Invariant Scattering Convolution Networks

Interpretation of, and Excerpts from, the paper:

http://arxiv.org/abs/1203.1513 by Joan Bruna, Stephane Mallat

Need for Invariant Representations

Various applications might benefit with representations of Input Images that are invariant to translations and deformations, at the same time. Eg.

-Digit Classification

-Texture Recognition

Towards Scattering Wavelet Transform

- Fourier is stable to translation but not deformation

- Wavelet is stable to deformation but not translation

- Wavelet transform may be cascaded by modulus, without causing information loss, due to the inherent redundancy in Wavelet transform.

- Integrating/Averaging the representation thus leads to something that possesses the desired properties of translation invariance along with deformation invariance.

-However Averaging cuts down the higher frequency components.

Towards Scattering Wavelet Transform

- Higher frequency components may be recovered if the result prior to averaging is once again transformed.

- Thus Image Data is transformed by a chain/path of wavelets and at each stage invariant information is extracted via the Averaging operator.

- The span of the averaging operator decides the scale up to which invariance is obtained, for complete invariance, the complete span may need to be covered, which may be equivalent to Full Integration. However, complete invariance is not required, and probably not even desired.

- The larger the scale of the invariance, and the larger the window of averaging, the more the loss of higher frequency components, with only the DC component being obtained at the first stage/layer, in case of full Integration.

- The larger the scale of the invariance, the more the number of stages needed to adequately represent the Higher frequency information in the Image. Since the energy progressively decreases, it may be reasonable to truncate at a suitable number of stages/layers “m”.

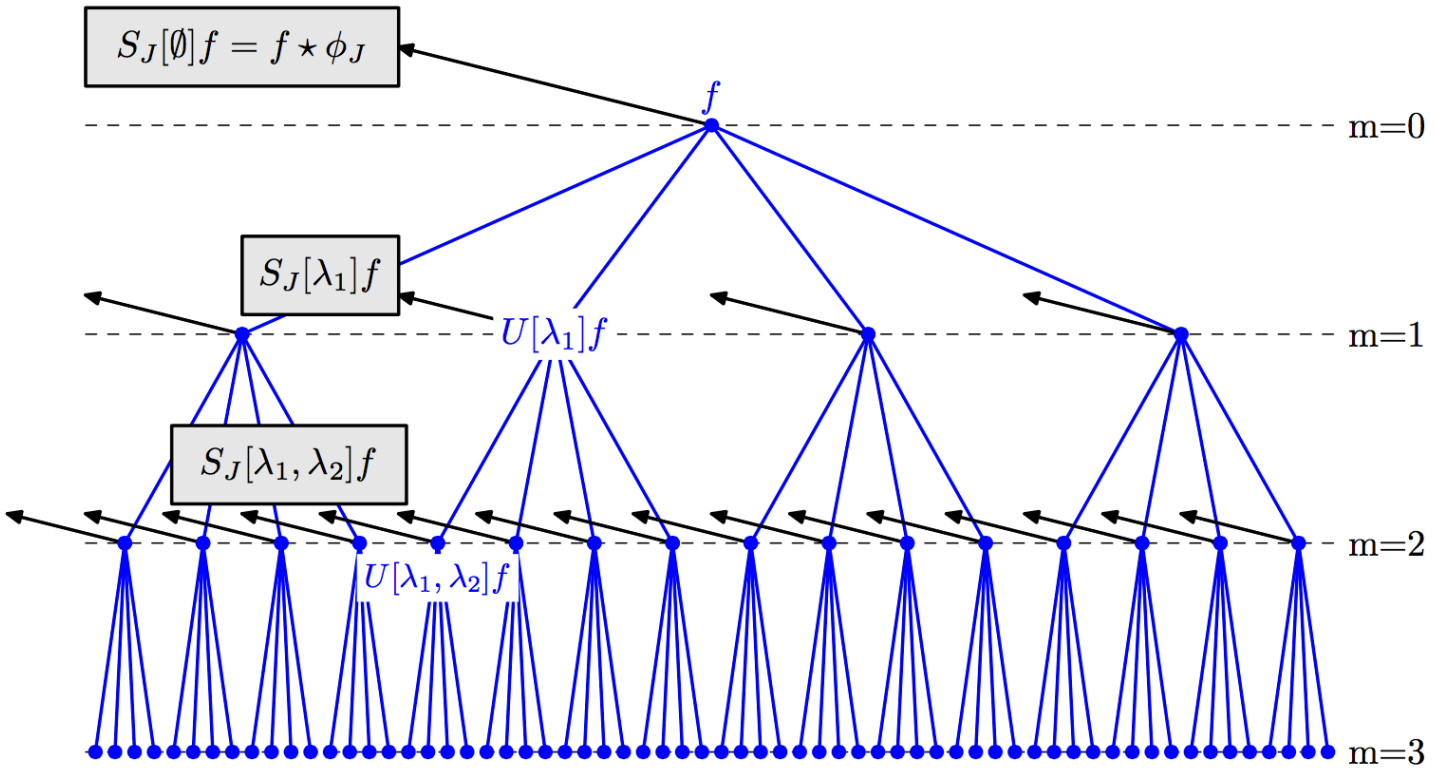

Fig. 2. A scattering propagator U_J applied to x computes each $U[\lambda_1]x = |x \star \psi_{\lambda_1}|$ and outputs $S_J[\emptyset]x = x \star \phi_{2^J}$ (black arrow). Applying U_J to each $U[\lambda_1]x$ computes all $U[\lambda_1, \lambda_2]x$ and outputs $S_J[\lambda_1] = U[\lambda_1] \star \phi_{2^J}$ (black arrows). Applying U_J iteratively to each $U[p]x$ outputs $S_J[p]x = U[p]x \star \phi_{2^J}$ (black arrows) and computes the next path layer.

Comparison to Deep Convolution Network

- Outputs are obtained at every layer, instead of just the last layer.

- Filters are predefined Wavelets, instead of being learned from data.

The operation at each stage is given as:

$$U_J x(u) = \left\{ x \star \phi_{2^J}(u) \ , \ |x \star \psi_\lambda(u)| \right\}_{\lambda \in \Lambda_J}$$

Towards Reduced Scattering Transform

- In the basic Scattering Transform, the set of outputs at every layer, are observed to be correlated.

- This motivates the use of KL basis to decorrelate the outputs, and preserve most of the information in a smaller amount of data. Thus instead of U_J we use:

$$U_{k,J}x = \left\{ x \star \phi_J(2^J \alpha n) \ , \ |x \star \psi_{2^j r}(2^j \alpha n)| \right\}_{-J<j\leq k, r\in G^+}$$

- This leads to the Reduced Scattering Transform, which is similar to the implementation of the basic Scattering Transform network, just with use of the above defined $U_{k,J}$ operation instead of U_J

Algorithm: Reduced Scattering Transform

Compute $U_{0,J}(x)$
Output $x \star \phi_{2^J}(2^J \alpha n)$
for $m = 1$ to $m_{\max} - 1$ **do**
  for all $0 \geq j_1 > ... > j_m > -J$ **do**
    for all $(r_1, ..., r_q) \in G^{+m}$ **do**
      if $m = m_{\max} - 1$ **then**
        Compute $|||x \star \psi_{2^{j_1} r_1}| \star ... | \star \psi_{2^{j_m} r_m}| \star \phi_{2^J}(2^J \alpha n)$
      else
        Compute $U_{j_m,J}(|||x \star \psi_{2^{j_1} r_1}| \star ...| \star \psi_{2^{j_m} r_m}|)$
      end if
      Output $|||x \star \psi_{j_1,\gamma_1}| \star ...| \star \psi_{j_q,\gamma_q}| \star \phi_J(2^J \alpha n)$
    end for
  end for
end for

Configurable Params

- filt_opt contains filters-related options
- filt_opt.J (default 4) is the number of scale j in the filter bank $\psi_{j,\theta}$. Note that increasing J will also increase the range of translation invariance.
- filt_opt.L (default 8) is the number of orientations. Note that increasing L will also increase the angular selectivity of filters.
- filt_opt.Q (default 1) is the number of scale per octave. Not that increasing Q will decrease the range of translation invariance and increase the scale selectivity of filters.

- scat_opt contains scattering-related options
- scat_opt.M (default 2) is the maximum scattering order. The scat function will compute scattering coefficient of order 0 to M, that is $S_0x,\ldots,S_Mx$
- scat_opt.oversampling (default 1) : scattering will be oversampled by up to a power of 2.