# Deep Learning Motivated By Numerical Analysis

Martin Do Pham
Supervisor: Prof. Giang Tran

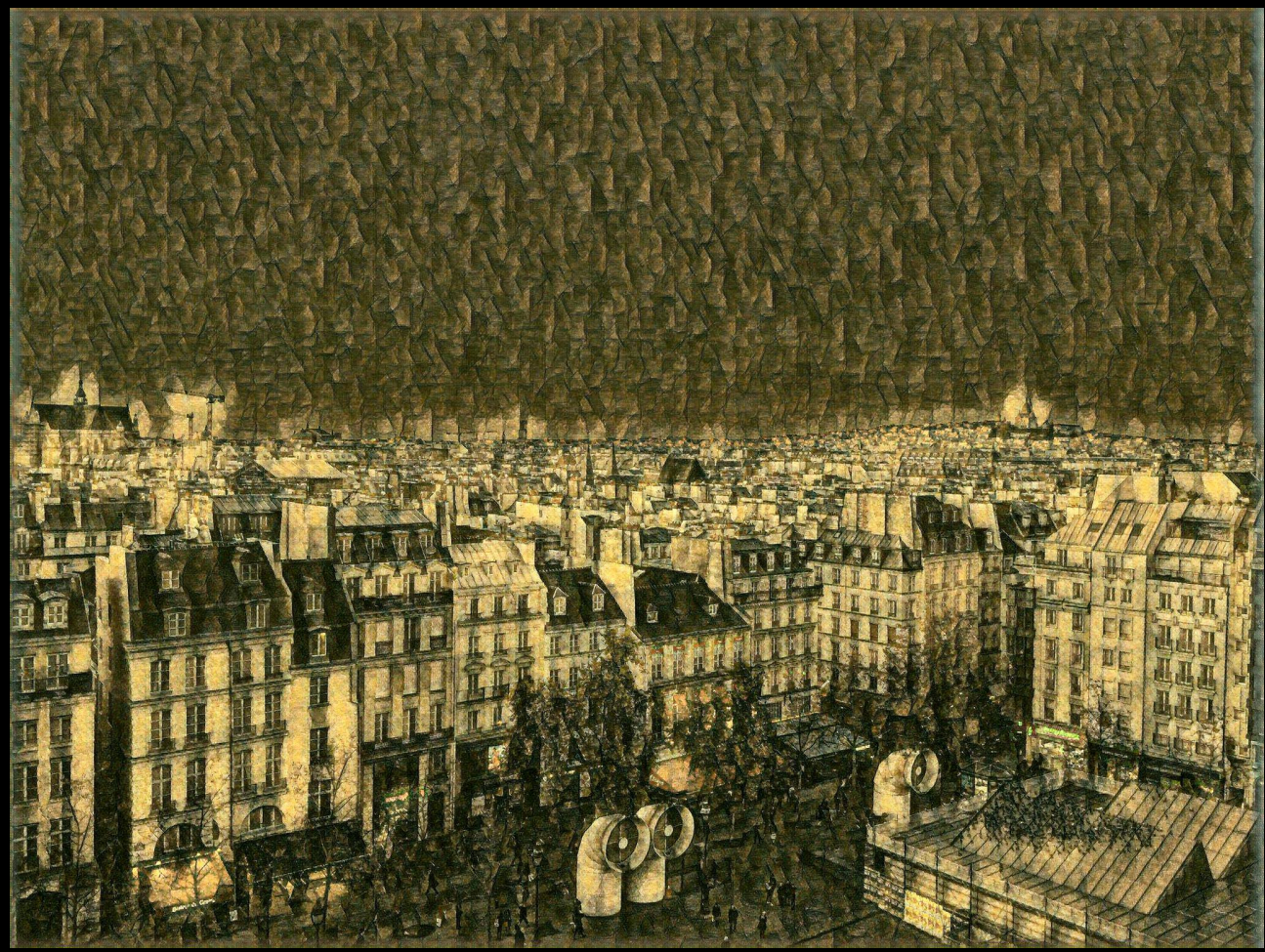
University of Waterloo Applied Math Dept.

12 September 2018

# Coote's Paradise + Mosaic

# Coote's Paradise + Ma Jolie

# Paris + Ma Jolie



# @diananguyenart

# Outline

- Motivation

# Outline

- Motivation
- Numerical Analysis

# Outline

- Motivation
- Numerical Analysis
- ResNet and FractalNet

# Outline

- Motivation
- Numerical Analysis
- ResNet and FractalNet
- Style Transfer to Finance

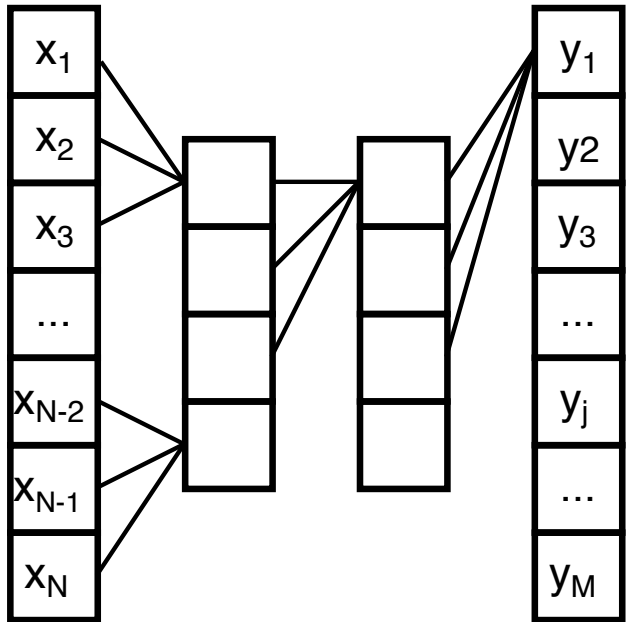
# Outline

- **Motivation**
- Numerical Analysis
- ResNet and FractalNet
- Style Transfer to Finance

# Some basics of (Convolutional) Neural Networks

# Convolutional Neural Networks (CNNs)

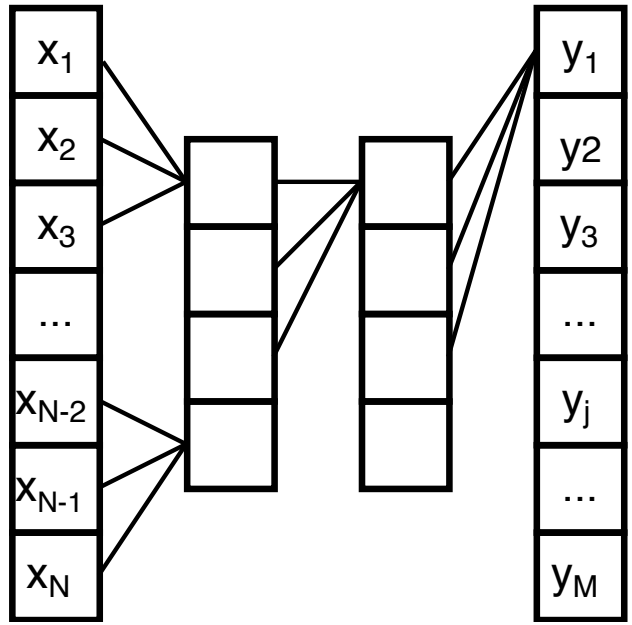
- Computational model inspired by human visual system




A two-layer CNN

# Convolutional Neural Networks (CNNs)

- Computational model inspired by human visual system

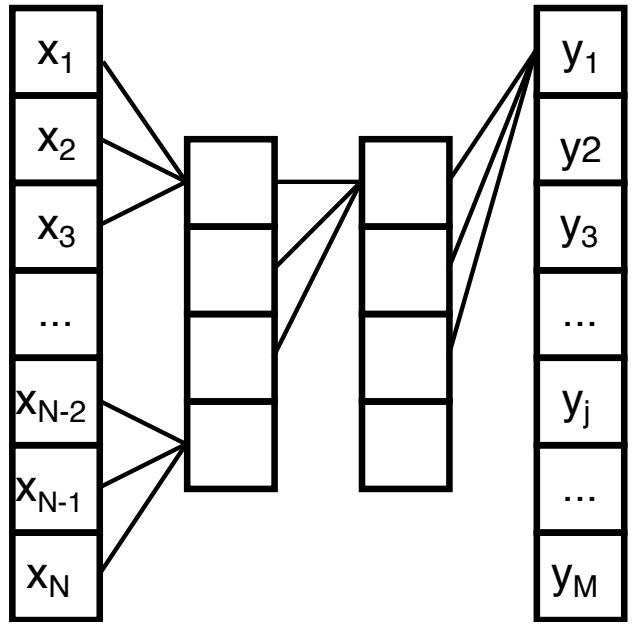



A two-layer CNN

- Use data to **learn parameters** (connections between neurons)

# Convolutional Neural Networks (CNNs)

- Computational model inspired by human visual system

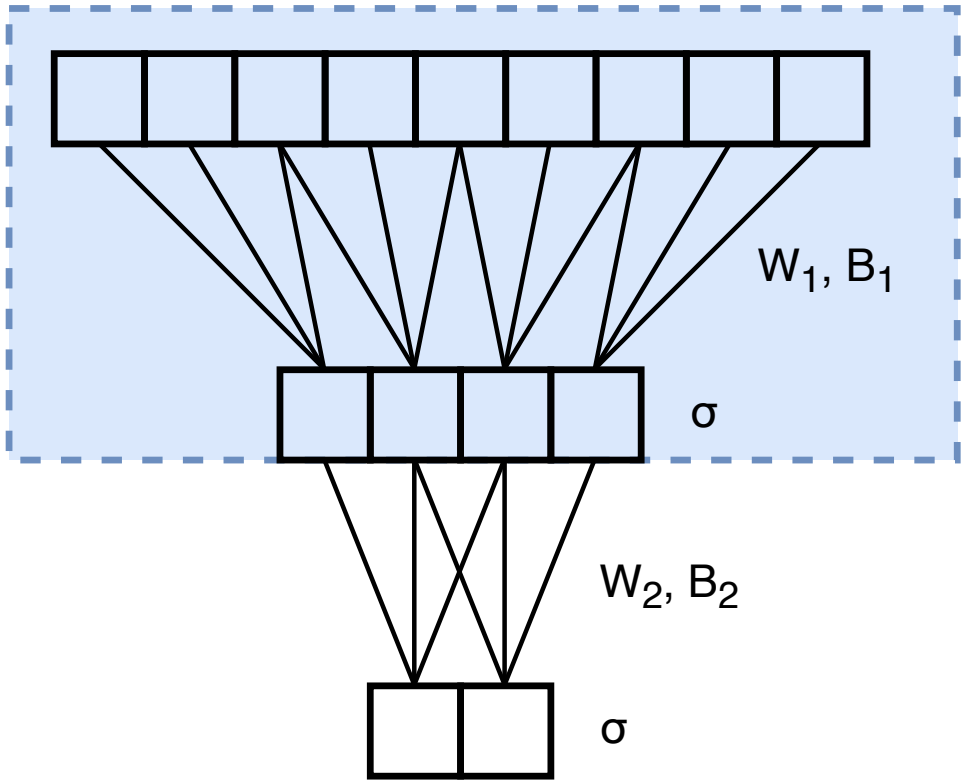



A two-layer CNN

- Use data to **learn parameters** (connections between neurons)
- Applications: colourization, caption generation, translation...

# CNNs

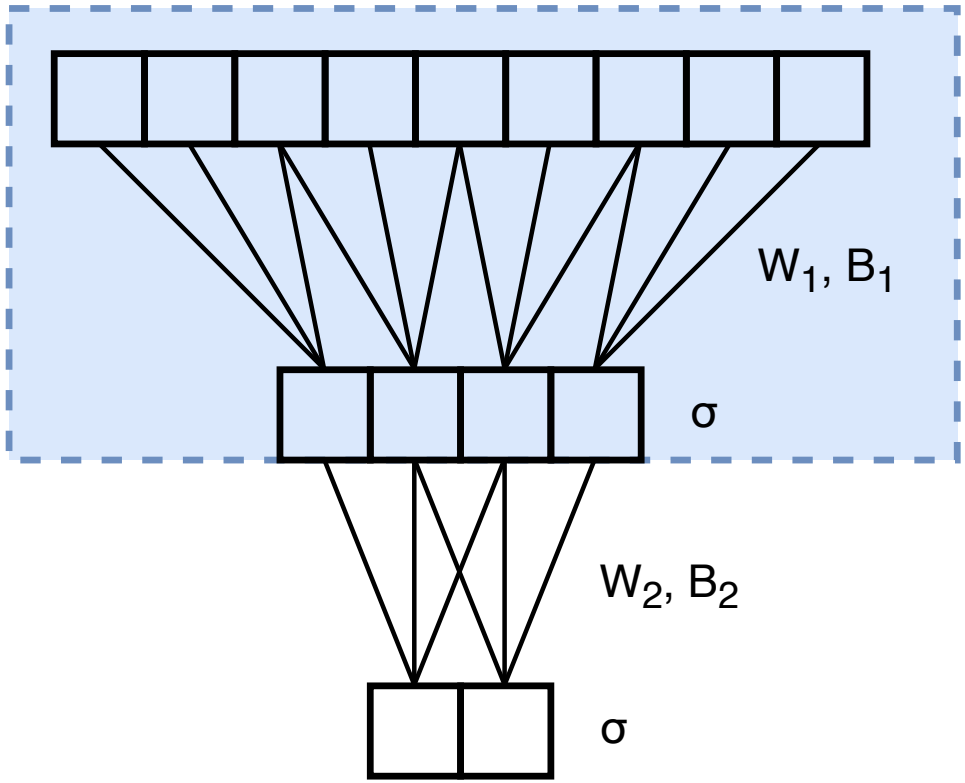
- Conv block: convolution $\{W, b\}$, non-linear activation $\sigma$




$$f_j(x) = \sigma(W_j x + b_j)$$

# CNNs

- Conv block: convolution $\{W, b\}$, non-linear activation $\sigma$




$$f_j(x) = \sigma(W_j x + b_j)$$

- Network can be represented as **composition of conv blocks**

$$F_{\mathbf{W},\mathbf{b}}(x) = (f_n \circ ... \circ f_j \circ ... \circ f_1)(x)$$

# Challenges

- **Training**: data, optimization technique, speedup, memory
- **Design**: architecture, justification, loss function

# Challenges

- **Training**: data, optimization technique, speedup, memory
- **Design**: architecture, justification, loss function

We will focus on the problem of *design* ...

# Outline

- Motivation
- **Numerical Analysis**
- ResNet and FractalNet
- Style Transfer to Finance

# Connections between Neural Networks and Numerical Analysis

# Infinite-layer neural networks

- Let $F_{\mathbf{W},\mathbf{b}}$ be an **infinite-layer neural network**

Ruthotto, L., & Haber, E. Stable Architectures for Deep Neural Networks (2017)

# Infinite-layer neural networks

- Let $F_{\mathbf{W},\mathbf{b}}$ be an **infinite-layer neural network**
- Consider the first-order differential equation (DE), $t \in [0, 1]$

$$\frac{dY}{dt}(t) = F_{\mathbf{W},\mathbf{b}}(Y(t), t)$$
$$Y(0) = Y_0$$

Ruthotto, L., & Haber, E. Stable Architectures for Deep Neural Networks (2017)

# Infinite-layer neural networks

- Let $F_{\mathbf{W},\mathbf{b}}$ be an **infinite-layer neural network**
- Consider the first-order differential equation (DE), $t \in [0, 1]$

$$\frac{dY}{dt}(t) = F_{\mathbf{W},\mathbf{b}}(Y(t), t)$$
$$Y(0) = Y_0$$

- DE represents propagation of signal $Y$ through network

---

Ruthotto, L., & Haber, E. Stable Architectures for Deep Neural Networks (2017)

# Infinite-layer neural networks

- Let $F_{\mathbf{W},\mathbf{b}}$ be an **infinite-layer neural network**
- Consider the first-order differential equation (DE), $t \in [0, 1]$

$$\frac{dY}{dt}(t) = F_{\mathbf{W},\mathbf{b}}(Y(t), t)$$
$$Y(0) = Y_0$$

- DE represents propagation of signal $Y$ through network
- Goal: find an approximate numerical solution $Y_{t_n} \approx Y(t_n)$

Ruthotto, L., & Haber, E. Stable Architectures for Deep Neural Networks (2017)

# Euler Method

- Consider finite difference definition of derivative

$$\frac{dY}{dt}(t) = \lim_{h \to 0} \frac{Y(t+h) - Y(t)}{h}$$

# Euler Method

- Consider finite difference definition of derivative

$$\frac{dY}{dt}(t) = \lim_{h \to 0} \frac{Y(t+h) - Y(t)}{h}$$

- Replace LHS with DE, approximate RHS with steps $\Delta h$

$$F_{\mathbf{W},\mathbf{b}}(Y(t), t) = \frac{Y(t+h) - Y(t)}{\Delta h}$$

# Euler Method

- Consider finite difference definition of derivative

$$\frac{dY}{dt}(t) = \lim_{h \to 0} \frac{Y(t+h) - Y(t)}{h}$$

- Replace LHS with DE, approximate RHS with steps $\Delta h$

$$F_{\mathbf{W},\mathbf{b}}(Y(t), t) = \frac{Y(t+h) - Y(t)}{\Delta h}$$

- Rearrange to obtain **forward Euler method** numerical scheme

$$Y_{t_{n+1}} := Y(t_n + \Delta h) = Y_{t_n} + \Delta h F_{\mathbf{W},\mathbf{b}}(Y_{t_n}, t_n)$$

# How is this useful?

We have an approximation of the propagation of our signal through the network $F_{\mathbf{W},\mathbf{b}}$

We can ask a few things...

# Properties of numerical methods

- **Consistency**: analytic solution satisfies numerical scheme

# Properties of numerical methods

- **Consistency**: analytic solution satisfies numerical scheme
- **Stability**: growth rate of errors do not diverge

# Properties of numerical methods

- **Consistency**: analytic solution satisfies numerical scheme
- **Stability**: growth rate of errors do not diverge
- **Convergence**: refining discretization recovers exact solution

# Properties of numerical methods

- **Consistency**: analytic solution satisfies numerical scheme
- **Stability**: growth rate of errors do not diverge
- **Convergence**: refining discretization recovers exact solution

Many neural network problems are ill-posed but stability is still an important property for robustness of solution

# Outline

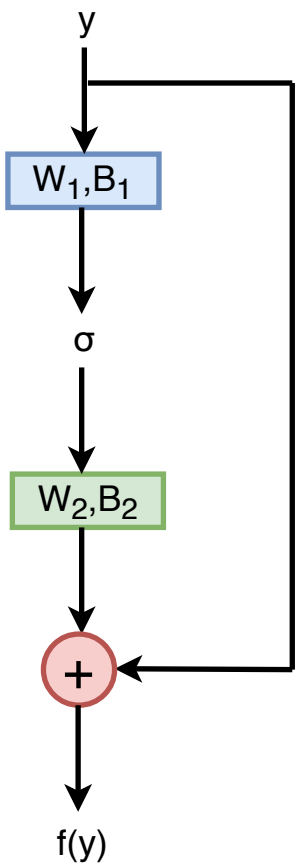
- Motivation
- Numerical Analysis
- **ResNet and FractalNet**
- Style Transfer to Finance

# What makes a “good” architecture?

Not clear... but numerical analysis can help compare two examples

# Architecture 1: ResNet

Introduce a skip connection to learn a residual function




ResNet block:

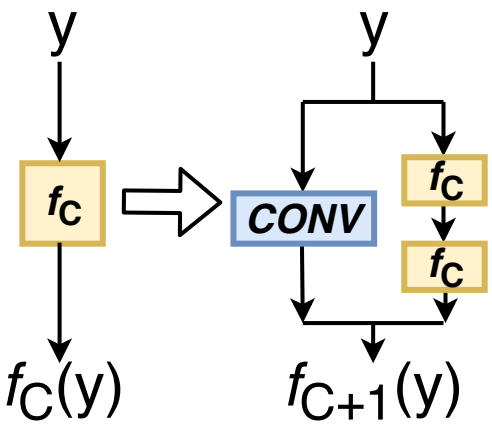
$$f^{res}(y) = y + \underbrace{\left(W_2\sigma(W_1 y + b_1) + b_2\right)}_{F}$$

Define propagation of signal as:

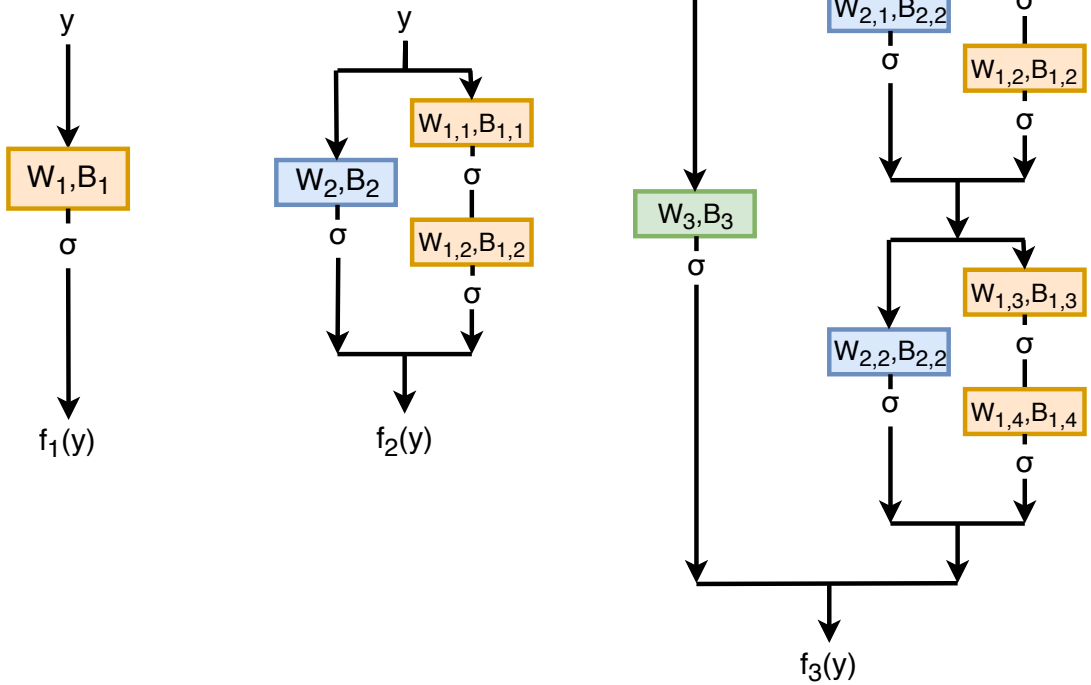
$$y_{j+1} = f^{res}(y_j) := y_j + F(y_j, \mathbf{W}^j)$$

# Architecture 2: FractalNet

Generate layers of convolution using an expansion rule

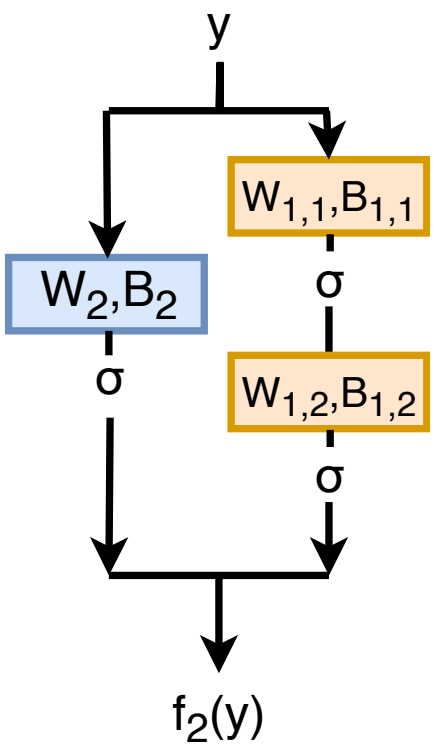



Expansion rule




FractalNet blocks up to $C = 3$

# Architecture 2: FractalNet




FractalNet block, $C = 2$:

$$f_2^{frac}(y) = (W_2 y + b_2) + \left( W_{1,2}\sigma(W_{1,1}y + b_{1,1}) + b_{1,2} \right)$$

Rewrite:

$$f_2^{frac}(y) = W_2 \left( y + \underbrace{W_2^{-1} \left( W_{1,2}\sigma(W_{1,1}y + b_{1,1}) + b_{1,2} + b_2 \right)}_{F} \right)$$

Define propagation of signal as:

$$y_{j+1} = f_2^{frac}(y_j) := W_2 \left( y_j + F(y_j, \mathbf{W}^j) \right)$$

# Comparison

## ResNet

$$f^{res}(y) = y + (W_2\sigma(W_1 y + b_1) + b_2)$$

$$\downarrow$$

$$f^{res}(y_j) = y_j + F(y_j, \mathbf{W}^j)$$

## FractalNet

$$f_2^{frac}(y) = (W_2 y + b_2) + (W_{1,2}\sigma(W_{1,1} y + b_{1,1}) + b_{1,2})$$

$$\downarrow$$

$$f_2^{frac}(y_j) = W_2\Big(y_j + F(y_j, \mathbf{W}^j)\Big)$$

Resemble Euler method for respective DEs

$$\frac{dY}{dt}(t, \mathbf{W}(t)) = F(Y(t), \mathbf{W}(t))$$

$$Y(0) = Y_0$$

---

Ruthotto, L., & Haber, E. Deep Neural Networks motivated by Partial Differential Equations (2018)

# Takeaway Conclusion

Training a neural network is thus equivalent to estimating the parameters of the analogous differential equation

i.e. find $\mathbf{W}(t)$ given network architecture $F$

# Outline

- Motivation
- Numerical Analysis
- ResNet and FractalNet
- **Style Transfer to Finance**

# Style Transfer

Goal: render an image $y_c$ in the style of $y_s$

# Style Transfer

Goal: render an image $y_c$ in the style of $y_s$

(**Preserve content** while **transferring texture, tone**)

# Style Transfer

Goal: render an image $y_c$ in the style of $y_s$

(**Preserve content** while **transferring texture, tone**)

Idea: train a neural network $F$ to do this

# Style Transfer DE

Find $\mathbf{W}(t)$ such that the evolution, $t \in [0, 1]$

$$\frac{dY}{dt}(t, \mathbf{W}(t)) = F(Y(t), \mathbf{W}(t))$$
$$Y(0) = Y_0 = y_c$$

has a final solution $Y(1)$ whose pixel distribution is similar to $y_s$

We choose $F$ to be FractalNet

# Numerical Results

# Coote's Paradise Leaf

Content

Style

# The Lovers II (Magritte)

Content

Style

# The Lovers II (Magritte)

# How is this useful for finance?

There are many DE models for finance...

# Towards Finance

- Black-Scholes:

$$\frac{\partial V}{\partial t} = rV - rS\frac{\partial V}{\partial S} - \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2}$$

# Towards Finance

- Black-Scholes:

$$\frac{\partial V}{\partial t} = rV - rS\frac{\partial V}{\partial S} - \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2}$$

- Traditionally solved using various numerical methods

# Towards Finance

- Black-Scholes:

$$\frac{\partial V}{\partial t} = rV - rS\frac{\partial V}{\partial S} - \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2}$$

- Traditionally solved using various numerical methods
- LHS should motivate the neural network architecture

# Takeaway Idea

Apply the DE parameter estimation framework!

# Takeaway Idea

Apply the DE parameter estimation framework!

Develop a neural network architecture analogous to a numerical scheme for financial models

# Takeaway Idea

Apply the DE parameter estimation framework!

Develop a neural network architecture analogous to a numerical scheme for financial models

Network will thus learn the underlying dynamics of evolution from the data

# Some possible challenges

- Many financial models are stochastic

# Some possible challenges

- Many financial models are stochastic
- What is the appropriate data to be used?

# Some possible challenges

- Many financial models are stochastic
- What is the appropriate data to be used?
- **Ethical concerns of algorithmic accountability**

# Conclusions

# Conclusions

- Can draw analogy between neural networks and DEs

# Conclusions

- Can draw analogy between neural networks and DEs
- Training neural networks is thus DE parameter estimation

# Conclusions

- Can draw analogy between neural networks and DEs
- Training neural networks is thus DE parameter estimation
- Can this parameter estimation problem be applied to finance?

Thanks!

# References


1. Johnson, Alahi, & Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. ECCV 2016
2. Larsson, Maire, & Shakhnarovich. Fractalnet: Ultra-deep neural networks without residuals. ICLR 2017
3. Ruthotto & Haber. Deep Neural Networks motivated by Partial Differential Equations. 2018
4. Ruthotto & Haber. Stable Architectures for Deep Neural Networks. 2017
5. Image source: Google Arts & Culture

# Style Transfer

1. Approximate stylizing by passing $x$ thru $F$

$$y = F(x, \mathbf{W})$$

# Style Transfer

1. Approximate stylizing by passing $x$ thru $F$

$$y = F(x, \mathbf{W})$$

2. Compute error between $y$ and $y_s$ using perceptual loss $L^\phi$
   $\rightarrow \phi$ is a fixed, pretrained network
   $\rightarrow L^\phi$ compares pixel distributions

$$E = L^\phi(y, y_s)$$

# Style Transfer

1. Approximate stylizing by passing $x$ thru $F$

$$y = F(x, \mathbf{W})$$

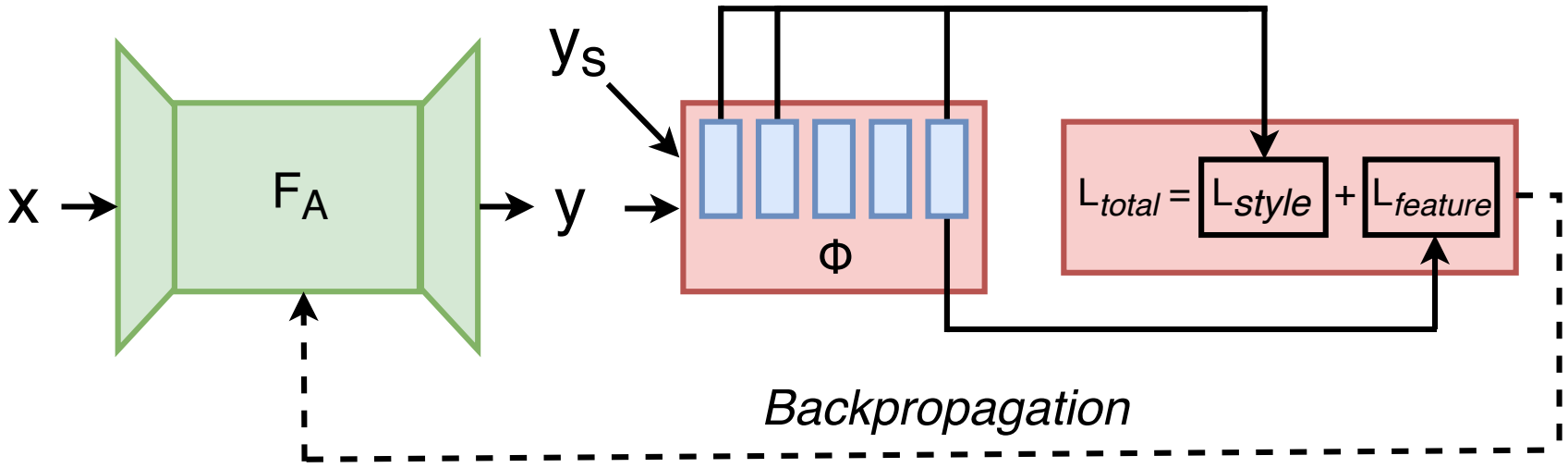
2. Compute error between $y$ and $y_s$ using perceptual loss $L^\phi$
   $\rightarrow \phi$ is a fixed, pretrained network
   $\rightarrow L^\phi$ compares pixel distributions

$$E = L^\phi(y, y_s)$$

3. Gradient descent on $\mathbf{W}$ until $L^\phi$ is minimized

$$\mathbf{W}_{next} = \mathbf{W}_{prev} - \eta \frac{\partial E}{\partial \mathbf{W}_{prev}}$$

# Training




Pass a content image $x$ thru network $F$ to stylize like $y_s$

Compute loss using different network $\phi$ and minimize by backpropagation

# Perceptual Losses

Let $\phi$ be a fixed and pretrained network;
$\phi_j(x) \in \mathbb{R}^{C_j \times H_j \times W_j}$ be the activations at layer $j$ given input $x$.
Define the perceptual loss function:

$$L^{\phi}_{total}(y_s, y) = L^{\phi}_{style}(y_s, y) + L^{\phi}_{feature}(y_s, y).$$

The network $F$ is trained to learn the style of $y_s$ using backpropagation and stochastic gradient descent.
This **minimizes the perceptual loss**

$$\mathbf{A}^* = \arg\min_{\mathbf{A}} \mathbf{E}_{x, y_s}\left[\sum_{i=1} L^{\phi}_{total}(F(x, \mathbf{A}), y_s))\right]$$

# Perceptual Losses

$L^{\phi}_{style}$: **correlations between activations within lower layers**:

$$L^{\phi}_{style}(y_s, y) = \sum_j ||G^{\phi}_j(y_s) - G^{\phi}_j(y)||^2_F$$

where $G^{\phi}_j(x) \in \mathbb{R}^{C_j \times C_j}$ has entries

$$G^{\phi}_j(x)_{c,c'} = \frac{1}{C_j H_j W_j} \sum_{h=1}^{H_j} \sum_{w=1}^{W_j} \phi_j(x)_{h,w,c} \phi_j(x)_{h,w,c'}$$

$L^{\phi}_{feature}$: **higher layer activations to capture spatial structure**:

$$L^{\phi}_{feature}(y_s, y) = \sum_j \frac{1}{C_j H_j W_j} ||\phi_j(y_s) - \phi_j(y)||^2_2$$