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

Contents

Linear Classification Problem

Notation

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

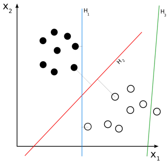
Linear Classification Problem

Notation

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

General Form

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



Source: Wikipedia

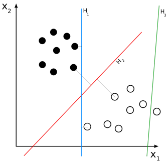
Linear Classification Problem

Notation

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

General Form

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



Source: Wikipedia

Learning Algorithm

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

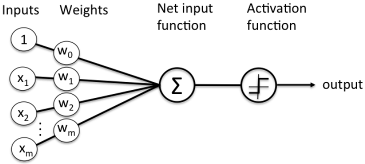
Several choices of loss $\mathcal{L}$: LDA, SVM, perceptron

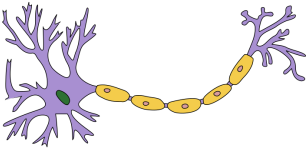
Linear Perceptron

- Simple linear classifier

$$y = \operatorname{sgn}\left(\sum_i w_i x_i\right) = GWx$$

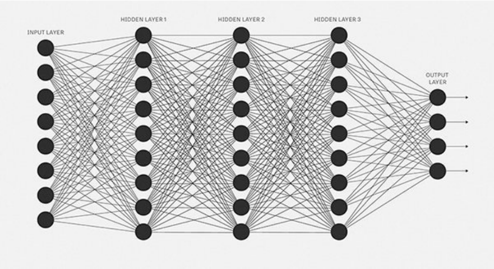
- W, G: linear, nonlinear operators
- Weight learning: gradient descent
- Simplified model of real neurons





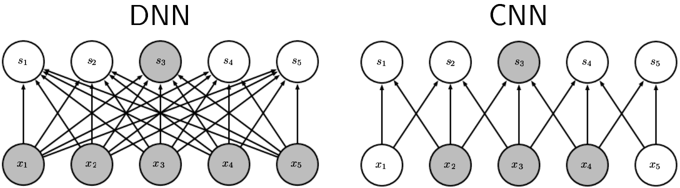
Deep Neural Networks (DNNs)

- Large array of perceptrons in many layers



- Output: $\hat{y} = \ G_M W_M \ \cdots G_2 W_2 \ G_1 W_1 x$
- Deep: $M \gg 2$
- Linear operators $\{W_m\}$ learned from data
  $\longrightarrow$ Error back-propagation algorithm

Convolutional Neural Networks (CNNs)



Source [Goodfellow, et al., 2016]

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

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

Contents

Scattering Transform

[Mallat, 2012]

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

Scattering Transform

[Mallat, 2012]

- 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

[Mallat, 2012]

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

Notation

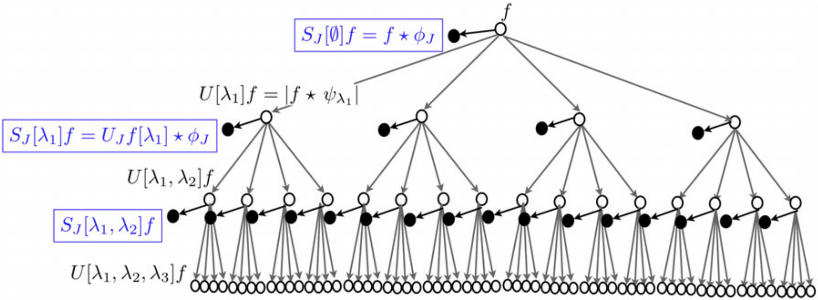
- $\lambda = \lambda(j, \theta) = a^{-j}\theta,\ j \in \mathbb{Z},\ \theta \in R \subset SO(d)$
- $p = (\lambda_1, \lambda_2, \ldots, \lambda_M)$
- $\psi_\lambda(u) = 2^{-dj}\psi(\lambda_i x)$
- $\phi_J(u) = 2^{-dJ}\phi(2^{-J}x)$

Scattering coefficients

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

$$S_m[P]X = \left(S_m[p]X\right)_{p \in P}$$

Illustration



Source: [Mallat, 2012]

Properties

Stability

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

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,

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

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,

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

Stability to deformations

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

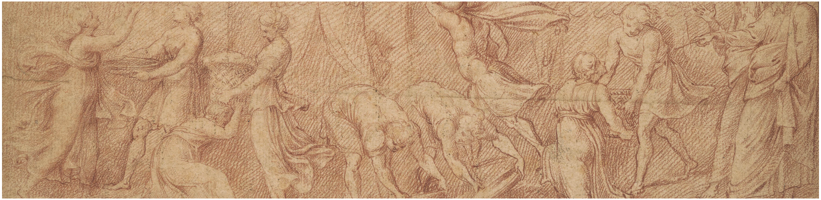
$$\forall X \in L^2(\mathbb{R}^d), \forall \tau \in C^2(\mathbb{R}^d), \qquad \|S[P]X - S[P]D_\tau X\| \leq C\|X\|\|\nabla \tau\|_\infty$$

[Mallat2012]

Contents

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: 1452×388 px to 5614×7381 px
- Database 2
  - 21 drawings by Raphael
  - 9 imitations
  - Size: 2188×3312 to 6330×4288 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

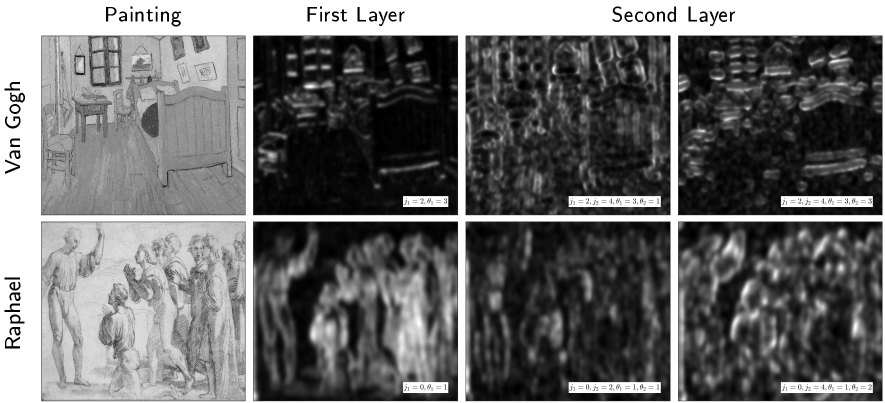
- Wavelets, custom frames, EMD
- SVM, clustering, Hidden Markov Models
- Accuracy $< 90\%$
- Single layer of features

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

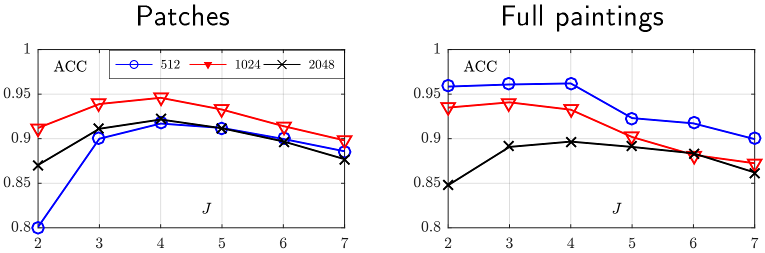
Analysis Setup

- 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: 512×512, 1024×1024 and 2048×2048
- 5-fold stratified cross-validation
- Linear classifiers:
  - PCA, LDA, SVM
  - **Sparse versions:** SSVM, SLDA $\longrightarrow$ ℓ_1 regularization

Example: Scattering Coefficients

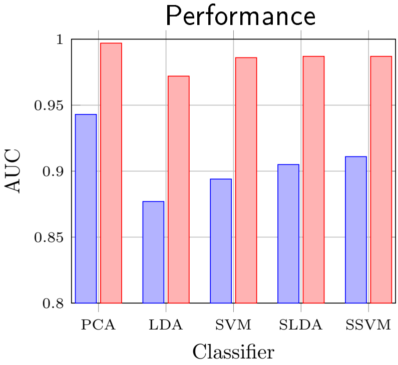


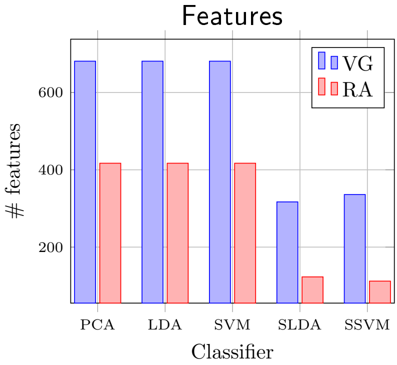
Influence of Patch Size and Averaging Scale



- Select 512×512 and $J = 4$.
- Fine-scale details preferred

Performance: Individual Patches

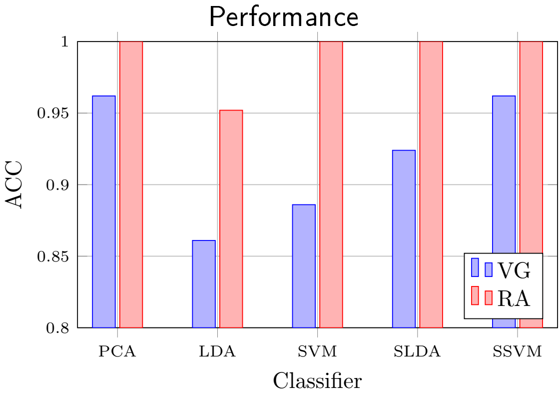




- Simple is better (PCA)
- Sparse is better (SLDA/SSVM)
- Raphael easier than van Gogh

Performance: Full Paintings

- Majority votes from patch decisions



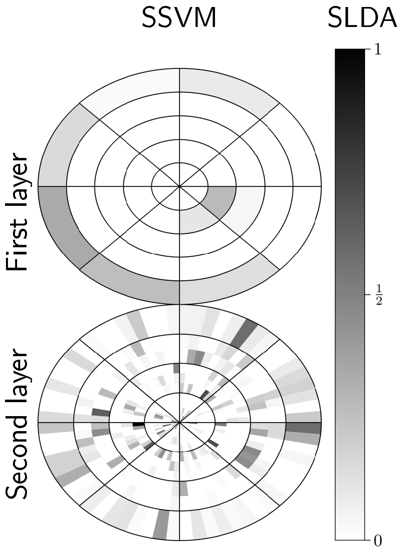
- SSVM: good performance & few features

Performance: Comparison with State of the Art

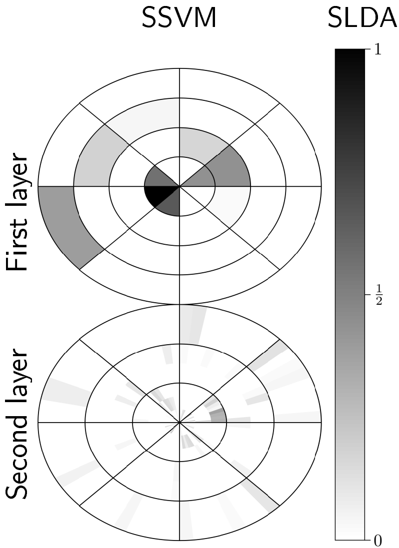
- Similar Van Gogh database

Reference	ACC	Data size (VG+NVG)	Validation
[Liu & Chan, 2016]	0.88	$64 + 15$	LOO
[Qi et al., 2013]	0.85	$65 + 15$	LOO
[Johnson et al., 2008]	0.84	$64 + 12$	LOO
Our results	0.96	$64 + 15$	5-CV

Feature Selection: van Gogh

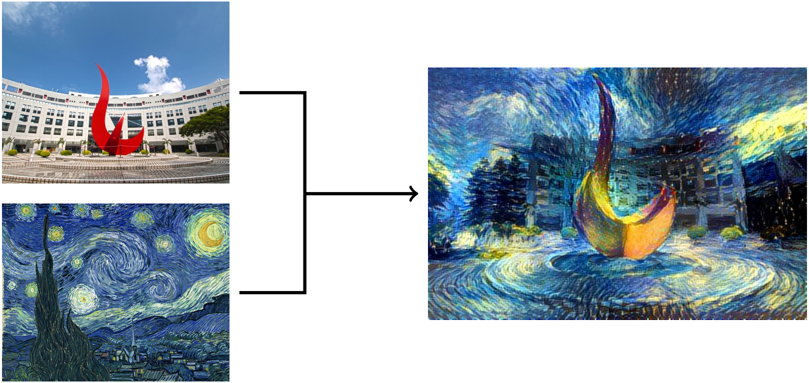


Feature Selection: Raphael



Contents

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:** $G_m(X) = \frac{1}{N_{pixels}} F_m(X) \big(F_m(X)\big)^T$

How Does It Work?

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

$$\mathcal{L}_{content}(X, m) = \|F_m(X) - F_m(X_{content})\|_F^2$$

How Does It Work?

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

$$\mathcal{L}_{content}(X, m) = \|F_m(X) - F_m(X_{content})\|_F^2$$

$$\mathcal{L}_{style}(X, m) = \|\hat{G}_m(X) - G_m(X_{style})\|_F^2$$

$$\mathcal{L}_{total}(X) = \alpha \mathcal{L}_{content}(X; m_0) + \beta \sum_m w_m \mathcal{L}_{style}(X; m)$$

How Does It Work?

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

$$\mathcal{L}_{content}(X, m) = \|F_m(X) - F_m(X_{content})\|_F^2$$

$$\mathcal{L}_{style}(X, m) = \|\hat{G}_m(X) - G_m(X_{style})\|_F^2$$

$$\mathcal{L}_{total}(X) = \alpha \mathcal{L}_{content}(X; m_0) + \beta \sum_m w_m \mathcal{L}_{style}(X; m)$$

- **Image synthesis:**

$$X \in \arg\min_X (\mathcal{L}_{total})$$

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

Procedure

Covariance Change

Notation

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

Procedure

1. Remove current style

$$F_m^{white} = \Lambda^{-1/2} U^T F_m$$

Covariance Change

Notation

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

Procedure

1. Remove current style

$$F_m^{white} = \Lambda^{-1/2} U^T F_m$$

2. Determine new covariance

$$\Sigma_m^{new} = \alpha \Sigma_m^{cont} + (1 - \alpha) \Sigma_m^{style}$$

Covariance Change

Notation

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

Procedure

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

Covariance Change

Notation

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

Procedure

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

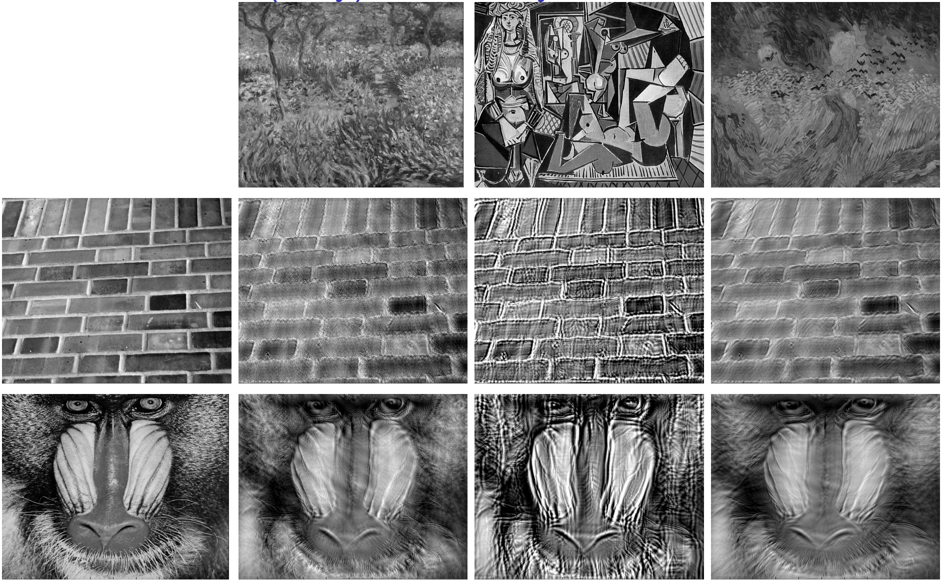
Phase Retrieval

- 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!