Deep Learning and Numerical PDEs
Image Classification and MgNet

Jinchao Xu

KAUST and Penn State

xu@multigrid.org

Morgan State University, June 21, 2023

CBMS Lecture Series

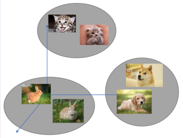
NSF DMS-2111387, KAUST Baseline Research Fund

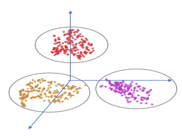
A basic AI problem: classification

- Can a machine (function) tell the difference ?

Supervised learning

- Function interpolation (data fitting)
  - Each image = a big vector of pixel values
    - ★ $d = 1280 \times 720 \times 3$ (width$\times$ height $\times$ RGB channel) $\approx$ 3M.
  - 3 different sets of points in $\mathbb{R}^d$, are they separable?





  - Mathematical problem: Find $f(\cdot;\Theta) : \mathbb{R}^d \to \mathbb{R}^3$ such that:

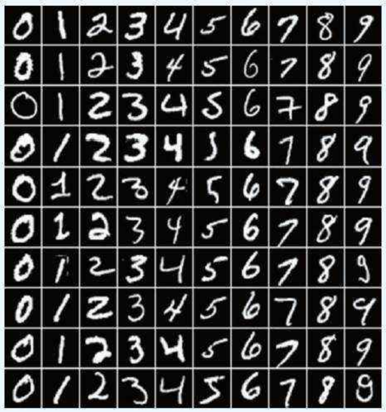
$f($  $;\Theta) \approx \begin{pmatrix}1\\0\\0\end{pmatrix}$ $f($  $;\Theta) \approx \begin{pmatrix}0\\1\\0\end{pmatrix}$ $f($  $;\Theta) \approx \begin{pmatrix}0\\0\\1\end{pmatrix}$

  - Test:

$$f(\boxed{?};\Theta) = \begin{pmatrix}0.7\\0.2\\0.1\end{pmatrix} \quad \Longrightarrow \quad \boxed{?} = \text{cat}$$

Example: MNIST data set

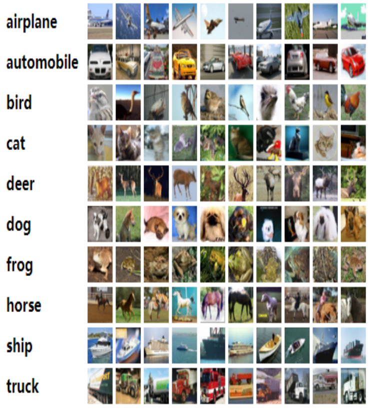
(Modified National Institute of Standards and Technology database)



Handwritten digits:

- Training set : 60,000
- Test set : 10,000
- Image size : 28*28*1
- Classes: 10

Example: CIFAR10



CIFAR10:

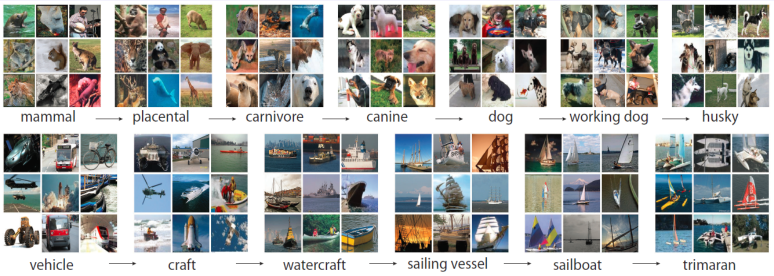
- Training set : 50,000
- Test set : 10,000
- Image size : 32*32*3
- Classes: 10

Example: CIFAR100

CIFAR100:

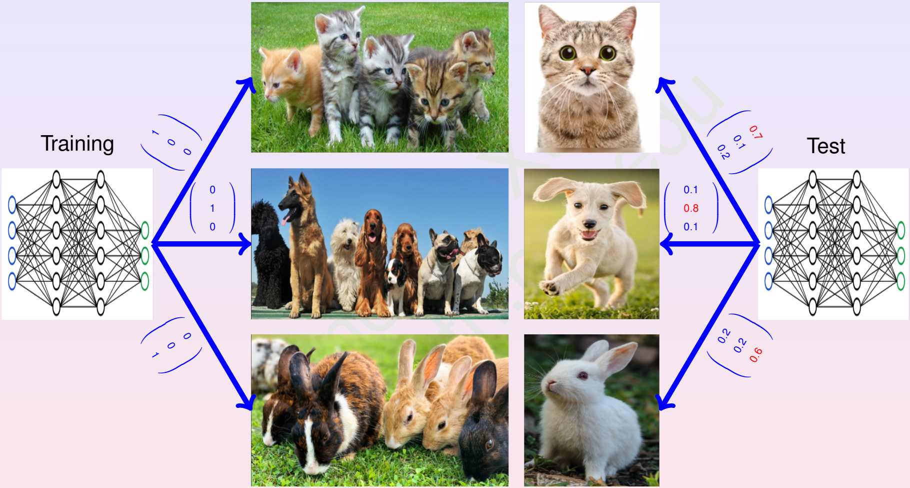
- Training set : 50,000
- Test set : 10,000
- Image size : 32*32*3
- Classes: 100

Example: ImageNet



ImageNet:

- Training set :1.2M
- Test set : 50,000
- Image size : The largest image resolution on ImageNet is $4288 \times 2848 \times 3$ pixels, the smallest image resolution is $75 \times 56 \times 3$ pixels, the average image resolution is $469 \times 387 \times 3$ pixels. Normally it's applied a pre-processing that samples them to a certain size, $224 \times 224 \times 3$ is used by most of the networks.
- Classes: 1000


Training
1 0 0
0 1 0
0 0 1
Test
0.7 0.1 0.2
0.1 0.8 0.1
0.2 0.2 0.6

Supervised Learning $\leftrightarrow$ Function Interpolation

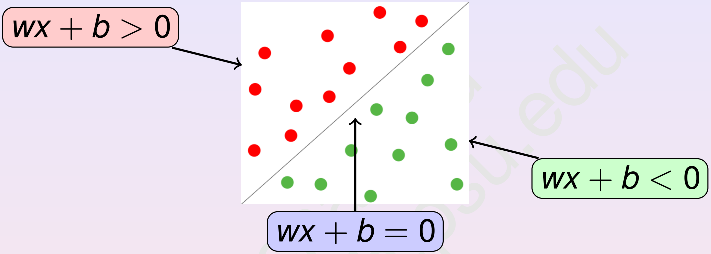
Consequence:

Machine Learning = High-Dimensional Numerical Analysis

Main question

What is a good function class for f?

Linearly separable sets: binary case

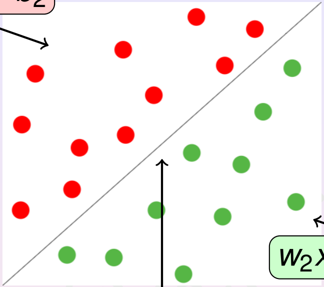


Definition: We say $A_1, A_2 \subset \mathbb{R}^n$ are **linearly separable**, if there exist a $w \in \mathbb{R}^{1\times n}$ and a $b \in \mathbb{R}$ such that

$$\begin{cases} wx + b > 0, \ \forall x \in A_1, \\ wx + b < 0, \ \forall x \in A_2. \end{cases} \tag{2}$$

Binary linearly separable: two equivalent definitions

$$w_1 x + b_1 > w_2 x + b_2$$



$$w_2 x + b_2 > w_1 x + b_1$$

$$(w_1 - w_2)x + (b_1 - b_2) = 0$$

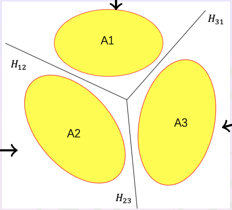
Definition (II)

We say $A_1, A_2 \subset \mathbb{R}^n$ are linearly separable, if there exist $\begin{pmatrix} w_1 \\ w_2 \end{pmatrix} \in \mathbb{R}^{2\times n}$, $\begin{pmatrix} b_1 \\ b_2 \end{pmatrix} \in \mathbb{R}^2$, such that

$$\begin{cases} w_1 x + b_1 > w_2 x + b_2, \ \forall x \in A_1, \\ w_2 x + b_2 > w_1 x + b_1, \ \forall x \in A_2. \end{cases} \tag{3}$$

Linearly separable sets: extension to k-class

$w_1 x + b_1 > w_2 x + b_2$
$w_1 x + b_1 > w_3 x + b_3$



$w_3 x + b_3 > w_1 x + b_1$
$w_3 x + b_3 > w_2 x + b_2$

$w_2 x + b_2 > w_1 x + b_1$
$w_2 x + b_2 > w_3 x + b_3$

Definition (Linearly Separable)

$A_1, ..., A_k \subset \mathbb{R}^n$ are linearly separable if there exist $W \in \mathbb{R}^{k\times d}$ and $b \in \mathbb{R}^k$ such that for each $1 \le i \le k$ and $x \in A_i$,

$$w_i x + b_i > w_j x + b_j,\ \forall j \neq i, \tag{4}$$

or equivalently,

$$p_i(x;\theta) > p_j(x;\theta),\ \forall j \neq i, \tag{5}$$

where

$$p_l(x;\theta) := \frac{e^{w_l x + b_l}}{\sum_{j=1}^k e^{w_j x + b_j}}, l = 1, 2, \cdots, k. \tag{6}$$

How to find a "classifiable" separating hyperplane?

- **Likelyhood function:**

$$\boxed{P(\boldsymbol{\theta}) = \prod_{i=1}^{k} \prod_{x \in A_i} p_i(x; \boldsymbol{\theta})}$$

- **An important observation:**

$$\boxed{P(\boldsymbol{\theta}) > \frac{1}{2} \Rightarrow \boldsymbol{\theta} \text{ is classifiable.}}$$

Maximize this function!
(Logistic Regression)

- **Transform to convex optimization problem:**

$$\boxed{\max_{\boldsymbol{\theta}} P(\boldsymbol{\theta}) \Leftrightarrow \min_{\boldsymbol{\theta}} -\log P(\boldsymbol{\theta})}$$

$P(\boldsymbol{\theta})$ **is not concave**

How to find a "classifiable" separating hyperplane?

- **Likelyhood function:**

$$\boxed{P(\boldsymbol{\theta}) = \prod_{i=1}^{k} \prod_{x \in A_i} p_i(x; \boldsymbol{\theta})}$$

- **An important observation:**

$$\boxed{P(\boldsymbol{\theta}) > \tfrac{1}{2} \Rightarrow \boldsymbol{\theta} \text{ is classifiable.}}$$

Maximize this function!
(Logistic Regression)

- **Transform to convex optimization problem:**

$$\boxed{\max_{\boldsymbol{\theta}} P(\boldsymbol{\theta}) \Leftrightarrow \min_{\boldsymbol{\theta}} -\log P(\boldsymbol{\theta})}$$

$-\log P(\boldsymbol{\theta})$ **is convex!**

Regularized Logistic Regression

Log-likelyhood Function

$$L(\boldsymbol{\theta}) = -\log P(\boldsymbol{\theta}) = \sum_{j=1}^{N} -\log p(x_j;\boldsymbol{\theta})\cdot y(x_j)$$

- $L(\boldsymbol{\theta}) > 0$.
- $\|\boldsymbol{\theta}\| \to +\infty$ when $L(\boldsymbol{\theta}) \to 0$.
- $L(\boldsymbol{\theta})$ has no global minimum.

Penalizing $\|\boldsymbol{\theta}\|$

Regularized Version

$$L_\lambda(\boldsymbol{\theta}) = L(\boldsymbol{\theta}) + \lambda\|\boldsymbol{\theta}\|^2$$

- $L_\lambda(\boldsymbol{\theta})$ is strongly convex.
- $L_\lambda(\boldsymbol{\theta})$ has global minimum.
- Will $\mathrm{argmin}_\theta\, L_\lambda(\boldsymbol{\theta})$ be classifiable?.

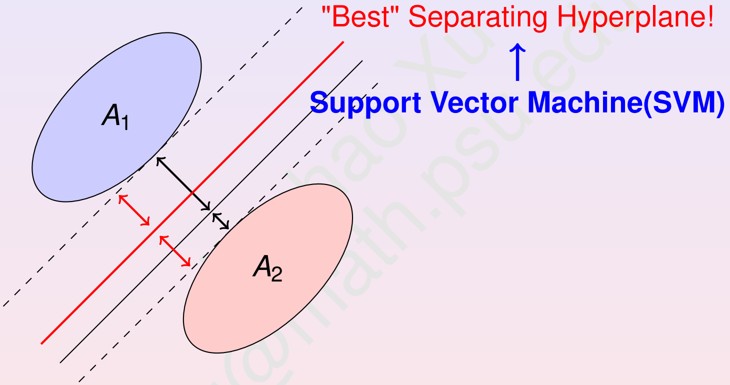
Theorem

Assume that $\{x_j\}_{j=1}^N = \cup_{i=1}^k A_i$ is linearly separable,

1. *If $\lambda = 0$, no global minimum.*
2. *If $\lambda > 0$ is sufficiently small, $W_\lambda x + b_\lambda$ classifies the sets correctly.*

Why do we need "Regularization"?

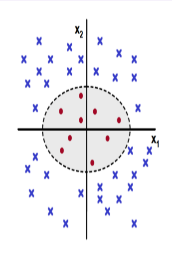
Regularized LR will approximate SVM when $\lambda \to 0$



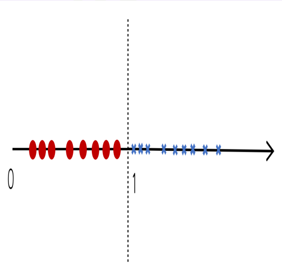
Note: Regularization help improve generalization accuracy!

Nonlinear classification models

$A_1, \cdots, A_n$ are linearly non-separable $\xrightarrow{\text{Nonlinear Map } \phi(\cdot)}$ $\phi(A_1), \cdots, \phi(A_n)$ are linearly separable!



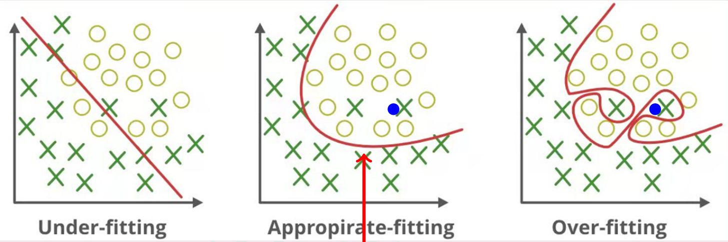
$$\xrightarrow[\text{feature map}]{\phi(x,y)=x^2+y^2}$$



Regularized loss with feature map

$$\mathcal{L}_\lambda(\boldsymbol{\theta}) = \sum_{j=1}^{N} -\log p(\phi(x_j;\theta_1);\theta_2)\cdot y(x_j) + \lambda R(\|\boldsymbol{\theta}\|) \tag{7}$$

How to find the nonlinear mapping $\phi(\cdot)$?



Desirable ϕ!

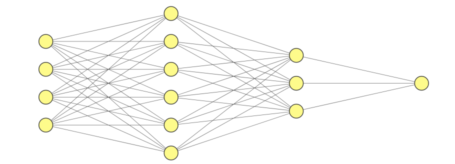
Popular choices of $\phi(\cdot)$

1. **Polynomials** $P(x) = \sum\limits_{|\alpha| \leqslant N} a_\alpha x^\alpha$.

2. **Kernel functions** in SVM ($k(x, y) = \langle \phi(x), \phi(y) \rangle$).

   e.g. Gaussian kernel: $k(x, y) = e^{-\gamma \|x-y\|^2}$, $\gamma > 0$.

3. **Deep neural networks**



Model Capacity (Representation Power)

Goal: Approximate any continuous function f on $\Omega \subset \mathbb{R}^d$.

Polynomials (Weierstrass Approximation Theorem):

$$\sum_{|\alpha| \leqslant N} a_\alpha x^\alpha \to f(x) \text{ as } N \to \infty.$$

Kernel Methods (Universal Kernel):

$$\sum_{i=1}^{N} a_i k(x_i, x) \to f(x) \text{ as } N \to \infty.$$

Neural Networks (Universal Approximation Theorem):

$$\sum_{i=1}^{N} a_i \, \sigma \, (w_i x + b_i) \to f(x) \text{ as } N \to \infty.$$

What if σ is a polynomial? **No!**

Model Capacity (Representation Power)

Goal: Approximate any continuous function f on $\Omega \subset \mathbb{R}^d$.

Polynomials (Weierstrass Approximation Theorem):

$$\sum_{|\alpha| \leqslant N} a_\alpha x^\alpha \to f(x) \text{ as } N \to \infty.$$

Kernel Methods (Universal Kernel):

$$\sum_{i=1}^{N} a_i k(x_i, x) \to f(x) \text{ as } N \to \infty.$$

Neural Networks (Universal Approximation Theorem):

$$\sum_{i=1}^{N} a_i \, \sigma \, (w_i x + b_i) \to f(x) \text{ as } N \to \infty.$$

Yes, if and only if σ is Not a polynomial!

Refs: Weierstrass 1885, Micchelli, Xu and Zhang 2006, Pinkus 1999.

Approximation Rates

$$\Sigma_n^\sigma = \left\{ \sum_{i=1}^n a_i \sigma(w_i x + b_i) : a_i, w_i, b_i \right\} \tag{8}$$

Theorem: Neural networks: *dimension independent* approximation rate

$$\inf_{f_n \in \Sigma_n^\sigma} \|f - f_n\|_{L^2} \lesssim n^{-\frac{1}{2}}. \tag{9}$$

Comparison: Piecewise polynomial: $O(n^{-\frac{1}{d}})$.

Theorem: When $\sigma = \mathrm{ReLU}^k$, we have the *sharp* rate

$$\inf_{f_n \in \Sigma_n^\sigma} \|f - f_n\|_{L^2} \lesssim n^{-\frac{1}{2}-\frac{2k+1}{2d}} \tag{10}$$

Refs: Makovoz 1996. Siegel and Xu 2021.

Examples on MNIST

$$\mathcal{L}_\lambda(\boldsymbol{\theta}) = \sum_{j=1}^{N} -\log p(\phi(x_j;\theta_1);\theta_2)\cdot y(x_j) + \lambda R(\|\boldsymbol{\theta}\|) \tag{11}$$

Model $(\phi(\cdot), R(\cdot))$	Training accuracy(%)	Test accuracy(%)
Logistic regression	93.46	92.56
SVM without kernel	87.91	87.73
SVM with polynomial kernel	99.16	97.71
One hidden layer NN (1000 neurons)	100.00	98.75

Table: Training and test accuracy of different models on MNIST.

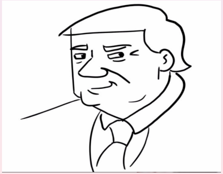
Images versus Features

Data space: g

Feature space: u

Feature extraction $\longrightarrow$

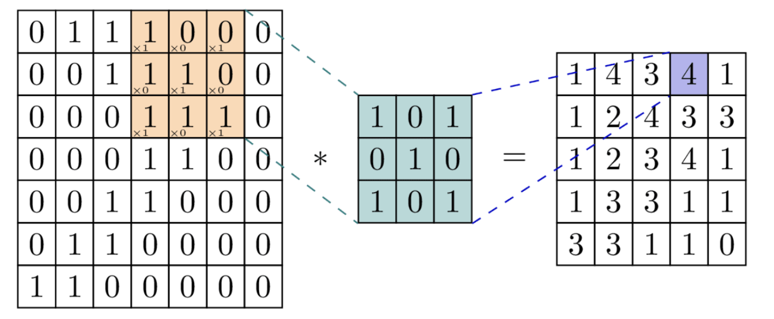
Feature extraction $\longrightarrow$



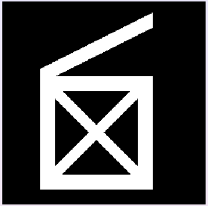
Convolution: a special class of linear functions

$$(K * x)_{i,j} := \sum_{\ell=1}^{c} \sum_{s,t=-k}^{k} K_{s+k+1,t+k+1,\ell} x_{i+s,j+t,\ell}, \tag{12}$$

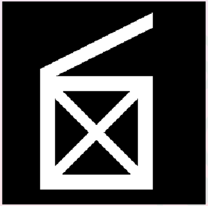
- (i, j): pixel points, ℓ: channel dimension



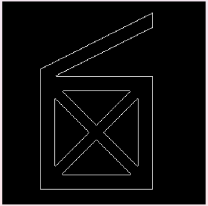
Convolution for edge detection: an example



$$\xrightarrow[K*x]{K=\begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix}}$$

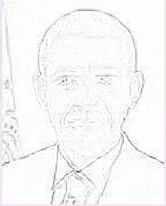


$$\xrightarrow[K*x]{K=\begin{pmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{pmatrix}}$$

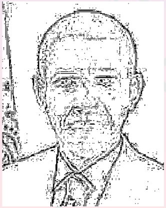


Example of feature extraction by convolution: Laplacian of Gaussian

$$\xrightarrow[\text{filter}]{\begin{pmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{pmatrix}}$$



$$\xrightarrow{\text{ReLU}}$$



$$\xrightarrow[\text{average}]{\begin{pmatrix} \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \end{pmatrix}}$$

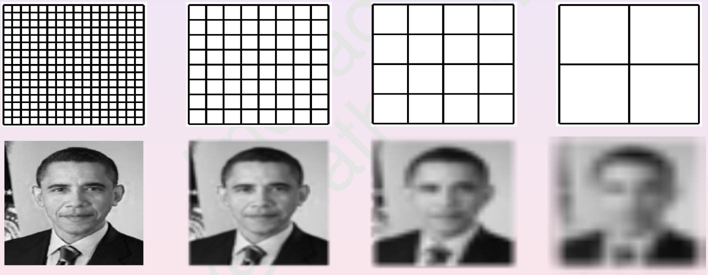
Images on multilevel grids

Images:

- Piecewise constant functions
- Initial grid $\mathcal{T}$ with size:

$$m = 2^s + 1, n = 2^t + 1$$

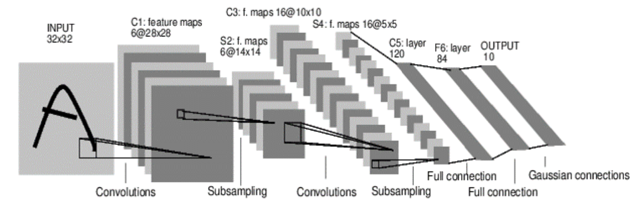
piecewise constant functions on multilevel grids



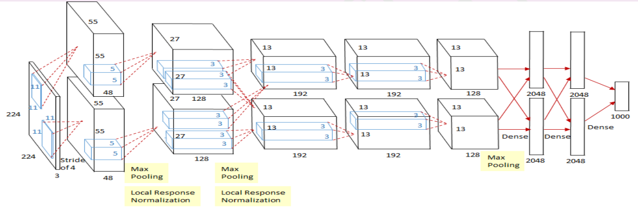
$\mathcal{V}_1$ $\mathcal{V}_2$ $\mathcal{V}_3$ $\mathcal{V}_4$

Goal: Multi-scale extraction of features

Two classical CNN examples: LeNet-5 & AlexNet



First successful CNN: LeNet-5 (Y. LeCun, L. Bottou, Y. Bengio, and P.Haffner, 1998)



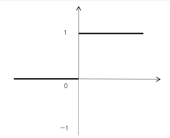
The beginning of new revolution: AlexNet (A. Krizhevsky, I. Sutskever, and G. Hinton, 2012)

Classic CNN

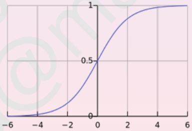
Classic CNN structure

1. Initialization of inputs: $f^{1,0} \leftarrow$ input image
2. For $\ell = 1 : J$
   - For $i = 1 : \nu_\ell$
$$f^{\ell,i} = \sigma\left(\theta^{\ell,i} * f^{\ell,i-1} + b^\ell \mathbf{1}\right) \tag{13}$$
   - Restriction-Pooling
$$f^{\ell+1,0} = R_\ell^{\ell+1} *_2 f^{\ell,\nu_\ell} \tag{14}$$
3. Output f^{J,ν_J}.

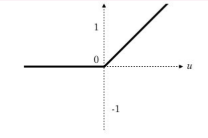
Neuron Activation Function



$\sigma = \mathrm{H}(x)$



$\sigma = \dfrac{1}{1+e^{-x}}$



$\sigma = \mathrm{ReLU}(x) := max(0, x)$

CNN: ResNet and Pre-act ResNet

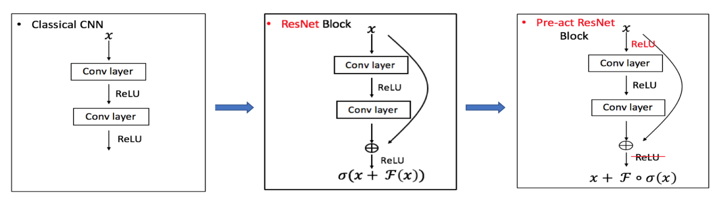


Figure: From classical CNNs to ResNet and Pre-act ResNet (K. He, X. Zhang, S. Ren and J. Sun, 2015 and 2016)

Mathematical formula

ResNet

$$r^{\ell,i} = \sigma\left(r^{\ell,i-1} + \alpha^{\ell,i} * \sigma \circ \beta^{\ell,i} * r^{\ell,i-1}\right). \tag{15}$$

Pre-act ResNet

$$r^{\ell,i} = r^{\ell,i-1} + \alpha^{\ell,i} * \sigma \circ \beta^{\ell,i} * \sigma(r^{\ell,i-1}). \tag{16}$$

Question

How to mathematically understand the feature extract process with convolution?

A mathematical model of feature extraction

$$g \xleftrightarrow{A*u=g} u$$

Model: given an image g, find its feature u satisfying

$$A * u = g \tag{17}$$

with a constraint

$$u \geq 0. \tag{18}$$

Questions:

1. What is A? (to be trained ...) $\longleftarrow$ data
2. How to solve (17)? (iterative methods) $\longleftarrow$ scientific computing

Ref: J. He and J. Xu 2019, J. He, J. Xu, L. Zhang and J. Zhu 2021.

Data and feature spaces

Partial differential equations

$$-\Delta u = g. \tag{19}$$

Constrained linear model: Given an image g, find its feature u such that

$$A * u = g. \tag{20}$$

Main idea:

- Use a geometric multigrid method for PDE (19) to solve the data-feature equation (20)!

Ref: J. He and J. Xu 2019, J. He, J. Xu, L. Zhang and J. Zhu 2021.

Iterative methods for $Au = g$: residual correction

$$u^0, u^1, \ldots, u^{k-1} \longrightarrow u^k$$

Basic ideas:

1. Form the residual: $r = g - Au^{k-1}$
2. Solve the residual eqn $Ae = r$ approximately $\hat{e} = Br$ with $B \approx A^{-1}$
3. Update $u^i = u^{i-1} + \hat{e}$

A basic iterative method:

$$u^i = u^{i-1} + B^i(g - Au^{i-1}) \tag{21}$$

An example: $A = D + L + U$ with $D = \mathrm{DIAG}(A)$

1. $B = \mathrm{DIAG}(A)^{-1}$ (Jacobi),
2. $B = \mathrm{TRIL}(A)^{-1}$ (Gauss-Seidel)

Iterative schemes for the constrained linear model

1. Recall iterative methods without constraint

$$u^i = u^{i-1} + B^i * (g - A * u^{i-1})$$

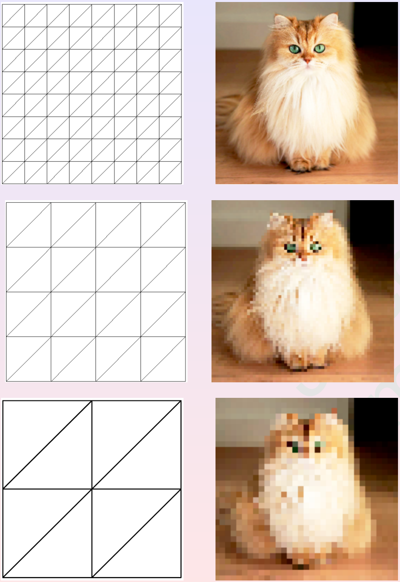
2. Image classification [$\sigma = \mathrm{ReLU}$: dropping the negative values]

$$u^i = u^{i-1} + \sigma * B^i * \sigma(g - A * u^{i-1})$$

or, in terms of residual

$$r^i = r^{i-1} - A * \sigma * B^i * \sigma(r^{i-1}).$$

MgNet: a “trained” multigrid method



Initialization of inputs:

$$g_1 \leftarrow \theta * g, \quad u_1 \leftarrow 0$$

Smoothing and restriction

- For $\ell = 1 : J$
  - For $i = 1 : \nu_\ell$

$$u^\ell \leftarrow u^\ell + \sigma \circ B^\ell * \sigma(g^\ell - A^\ell * u^\ell).$$

  - Form restricted residual and set initial guess:

$$u^{\ell+1,0} \leftarrow \sigma \circ \Pi_\ell^{\ell+1} *_2 u^\ell,$$

$$g^{\ell+1} \leftarrow \sigma \circ R_\ell^{\ell+1} *_2 (g^\ell - A^\ell * u^\ell) + A^{\ell+1} * u^{\ell+1,0}.$$

Output: . . .

$$\phi(g) \leftarrow u^J$$

Ref: He, J. & Xu, J. (2019)

Batch normalization

DNN as example:

- Original model

$$\begin{cases} f^1(x^i) & = W^1 x^i, \\ f^\ell & = W^\ell \sigma(f^{\ell-1}), \quad \ell = 2, \dots, L. \end{cases} \tag{22}$$

- Batch normalization:
  - For t-th step of SGD training on mini-batch $\mathcal{B}_t$
  - For the ℓ-th layer

$$\begin{aligned} \mu^\ell_{\mathcal{B}_t} &\leftarrow \frac{1}{m} \sum_{i \in \mathcal{B}_t} f^\ell(x^i) && \text{mini-batch mean} \\ \sigma^\ell_{\mathcal{B}_t} &\leftarrow \frac{1}{m} \sum_{i \in \mathcal{B}_t} (f^\ell(x^i) - \mu_{\mathcal{B}_t})^2 && \text{mini-batch variance} \\ \hat{f}^\ell(x) &\leftarrow \frac{f^\ell(x) - \mu^\ell_{\mathcal{B}_t}}{\sqrt{\sigma^\ell_{\mathcal{B}_t} + \epsilon}} && \text{normalize} \\ \mathrm{BN}_{\mathcal{B}_t}(f^\ell) &\leftarrow \gamma^\ell \hat{f}^\ell(x) + \beta^\ell && \text{scale and shift} \end{aligned} \tag{23}$$

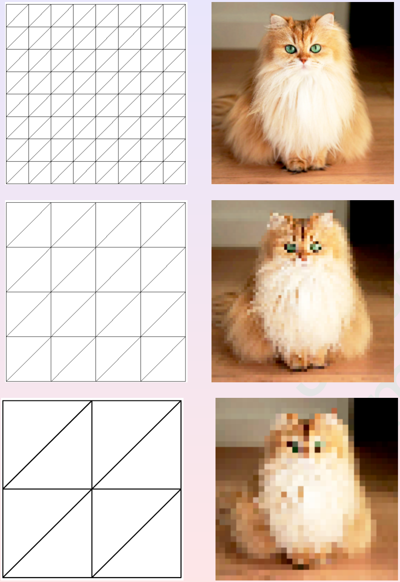
- Model with batch normalization

$$\begin{cases} \tilde{f}^1(x^i) & = W^1 x^i, \\ \tilde{f}^\ell & = W^\ell \sigma_{BN}(\tilde{f}^{\ell-1}), \quad \ell = 2, \dots, L, \end{cases} \tag{24}$$

where

$$\sigma_{BN}(f) = \sigma\left(\mathrm{BN}_{\mathcal{B}_t}(f)\right). \tag{25}$$

Practical MgNet with bath normalization



Initialization of inputs: $$g_1 \leftarrow \theta * g, \quad u_1 \leftarrow 0$$

Smoothing and restriction

- For $\ell = 1 : J$
  - For $i = 1 : \nu_\ell$

    $$u^\ell \leftarrow u^\ell + \sigma_{BN} \circ B^\ell * \sigma_{BN}(g^\ell - A^\ell * u^\ell).$$

  - Form restricted residual and set initial guess:

    $$u^{\ell+1,0} \leftarrow \sigma_{BN} \circ \Pi_\ell^{\ell+1} *_2 u^\ell,$$

    $$g^{\ell+1} \leftarrow \sigma_{BN} \circ R_\ell^{\ell+1} *_2 (g^\ell - A^\ell * u^\ell) + A^{\ell+1} * u^{\ell+1,0}.$$

- ...

Output:

$$\phi(g) \leftarrow u^J$$

Denote: $\sigma = \sigma_{BN}$ for CNNs in image classification by default.

- Ref: He, J. & Xu, J. (2019)

Pre-act ResNet: a "residual" version of MgNet

Theorem (MgNet and pre-act ResNet, He and Xu 2019)

The MgNet model recovers the pre-act ResNet (K. He et al 2016) as follows

$$r^{\ell,i} = r^{\ell,i-1} + A^{\ell,i} * \sigma \circ B^{\ell,i} * \sigma(r^{\ell,i-1}), \quad i = 1 : \nu_\ell, \tag{26}$$

where

$$r^{\ell,i} = g^\ell - A^\ell * u^{\ell,i}.$$

Proof.

$$\begin{aligned} u^{\ell,i} &= u^{\ell,i-1} + \sigma \circ B^{\ell,i} * \sigma(g^\ell - A^\ell * u^{\ell,i-1}), \\ \Rightarrow A^\ell * u^{\ell,i} &= A^\ell * u^{\ell,i-1} + A^\ell * \sigma \circ B^{\ell,i} * \sigma(g^\ell - A^\ell * u^{\ell,i-1}), \\ \Rightarrow g^\ell - A^\ell * u^{\ell,i} &= g^\ell - A^\ell * u^{\ell,i} - A^\ell * \sigma \circ B^{\ell,i} * \sigma(g^\ell - A^\ell * u^{\ell,i-1}), \\ \Rightarrow r_\ell^i = r^{\ell,i} &= r^{\ell,i-1} + A^{\ell,i} * \sigma \circ B^{\ell,i} * \sigma(r^{\ell,i-1}). \end{aligned} \tag{27}$$

$\square$

Modified Pre-act ResNet, ResNet

Modified Pre-act ResNet – Pre-act ResNet-A^ℓ

$$r^{\ell,i} = r^{\ell,i-1} + A^\ell * \sigma \circ B^{\ell,i} * \sigma(r^{i-1}). \tag{28}$$

Modified ResNet – ResNet-A^ℓ

$$r^{\ell,i} = \sigma\left(r^{\ell,i-1} + A^\ell * \sigma \circ B^{\ell,i} * r^{i-1}\right) \tag{29}$$

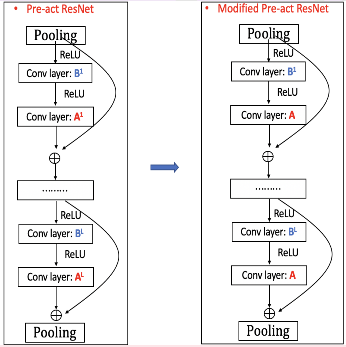


Figure: Diagram of modified models.

Modify (pre-act) ResNet numerical experiments

Table: Accuracy and number of parameters for ResNet, pre-act ResNet, and their variants of modified versions on CIFAR10 and CIFAR100.

Model	CIFAR10	CIFAR100	# Parameters
ResNet18-$A^{\ell,i}$-$B^{\ell,i}$	94.22	76.08	11M
ResNet18-A^{ℓ}-$B^{\ell,i}$	**94.34**	**76.32**	8.1M
ResNet18-$A^{\ell,i}$-B^{ℓ}	93.95	74.23	9.7M
ResNet18-A^{ℓ}-B^{ℓ}	93.30	74.85	6.6M
pre-act ResNet18-$A^{\ell,i}$-$B^{\ell,i}$	94.31	76.33	11M
pre-act ResNet18-A^{ℓ}-$B^{\ell,i}$	**94.54**	**76.43**	8.1M
pre-act ResNet18-$A^{\ell,i}$-B^{ℓ}	93.96	74.45	9.7M
pre-act ResNet18-A^{ℓ}-B^{ℓ}	93.63	74.46	6.6M
ResNet34-$A^{\ell,i}$-$B^{\ell,i}$	94.43	76.31	21M
ResNet34-A^{ℓ}-$B^{\ell,i}$	**94.78**	**76.44**	13M
ResNet34-$A^{\ell,i}$-B^{ℓ}	93.98	74.48	15M
ResNet34-A^{ℓ}-B^{ℓ}	93.55	74.46	6.7M
pre-act ResNet34-$A^{\ell,i}$-$B^{\ell,i}$	94.70	77.38	21M
pre-act ResNet34-A^{ℓ}-$B^{\ell,i}$	**94.91**	**77.41**	13M
pre-act ResNet34-$A^{\ell,i}$-B^{ℓ}	94.08	75.32	15M
pre-act ResNet34-A^{ℓ}-B^{ℓ}	94.01	74.12	6.7M

MgNet: From multigrid to CNN

Multigrid:

- A^ℓ, B^ℓ, R^ℓ are all given a priori

CNN:

- Almost identically same structure as multigrid!
- $A^{\ell,i}$, $B^{\ell,i}$, $R^{\ell,i}$ are all trained!
- Activation, ReLU, is introduced (to drop-off negative pixel values).
- Extra channels are introduced.

CNN versus multigrid: classic approaches versus MgNet

1. CNN:

   Classic: Almost all the components are unrelated and need to be trained
   MgNet: Most of the components are related and can be given a priori

2. Multigrid

   Classic: Almost all the components are a priori given
   MgNet: Some of components can be trained!

MgNet versus other CNNs

Model	Accuracy	# Parameters
ResNet18	95.28	11.2M
pre-act ResNet18	95.08	10.2M
MgNet[2,2,2,2],256	96.00	8.2M

Table: The comprison of MgNet and classical CNN on Cifar10

Model	Accuracy	# Parameters
ResNet18	77.54	11.2M
pre-act ResNet18	77.29	11.2M
MgNet[2,2,2,2],256	79.94	8.3M
MgNet[2,2,2,2],512	81.35	33.1M
MgNet[2,2,2,2],1024	82.46	132.2M

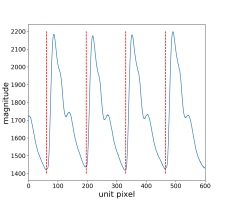
Table: The comprison of MgNet and classical CNN on Cifar100

Model	Accuracy	Parameters
ResNet18	72.12	11.2M
MgNet[2,2,2,2], [64,128,256,512]	73.36	13.0M
MgNet[3,4,6,3],[128,256,512,1024]	78.59	71.3M

Table: The comprison of MgNet and classical CNN on ImageNet.

Application: Pulse-Feeling

An ancient technique in Traditional Chinese Medication (TCM)



- It has been widely believed and claimed to be accurate
- No record of clinical trials nor quantitative studies
- it is a valid technique or it is ... ?

Deep learning for diagnosing pregnancy

model	test accuracy(%)	AUC(%)	size
ResNet	84.73	89.66	232,642
MgNet	84.68	91.04	3,450
SVM	78.08	71.32	
logistic regression	79.10	74.27	

Ref: Chen, Huang, Hao and Xu 2019

Two types of basis function

- Hat basis:

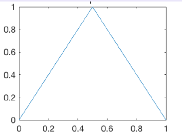
$$\varphi(x) = \begin{cases} 2x & x \in [0, \frac{1}{2}] \\ 2(1-x) & x \in [\frac{1}{2}, 1] \\ 0, & \text{others} \end{cases}.$$

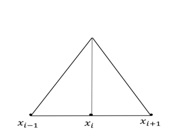
$$\varphi_i(x) = \varphi(\frac{x - x_{i-1}}{2h}) = \varphi(w_h x + b_i).$$

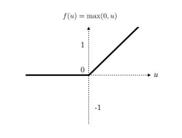
with $w_h = \frac{1}{2h}$, $\quad b_i = \frac{-x_{i-1}}{2h}$.

- ReLU basis: $\mathrm{ReLU}(x) = \max(0, x)$ and

$$r_i(x) = \mathrm{ReLU}(\frac{x - x_{i-1}}{2h}) = \mathrm{ReLU}(w_h x + b_i)$$







Background

Regularization:

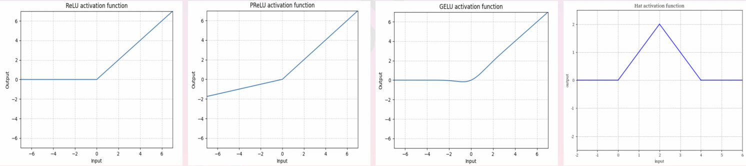
- Frequency principle claims that ReLU makes the networks prioritize learning the low frequency modes, which is one main reason for good generalization accuracy.

Ref: N. Rahaman, A. Baratin, Y. Bengio, A. Courville (2019).

Vanish gradient:

- A small support is thought to be more likely to cause vanish gradient problem.

PReLU, GELU: Variants of ReLU to improve the vanish gradient problem.



ReLU *PReLU* *GELU* *Hat*

Motivation

Our consideration:

- The convergence of frequency components of error for Hat neural networks is different from ReLU.
- The variation of frequency is also important in CNN and image.

Main idea:

- Make use of the property of Hat function in CNN.
- Give the design of CNN models from FEM and Multigrid.

Hat-CNN-MgNet versus ReLU-CNN-MgNet

Dataset	Model	#Parameters	Activation function	Test accuracy
CIFAR10	ResNet	11.2M	ReLU Hat	94.64 **94.79**
	MgNet	3.1M	ReLU Hat	94.74 **94.95**
CIFAR100	ResNet	11.2M	ReLU Hat	76.21 **76.47**
	MgNet	5.7M	ReLU Hat	77.07 **77.54**
ImageNet	ResNet	11.2M	ReLU Hat	72.12 **72.46**
	MgNet	9.9M	ReLU Hat	72.36 **72.69**

Table: Comparison of Hat-CNNs and ReLU-CNNs for image classification.

Observations:

- MgNet compares competitively with classic CNN
- Using hat functions gives a **comparable or even better** accuracy.

Ref: He, J. & Xu, J. (2019), He, J., Xu, J., Zhang, L. & Zhu, J. (2021), Wang, J., Xu, J. & Zhu, J. (2022).

Experiment: CNNs with Hat function

Table: Comparison of different support setting of Hat function for MgNet.

Dataset	Activation function	Test accuracy
MNIST	Hat-[5,10,15,20]	**99.68**
	Hat-[20,15,10,5]	99.64
CIFAR10	Hat-[5,10,15,20]	**93.23**
	Hat-[20,15,10,5]	92.87
CIFAR100	Hat-[5,10,15,20]	**70.96**
	Hat-[20,15,10,5]	70.56
ImageNet	Hat-[10,20,30,40]	**72.69**
	Hat-[40,30,20,10]	71.87

Observation:

Coarser resolution (deeper) layer needs a larger support.

Experiment: CNNs with Hat function

Table: MgNet with Hat function of trainable support.

Dataset	Test accuracy	Initial support	Final support
CIFAR10	**94.15**	[5,10,15,20]	[1.05,1.69,1.77,3.25]
	93.99	[20,15,10,5]	[1.26,1.75,2.83,3.26]
	93.23 for fix support [5,10,15,20]		
CIFAR100	72.24	[5,10,15,20]	[1.42,2.54,2.74,9.64]
	72.18	[20,15,10,5]	[2.00,2.43,2.73,9.80]
	70.96 for fix support [5,10,15,20]		

Observation:

Trainable Hat function gives better results and even smaller support.

A summary of MgNet

- J. Xu, Deep Neural Networks and Multigrid Methods, (Lecture Notes at Penn State and KAUST), 2023.
- J. He, J. Xu. MgNet: A Unified Framework of Multigrid and Convolutional Neural Network. Sci China Math, 2019, 62: 1331-1354.
- Y. Chen, B. Dong, J. Xu. Meta-MgNet: Meta Multigrid Networks for Solving Parameterized Partial Differential Equations Image Classification. Journal of Computational Physics, 2022,455.
- J. He, L. Li, J. Xu. Approximation Properties of Deep ReLU CNNs. Research in the Mathematical Sciences, 2022, 9(3).
- J. He, J .Xu, L. Zhang, J. Zhu. An interpretive constrained linear model for ResNet and MgNet. Neural Networks, 2023, 162: 384-392.

1. A uniform framework for understanding and designing CNNs: ResNet, U-Net, DenseNet . . .
2. Construct the new
   - reduce the free parameters 1%, .1%, .01%...?
   - increase the generalization accuracy,
   - accelerate the training speed,
   - extend to general graph models,
   - ...
3. Applications:
   - image problems,
   - time series forecasting,
   - numerical PDEs, in particular for operator learning,
   - ...

Challenging Objective: MgNet vs Transformer

Can MgNet outperform Transformer?

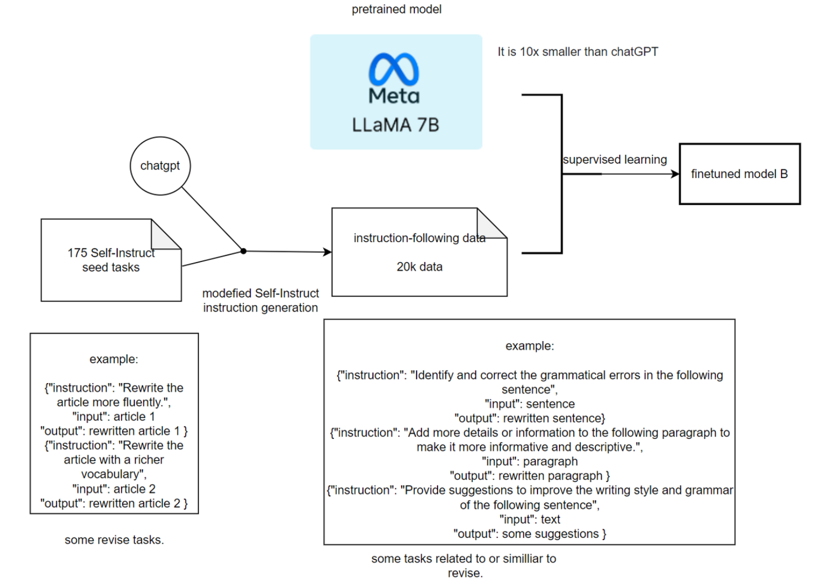
Model	Type	Accuracy	Parameters
DeiT-Small	Transformer	79.8	22.1M
PVT-Small	Transformer	79.8	24.5M
ConvMixer	Transformer	80.2	21.1M
CrossViT-Small	Transformer	81.0	26.7M
Swin-Tiny	Transformer	81.2	28.3M
CvT-13	Transformer	81.6	20.0M
CoAtNet-0	Transformer	81.6	25.0M
CaiT-XS-24	Transformer	81.8	26.6M
ResNet-50	CNN	80.4	25.0M
MgNet-small	CNN	**81.0**	26.1M
MgNet	CNN	**82.0**	39.3M
CMT-XS	CNN+Transformer	81.8	15.2M
MgNet-CMT-XS	CNN +Transformer	**82.6**	17.9M
MgNet-CMT	CNN +Transformer	**83.4**	30.1M

Table: ImageNet results of transformers and CNNs

Observation:

MgNet has competitive performance with transformer models.

An ongoing project: ReviseGPT



An ongoing project: ReviseGPT

Average score	Input	ChatGPT-turbo	ReviseGPT
Testing data(500 arxiv abstract)	82.65	94.23	94.08
Training data(500 arxiv abstract)	85.33	94.83	95.2
Testing data(20 general text: part of blog, news)	83.7	95.8	94.85

* What can we do?
* Model, training algorithms, applications

- Evaluation: using the average score generated by Grammarly in a manually-created dataset, Grammarly is a website that can score texts in terms of grammar, spelling and clarity
- ReviseGPT is a model trained only for revising English texts