Method that was used in *Product Recommendation based on images*:

1. Get embedding: ResNet

ResNet solves the notorious problem of vanishing gradient and not convergence in normal neural network problems. It helped reduce error/loss as the number of layers increases.

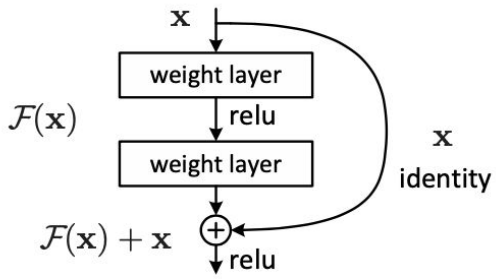


Figure 2. Residual learning: a building block.

From the construction of ResNet, we can see how it works. The idea is to add an outlet for skipping layers. It is institutionally understandable that ResNet construction provides an option of skipping layers when it doesn't reduce loss. But how to explain it in theory? How do we theoretically prove that ResNet could overcome the problem of no convergence as hidden layers explode?

As is mentioned in "Related Work" of ResNet essay [4]: In low-level vision and computer graphics, for solving Partial Differential Equations (PDEs), the widely used Multigrid method reformulates the system as subprob- lems at multiple scales, where each subproblem is respon- sible for the residual solution between a coarser and a finer scale. An alternative to Multigrid is hierarchical basis pre- conditioning, which relies on variables that repre- sent residual vectors between two scales. It has been shown that these solvers converge much faster than standard solvers that are unaware of the residual nature of the solutions. These methods suggest that a good reformulation or preconditioning can simplify the optimization.

Remember in normal neural network problem, target function $f = h_1(a_1 h_2 + b_1)$ and $h_1 = h_2(a_2 h_3 + b_2)$, etc… which causes the problem of vanishing gradient in backpropagation as the hidden layer explodes. The normal neural network is stacking layers and minimizing the loss function of f. However, ResNet minimizes the loss of residual, because each hidden layer, as in figure above, is $H(x) = F(x) + x$. Adding this shortcut connection *x* can transform the target as minimizing residuals and doesn't add parameters as well.

Let's explain the basic logics of why $H(x) = F(x) + x$ works:

Image processing is essentially a discretized Boundary Value Problem (BVP), because we need to minimize loss of “interpolation” in each pixel boundary. And in auto-encoder problems, like in our image processing, we are seeking to restitute the input image in a lower-embedding projection. The target function *f* makes the output smoother and more close to the original pixels. As said in [3], Many problems in computer graphics can be formulated using a variational approach, which involves defining a continuous two dimensional optimization problem, adding appropriate regularization terms, and then discretizing the problem on a regular grid [Gortler and Cohen 1995; Zhang et al. 2002; Fattal et al. 2002; P´erez et al. 2003; Levin et al. 2004a; Levin et al. 2004b].

To solve the BVP problem, there are two alternative methods: multigrid method & hierarchical basis preconditioning. Multigrid method shows the residual function can reach convergence at a rate of log, which is totally different from the unreferenced function. And that theory supports the idea of ResNet construction.

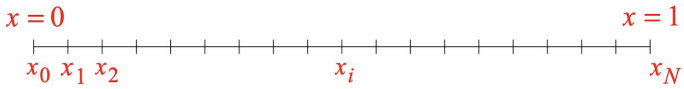
Proof of convergence:

Model Problems

- One-dimensional boundary value problem:

$$-u''(x) + \sigma u(x) = f(x) \qquad 0 < x < 1, \qquad \sigma > 0$$
$$u(0) = u(1) = 0$$

- Grid: $h = \frac{1}{N}$, $x_i = ih$, $i = 0, 1, \ldots N$



- Let $v_i \approx u(x_i)$ and $f_i \approx f(x_i)$ for $i = 0, 1, \ldots N$

We use Taylor Series to derive an approximation to u''(x)

- We approximate the second derivative using Taylor series:

$$u(x_{i+1}) = u(x_i) + hu'(x_i) + \frac{h^2}{2!}u''(x_i) + \frac{h^3}{3!}u'''(x_i) + O(h^4)$$

$$u(x_{i-1}) = u(x_i) - hu'(x_i) + \frac{h^2}{2!}u''(x_i) - \frac{h^3}{3!}u'''(x_i) + O(h^4)$$

- Summing and solving,

$$u''(x_i) = \frac{u(x_{i+1}) - 2u(x_i) + u(x_{i-1})}{h^2} + O(h^2)$$

We approximate the equation with a finite difference scheme

- We approximate the BVP

$$-u''(x) + \sigma u(x) = f(x) \qquad 0 < x < 1, \qquad \sigma > 0$$
$$u(0) = u(1) = 0$$

with the finite difference scheme:

$$\frac{-v_{i-1} + 2v_i - v_{i+1}}{h^2} + \sigma v_i = f_i \qquad i = 1, 2, \ldots N-1$$

$$v_0 = v_N = 0$$

The discrete model problem

- Letting $\boldsymbol{v} = (v_1, v_2, \ldots, v_{N-1})^T$ and

$$\boldsymbol{f} = (f_1, f_2, \ldots, f_{N-1})^T$$

we obtain the matrix equation $A\boldsymbol{v} = \boldsymbol{f}$ where A is (N-1) x (N-1), symmetric, positive definite, and

$$A = \frac{1}{h^2}\begin{pmatrix} 2+\sigma h^2 & -1 & & & & \\ -1 & 2+\sigma h^2 & -1 & & & \\ & -1 & 2+\sigma h^2 & -1 & & \\ & & & -1 & 2+\sigma h^2 & -1 \\ & & & & -1 & 2+\sigma h^2 \end{pmatrix}, \quad \boldsymbol{v} = \begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ \\ v_{N-2} \\ v_{N-1} \end{pmatrix}, \quad f = \begin{pmatrix} f_1 \\ f_2 \\ f_3 \\ \\ f_{N-2} \\ f_{N-1} \end{pmatrix}$$

Start from one-dimensional BVP, and generate the linear system. We then have many methods to solve it, including:
Direct – Gaussian elimination – Factorization
Iterative – Jacobi – Gauss-Seidel – Conjugate Gradient, etc.

Next, we move on to the 2-dimensional BVP problem.

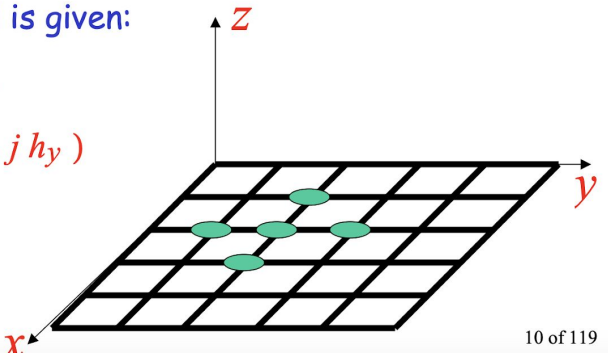
A two-dimensional boundary value problem

- Consider the problem:

$$-u_{xx} - u_{yy} + \sigma u = f(x,y), \quad 0 < x < 1, \quad 0 < y < 1$$
$$u = 0, \; x = 0, x = 1, \; y = 0, y = 1; \qquad \sigma > 0$$

- Where the grid is given:

$$h_x = \frac{1}{M}, \quad h_y = \frac{1}{N},$$
$$(x_i, y_j) = (i\,h_x, j\,h_y)$$
$$0 \le i \le M$$
$$0 \le j \le N$$



Yields the linear system

- We obtain a block-tridiagonal system $Av = f$:

$$\begin{pmatrix} A_1 & -I_y & & & \\ -I_y & A_2 & -I_y & & \\ & -I_y & A_3 & -I_y & \\ & & & -I_y \; A_{N-2} & -I_y \\ & & & & -I_y \; A_{N-1} \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ \\ v_{N-1} \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \\ f_3 \\ \\ f_{N-1} \end{pmatrix}$$

where I_y is a diagonal matrix with $\frac{1}{h_y^2}$ on the diagonal and

$$A_i = \begin{pmatrix} \frac{1}{h_x^2}+\frac{1}{h_y^2}+\sigma & -\frac{1}{h_x^2} & & & \\ -\frac{1}{h_x^2} & \frac{1}{h_x^2}+\frac{1}{h_y^2}+\sigma & -\frac{1}{h_x^2} & & \\ & -\frac{1}{h_x^2} & \frac{1}{h_x^2}+\frac{1}{h_y^2}+\sigma & -\frac{1}{h_x^2} & \\ & & \ddots & \ddots & \ddots \\ & & & -\frac{1}{h_x^2} & \frac{1}{h_x^2}+\frac{1}{h_y^2}+\sigma \end{pmatrix}$$

To solve the constraint optimization problem, we use iterative method.

Iterative Methods for Linear Systems

- Consider $Au = f$ where A is NxN and let v be an approximation to u.

- Two important measures:
  - The Error: $e = u - v$, with norms

$$\|e\|_\infty = \max |e_i| \qquad \|e\|_2 = \sqrt{\sum_{i=1}^{N} e_i^2}$$

  - The Residual: $r = f - Av$ with

$$\|r\|_\infty \qquad \|r\|_2$$

Relaxation Schemes

- Consider the 1D model problem

$$-u_{i-1} + 2u_i - u_{i+1} = h^2 f_i \qquad 1 \le i \le N-1 \qquad u_0 = u_N = 0$$

- Jacobi Method (simultaneous displacement): Solve the i^{th} equation for v_i holding other variables fixed:

$$v_i^{(new)} = \frac{1}{2}\left(v_{i-1}^{(old)} + v_{i+1}^{(old)} + h^2 f_i \right) \qquad 1 \le i \le N-1$$

"Fundamental theorem of iteration"

- R is convergent (that is, $R^n \to 0$ as $n \to \infty$) if and only if $\rho(R) < 1$, where

$$\rho(R) = \max \{ |\lambda_1|, |\lambda_2|, \ldots, |\lambda_N| \}$$

therefore, for any initial vector $v^{(0)}$, we see that $e^{(n)} \to 0$ as $n \to \infty$ if and only if $\rho(R) < 1$.

- $\rho(R) < 1$ assures the convergence of the iteration given by R and $\rho(R)$ is called the *convergence factor* for the iteration.

Convergence Rate

- How many iterations are needed to reduce the initial error by 10^{-d} ?

$$\frac{\|e^{(M)}\|}{\|e^{(0)}\|} \leq \|R^M\| \sim (\rho(R))^M \sim 10^{-d}$$

- So, we have $M = \dfrac{d}{\log_{10}\left(\dfrac{1}{\rho(R)}\right)}$.

- The *convergence rate* is given:

$$\text{rate} = \log_{10}\left(\frac{1}{\rho(R)}\right) = -\log_{10}(\rho(R)) \ \frac{\text{digits}}{\text{iteration}}$$

Convergence analysis for weighted Jacobi on 1D model

$$R_\omega = (1-\omega)I + \omega D^{-1}(L+U)$$
$$= I - \omega D^{-1}A$$

$$R_\omega = I - \frac{\omega}{2}\begin{pmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & -1 & 2 & -1 & \\ & & \ddots & \ddots & \ddots \\ & & & -1 & 2 \end{pmatrix}$$

$$\lambda(R_\omega) = 1 - \frac{\omega}{2}\lambda(A)$$

For the 1D model problem, he eigenvectors of the weighted Jacobi iteration and the eigenvectors of the matrix A are the same! The eigenvalues are related as well.

2. Compute similarity: Cosine

Cosine metric is used in this project to measure similarities between images. Future works should focus on Wasserstein distance, which results from a partial differential equa- tion (PDE) formulation of Monge's optimal transport problem, and was proved to

be an efficient numerical solution method for solving Monge's problem and image comparison.

Reference：

1. https://arxiv.org/pdf/1612.00181.pdf%20https://en.wikipedia.org/wiki/Scale-invariant_feature_transform Monge's Optimal Transport Distance for Image Classification
2. W. L. Briggs, S. F. McCormick, et al. A Multigrid Tutorial. Siam, 2000.
3. R. Szeliski. Locally adapted hierarchical basis preconditioning. In SIGGRAPH, 2006.
4. Kaiming He, et, Deep Residual Learning for Image Recognition