Scattering Transform for Art Investigation

Yang Wang

Department of Mathematics
Hong Kong University of Science and Technology

Joint Work with Roberto Leonarduzzi and Haixia Liu

Celebrating the 80th Birthday of John Benedetto
Scattering Transform for Art Investigation

Yang B. Wang

Department of Mathematics
Hong Kong University of Science and Technology

Joint Work with Roberto Leonarduzzi and Haixia Liu

Celebrating the 80th Birthday of John Benedetto
Content

Deep Neural Networks

Scattering Transform

Art Authentication

Neural style transfer
Contents

Deep Neural Networks

Scattering Transform

Art Authentication

Neural style transfer
Linear Classification Problem

Notation

- **Feature vector**: $f \in \mathbb{R}^p$
- **Output**: $y \in \{1, 2\}$
- **Training data**: $\mathcal{T} = \{f_i, y_i = y(f_i)\}_{i=1,...,N}$
Linear Classification Problem

Notation

- **Feature vector**: $f \in \mathbb{R}^p$
- **Output**: $y \in \{1, 2\}$
- **Training data**: $\mathcal{T} = \{f_i, y_i = y(f_i)\}_{i=1,...,N}$

General Form

$$\hat{y}(f) = \begin{cases} 
1 & \text{if } \hat{w}^T f > \hat{t} \\
2 & \text{otherwise,}
\end{cases}$$
Linear Classification Problem

Notation

- **Feature vector**: \( f \in \mathbb{R}^p \)
- **Output**: \( y \in \{1, 2\} \)
- **Training data**: \( \mathcal{T} = \{ f_i, y_i = y(f_i) \}_{i=1,...,N} \)

General Form

\[
\hat{y}(f) = \begin{cases} 
1 & \text{if } \hat{w}^T f > \hat{t} \\
2 & \text{otherwise,}
\end{cases}
\]

Learning Algorithm

\[
(\hat{w}, \hat{t}) \in \arg \min \mathcal{L}(y_i, \hat{y}(f_i)) + \gamma \|w\|_1
\]

Several choices of loss \( \mathcal{L} \): LDA, SVM, perceptron
Linear Perceptron

- Simple linear classifier

\[ y = \text{sgn} \left( \sum_i w_i x_i \right) = GWx \]

- \( W, G \): linear, nonlinear operators
- Weight learning: gradient descent
- Simplified model of real neurons

Y. Wang
Deep Neural Networks (DNNs)

- Large array of perceptrons in many layers

\[
\hat{y} = G_M W_M \cdots G_2 W_2 G_1 W_1 x
\]

- Deep: \( M \gg 2 \)

- Linear operators \( \{W_m\} \) learned from data
  \[ \longrightarrow \text{Error back-propagation algorithm} \]
Convolutional Neural Networks (CNNs)

- “Parameter sharing” and “sparse activations”
- Operators $W_m$ are convolutions
- Easier to learn (less weights)
- Successfully used in image processing tasks

Source [Goodfellow, et al., 2016]

[LeCun et al., 1989][Ciresan et al., 2012][Krizhevsky et al., 2012]
Contents

Deep Neural Networks

Scattering Transform

Art Authentication

Neural style transfer
Scattering Transform

- Structure of Convolutional Neural Network
- Replace linear filters by wavelets
- Use modulus as nonlinearity

[Mallat, 2012]
Scattering Transform

- Structure of Convolutional Neural Network
- Replace linear filters by wavelets
- Use modulus as nonlinearity

Notation

- $\lambda = \lambda(j, \theta) = a^{-j}\theta$, $j \in \mathbb{Z}$, $\theta \in R \subset SO(d)$
- $p = (\lambda_1, \lambda_2, \ldots, \lambda_M)$
- $\psi_\lambda(u) = 2^{-dj}\psi(\lambda_i x)$
- $\phi_J(u) = 2^{-dJ}\phi(2^{-J}x)$
Scattering Transform

- Structure of Convolutional Neural Network
- Replace linear filters by wavelets
- Use modulus as nonlinearity

Notation

- \( \lambda = \lambda(j, \theta) = a^{-j} \theta, \ j \in \mathbb{Z}, \ \theta \in R \subset SO(d) \)
- \( p = (\lambda_1, \lambda_2, \ldots, \lambda_M) \)
- \( \psi_\lambda(u) = 2^{-dj} \psi(\lambda_i x) \)
- \( \phi_J(u) = 2^{-dJ} \phi(2^{-J} x) \)

Scattering coefficients

\[ S_m[p]X(u) = | | | | X \ast \psi_{\lambda_1} \ast \psi_{\lambda_2} \ast \cdots \ast \ast \psi_{\lambda_m} \ast \phi_J(u) \]

\[ S_m[P]X = (S_m[p]X)_{p \in P} \]
Illustration

\[
S_J[\emptyset]f = f \ast \phi_J
\]

\[
S_J[\lambda_1]f = U_J f[\lambda_1] \ast \phi_J
\]

\[
U[\lambda_1]f = |f \ast \psi_{\lambda_1}|
\]

\[
S_J[\lambda_1, \lambda_2]f
\]

\[
U[\lambda_1, \lambda_2]f
\]

\[
U[\lambda_1, \lambda_2, \lambda_3]f
\]

Source: [Mallat, 2012]
Properties

Stability

∀X, Y ∈ L^2(\mathbb{R}^d), \quad \|S[P]X - S[P]Y\| \leq \|X - Y\|
Properties

Stability

\( \forall X, Y \in L^2(\mathbb{R}^d), \quad \| S[P]X - S[P]Y \| \leq \| X - Y \| \)

Translation invariance

Let \( T_c X(u) = X(u - c) \). Then,

\( \forall X \in L^2(\mathbb{R}^d), \forall c \in \mathbb{R}^d, \quad S[P] T_c X = S[P]X, \quad \text{when} \ J \rightarrow \infty \)
Properties

Stability

∀X, Y ∈ L^2(\mathbb{R}^d), \quad \|S[P]X - S[P]Y\| \leq \|X - Y\|

Translation invariance
Let \( T_cX(u) = X(u - c) \). Then,

∀X ∈ L^2(\mathbb{R}^d), \forall c ∈ \mathbb{R}^d, \quad S[P]T_cX = S[P]X, \quad \text{when } J \to \infty

Stability to deformations
Let \( D_\tau X(u) = X(u - \tau(u)) \) with \( \|\nabla \tau\|_\infty \leq \frac{1}{2} \). Then,

∀X ∈ L^2(\mathbb{R}^d), \forall \tau ∈ C^2(\mathbb{R}^d), \quad \|S[P]X - S[P]D_\tau X\| \leq C\|X\|\|\nabla \tau\|_\infty

[Mallat2012]
Contents

Deep Neural Networks

Scattering Transform

Art Authentication

Neural style transfer
Art Authentication: Is It a Raphael?

In 2013 I have received a request from a collector Edward Rosser in Boston, asking me whether I could tell the following drawing was a genuine Raphael.
Art Authentication Problem

- Detect art objects from forgeries or imitations
- Style vs content
- Database 1
  - 64 van Gogh paintings (several periods)
  - 15 forgeries or contemporaries in same style
  - Size: $1452 \times 388$ px to $5614 \times 7381$ px
- Database 2
  - 21 drawings by Raphael
  - 9 imitations
  - Size: $2188 \times 3312$ to $6330 \times 4288$ pixels
Sample Images

van Gogh (VG)  non van Gogh (NVG)

Raphael (RA)  non Raphael (NRA)
State of the Art

Van Gogh dataset: Features and classifiers

- Wavelets, custom frames, EMD
- SVM, clustering, Hidden Markov Models
- Accuracy < 90%
- Single layer of features

[Berezhnoy et al., 2007], [Johnson et al., 2008], [Qi et al., 2013], [Liu & Chan, 2016]
Analysis Setup

- Preprocessing: grayscale, [0, 1] double-precision
- Automatic removal of canvas edges (max 100 px)
- Morlet wavelets, \( a = 2 \), 8 rotations
- \( J = 3, 4, \ldots, 7 \)
- Analysis by patches: 512 \( \times \) 512, 1024 \( \times \) 1024 and 2048 \( \times \) 2048
- 5-fold stratified cross-validation
- Linear classifiers:
  - PCA, LDA, SVM
  - **Sparse versions:** SSVM, SLDA \( \rightarrow \) \( \ell_1 \) regularization
Example: Scattering Coefficients

Van Gogh

Raphael
Influence of Patch Size and Averaging Scale

- Select $512 \times 512$ and $J = 4$.
- Fine-scale details preferred
Performance: Individual Patches

- Simple is better (PCA)
- Sparse is better (SLDA/SSVM)
- Raphael easier than van Gogh
Performance: Full Paintings

- Majority votes from patch decisions

![Performance Graph]

- SSVM: good performance & few features
Performance: Comparison with State of the Art

- Similar Van Gogh database

<table>
<thead>
<tr>
<th>Reference</th>
<th>ACC</th>
<th>Data size (VG+NVG)</th>
<th>Validation</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Liu &amp; Chan, 2016]</td>
<td>0.88</td>
<td>64 + 15</td>
<td>LOO</td>
</tr>
<tr>
<td>[Qi et al., 2013]</td>
<td>0.85</td>
<td>65 + 15</td>
<td>LOO</td>
</tr>
<tr>
<td>[Johnson et al., 2008]</td>
<td>0.84</td>
<td>64 + 12</td>
<td>LOO</td>
</tr>
<tr>
<td>Our results</td>
<td>0.96</td>
<td>64 + 15</td>
<td>5-CV</td>
</tr>
</tbody>
</table>
Feature Selection: van Gogh

SSVM

SLDA

First layer

Second layer
Feature Selection: Raphael

SSVM

SLDA

First layer

Second layer

Y. Wang
Scattering Transform for Art Investigations
Contents

Deep Neural Networks

Scattering Transform

Art Authentication

Neural style transfer
Neural Style Transfer
How Does It Work?

- Pretrained convolutional neural network
- **Coefficient matrix:** $F_m(X) \in \mathbb{R}^{N_{filt} \times N_{pixels}}$
- **Correlation matrix:** $G_m(X) = \frac{1}{N_{pixels}} F_m(X)(F_m(X))^T$
How Does It Work?

- **Pretrained convolutional neural network**
- **Coefficient matrix:** $F_m(X) \in \mathbb{R}^{N_{filt} \times N_{pixels}}$
- **Correlation matrix:** $G_m(X) = \frac{1}{N_{pixels}} F_m(X) (F_m(X))^T$
- **Loss functions:**

$$L_{content}(X, m) = \| F_m(X) - F_m(X_{content}) \|_F^2$$
How Does It Work?

- Pretrained convolutional neural network
- **Coefficient matrix:** $F_m(X) \in \mathbb{R}^{N_{filt} \times N_{pixels}}$
- **Correlation matrix:** $G_m(X) = \frac{1}{N_{pixels}} F_m(X) (F_m(X))^T$
- **Loss functions:**
  \[
  \mathcal{L}_{content}(X, m) = \| F_m(X) - F_m(X_{content}) \|_F^2
  \]
  \[
  \mathcal{L}_{style}(X, m) = \| \hat{G}_m(X) - G_m(X_{style}) \|_F^2
  \]
How Does It Work?

- Pretrained convolutional neural network
- **Coefficient matrix**: \( F_m(X) \in \mathbb{R}^{N_{\text{filt}} \times N_{\text{pixels}}} \)
- **Correlation matrix**: \( G_m(X) = \frac{1}{N_{\text{pixels}}} F_m(X)(F_m(X))^T \)
- **Loss functions**:

\[
\mathcal{L}_{\text{content}}(X, m) = \| F_m(X) - F_m(X_{\text{content}}) \|_F^2
\]
\[
\mathcal{L}_{\text{style}}(X, m) = \| \hat{G}_m(X) - G_m(X_{\text{style}}) \|_F^2
\]
\[
\mathcal{L}_{\text{total}}(X) = \alpha \mathcal{L}_{\text{content}}(X ; m_0) + \beta \sum_m w_m \mathcal{L}_{\text{style}}(X ; m)
\]
How Does It Work?

- Pretrained convolutional neural network
- **Coefficient matrix**: $F_m(X) \in \mathbb{R}^{N_{filt} \times N_{pixels}}$
- **Correlation matrix**: $G_m(X) = \frac{1}{N_{pixels}} F_m(X)(F_m(X))^T$
- **Loss functions**:
  
  $$L_{content}(X, m) = \|F_m(X) - F_m(X_{content})\|_F^2$$
  $$L_{style}(X, m) = \|\hat{G}_m(X) - G_m(X_{style})\|_F^2$$
  $$L_{total}(X) = \alpha L_{content}(X; m_0) + \beta \sum_m w_m L_{style}(X; m)$$

- **Image synthesis**:
  
  $$X \in \arg \min_X (L_{total})$$

[ Gatys et al., 2015 ]
Style Transfer: Scattering

Ideas

1. Simple manipulation of coefficients
2. Scattering can be inverted
Style Transfer: Scattering

Ideas

1. Simple manipulation of coefficients
2. Scattering can be inverted
   • Phase retrieval + Pseudoinverse
Style Transfer: Scattering

Ideas

1. Simple manipulation of coefficients
2. Scattering can be inverted
   - Phase retrieval + Pseudoinverse
   - Seems easy...
Style Transfer: Scattering

Ideas

1. Simple manipulation of coefficients
2. Scattering can be inverted
   - Phase retrieval + Pseudoinverse
   - Seems easy...
   - ...it’s not
   - Limitation: current phase retrieval algorithms
Covariance Change

Notation

- **Coefficient matrix:** $F_m \in \mathbb{R}^{N_{\text{filt}} \times N_{\text{pixels}}}$

- $\Sigma_m = \text{cov}(F_m) = \frac{1}{N_{\text{pixels}}} F_m F_m^T = U\Lambda U^T$

Procedure
Covariance Change

Notation

- **Coefficient matrix:** \( F_m \in \mathbb{R}^{N_{filt} \times N_{pixels}} \)
- \( \Sigma_m = \text{cov}(F_m) = \frac{1}{N_{pixels}} F_m F_m^T = U \Lambda U^T \)

Procedure

1. Remove current style
   \[ F^{\text{white}}_m = \Lambda^{-1/2} U^T F_m \]
Covariance Change

Notation

- **Coefficient matrix:** \( F_m \in \mathbb{R}^{N_{\text{filt}} \times N_{\text{pixels}}} \)
- \( \Sigma_m = \text{cov}(F_m) = \frac{1}{N_{\text{pixels}}} F_m F_m^T = U \Lambda U^T \)

Procedure

1. Remove current style
   \[ F_m^{\text{white}} = \Lambda^{-1/2} U^T F_m \]
2. Determine new covariance
   \[ \Sigma_m^{\text{new}} = \alpha \Sigma_m^{\text{cont}} + (1 - \alpha) \Sigma_m^{\text{style}} \]
Covariance Change

Notation

- **Coefficient matrix**: \( F_m \in \mathbb{R}^{N_{\text{filt}} \times N_{\text{pixels}}} \)
- \( \Sigma_m = \text{cov}(F_m) = \frac{1}{N_{\text{pixels}}} F_m F_m^T = U \Lambda U^T \)

Procedure

1. Remove current style
   \[
   F_m^{\text{white}} = \Lambda^{-1/2} U^T F_m
   \]
2. Determine new covariance
   \[
   \Sigma_m^{\text{new}} = \alpha \Sigma_m^{\text{cont}} + (1 - \alpha) \Sigma_m^{\text{style}}
   \]
3. Transfer style
   \[
   F_m^{\text{new}} = \left( \alpha \Sigma_m^{\text{cont}} + (1 - \alpha) \Sigma_m^{\text{style}} \right)^{1/2} \Lambda^{-1/2} U^T F_m
   \]
Covariance Change

Notation

• **Coefficient matrix:** \( F_m \in \mathbb{R}^{N_{\text{filt}} \times N_{\text{pixels}}} \)
• \( \Sigma_m = \text{cov}(F_m) = \frac{1}{N_{\text{pixels}}} F_m F_m^T = U \Lambda U^T \)

Procedure

1. Remove current style
   \[
   F_m^{\text{white}} = \Lambda^{-1/2} U^T F_m
   \]
2. Determine new covariance
   \[
   \Sigma_m^{\text{new}} = \alpha \Sigma_m^{\text{cont}} + (1 - \alpha) \Sigma_m^{\text{style}}
   \]
3. Transfer style
   \[
   F_m^{\text{new}} = \left( \alpha \Sigma_m^{\text{cont}} + (1 - \alpha) \Sigma_m^{\text{style}} \right)^{1/2} \Lambda^{-1/2} U^T F_m
   \]
4. Invert \( F_m^{\text{white}} \)
Phase Retrieval

- Gerchberg-Saxton algorithm
  - Recover $x$ such that $|Ax| = b$
  - Input: $y^{(1)} \in \mathbb{C}$ such that $|y^{(1)}| = b$
  - Iteration:
    $$y_i^{(k+1)} = b_i \frac{(AA^\dagger y^{(k)})_i}{|(AA^\dagger y^{(k)})_i|}$$
  - Not guaranteed to converge to solution
  - Low computational complexity
(Very) Preliminary Results
Happy 80th Birthday John!