. Under distribution theory, tempered distributions is the correct domain of the Fourier transform for which the tempered distributions set is stable. All Fourier transforms are trivial sub-cases of the general transform. Hence, all proofs of this sections are trivial corollaries of the general Fourier theory.

It is easy to see that under the definition of the $\mathcal{F}$ symbol used in the article, that is, the normalized DFT, the transformation is a unitary transformation $\mathcal{F}$ from one vector space to itself, hence we have Parseval formula (2).

However, formula (1) in the article is **false**, **wrong**, since it lacks the a coefficient. The formula is only true as is with the normalized version of DFT. But with a good deep learning computation framework which supports complex gradients and smart gradient between real values and complex values, there is **NO** need to even think about formula (1). In our case though, Theano **does not** support complex gradients and in order to make the computations, we have to derive and implement smart gradient flows, playing through numpy and theano intertwined computations.

Moreover, formula (3) (hermitian symmetry for real signals) can be reformulated under the periodisation of the signal as a **trivial and more human interpretable** formula : $\mathcal{F}(x)(-(m,n)) = \overline{\mathcal{F}(x)((m,n))}$

3 Spectral Pooling

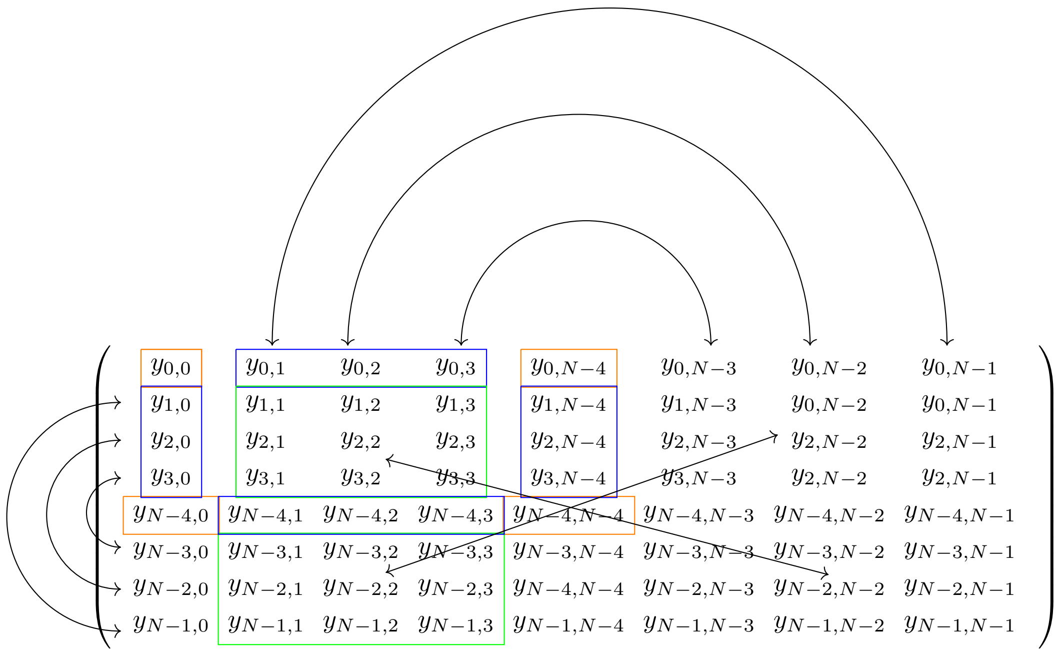
The article introduces a new way of pooling by reducing the dimension of the spectral representation. Given an input $\mathbf{x} \in \mathbb{R}^{N\times M}$ and its spectral representation $\mathbf{y} = \mathcal{F}(\mathbf{x}) \in \mathbb{C}^{N\times M}$, we crop $\mathbf{y}$ to $\hat{\mathbf{y}} \in \mathbb{C}^{H\times W}$ by removing the lowest frequencies. To perform the cropping, the DC components is supposed to to be at the center of $\mathbf{y}$, then we take the extracted matrix around the DC value. By the conjugate symmetry property, and as $y_{0,0}$ is the DC component, to crop $\mathbf{y}$ is equivalent to take the corners of the matrix.

$$\begin{pmatrix} y_{0,0} & y_{0,1} & \cdots & y_{0,N-2} & y_{0,N-1} \\ y_{1,0} & y_{1,1} & \cdots & y_{1,N-2} & y_{1,N-1} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ y_{M-2,0} & y_{M-2,1} & \cdots & y_{M-2,N-2} & y_{M-2,N-1} \\ y_{M-1,0} & y_{M-1,1} & \cdots & y_{M-1,N-2} & y_{M-1,N-1} \end{pmatrix}$$

$$\hat{\mathbf{y}} = \begin{pmatrix} y_{0,0} & y_{0,1} & y_{0,N-2} & y_{0,N-1} \\ y_{1,0} & y_{1,1} & y_{1,N-2} & y_{1,N-1} \\ y_{M-2,0} & y_{M-2,1} & y_{M-2,N-2} & y_{M-2,N-1} \\ y_{M-1,0} & y_{M-1,1} & y_{M-1,N-2} & y_{M-1,N-1} \end{pmatrix}$$

But why not pooling the image not based on the frequencies but based on the energies? Instead of selecting the frequencies around the DC component we can think of taking the $H \times W$ values of $\mathbf{y}$ with the highest module. However, $\hat{\mathbf{y}}$ will not necessary respect the property of conjugate symmetry.
This property is crucial in order to reconstruct a real cropped input $\hat{\mathbf{x}} = \mathcal{F}^{-1}(\hat{\mathbf{y}}) \in \mathbb{R}^{H\times W}$. Therefore we need to check for each case of the cropping (regarding the parity of M, N, H and W) if the conjugate symmetry is verified. For simplicity, we will focus on the case of square matrices : $M = N$ and $H = W$.

- N and H even
  The matrix $\mathbf{y}$ will look like this :

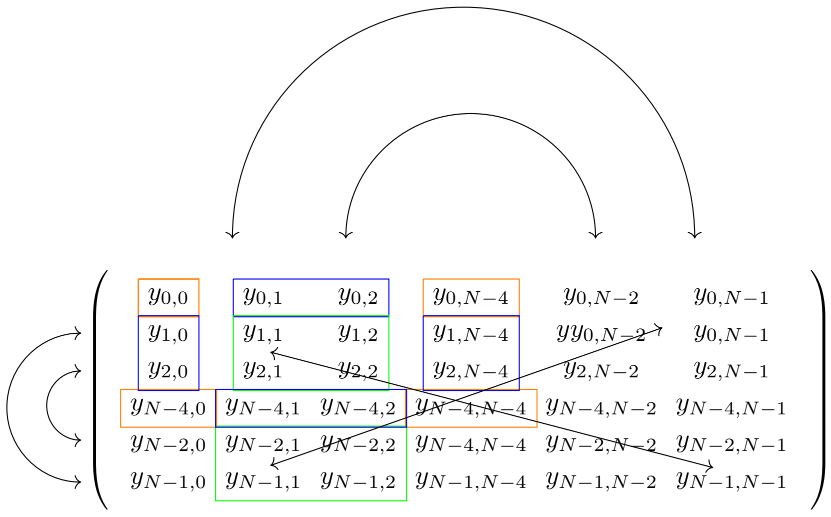


$y_{0,0}$ $y_{0,1}$ $y_{0,2}$ $y_{0,3}$ $y_{0,N-4}$ $y_{0,N-3}$ $y_{0,N-2}$ $y_{0,N-1}$
$y_{1,0}$ $y_{1,1}$ $y_{1,2}$ $y_{1,3}$ $y_{1,N-4}$ $y_{1,N-3}$ $y_{0,N-2}$ $y_{0,N-1}$
$y_{2,0}$ $y_{2,1}$ $y_{2,2}$ $y_{2,3}$ $y_{2,N-4}$ $y_{2,N-3}$ $y_{2,N-2}$ $y_{2,N-1}$
$y_{3,0}$ $y_{3,1}$ $y_{3,2}$ $y_{3,3}$ $y_{3,N-4}$ $y_{3,N-3}$ $y_{2,N-2}$ $y_{2,N-1}$
$y_{N-4,0}$ $y_{N-4,1}$ $y_{N-4,2}$ $y_{N-4,3}$ $y_{N-4,N-4}$ $y_{N-4,N-3}$ $y_{N-4,N-2}$ $y_{N-4,N-1}$
$y_{N-3,0}$ $y_{N-3,1}$ $y_{N-3,2}$ $y_{N-3,3}$ $y_{N-3,N-4}$ $y_{N-3,N-3}$ $y_{N-3,N-2}$ $y_{N-3,N-1}$
$y_{N-2,0}$ $y_{N-2,1}$ $y_{N-2,2}$ $y_{N-2,3}$ $y_{N-4,N-4}$ $y_{N-2,N-3}$ $y_{N-2,N-2}$ $y_{N-2,N-1}$
$y_{N-1,0}$ $y_{N-1,1}$ $y_{N-1,2}$ $y_{N-1,3}$ $y_{N-1,N-4}$ $y_{N-1,N-3}$ $y_{N-1,N-2}$ $y_{N-1,N-1}$

An orange square means that the value is in $\mathbb{R}$. An arrow means that the two entries are conjugated. A blue rectangle is for an axial conjugate symmetry, a green rectangle is for central conjugate symmetry around the center of the matrix.

First, let assume that we want to remove 2 lines and 2 columns ($H = N - 2$). To do so we remove the lines and columns 3 and $N - 3$.

The resulting matrix $\hat{\mathbf{y}}$ is

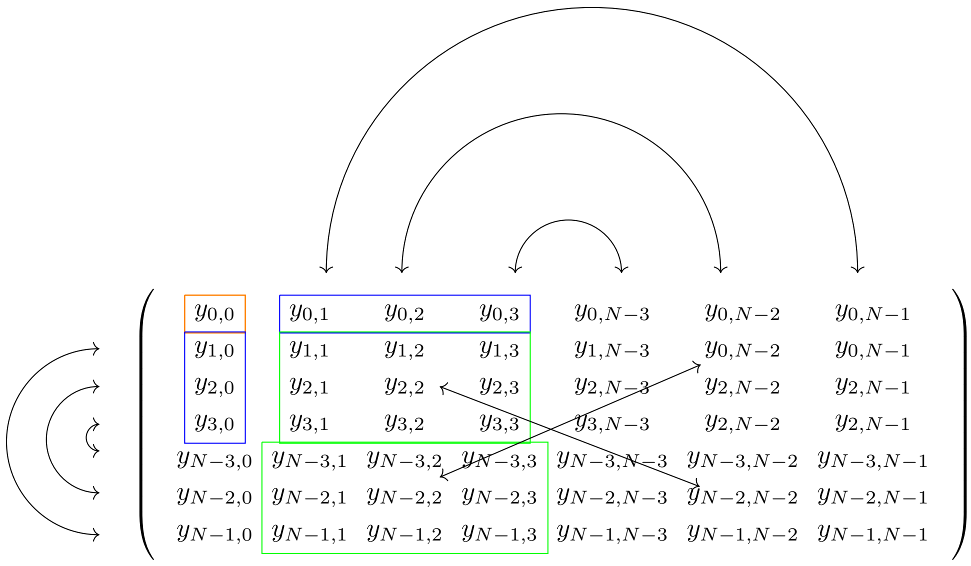


$$\begin{pmatrix} y_{0,0} & y_{0,1} & y_{0,2} & y_{0,N-4} & y_{0,N-2} & y_{0,N-1} \\ y_{1,0} & y_{1,1} & y_{1,2} & y_{1,N-4} & yy_{0,N-2} & y_{0,N-1} \\ y_{2,0} & y_{2,1} & y_{2,2} & y_{2,N-4} & y_{2,N-2} & y_{2,N-1} \\ y_{N-4,0} & y_{N-4,1} & y_{N-4,2} & y_{N-4,N-4} & y_{N-4,N-2} & y_{N-4,N-1} \\ y_{N-2,0} & y_{N-2,1} & y_{N-2,2} & y_{N-4,N-4} & y_{N-2,N-2} & y_{N-2,N-1} \\ y_{N-1,0} & y_{N-1,1} & y_{N-1,2} & y_{N-1,N-4} & y_{N-1,N-2} & y_{N-1,N-1} \end{pmatrix}$$

$\hat{\mathbf{y}}$ is still conjugate symmetric. In general, for H so that $k = \frac{N-H}{2}$, we remove the lines and columns $(\frac{N}{2} - 1, \frac{N}{2} + 1), (\frac{N}{2} - 2, \frac{N}{2} + 2), ..., (\frac{N}{2} - k, \frac{N}{2} + k)$.

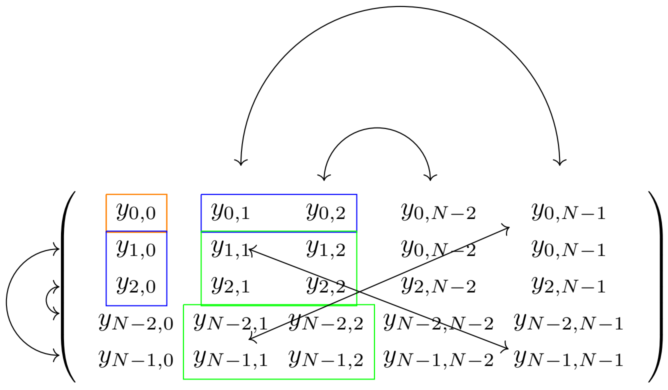
- N and H odd

The matrix $\mathbf{y}$ will look like this :



$$\begin{pmatrix} y_{0,0} & y_{0,1} & y_{0,2} & y_{0,3} & y_{0,N-3} & y_{0,N-2} & y_{0,N-1} \\ y_{1,0} & y_{1,1} & y_{1,2} & y_{1,3} & y_{1,N-3} & y_{0,N-2} & y_{0,N-1} \\ y_{2,0} & y_{2,1} & y_{2,2} & y_{2,3} & y_{2,N-3} & y_{2,N-2} & y_{2,N-1} \\ y_{3,0} & y_{3,1} & y_{3,2} & y_{3,3} & y_{3,N-3} & y_{2,N-2} & y_{2,N-1} \\ y_{N-3,0} & y_{N-3,1} & y_{N-3,2} & y_{N-3,3} & y_{N-3,N-3} & y_{N-3,N-2} & y_{N-3,N-1} \\ y_{N-2,0} & y_{N-2,1} & y_{N-2,2} & y_{N-2,3} & y_{N-2,N-3} & y_{N-2,N-2} & y_{N-2,N-1} \\ y_{N-1,0} & y_{N-1,1} & y_{N-1,2} & y_{N-1,3} & y_{N-1,N-3} & y_{N-1,N-2} & y_{N-1,N-1} \end{pmatrix}$$

If we remove 2 lines and columns, by removing $(3, N-3)$ we get $\hat{\mathbf{y}}$ as follows :



$$\begin{pmatrix} y_{0,0} & y_{0,1} & y_{0,2} & y_{0,N-2} & y_{0,N-1} \\ y_{1,0} & y_{1,1} & y_{1,2} & y_{0,N-2} & y_{0,N-1} \\ y_{2,0} & y_{2,1} & y_{2,2} & y_{2,N-2} & y_{2,N-1} \\ y_{N-2,0} & y_{N-2,1} & y_{N-2,2} & y_{N-2,N-2} & y_{N-2,N-1} \\ y_{N-1,0} & y_{N-1,1} & y_{N-1,2} & y_{N-1,N-2} & y_{N-1,N-1} \end{pmatrix}$$

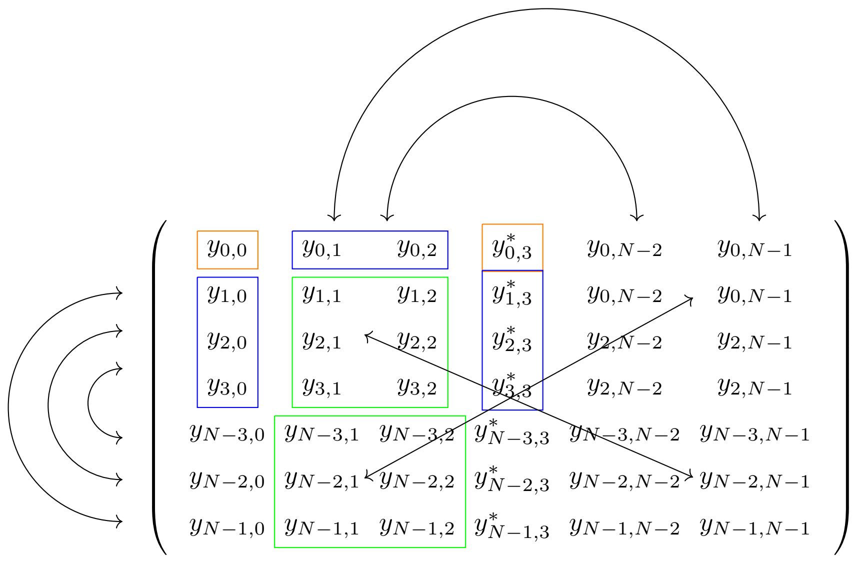
$\hat{\mathbf{y}}$ is still conjugate symmetric. In general, for H so that $k = \frac{N-H}{2}$, we remove the lines and columns $(\frac{N-1}{2}, \frac{N-1}{2}+1), (\frac{N-2}{2}-1, \frac{N}{2}+2), ..., (\frac{N-1}{2}-k+1, \frac{N-1}{2}+k)$.

- N even and H odd

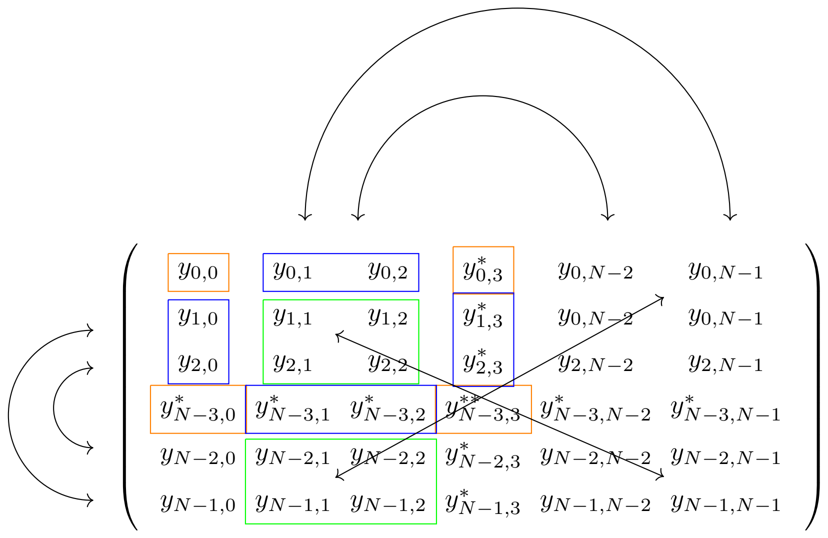
  We have an odd number of lines and columns to remove in $\mathbf{y}$. To crop the matrix , we remove the central line and column (index $\frac{N}{2}$). It will give an odd matrix that will still respect the conjugate symmetry property. This boils down to the case of N odd and H odd.

- N odd and H even

  This part is a little bit tricky. Assume we want to remove one line and one column. By removing the column $\frac{N+1}{2}$ and by adding this column to the column $\frac{N-1}{2}$,we get :



with $y^*_{i,\frac{N-1}{2}} = y_{i,\frac{N-1}{2}} + y_{i,\frac{N+1}{2}}$. We notice that $y^*_{0,\frac{N-1}{2}} \in \mathbb{R}$.
Now, by removing the line $\frac{N-1}{2}$ and by adding this to the line $\frac{N+1}{2}$,we obtain the matrix :



with $y^*_{\frac{N+1}{2},j} = y_{\frac{N+1}{2},j} + y_{\frac{N-1}{2},j}$. We notice that $y^*_{0,\frac{N+1}{2}} \in \mathbb{R}$. Besides,

$$\begin{aligned} y^{**}_{\frac{N+1}{2},\frac{N-1}{2}} &= y^*_{\frac{N+1}{2},\frac{N-1}{2}} + y^*_{\frac{N-1}{2},\frac{N-1}{2}} \\ &= y_{\frac{N+1}{2},\frac{N-1}{2}} + y_{\frac{N+1}{2},\frac{N+1}{2}} + y_{\frac{N-1}{2},\frac{N-1}{2}} + y_{\frac{N-1}{2},\frac{N+1}{2}} \\ &= y_{\frac{N+1}{2},\frac{N-1}{2}} + y_{\frac{N+1}{2},\frac{N+1}{2}} + \bar{y}_{\frac{N+1}{2},\frac{N+1}{2}} + \bar{y}_{\frac{N+1}{2},\frac{N-1}{2}} \\ &= 2\mathrm{Re}(y_{\frac{N+1}{2},\frac{N-1}{2}} + y_{\frac{N+1}{2},\frac{N+1}{2}}) \in \mathbb{R} \end{aligned}$$

Therefore this matrix is conjugate symmetric. In general, for H so that $k = N - H$, we do the process above k times. The reduced input $\hat{\mathbf{y}}$ will not take exact information from $\mathbf{y}$ as are performing additions of lines and columns.

Computing the gradient is pretty straightforward. Indeed the cropping can be seen as matrix multiplications. For instance, if we want to remove $k = N - H$ dimensions :

$$\hat{\mathbf{y}} = S^T \mathbf{y} S$$

where $S = \begin{pmatrix} 1 & 0 & 0 & ... & 0 & 0 & 0 \\ 0 & 1 & 0 & ... & 0 & 0 & 0 \\ 0 & 0 & 1 & ... & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & ... & 1 & 0 & 0 \\ 0 & 0 & 0 & ... & 0 & 1 & 0 \\ 0 & 0 & 0 & ... & 0 & 0 & 1 \end{pmatrix} \in \mathbb{R}^{N \times H}$

For the case N odd and H even, the line/column addition can be made by using shear matrices.

Spectral pooling preserves much more information than max-pooling as we can choose are output dimensionality. Truncating the frequency around the DC components has the effect of cutting low energies, thus, by Parseval's theorem, the l_2 reconstruction distortion will not be dramatically diminished. However, as noticed above the information of the output $\hat{\mathbf{x}}$ will be a little bit altered for the case N odd and H even.

4 Adam

Adam is the optimization algorithm used in this paper and has the good property (to make the article interesting) of being axis racist. Adam has also on top of it an essential property that is it can be performed axis by axis, completely ignoring the other axis. Then, it can be easily split into multiple non connected sub graphs which is crucial as we will see in the CNN section.

5 CNN

The challenge here was to extend the code of the convolutional layer in order to include a under the hood spectral parametrization of the layer's filters. Each Convolutional layer in our implementation hence have a real filter tensor and complex spectral filter tensor.

In the article, it seems that the filters' tensors stored have same dimensions as the size of the kernel. However, the **true** paradigm under which the formula (1) (unused by the article most probably) can be used to compute as is the convolutional layers is the one where convolutions are zero padded such as to emulate a true circular convolution and where filters have size $image_{shape} + kernel_{size} - 1$. In order to respect Theano's implementation, we have to have a kernel size tensor to represent filters though (the rest, in the $image_{shape} + kernel_{size} - 1$ size representation are zero paddings). But it is of utmost importance to have the right size ($image_{shape} + kernel_{size} - 1$) for the spectral representation since zero paddings in real domain doesn't give zero values at all in the spectral domain !

Theano doesn't support any form of complex gradients anyway and all our implementations had to be a smart interleaving of manual numpy information transmission between theano's different split graphs wherever complex gradient flows are needed. This need was even accentuated with the fact that Theano has no fft operations as of date.

In architecture (6) and (7), only the end nodes filters (in the gradient backdrop perspective) have to be such manually linked through numpy. However, architecture (5) needs way more linkage and manual gradient flow through numpy since theano can only give automatic differentiation up to the SP layer. The rest, has to be done by hands (numpy).

There is a slight subtlety in the manual gradient flow in the case of (5) since, with the right implementation paradigm for convolutions (which is not the case most probably in the article), there is no need to compute the real filter gradient and we can directly use formula (1) to compute the complex filter update gradient using the flowed through SP layer gradient.

Manual gradient flow makes the Adam updates split. However, since Adam has the good property of being axis splitable (no division by any norm or anything requiring information across all axis is needed ever), the hand crafted computations are equivalent as if Theano did support complex gradients and ffts.

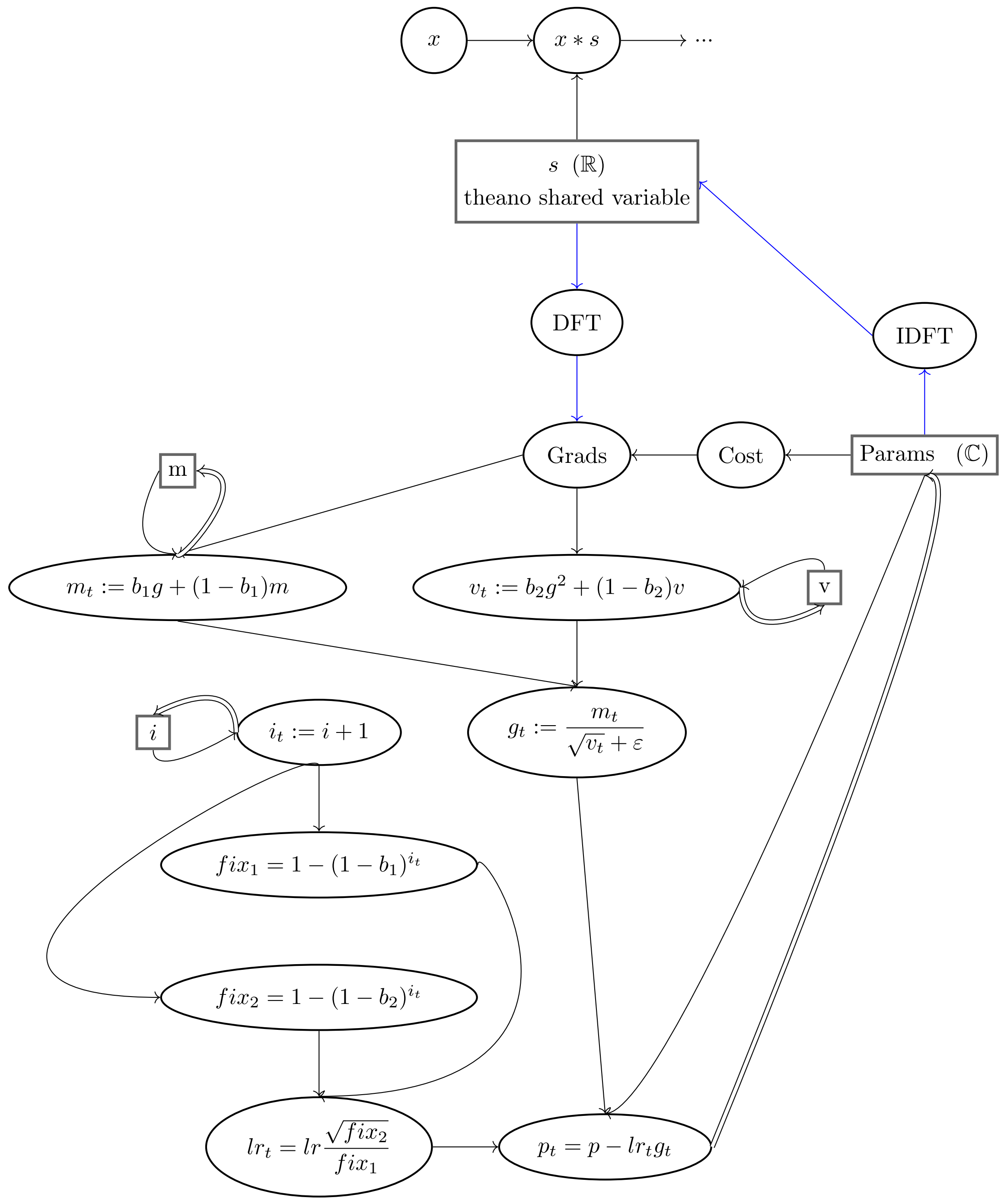
6 Implementation

There are 4 files which extend the homework 3 structure, at least keeps the global structure idea of the code :

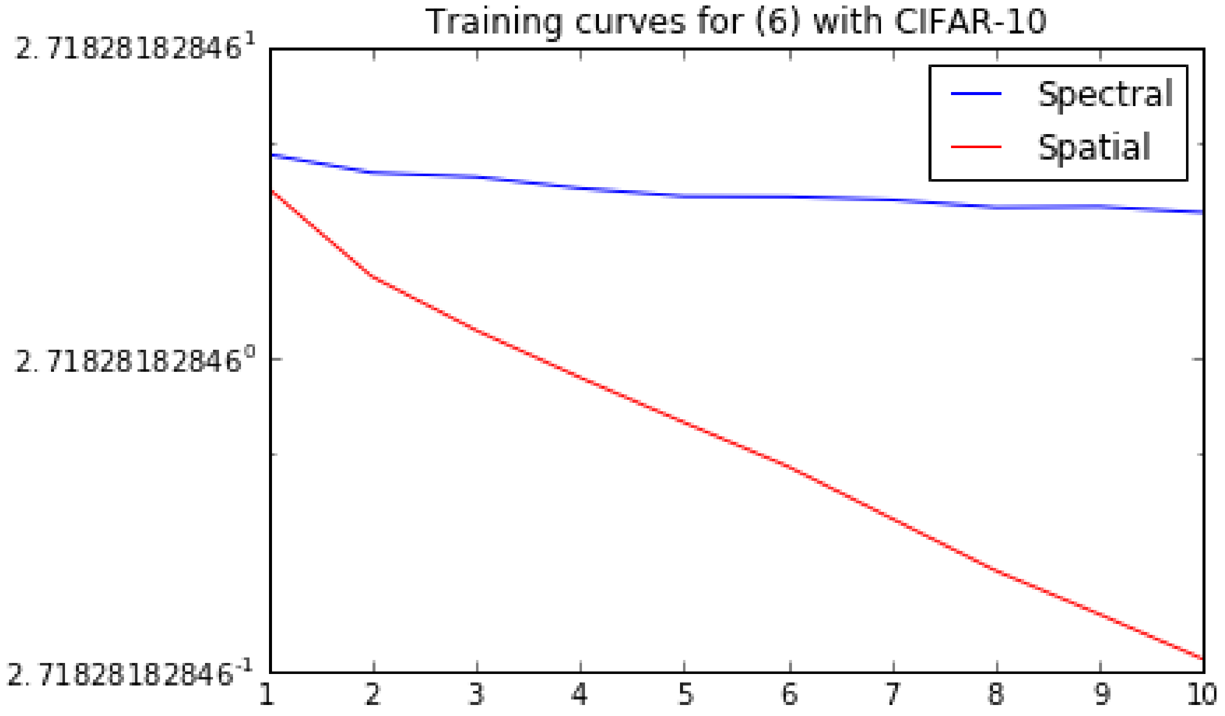
- project_utils.py → loads CIFAR 10 and CIFAR 100 data sets
- project_nn.py → contains layer classes and a variety of functions
- networks.py → contains the architectures to run
- project_info_loss.py → contains the information loss comparison MP/SP

6.1 CNN implementation with spectral parametrization

Below we represent the graph on which we based the implementation of a CNN layer using spectral parametrization.
The ellipsoidal nodes represent theano computation graph nodes.
The rectangle nodes represent theano shared variables.
The blue arrows represent transfers and computations done outside of theano (in numpy). We had to use this trick because theano does not support gradients over complex variables and hence cannot perform a FFT.
The big arrows represent updates of shared variables (with set_value())


x
$x * s$
$\cdots$
s $(\mathbb{R})$
theano shared variable
DFT
IDFT
Grads
Cost
Params $(\mathbb{C})$
m
$m_t := b_1 g + (1 - b_1) m$
$v_t := b_2 g^2 + (1 - b_2) v$
v
i
$i_t := i + 1$
$g_t := \frac{m_t}{\sqrt{v_t} + \varepsilon}$
$fix_1 = 1 - (1 - b_1)^{i_t}$
$fix_2 = 1 - (1 - b_2)^{i_t}$
$lr_t = lr \frac{\sqrt{fix_2}}{fix_1}$
$p_t = p - lr_t g_t$

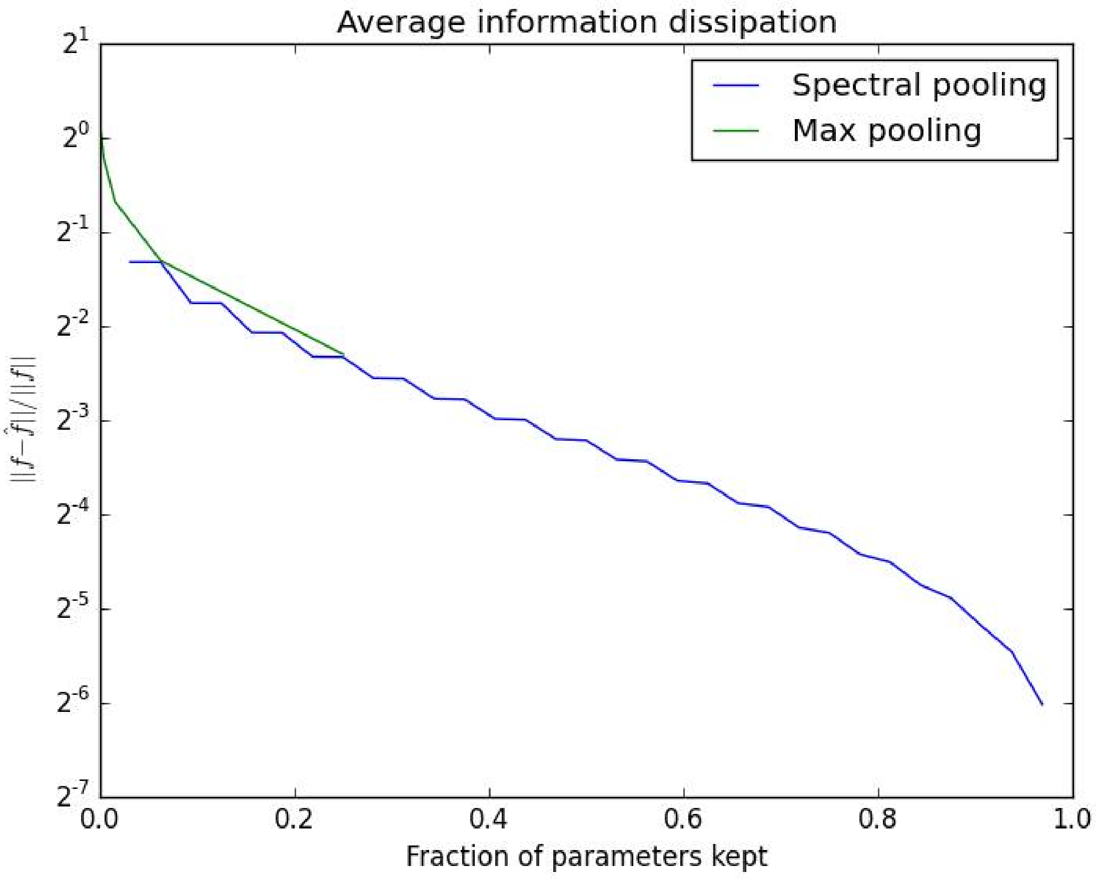
However running the simulations of Figure 5 of the article gives unexpected results. For instance, the architecture (6) applied to CIFAR-10 for 10 epochs gives these training curves :



The spatial CNN performs much better than the new CNN. We could not run with 200 epochs because of a lack of time.

6.2 Information dissipation

Average information dissipation for the CIFAR10 validation set as a function of gamma: the fraction of parameters kept measured in Frobenius norm.



We observe a graph that is similar to Figure (a) in section 4.1. of the article

7 Improvements - CCL

7.1 Convolutional filter parametrization

Using a particular new parametrizzation for the filters other than the natural one corresponds to having new (in this case orthonormal) base on which the minimization of the negative likelihood (cost) is made. This idea would be **worthless** if we were using a non axis adaptive technique such as pure SGD. However, with the use of axis adaptive acceleration such as the one used in the paper with Adam, axis of parametrizzation over the space of exploration plays a role.

Fourier basis was the choice made here. However a better idea would be maybe, to let a train be done in the natural base and analyze closely the dynamic and convergence point in the parameter space and see what base it seems to follow. That is, project the parameter vector onto a candidate base and look at the coefficients distribution. For instance, if the coefficients tend to be highly concentrated, we can think that the base has some adequation with the problem. Maybe some crazy things can happen, the dynamic changing the clinging space over epochs, a mixture of bases, etc...

Fourier is merely a first natural guess for another base. However, there are tons when we start looking toward wavelets and other transformations which have bases under the hood. Moreover, some work can also be done with over complete bases.

7.2 Pooling layers

The idea here in this paper was to reduce dimensions under a specific other base which tend (empirically) to concentrate on some vectors of the base. Indeed, it has been seen empirically that the energy of images tend to concentrate on the border of the fourier transform matrix unshifted values. Hence a natural urge to take out the low energy values to reduce dimensions.

Taking out values in the constrained pattern exposed in the paper have the nice property to keep a fourier invertible matrix back to a real image. However, there may be other patterns of taking out values respecting such a property. And since, the "image" imported back to the real domain has messed up with the frequencies, there is absolutely no need to keep any human image interpretability property. Only math hermitian symmetry has to be respected with the surgically reduced spectral matrix.

However, let us think about the fact that taking out all the $N^2 - H^2$ minimum values makes it hard to repsect hermitian symmetry and thus seems not usable.