Tsinghua University

Advanced Computer Vision

THU×SENSETIME – 80231202

sensetime

Chapter 1 - Section 4

CNN & High-level Feature Extraction

Dr. Liu Yu

Friday, March 12, 2021

Acknowledge : Song Guanglu , Liu Boxiao , Zhang Manyuan

Assignment Introduction

- **Assignment**
  - All assignments should be finished by one person
  - You can finish assignment on your local machines or on clusters provided by SenseTime
  - More details will be update on Course Homepage

Assignment	Released Date	Due Date	Topic
Assignment 1	Mar. 19	Apr. 2	Deep learning training framework and model optimization implementation
Assignment 2	Apr. 2	May. 7	Advanced Computer Vision Tasks
Assignment 3	May. 7	May. 30	Lightweight Model Quantization and Model Pruning

Assignment 1

实验目标：特定场景和限制下的模型设计与优化

实验描述：

在给定网络结构Flops的限制下，实现对于给定数据集（人脸识别子集）上的模型设计、训练以及验证调优。数据集中会有当前工业界和学术集通用的研究问题，如类别不均衡、数据噪声等因素存在。

实验内容

实验考察对于神经网络标准训练流程的设计与处理，包括数据分析处理、模型设计、训练算法设计、模型调优、结果分析等基本能力，需要基于给定的数据集实现对于评测集的性能调优，记录实验过程及调优方案并且完成一些问答题目，包括：

1.熟悉标准的神经网络训练代码搭建；

2.标准数据集的预处理流程；

3.分析数据集分布，确定Optimizer的配置、Augmentation的选取、训练优化的配置，并实践运用；

4.熟悉模型Flops和参数量计算，网络模型结构设计与调优；

5.掌握常用Evaluation Metrics的计算方式，了解基本的模型性能分析方法以及特征可视化方法

Description

Training data samples:

The number of images is on the order of 10^5.

The training data contains label noise:

intra-class noise: different identities with same label ID.

inter-class noise: same identity with different label IDs.

Test data samples:

The number of images is on the order of 10^4.

Evaluation protocol:

TPR @ FPR1e-5

TPR = TP(true positives) / (TP(true positives) + FN(false negatives))

FPR = FP(false positives) / (FP(false positives) + TN(true negatives))

Constraint:

1.The Flops of submitted model should be less than 500M madds (single image inference).

2.External training data is not allowed.

Timeline

报名分组截止	2021.3.18
数据下载开放	2021.3.15 [包含训练样例代码]
提交开放	2021.3.19
	每人每天有2次提交机会
提交截止	2021.4.2

实验资源使用说明

使用原则

- 集群仅能通过清华校园网进行访问和使用
- 仅供完成实验作业及大作业使用，请勿用于其他用途
- 优先供已选课同学使用

集群资源分组

- 分为6组，每组不超过10人
- 每组设1名组长，由组长对本组服务器进行统一协调和管理
- 右侧扫码报名分组

集群使用

- 访问方式：跳板机+VM
- 每组一套账号及秘钥
- 请妥善保管，不可外泄
- 集群使用如有问题，请在群内及时反馈

其他说明

- 实验作业不会占用太多计算资源
- 如同学们个人或所在实验室有GPU服务器资源，建议可以把公共资源留给更需要的同学
- 校外已选课同学如无法访问清华校园网，且个人无法解决服务器资源，请在问卷中注明

4.1 Neural Network Basics

Dr. Liu Yu

Friday, March 12, 2021

Outline

Part 1	Neural Network Overview
Part 2	Activation functions and gradient descent
Part 3	Deep L-layer neural network
Part 4	Regularization and optimization

Highlights

Learn the forward and backward propagation of neural network

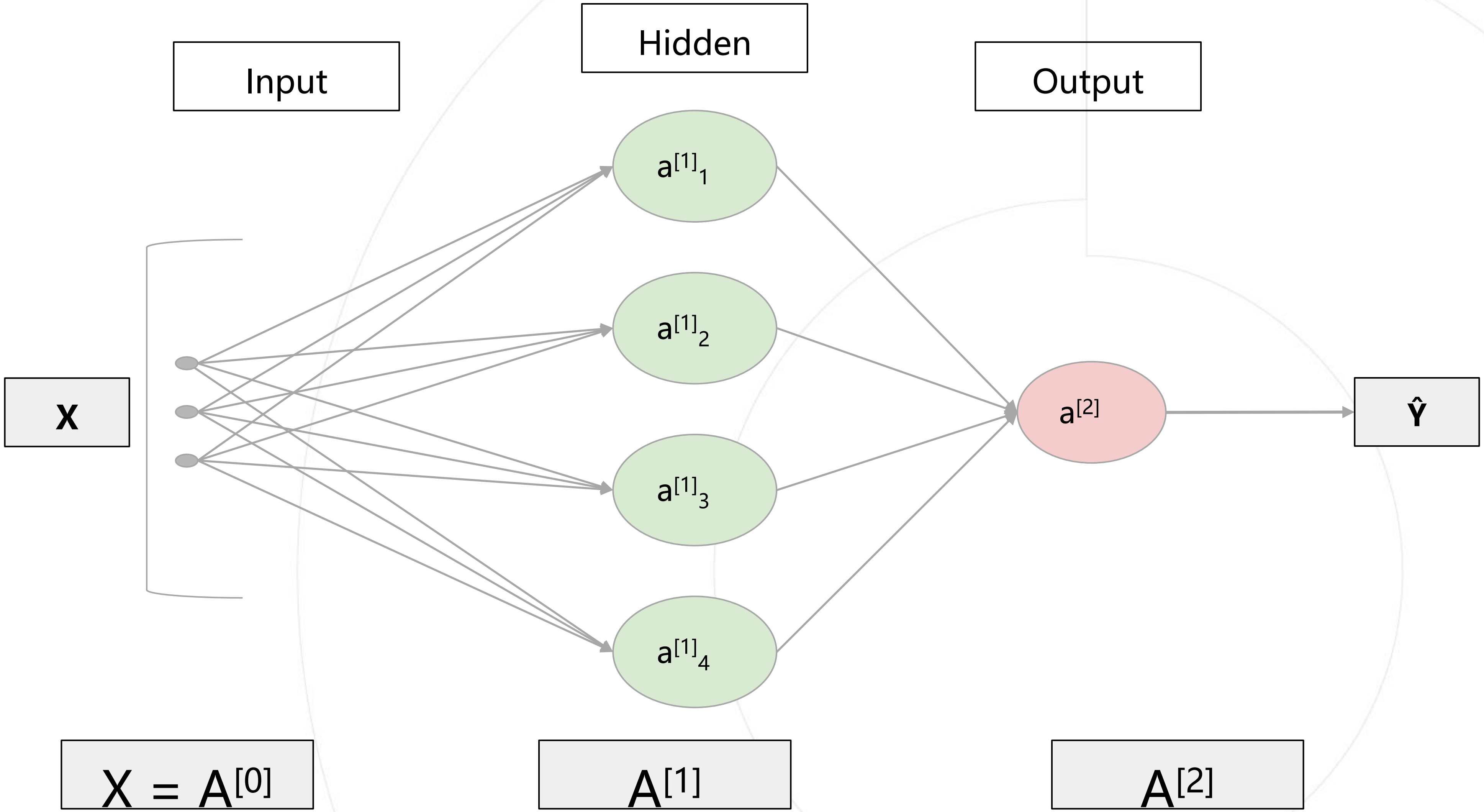
Learn the activation functions in neural networks

Learn the hyper-parameters in neural network training

Understand the regularization methods and optimization methods in neural network

Understand the phenomenon of overfitting in neural network training and its solution

Neural Network Overview



Neural Network Overview

Input → Hidden

$$\left.\begin{matrix} x \\ W^{[1]} \\ b^{[1]} \end{matrix}\right\} = z^{[1]} = W^{[1]}x + b^{[1]} \Longrightarrow a^{[1]} = \sigma\left(z^{[1]}\right)$$

Hidden → Output

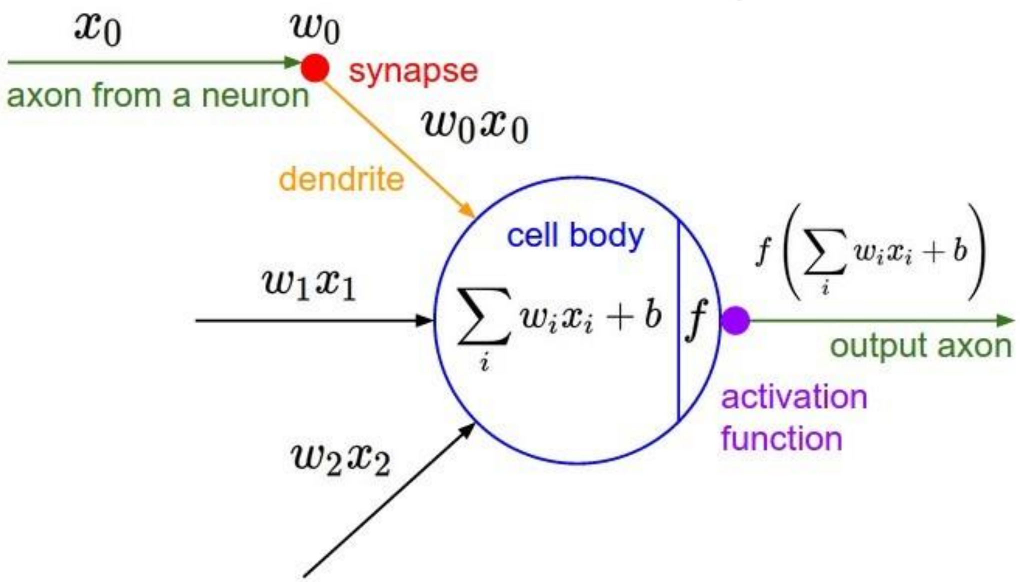
$$\left.\begin{matrix} a^{[1]} = \sigma\left(z^{[1]}\right) \\ W^{[2]} \\ b^{[2]} \end{matrix}\right\} \Rightarrow z^{[2]} = W^{[2]}a^{[1]} + b^{[2]} \Longrightarrow a^{[2]} = \sigma\left(z^{[2]}\right) \Longrightarrow L\left(a^{[2]}, y\right)$$

Derivation

$$\left.\begin{matrix} da^{[1]} = d\sigma\left(z^{[1]}\right) \\ dW^{[2]} \\ db^{[2]} \end{matrix}\right\} \Longleftarrow dz^{[2]} = d\left(W^{[2]}\alpha^{[1]} + b^{[2]}\right) \Longleftarrow da^{[2]} = d\sigma\left(z^{[2]}\right) \Longleftarrow dL\left(a^{[2]}, y\right)$$

Neural Network Overview

- Computing a Neural Network's output



$$z = w^T x + b$$

$$a = \sigma(z)$$

$$a^{[1]} = \begin{bmatrix} a_1^{[1]} \\ a_2^{[1]} \\ a_3^{[1]} \\ a_4^{[1]} \end{bmatrix} = \sigma\left(z^{[1]}\right)$$

$$\begin{bmatrix} z_1^{[1]} \\ z_2^{[1]} \\ z_3^{[1]} \\ z_4^{[1]} \end{bmatrix} = \overbrace{\begin{bmatrix} \ldots W_1^{[1]T} \ldots \\ \ldots W_2^{[1]T} \ldots \\ \ldots W_3^{[1]T} \ldots \\ \ldots W_4^{[1]T} \ldots \end{bmatrix}}^{W^{[1]}} * \overset{input}{\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}} + \overbrace{\begin{bmatrix} b_1^{[1]} \\ b_2^{[1]} \\ b_3^{[1]} \\ b_4^{[1]} \end{bmatrix}}^{b^{[1]}}$$

Neural Network Overview

- Vectorizing across multiple examples

$$x = \begin{bmatrix} \vdots & \vdots & \vdots & \vdots \\ x^{(1)} & x^{(2)} & \dots & x^{(m)} \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix} \qquad A^{[1]} = \begin{bmatrix} \vdots & \vdots & \vdots & \vdots \\ \alpha^{[1](1)} & \alpha^{[1](2)} & \dots & \alpha^{[1](m)} \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix} \qquad Z^{[1]} = \begin{bmatrix} \vdots & \vdots & \vdots & \vdots \\ z^{[1](1)} & z^{[1](2)} & \dots & z^{[1](m)} \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix}$$

$$W^{[1]}x = \begin{bmatrix} \cdots \\ \cdots \\ \cdots \end{bmatrix} \begin{bmatrix} \vdots & \vdots & \vdots & \vdots \\ x^{(1)} & x^{(2)} & x^{(3)} & \vdots \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix} = \begin{bmatrix} \vdots & \vdots & \vdots & \vdots \\ w^{(1)}x^{(1)} & w^{(1)}x^{(2)} & w^{(1)}x^{(3)} & \vdots \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix} = \begin{bmatrix} \vdots & \vdots & \vdots & \vdots \\ z^{[1](1)} & z^{[1](2)} & z^{[1](3)} & \vdots \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix} = Z^{[1]}$$

$$\left.\begin{array}{c} z^{[1](i)} = W^{[1](i)}x^{(i)} + b^{[1]} \\ \alpha^{[1](i)} = \sigma\left(z^{[1](i)}\right) \\ z^{[2](i)} = W^{[2](i)}\alpha^{[1](i)} + b^{[2]} \\ \alpha^{[2](i)} = \sigma\left(z^{[2](i)}\right) \end{array}\right\} \Rightarrow \begin{cases} A^{[1]} = \sigma\left(z^{[1]}\right) \\ z^{[2]} = W^{[2]}A^{[1]} + b^{[2]} \\ A^{[2]} = \sigma\left(z^{[2]}\right) \end{cases}$$

$a^{[2](i)}$, (i) 指第个i 训练样本，而[2]是指第二层

Outline

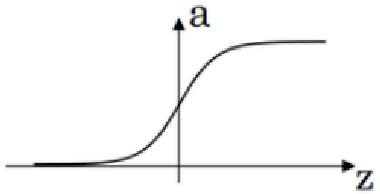
Part 1	Neural Network Overview
Part 2	Activation functions and gradient descent
Part 3	Deep L-layer neural network
Part 4	Regularization and optimization

Activation functions

- Four activation functions

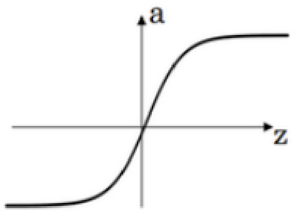
1） sigmoid activation function

$$g(z) = \frac{1}{1+e^{-z}}$$



2) Tanh activation function

$$g(z) = \tanh(z) = \frac{e^{z} - e^{z}}{e^{y} + e^{-z}}$$

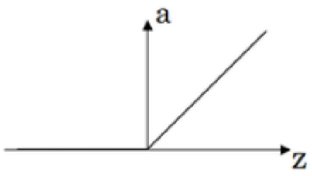


$$a^{[1]} = \sigma\left(z^{[1]}\right)$$

Activation functions

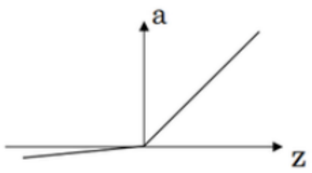
3） Rectified Linear Unit (ReLU)

$$g(z) = \max(0, z)$$



4） Leaky linear unit (Leaky ReLU)

$$g(z) = \max(0.01z, z)$$

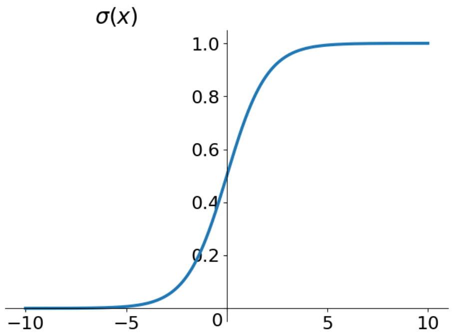


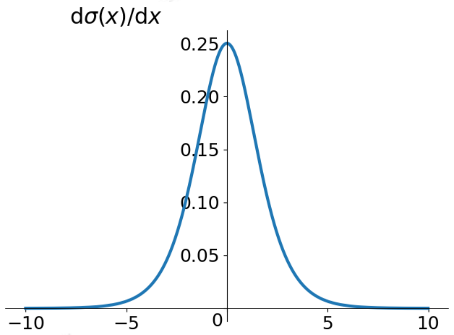
Activation functions

- Derivatives of activation functions

1) sigmoid activation function

$$\frac{d}{dz}g(z) = \frac{1}{1+e^{-z}}\left(1-\frac{1}{1+e^{-z}}\right) = g(z)(1-g(z))$$



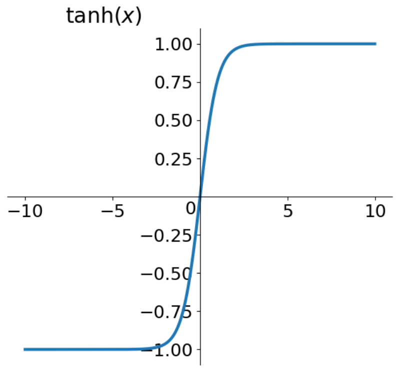


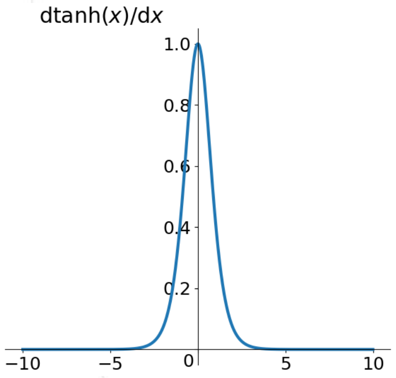
Drawbacks:

(1) gradient vanishing
(2) output is not zero-centered (slow the convergence)
(3) power operation is time-consuming

2) Tanh activation function

$$\frac{d}{dz}g(z) = 1-(\tanh(z))^2$$





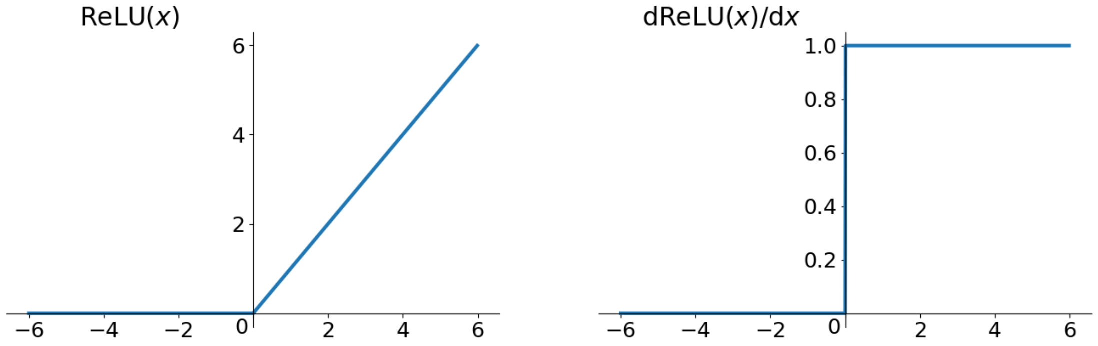
Drawbacks:

(1) gradient vanishing
(2) power operation is time-consuming

Activation functions

- Derivatives of activation functions

3) Rectified Linear Unit (ReLU)



$$g(z)' = \begin{cases} 0 & \text{if } z < 0 \\ 1 & \text{if } z > 0 \\ \text{undefined} & \text{if } z = 0 \end{cases}$$

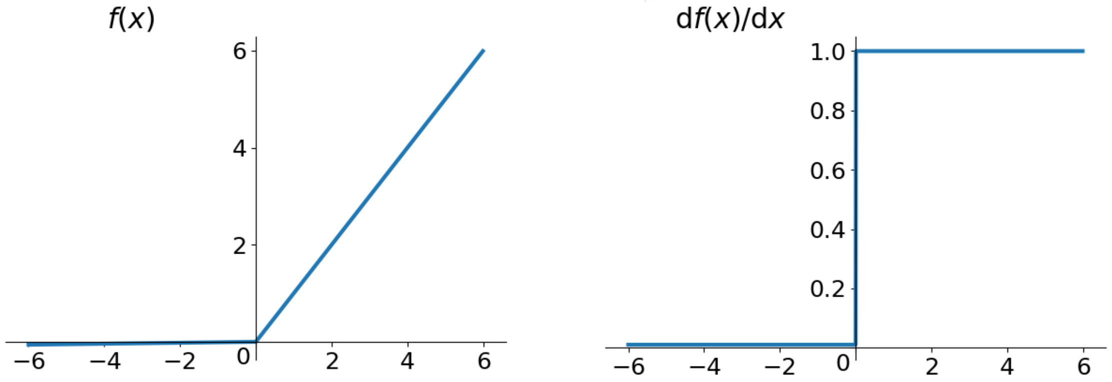
Features:

(1) eliminate gradient vanishing
(2) fast computation
(3) fast convergence than sigmoid and tanh

Drawbacks:

(1) Dead ReLU Problem

4) Leaky linear unit (Leaky ReLU)



$$g(z)' = \begin{cases} 0.01 & \text{if } z < 0 \\ 1 & \text{if } z > 0 \\ \text{undefined} & \text{if } z = 0 \end{cases}$$

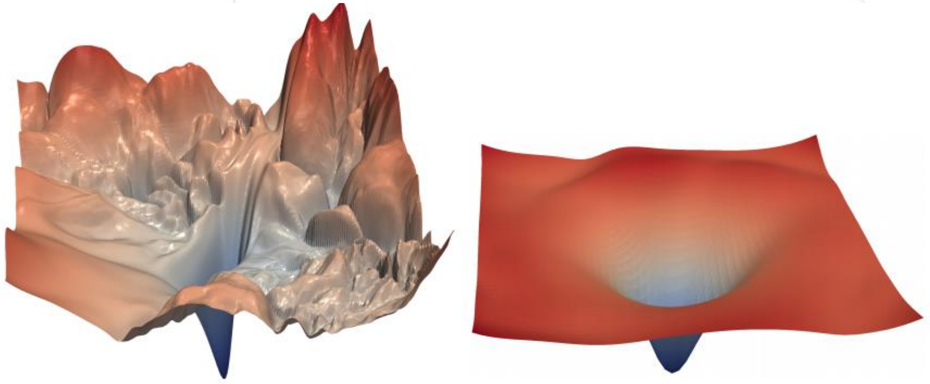
Gradient descent

- Gradient descent for one layer neural networks

For one layer neural network

$$\hat{y} = \sigma\left(w^T x + b\right)$$
$$\sigma(z) = \frac{1}{1+e^{-z}}$$
$$J(w,b) = \frac{1}{m}\sum_{i=1}^{m} \mathcal{L}\left(\hat{y}^{(i)}, y^{(i)}\right)$$
$$= -\frac{1}{m}\sum_{i=1}^{m} y^{(i)} \log \hat{y}^{(i)} + \left(1 - y^{(i)}\right) \log\left(1 - \hat{y}^{(i)}\right)$$

maximum likelihood



The loss surfaces of ResNet-56 with/without skip connections.

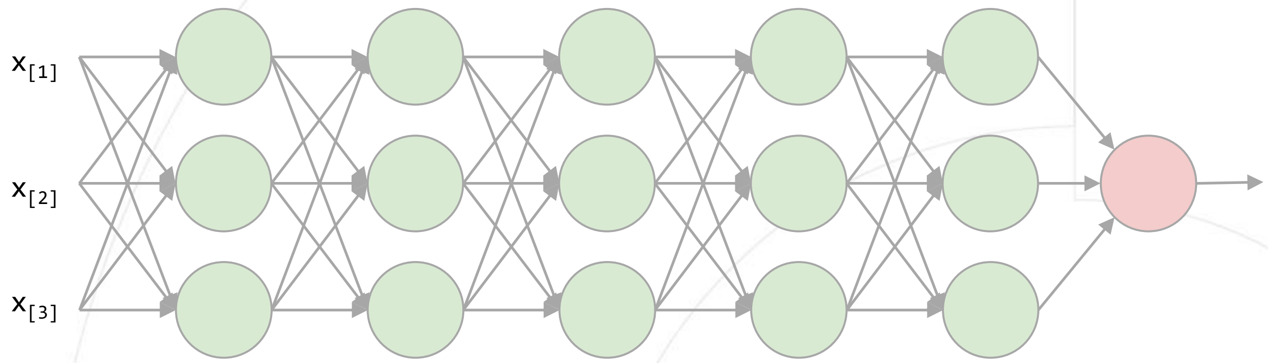
$$\underbrace{x, w, b}_{dw=dz\cdot x, db=dz} \quad \Longleftrightarrow \quad \underbrace{z = w^T x + b}_{dz=da\cdot g'(z), g(z)=\sigma(z), \frac{dL}{dz}=\frac{dL}{da}\cdot\frac{da}{dz}, \frac{d}{dz}g(z)=g'(z)} \quad \Longleftrightarrow \quad \underbrace{a = \sigma(z) \Longleftarrow L(a,y)}_{da=\frac{d}{da}L(a,y)=(-y\log\alpha-(1-y)\log(1-a))'=-\frac{y}{a}+\frac{1-y}{1-a}}$$

Outline

Part 1	Neural Network Overview
Part 2	Activation functions and gradient descent
Part 3	Deep L-layer neural network
Part 4	Regularization and optimization

Tsinghua University sensetime

- Forward and backward for Deep L-layer neural network



Forward

$input: a^{[l-1]}$

$output: a^{[l]}$

$z^{[l]} = W^{[l]} \cdot a^{[l-1]} + b^{[l]}$

$a^{[l]} = g^{[l]}(z^{[l]})$

Backward

$input: da^{[l]}$

$output: da^{[l-1]}, dw^{[l]}, db^{[l]}$

$dz^{[l]} = da^{[l]} * g^{[l]'}\left(z^{[l]}\right)$

$dw^{[l]} = dz^{[l]} \cdot a^{[l-1]}$

$db^{[l]} = dz^{[l]}$

$da^{[l-1]} = w^{[l]T} \cdot dz^{[l]}$

$dz^{[l]} = w^{[l+1]T} dz^{[l+1]} \cdot g^{[l]'}\left(z^{[l]}\right)$

Outline

Part 1	Neural Network Overview
Part 2	Activation functions and gradient descent
Part 3	Deep L-layer neural network
Part 4	Regularization and optimization

Regularization and optimization

- Parameters vs Hyperparameters

- Parameters

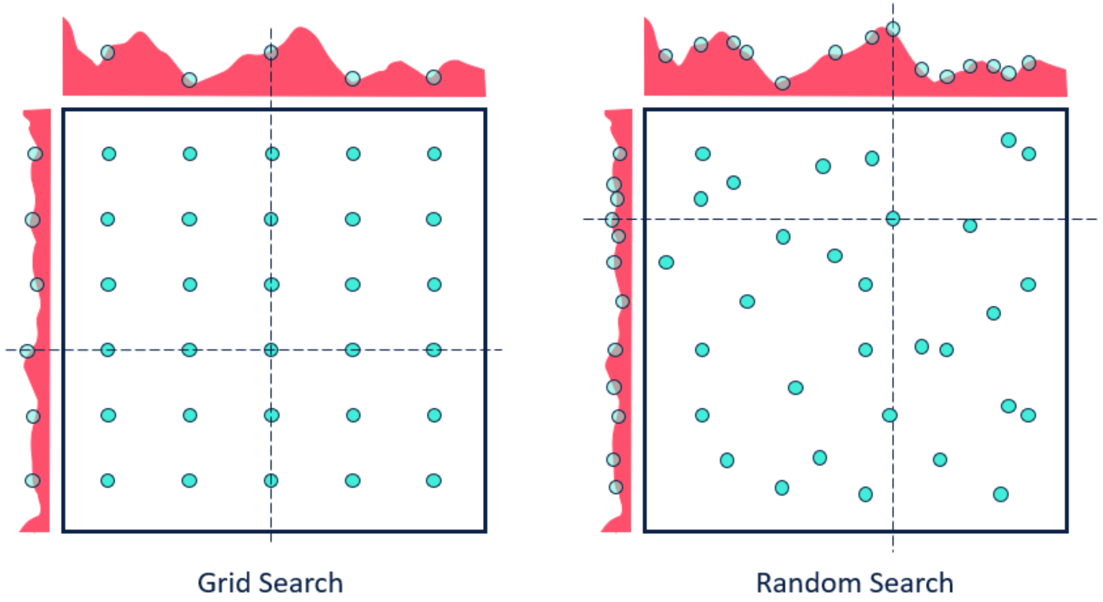
$$W^{[1]}, b^{[1]}, W^{[2]}, b^{[2]}, W^{[3]}, b^{[3]} \dots$$

- Hyperparameters

  - learning rate
  - iterations
  - *L* (Number of hidden layers)
  - *n* (Number of hidden layer neurons)
  - choice of activation function
  - momentum
  - mini batch size
  - regularization parameters

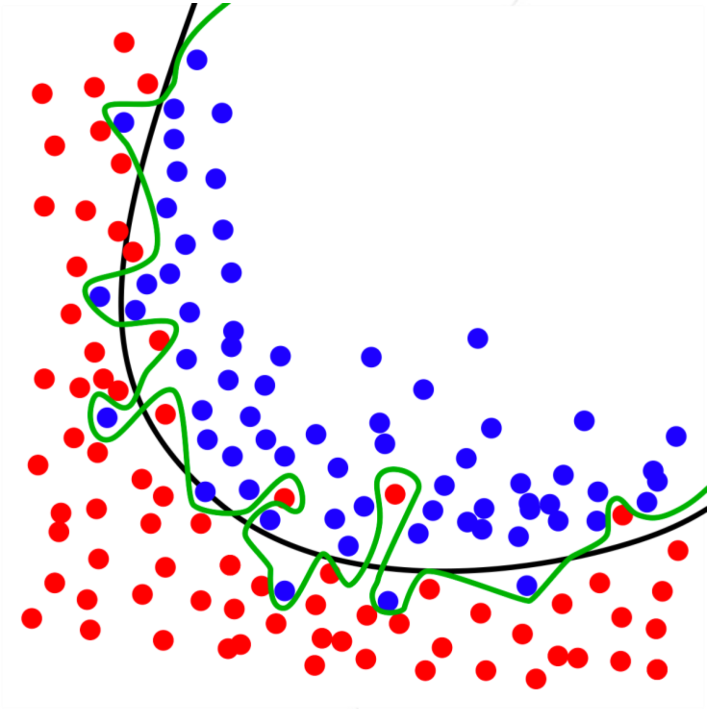
  … …

- Tune hyperparameters with grid search or random search

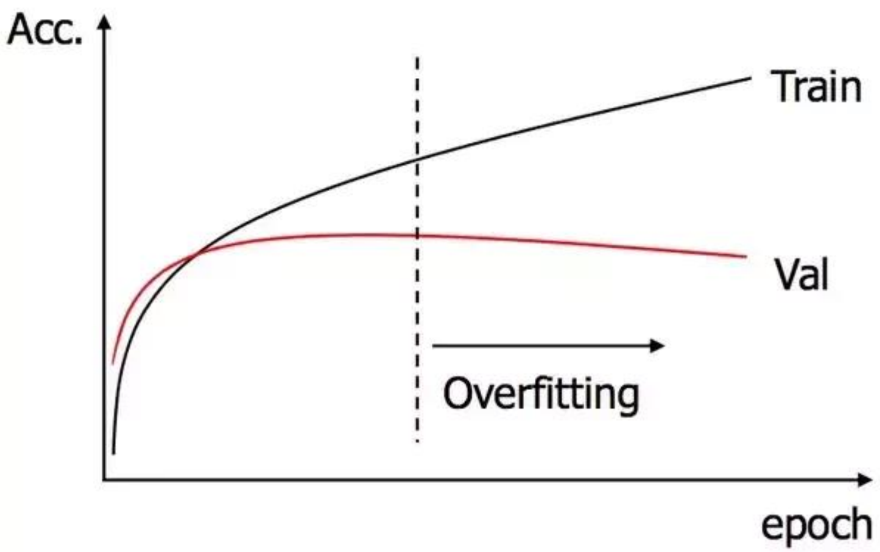


https://community.alteryx.com/t5/Data-Science/Hyperparameter-Tuning-Black-Magic/ba-p/449289

- Overfitting



https://en.wikipedia.org/wiki/Overfitting



https://www.quora.com/Which-signals-do-indicate-that-the-convolutional-neural-network-is-overfitted

Regularization and optimization

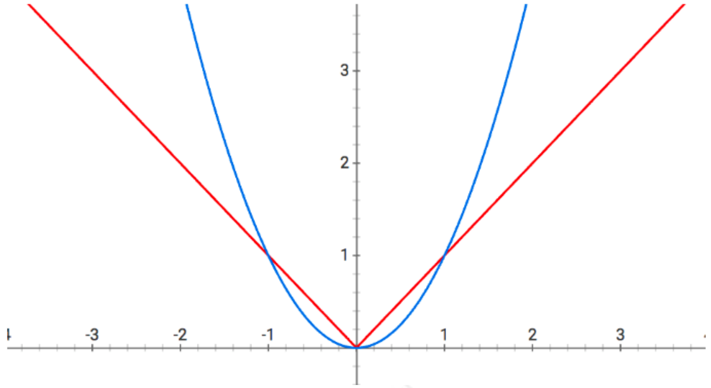
- Regularization

L1 penalty

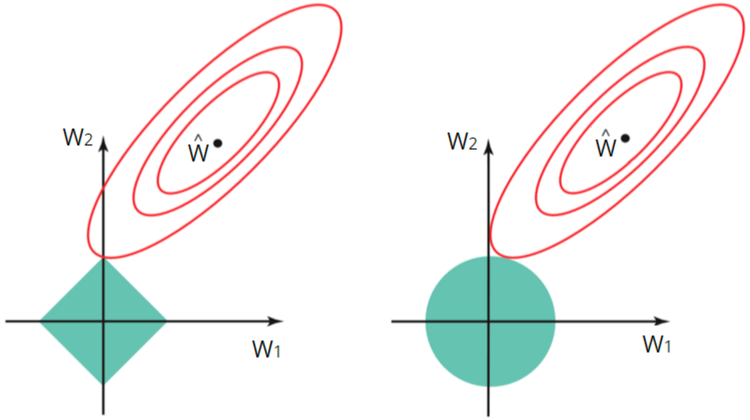
$$L(x,y) \equiv \sum_{i=1}^{n}(y_i - h_\theta(x_i))^2 + \lambda \boxed{\sum_{i=1}^{n}|\theta_i|}$$

L2 penalty

$$L(x,y) \equiv \sum_{i=1}^{n}(y_i - h_\theta(x_i))^2 + \boxed{\lambda \sum_{i=1}^{n}\theta_i^2}$$



L1 penalty	**L2 penalty**
L1 penalizes sum of absolute values of weights.	L2 penalizes sum of square values of weights.
L1 generates model that is simple and interpretable.	L2 regularization is able to learn complex data patterns.
L1 is robust to outliers.	L2 is not robust to outliers.



Source: An Introduction to Statistical Learning by Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani

Tsinghua University sensetime

- Weight Initialization matters

  - $W^{[l]}$-weight matrix of dimension (size of layer l, size of layer l-1)
  - $b^{[l]}$-bias vectors of dimension (size of layer l)

- Initialization with value 0?

  No!
  If the weight is zero, the outputs of all neural node are same.
  The gradient is same! The weight update is same!
  We can't accept this.

Regularization and optimization

- Weight Initialization matters

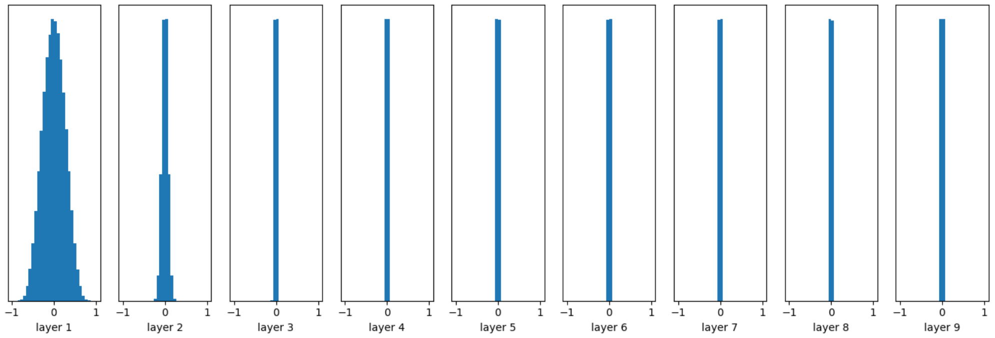
  - $W^{[l]}$-weight matrix of dimension (size of layer l, size of layer l-1)

  - $b^{[l]}$-bias vectors of dimension (size of layer l)

- Random initialization

```
W = np.random.randn(node_in, node_out)
```

e.g. We create a neural network with 10 layers and adopt the tanh activation function.
We initialize the W with a mean of 0 and a standard deviation of 0.01.



At the end of neural network, the output is close to 0.
This leads to a small gradient and hard to update the W.

Regularization and optimization

- Weight Initialization matters

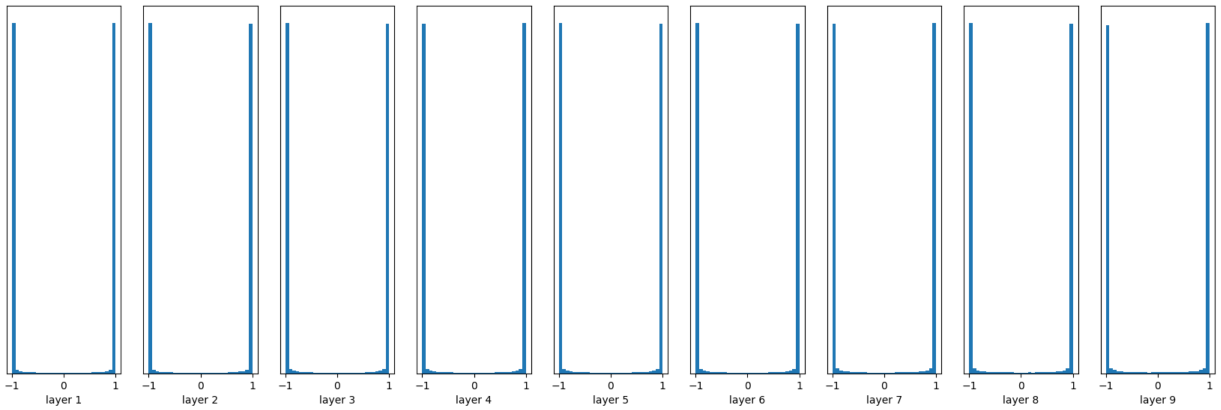
  - $W^{[l]}$-weight matrix of dimension (size of layer l, size of layer l-1)

  - $b^{[l]}$-bias vectors of dimension (size of layer l)

- Random initialization

```
W = np.random.randn(node_in, node_out)
```

e.g. We create a neural network with 10 layers and adopt the tanh activation function.
We initialize the W with a mean of 0 and a standard deviation of 1.



The output is close to -1 or 1.
The gradient of tanh is close to 0
This leads to a small gradient and hard to update the W.

Regularization and optimization

- Analysis

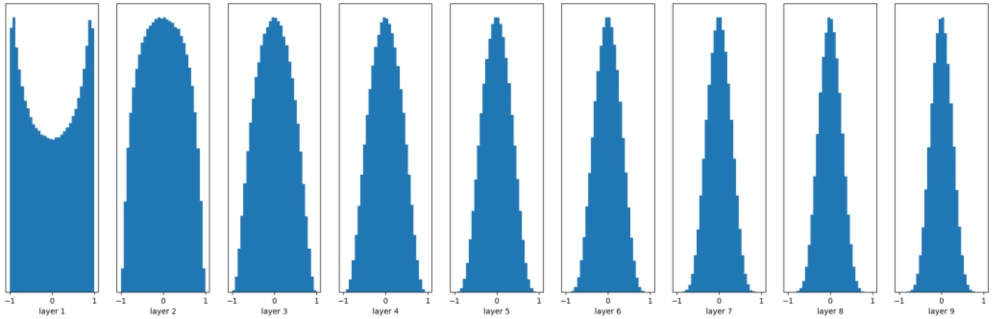
  Deep NN models have difficulties in converging when the weights are initialized using Normal Distribution with *fixed standard deviation*. This is because the variance of weights is not taken care of, which leads to very large or small activation values, resulting in exploding or vanishing gradient problem during backpropagation. This problem worsens as the depth of NN is increased.

- Xavier initialization

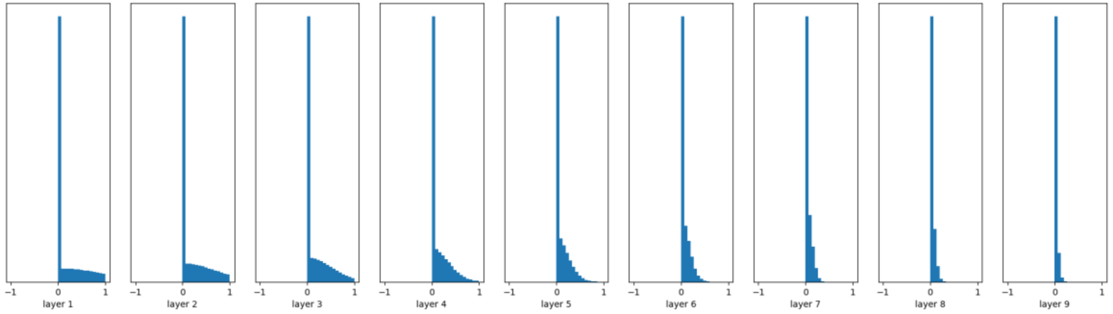
  It tries to keep variance of all the layers equal.

```
np.random.randn(node_in, node_out) / np.sqrt(node_in)
```

For tanh



For ReLU



- Xavier initialization – simple **derivation**

(1) It tries to keep variance of all the layers equal.

```
np.random.randn(node_in, node_out) / np.sqrt(node_in)
```

$$\boldsymbol{y} = \boldsymbol{w}\boldsymbol{x} = w_1 x_1 + w_2 x_2 + \ldots + w_n x_n$$

Input x and output y

$$\operatorname{Var}(W_i X_i) = [E(X_i)]^2 \operatorname{Var}(W_i) + [E(W_i)]^2 \operatorname{Var}(X_i) + \operatorname{Var}(X_i) \operatorname{Var}(W_i)$$

We have E(x) = 0, E(W) = 0

$$\operatorname{Var}(W_i X_i) = \operatorname{Var}(X_i) \operatorname{Var}(W_i)$$

$$\operatorname{Var}(\boldsymbol{y}) = \operatorname{Var}\left(\sum_i w_i x_i\right) = \sum_i \operatorname{Var}(w_i x_i) = \sum_i \operatorname{Var}(x_i) \operatorname{Var}(w_i) = n \operatorname{Var}(x_i) \operatorname{Var}(w_i)$$

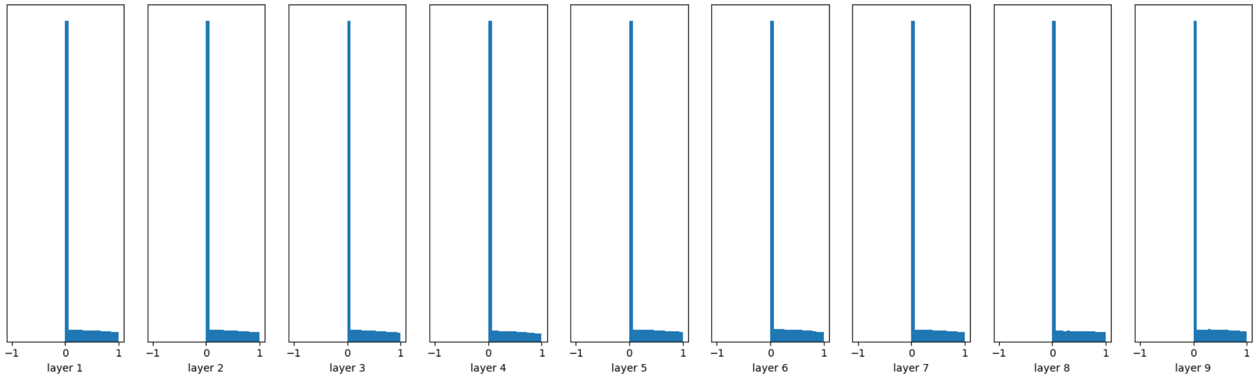
To achieve (1), we have

$$\operatorname{Var}(w_i) = \frac{1}{n} = \frac{1}{n_{\text{in}}}$$

- He initialization

```
np.random.randn(node_in,node_out) / np.sqrt(node_in/2)
```

For ReLU



Assumptions (valid for each layer k)-

1. All elements in W^k share the same distribution and are independent of each other. Similarly for x^k and y^k.
2. each element of W^k and each element of x^k are independent of each other.
3. W^k and y^k have zero mean and are symmetrical around zero.
4. b^k is initialized to zero vector as we don't require any bias initially.

Regularization and optimization

- Derivation of Kaiming He initialization

Keep in mind

$$Var(X+Y) = Var(X) + Var(Y)$$
$$Var(XY) = Var(X)Var(Y) + (E[X])^2 Var(Y) + Var(X)(E[Y])^2$$

Assume, $y^k = W^k x^k + b^k$ and $x^{k+1} = f(y^k)$; k is layer number and f is activation

$$y_i = W_{i,1}x_1 + W_{i,2}x_2 + \cdots + W_{i,n}x_n + b_i$$ //n is size of input activation at current layer

$$\begin{aligned} So, Var(y_i) &= Var(W_{i,1}x_1 + W_{i,2}x_2 + \cdots + W_{i,n}x_n) \\ &= n * Var(W_{i,j}x_j) \\ &= n * (Var(W_{i,j})Var(x_j) + (E[W_{i,j}])^2 Var(x_j) + Var(W_{i,j})(E[x_j])^2) \\ &= n * (Var(W_{i,j})Var(x_j) + (0)^2 Var(x_j) + Var(W_{i,j})(E[x_j])^2) \\ &= n * Var(W_{i,j}) * (Var(x_j) + (E[x_j])^2) \\ &= n * Var(W_{i,j}) * E[x_j^2] \end{aligned}$$

Remember that $E[x_j^2] \neq Var(x_j)$ unless $E[X_j]=0$. This is because of ReLU which does not have zero mean.

Regularization and optimization

- Derivation of Kaiming He initialization

Assume, $y^k = W^k x^k + b^k$ and $x^{k+1} = f(y^k)$; k is layer number and f is activation

$$
\begin{aligned}
E[x^2] &= \int_{-\infty}^{\infty} x^2 P(x)dx \\
&= \int_{-\infty}^{\infty} max(0,y)^2 P(y)dy \\
&= \int_{0}^{\infty} y^2 P(y)dy \\
&= 0.5 * \int_{-\infty}^{\infty} y^2 P(y)dy \\
&= 0.5 * Var(y)
\end{aligned}
\longrightarrow
$$

$$Var(y_i^l) = 0.5 * n^l * W_{i,j}^l * Var(y_j^{l-1})$$

$$Var(y^l) = 0.5 * n^l * W^l * Var(y^{l-1})$$

$$Var\left(y^L\right) = Var\left(y^1\right)\left(\prod_{l=2}^{L} \frac{n^l}{2} Var\left(W^l\right)\right)$$

$$\frac{n^l}{2} \operatorname{Var}\left(W^l\right) = 1, \quad \forall l$$

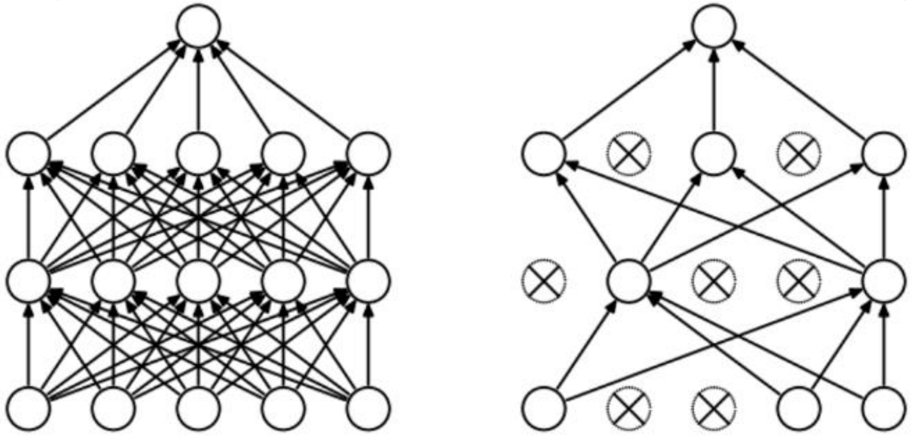
a=initialized slope of PReLU.

$$W \sim \mathcal{N}\left(0, \frac{2}{n^l}\right) \longrightarrow \frac{1}{2}\left(1+a^2\right) n^l \operatorname{Var}\left(W^l\right) = 1, \quad \forall l$$

- if a=0, we get ReLU case
- if a=1, we get linear case

Tsinghua University sensetime

- Dropout Regularization



(a) Standard Neural Net

(b) After applying dropout.

Training Phase

For each hidden layer, for each training sample, for each iteration, ignore (zero out) a random fraction, p, of nodes (and corresponding activations).

Testing Phase

Use all activations, but reduce them by a factor p (to account for the missing activations during training).

[1] Srivastava, Nitish, et al. "Dropout: a simple way to prevent neural networks from overfitting", JMLR 2014

Tsinghua University sensetime

Regularization and optimization

- Batch Normalization

why use batch normalization?

- reduce internal covariate shift
- network can use higher learning rate without vanishing or exploding gradients
- regularizing effect
- network becomes more robust to different initialization schemes and learning rates.

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1\ldots m}\}$;
Parameters to be learned: γ, β
Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_\mathcal{B} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma_\mathcal{B}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_\mathcal{B})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_\mathcal{B}}{\sqrt{\sigma_\mathcal{B}^2 + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Algorithm 1: Batch Normalizing Transform, applied to activation x over a mini-batch.

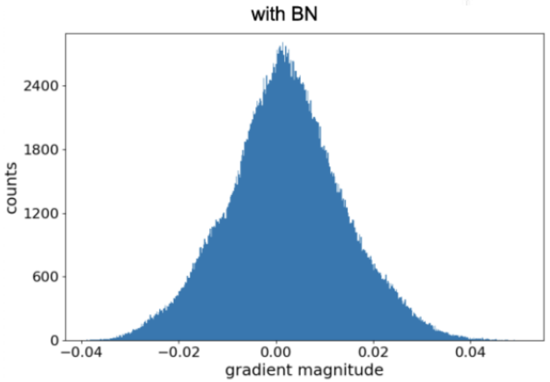
Input: Network N with trainable parameters Θ;
subset of activations $\{x^{(k)}\}_{k=1}^{K}$
Output: Batch-normalized network for inference, $N_{\mathrm{BN}}^{\mathrm{inf}}$

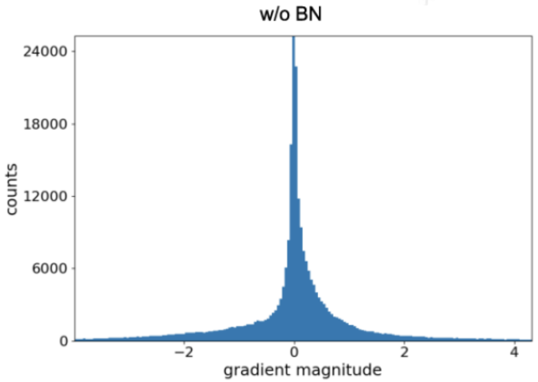
$N_{\mathrm{BN}}^{\mathrm{tr}} \leftarrow N$ // Training BN network
for $k = 1 \ldots K$ **do**
  Add transformation $y^{(k)} = \mathrm{BN}_{\gamma^{(k)},\beta^{(k)}}(x^{(k)})$ to $N_{\mathrm{BN}}^{\mathrm{tr}}$ (Alg. 1)
  Modify each layer in $N_{\mathrm{BN}}^{\mathrm{tr}}$ with input $x^{(k)}$ to take $y^{(k)}$ instead
end for
Train $N_{\mathrm{BN}}^{\mathrm{tr}}$ to optimize the parameters $\Theta \cup \{\gamma^{(k)}, \beta^{(k)}\}_{k=1}^{K}$
$N_{\mathrm{BN}}^{\mathrm{inf}} \leftarrow N_{\mathrm{BN}}^{\mathrm{tr}}$ // Inference BN network with frozen // parameters
for $k = 1 \ldots K$ **do**
  // For clarity, $x \equiv x^{(k)}, \gamma \equiv \gamma^{(k)}, \mu_\mathcal{B} \equiv \mu_\mathcal{B}^{(k)}$, etc.
  Process multiple training mini-batches $\mathcal{B}$, each of size m, and average over them:

$$\mathrm{E}[x] \leftarrow \mathrm{E}_\mathcal{B}[\mu_\mathcal{B}]$$
$$\mathrm{Var}[x] \leftarrow \frac{m}{m-1}\mathrm{E}_\mathcal{B}[\sigma_\mathcal{B}^2]$$

  In $N_{\mathrm{BN}}^{\mathrm{inf}}$, replace the transform $y = \mathrm{BN}_{\gamma,\beta}(x)$ with $y = \frac{\gamma}{\sqrt{\mathrm{Var}[x]+\epsilon}} \cdot x + \left(\beta - \frac{\gamma\,\mathrm{E}[x]}{\sqrt{\mathrm{Var}[x]+\epsilon}}\right)$
end for

Algorithm 2: Training a Batch-Normalized Network





Ioffe S , Szegedy C . Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift[J]. 2015.

4.2 Convolutional neural network and feature extraction

Dr. Liu Yu

Friday, March 12, 2021

Outline

Part 1	Introduction to CNN
Part 2	The Progress of CNN
Part 3	Analysis

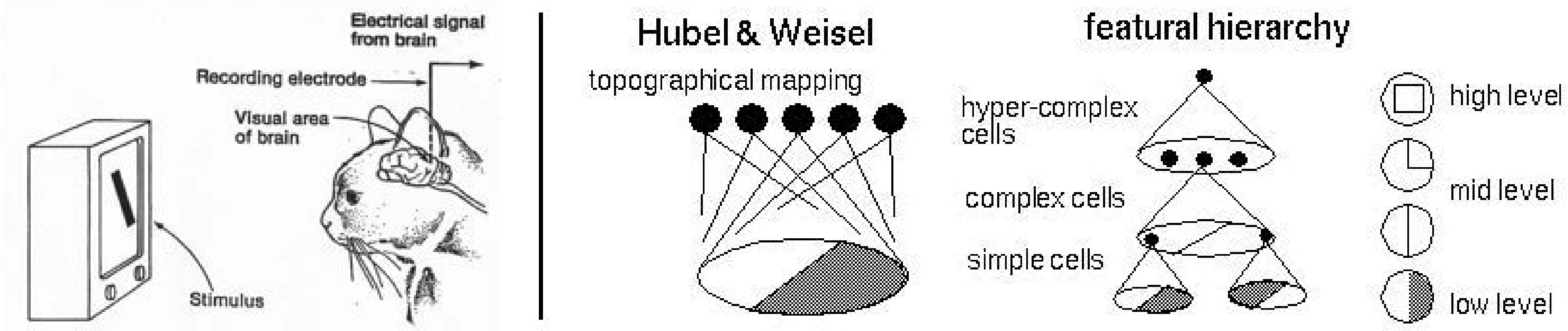
商汤 sensetime

清華大學 Tsinghua University

Highlights

- Learn the basic operators in CNN
- Learn the development process of convolutional neural network
- Learn the training process of CNN
- Understand the methods of neural network feature visualization
- Understand the problems of large-batch training

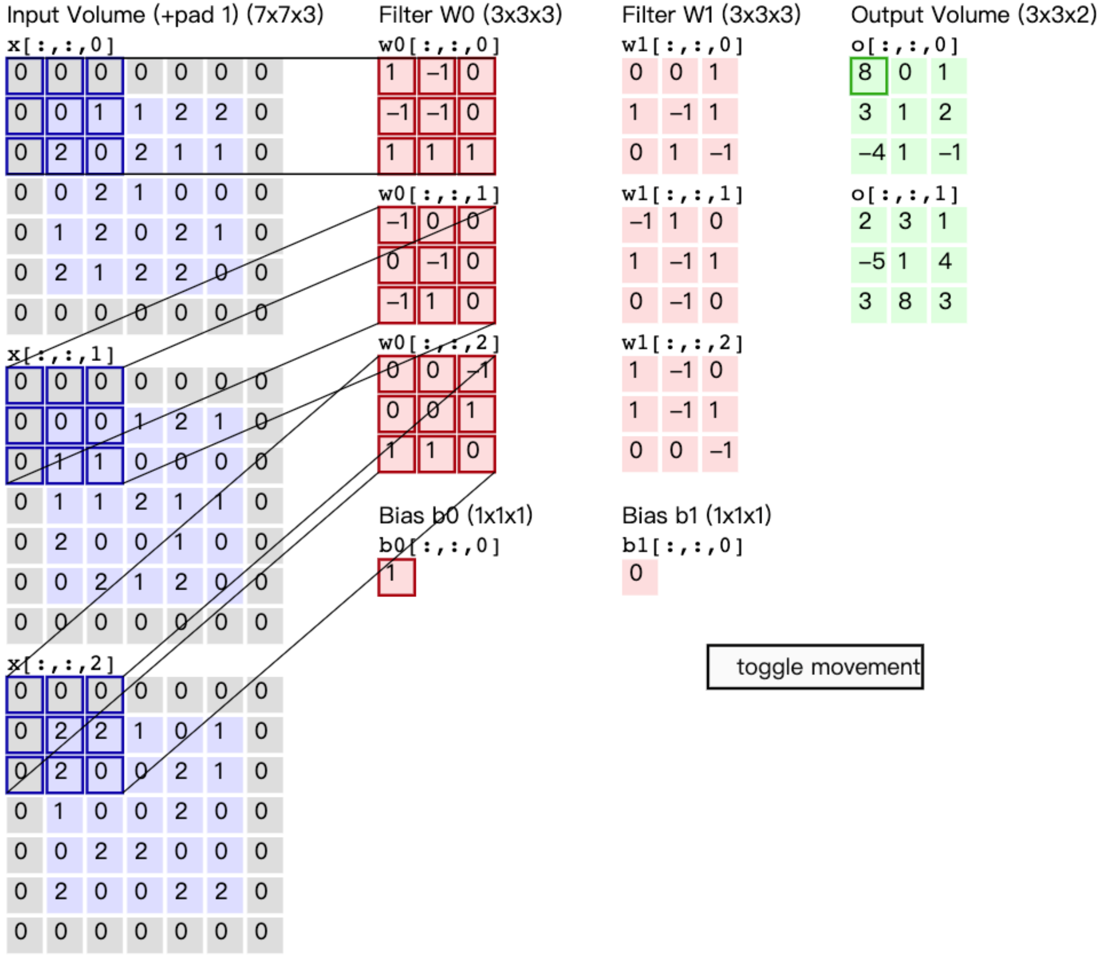
- Inspired by nature



Hubel D H , Wiesel T N . Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. Journal of Physiology, 1962.

Introduction to CNN

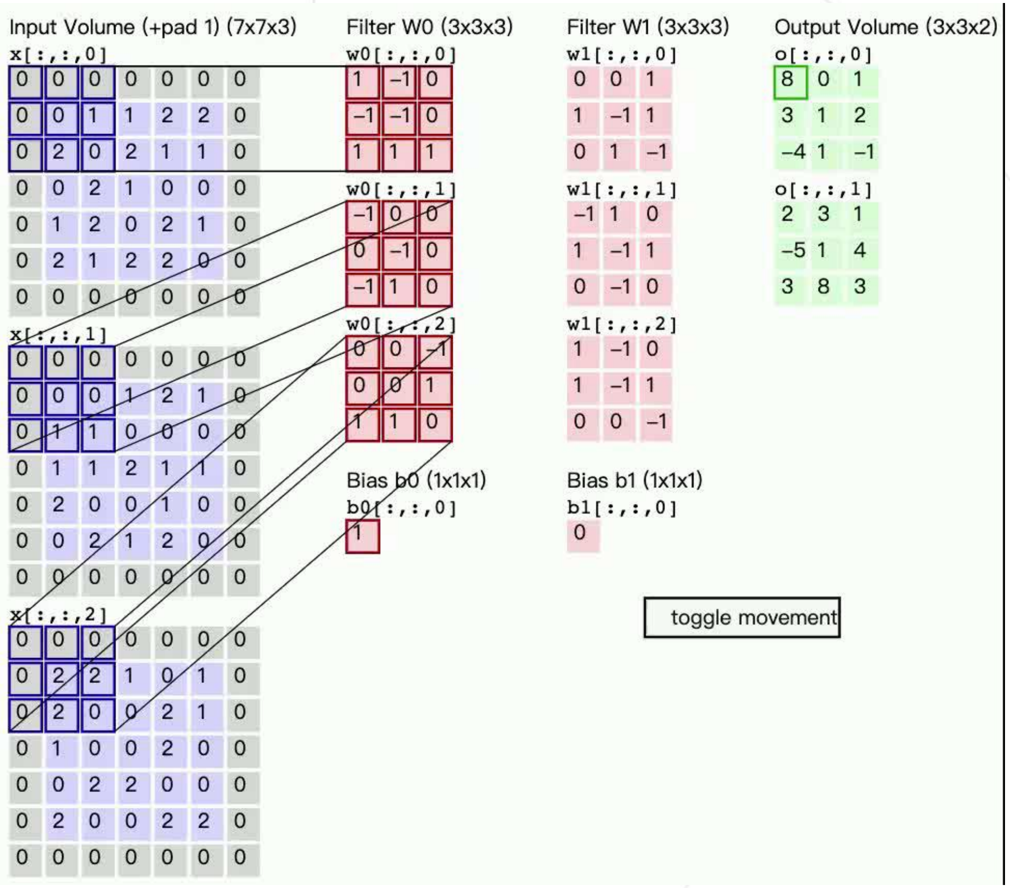
- Convolution



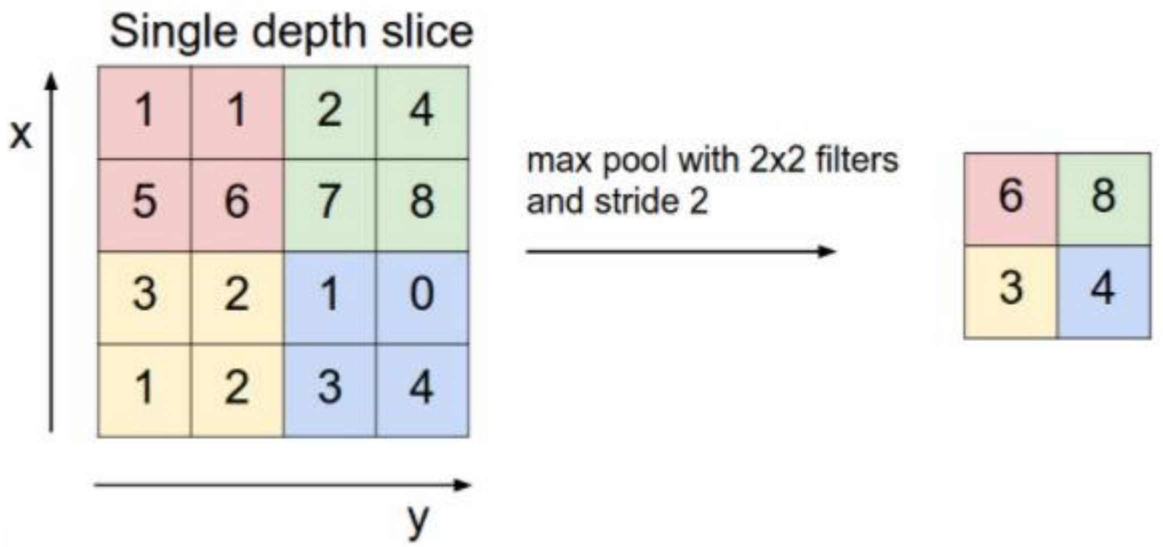
https://cs231n.github.io/convolutional-networks/

Introduction to CNN

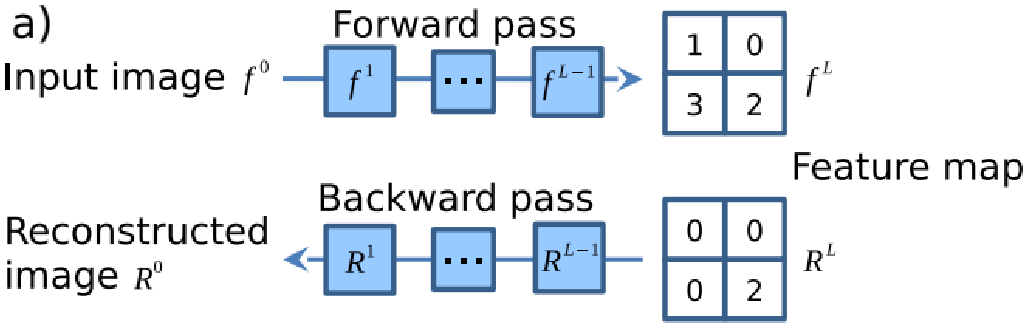
- Features of convolutional layer
  - Sparse interactions
  - Parameter sharing
  - Equivariant representations
- kernel size
  - 3x3, 5x5, ...
- padding
  - zero padding
- stride
  - =2: input size: 5x5 -> output size: 3x3
  - $output\_size = \frac{input\_size + 2 * padding - kernel\_size}{stride} + 1$



- Pooling


Single depth slice
x
y
1
1
2
4
5
6
7
8
3
2
1
0
1
2
3
4
max pool with 2x2 filters
and stride 2
6
8
3
4

- Schematic of visualizing the activations of high layer neurons



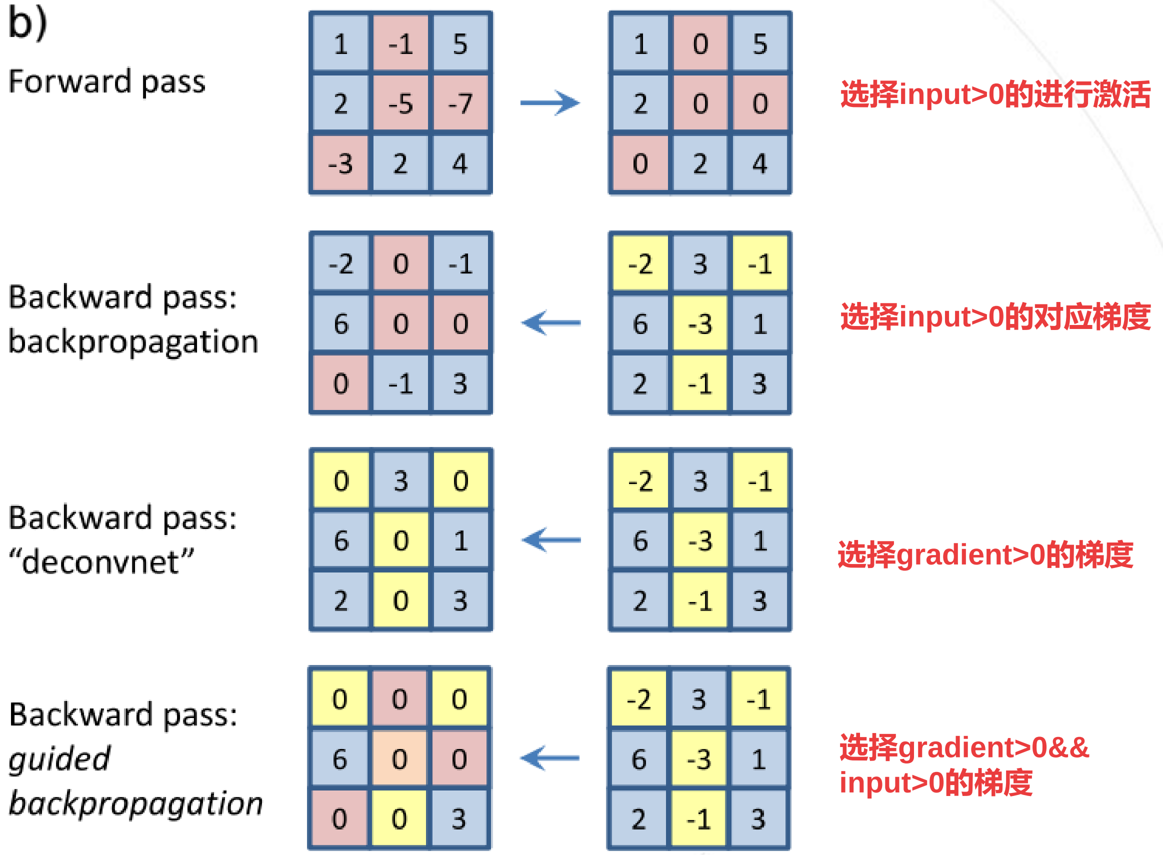
c)

activation: $f_i^{l+1} = relu(f_i^l) = \max(f_i^l, 0)$

backpropagation: $R_i^l = (f_i^l > 0) \cdot R_i^{l+1}$, where $R_i^{l+1} = \frac{\partial f^{out}}{\partial f_i^{l+1}}$

backward 'deconvnet': $R_i^l = (R_i^{l+1} > 0) \cdot R_i^{l+1}$

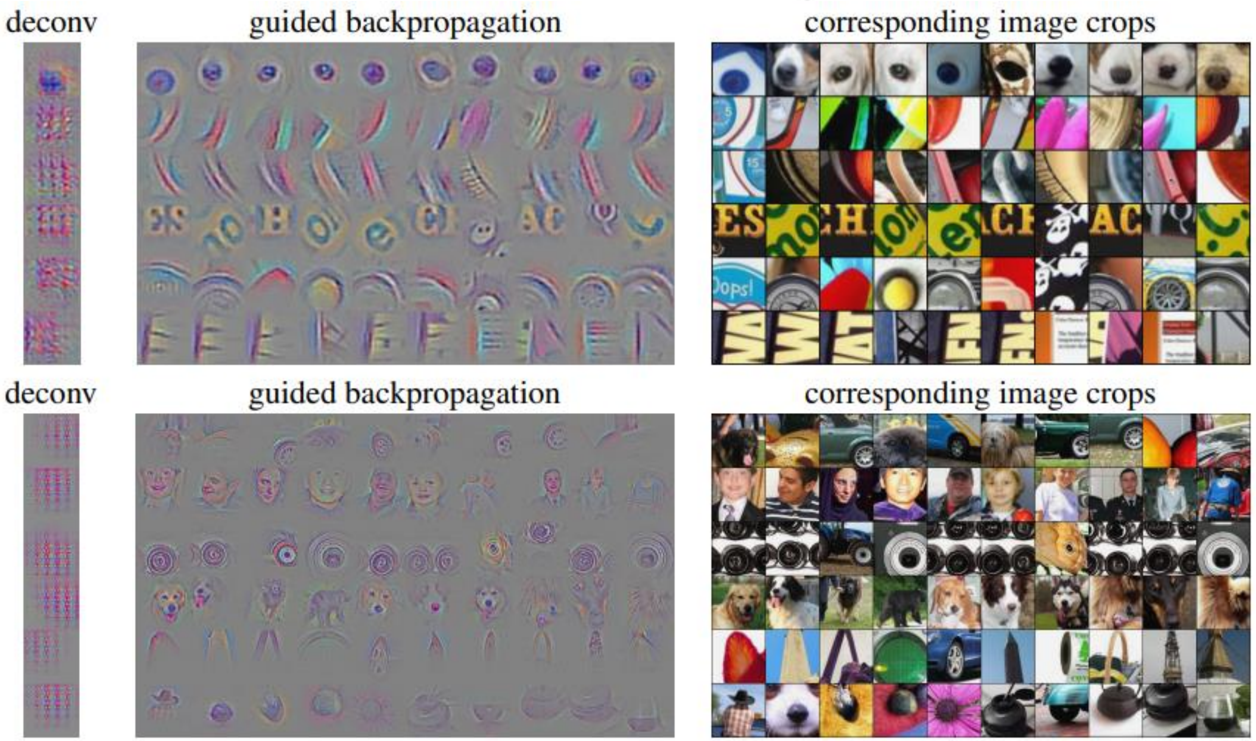
guided backpropagation: $R_i^l = (f_i^l > 0) \cdot (R_i^{l+1} > 0) \cdot R_i^{l+1}$

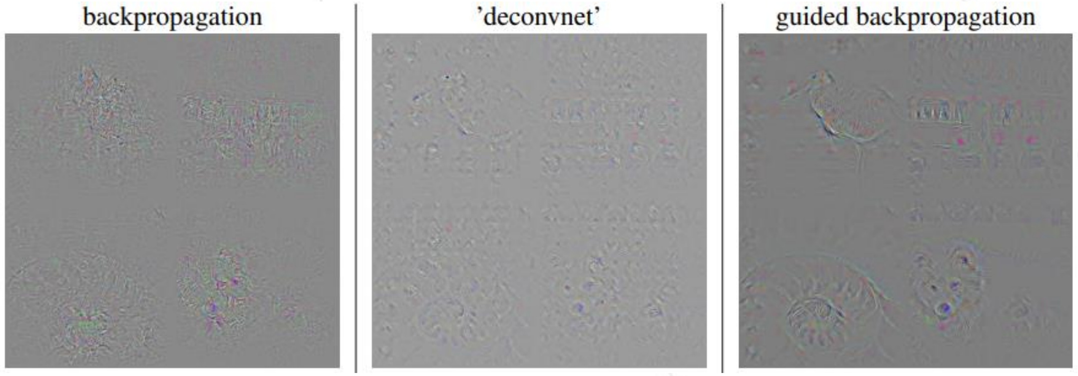


Springenberg J T, Dosovitskiy A, Brox T, et al. Striving for simplicity: The all convolutional net[J]. arXiv preprint arXiv:1412.6806, 2014.

Introduction to CNN

- Schematic of visualizing the activations of high layer neurons





There is a lot of noise in the image, so you can't see what the model has learned

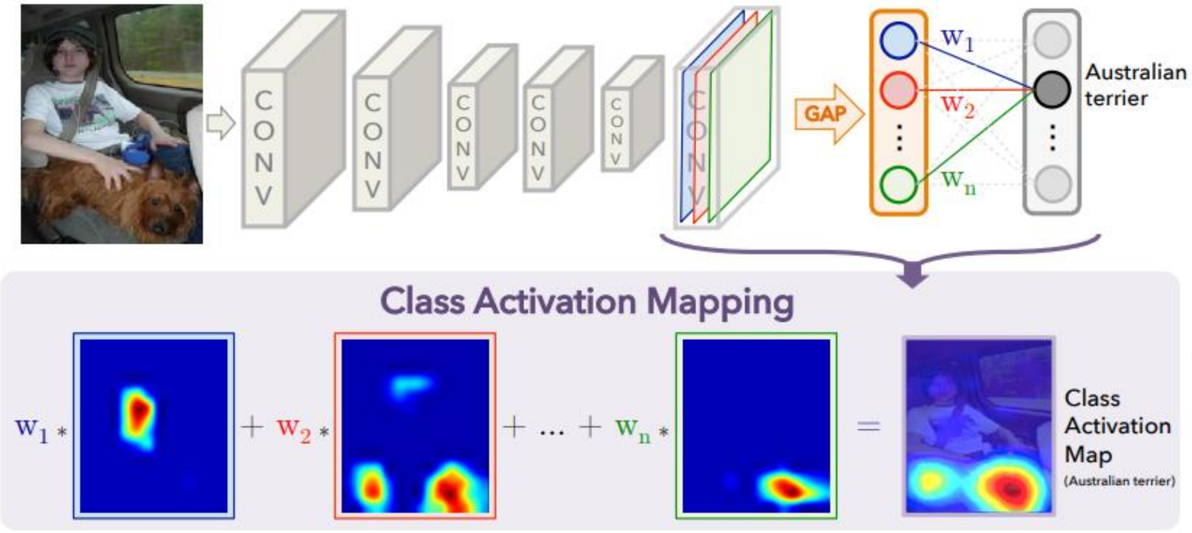
There is a rough outline, but there is still a lot of noise.

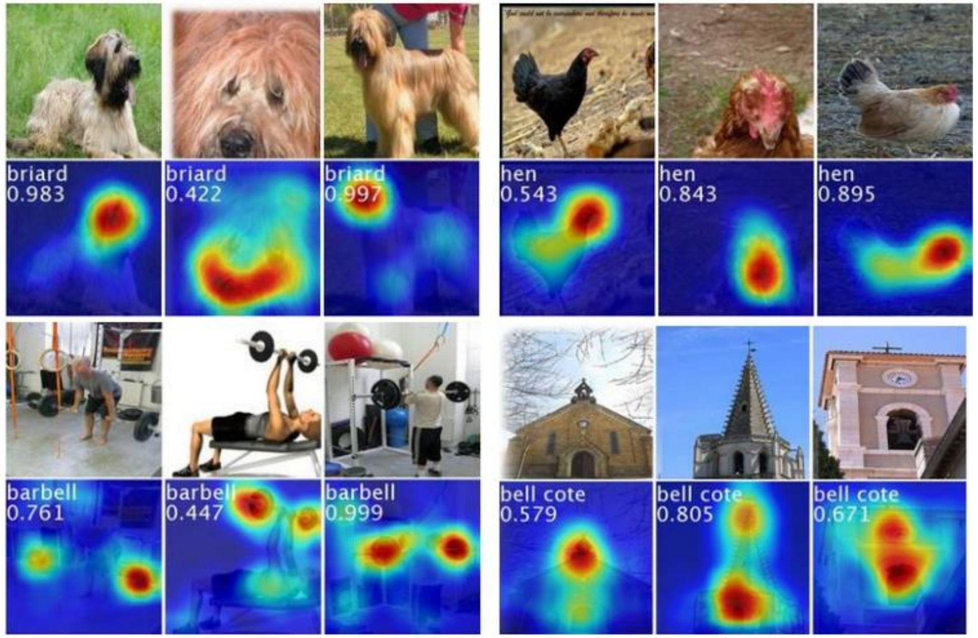
obvious features

We "see" the mysterious inside of the CNN models, but they can't be used to explain the results of the classification, because they are insensitive to categories and simply show all the features that can be extracted.

Springenberg J T, Dosovitskiy A, Brox T, et al. Striving for simplicity: The all convolutional net[J]. arXiv preprint arXiv:1412.6806, 2014.

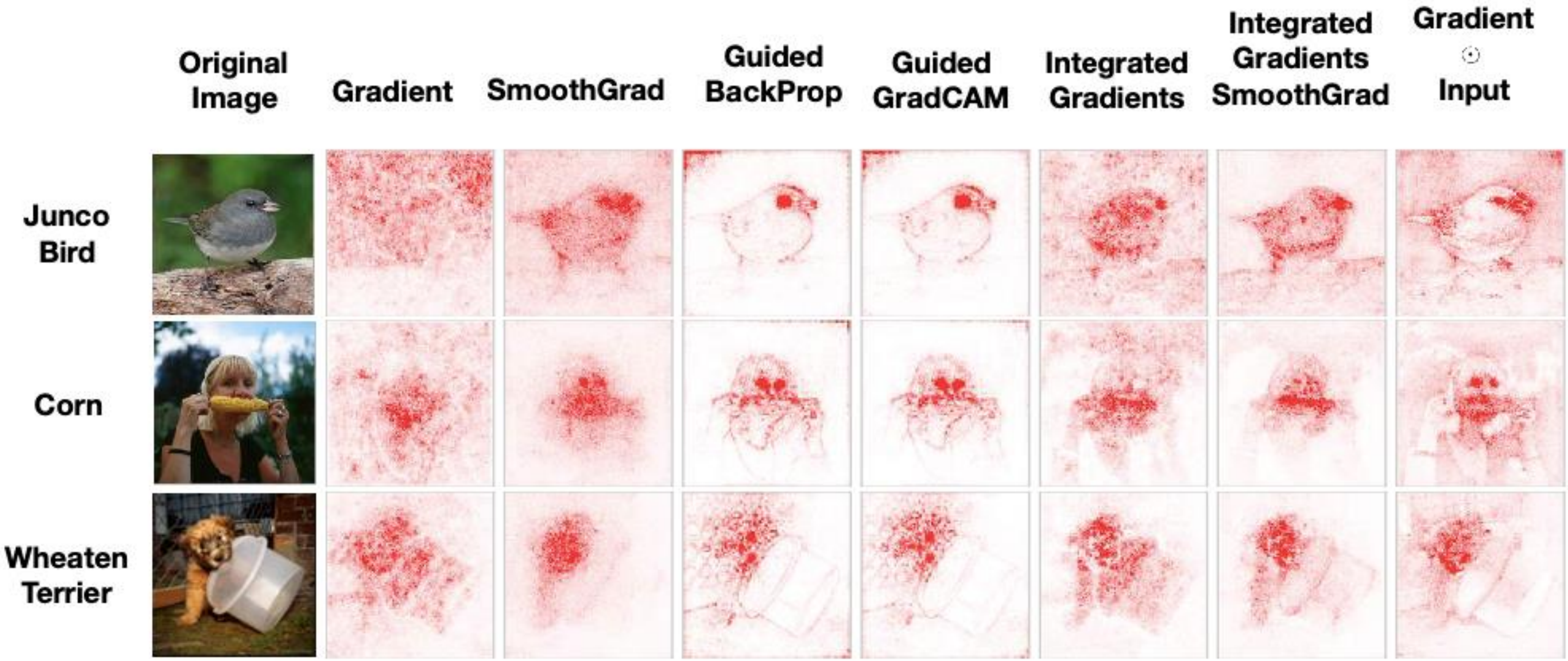
- CAM(class activation map)





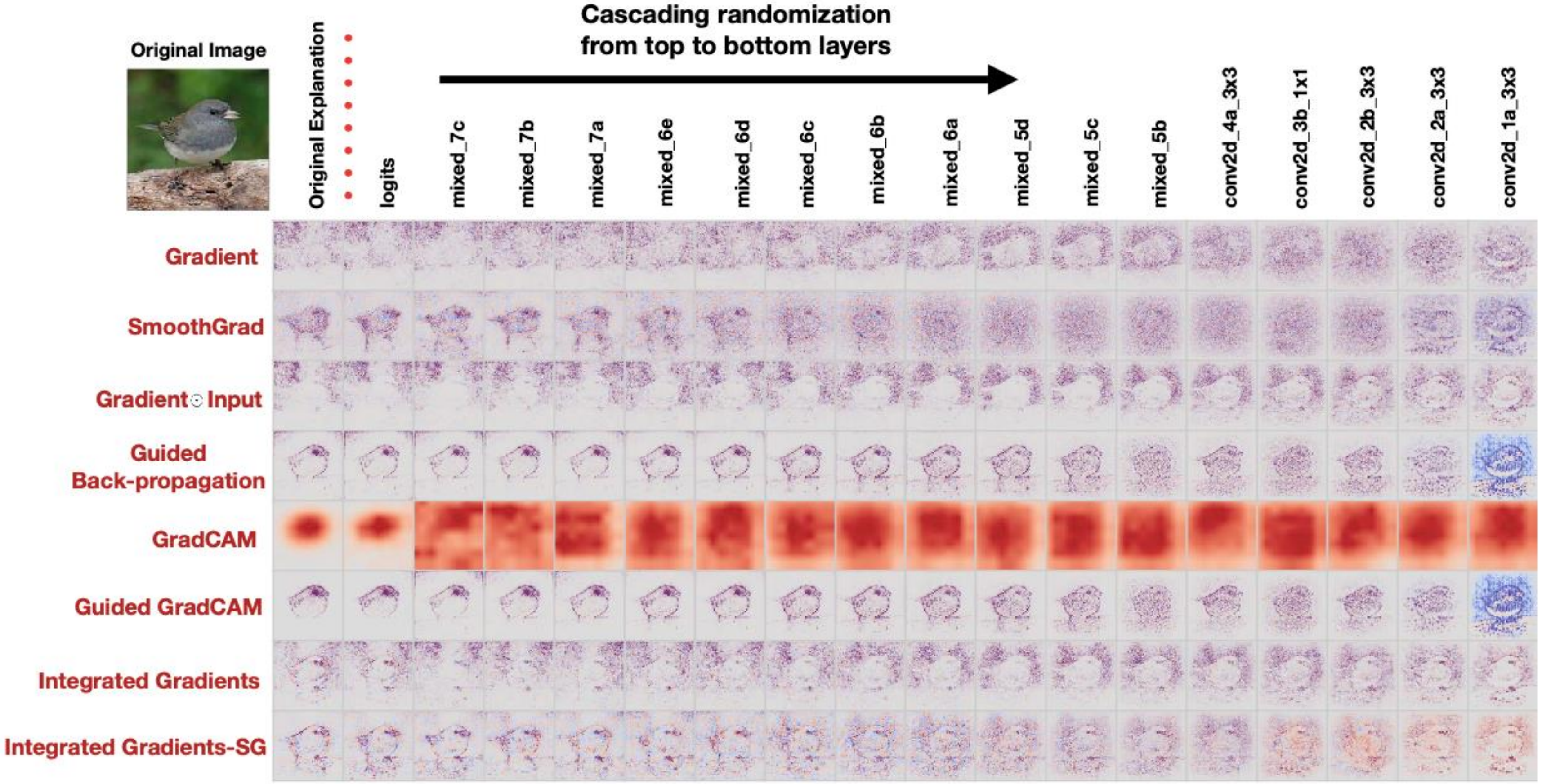
Zhou B, Khosla A, Lapedriza A, et al. Learning deep features for discriminative localization[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 2921-2929.

- The interpretability of CNN
  - Saliency map can tell us where the CNN focus in an image.

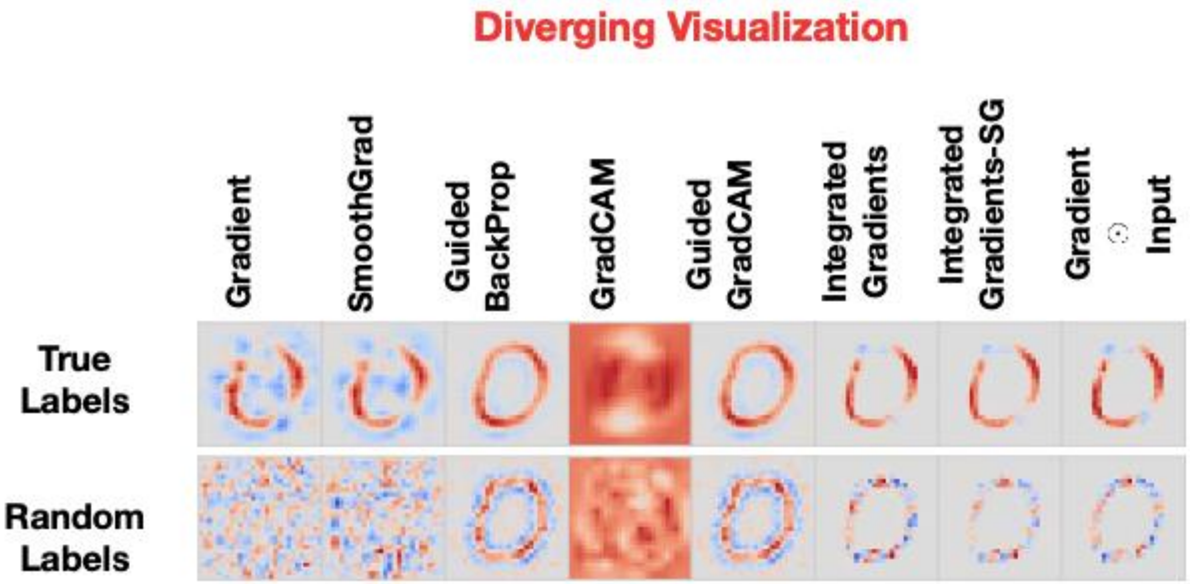


Adebayo J, Gilmer J, et al. Sanity Checks for Saliency Maps. NIPS, 2018.

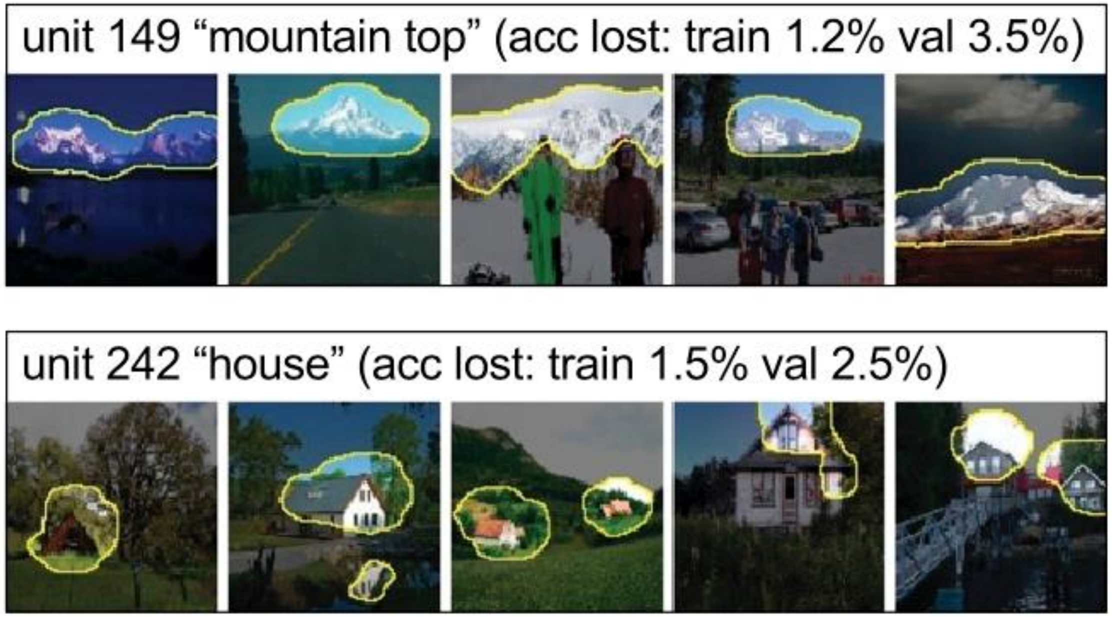
- The interpretability of CNN
  - Shockingly, the saliency map seems unchanged under random parameters …



- The interpretability of CNN
  - ... or random label.



- The interpretability of CNN
  - Just because it "makes sense" to humans, doesn't mean it reflects the evidence for prediction.
  - Concept matters!



Baua D, Zhu J, et al. Understanding the role of individual units in a deep neural network. PNAS, 2020.

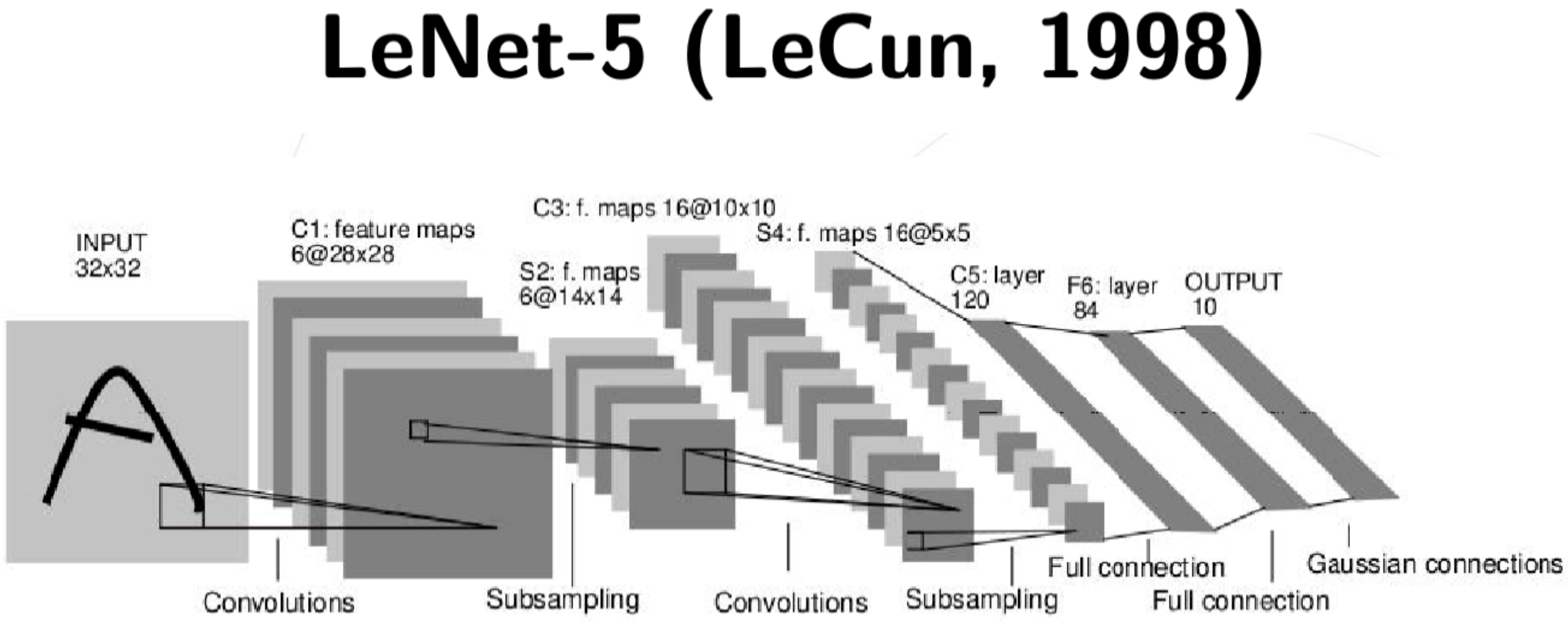
Outline

Part 1	Introduction to CNN
Part 2	The Progress of CNN Architecture
Part 3	Analysis

- **Progress**

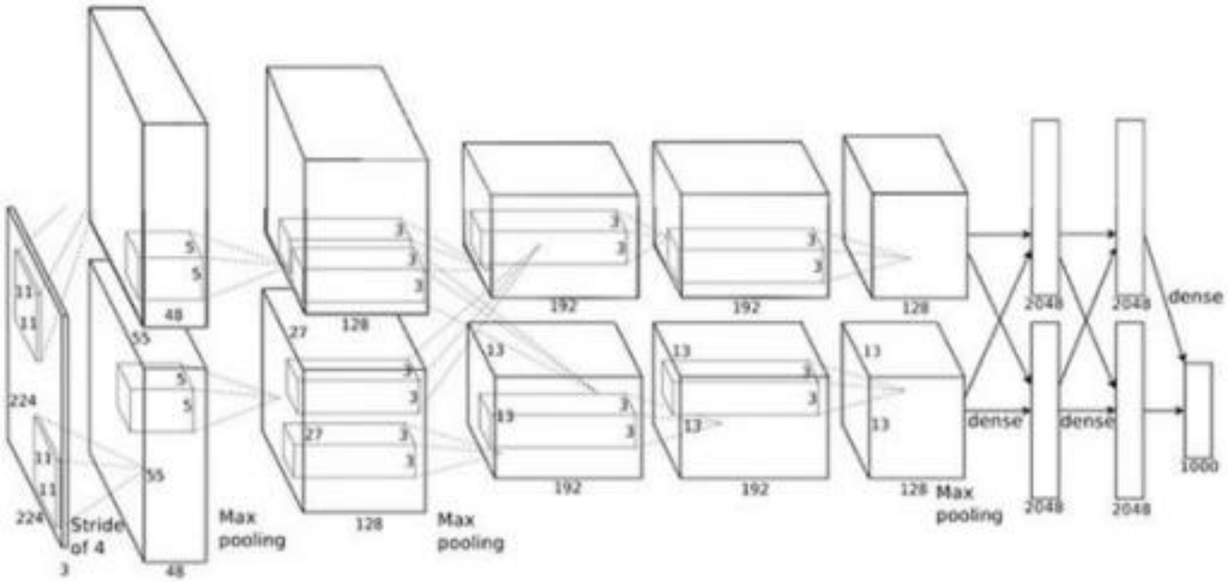
LeNet	AlexNet	NIN	VGG	GoogLeNet	ResNet	...
1998	2012	2013	2014	2015	2016	now

- Practical CNN for Document Recognition



Lecun Y , Bottou L . Gradient-based learning applied to document recognition. Proceedings of the IEEE, 1998.

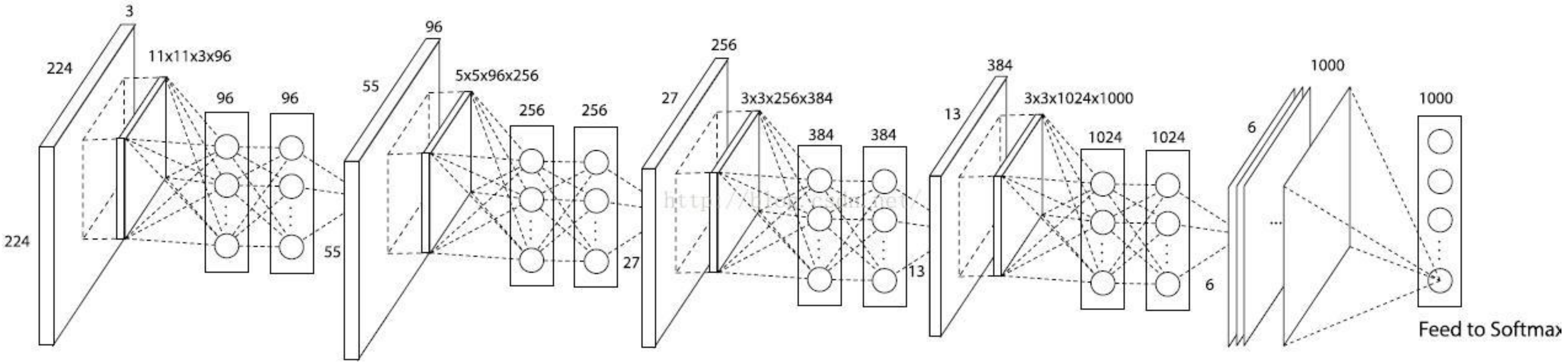
- **AlexNet**
  - Deeper
    - 8 layers
  - ReLU
    - alleviate the problem of saturation
  - Dropout
    - prevent over-fitting



Krizhevsky A, Sutskever I, Hinton G E. Imagenet classification with deep convolutional neural networks. NIPS, 2012.

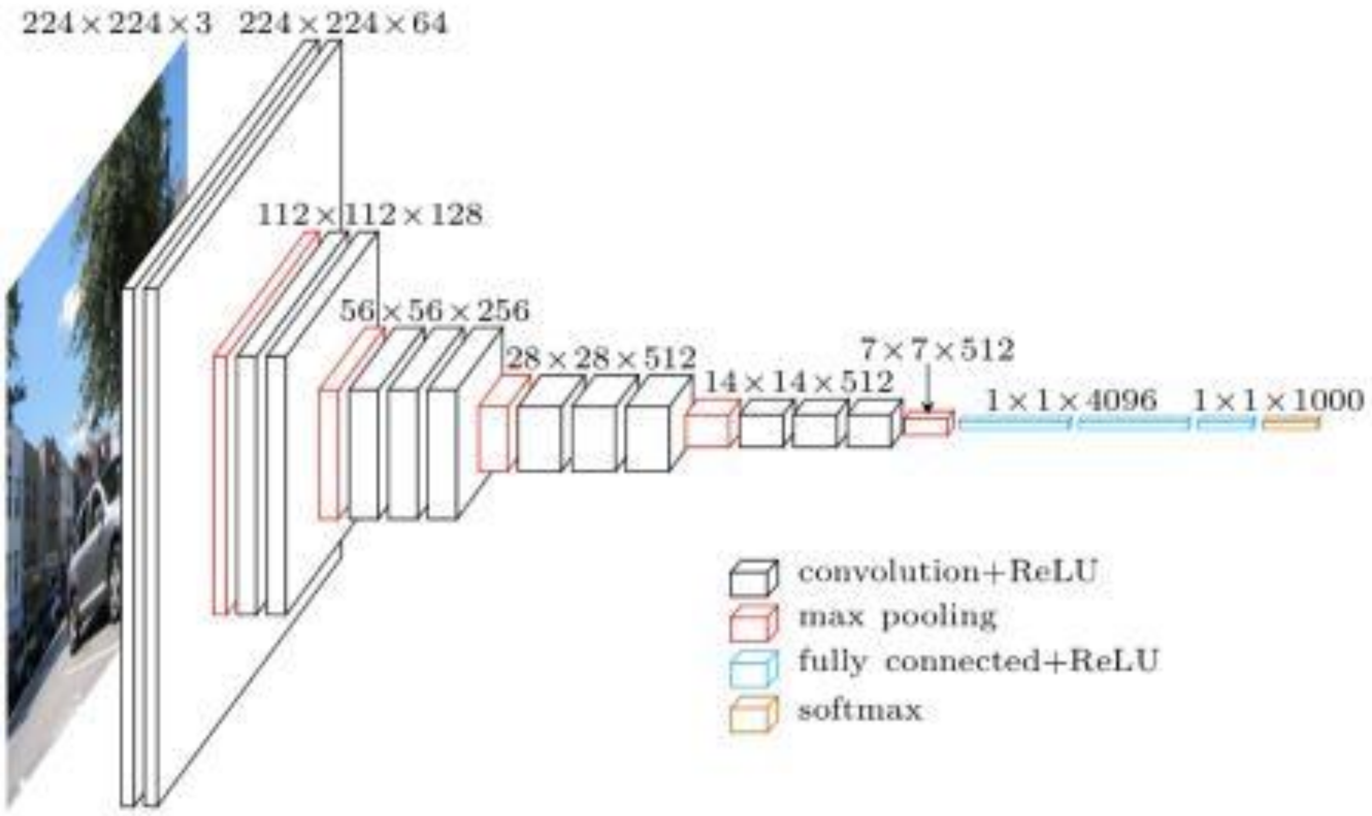
The Progress of CNN

- **NIN**
  - 1x1 conv
    - Improve the flexibility and capacity of CNN
  - Global average pooling (for image classification)
    - Efficient extraction of image feature



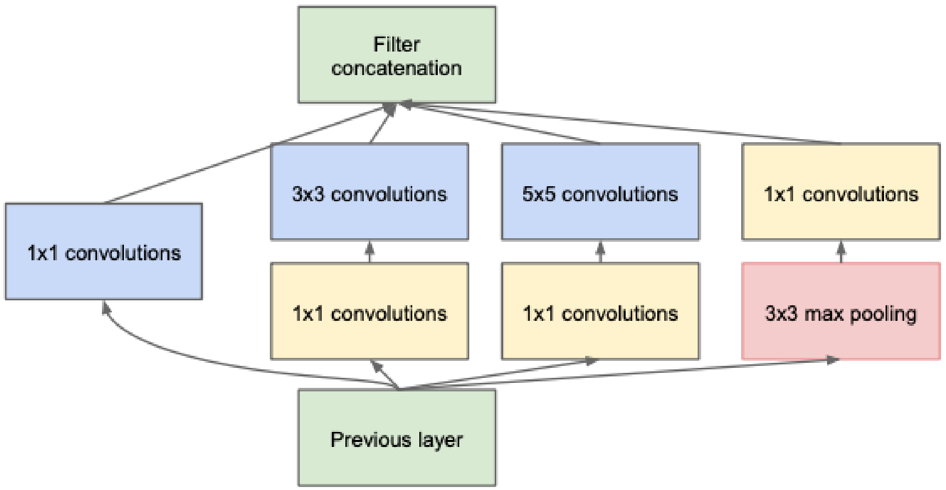
Lin M, Chen Q, Yan S. Network In Network. ICLR, 2014.

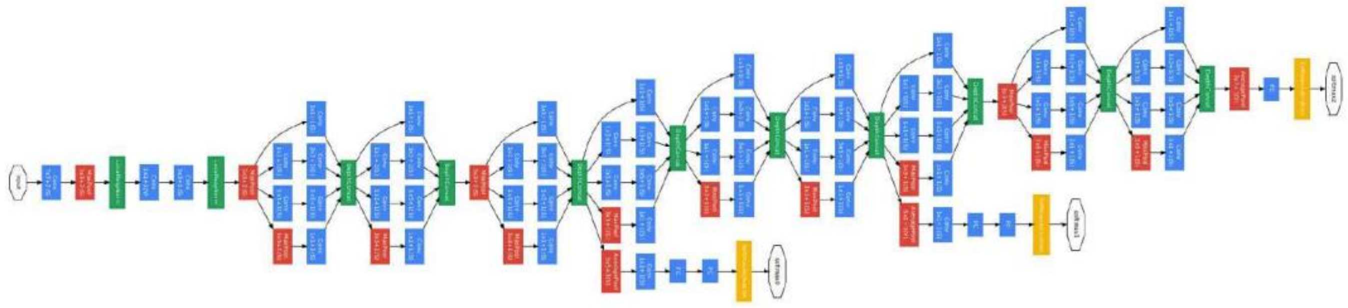
- **VGG**
  - Deeper
    - 16 layers typically
  - Small kernel
    - less parameters
    - more non-linearity
    - larger receptive-field



Simonyan K, Zisserman A. Very Deep Convolutional Networks for Large-Scale Image Recognition. ICLR, 2015.

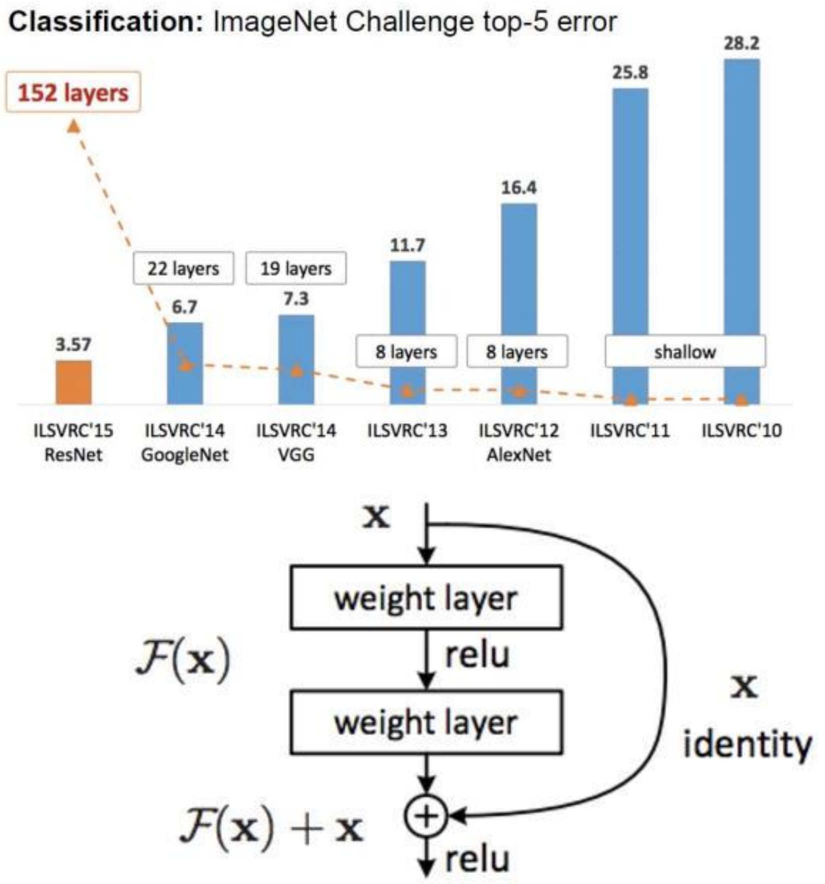
- **GoogLeNet**
  - Deeper
    - 22 layers
  - Inception Module
    - Increase in-block diversity
  - Deep supervision
    - alleviate gradient vanishment and explosion





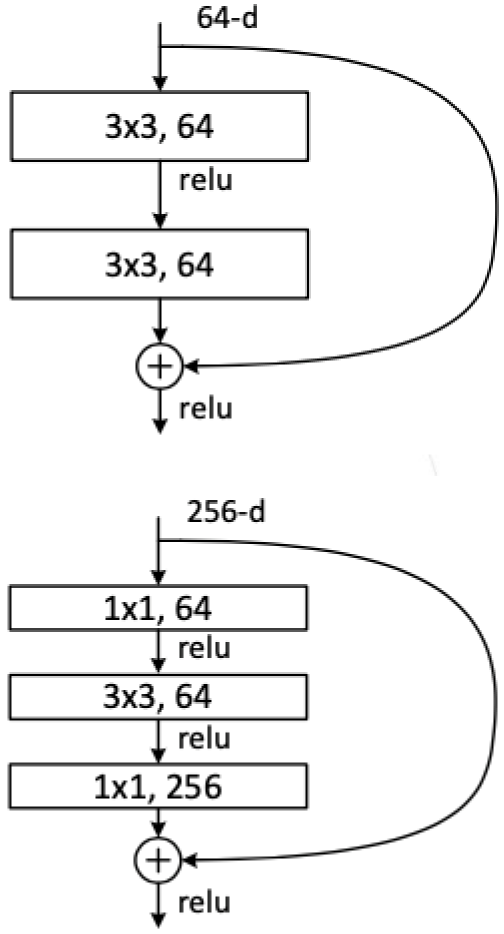
Szegedy C, Liu Q, Jia Y, et al. Going Deeper with Convolutions. CVPR, 2015.

- **ResNet**
  - Much Deeper
    - up to ~1000 layers
  - Residual learning
    - Difficulty of learning identity mapping
    - Similar behavior like an ensemble of many shallow networks



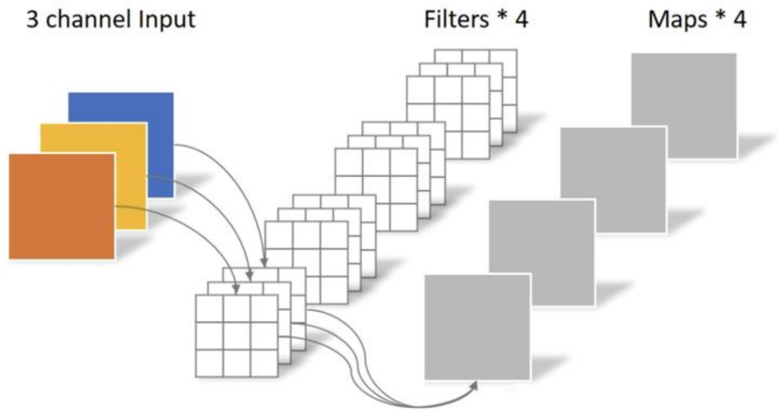
He K, Zhang X, Ren S, et al. Deep Residual Learning for Image Recognition. CVPR, 2016.

- **ResNet**
  - BasicBlock
    - Simple
    - Efficient utilization of GPU memory
  - Bottleneck
    - More channels in residual path to carry more information
    - Balance between 1x1 conv (channel) and 3x3 conv (spatial)

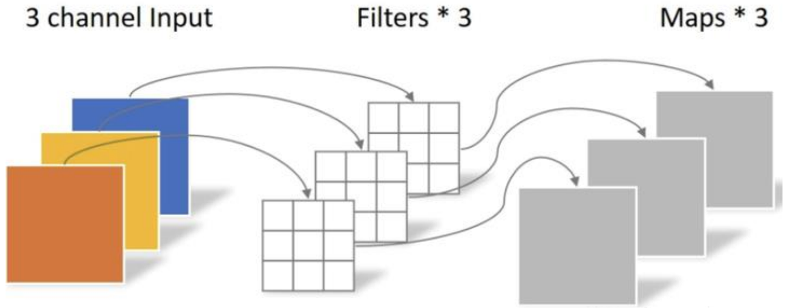


He K, Zhang X, Ren S, et al. Deep Residual Learning for Image Recognition. CVPR, 2016.

- **Light CNN**
  - MobileNet V1
  - Depthwise Separable Convolution
    - Significant reduction of computation
    - Foundamental component of light CNN
    - 3x3 DS conv + 1x1 conv ~ 3x3 conv



$$d_{out} \times d_{in} \times k \times k$$

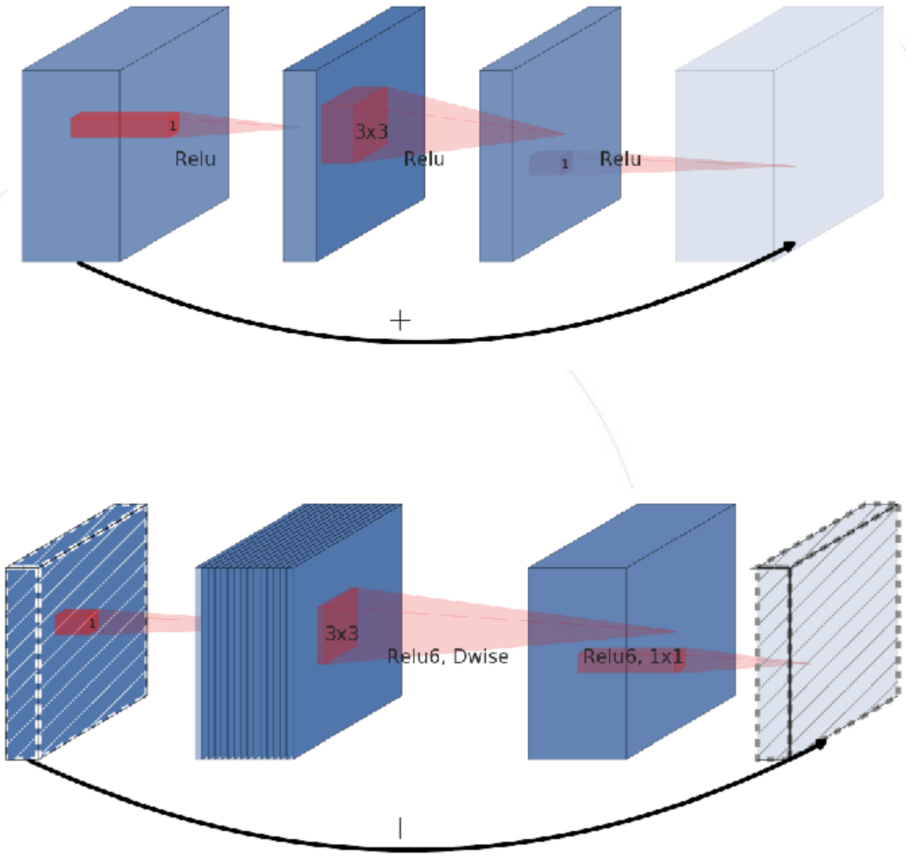


$$d_{in} \times 1 \times k \times k$$

Howard A, Zhu M, Chen B, et al. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. CVPR, 2017.

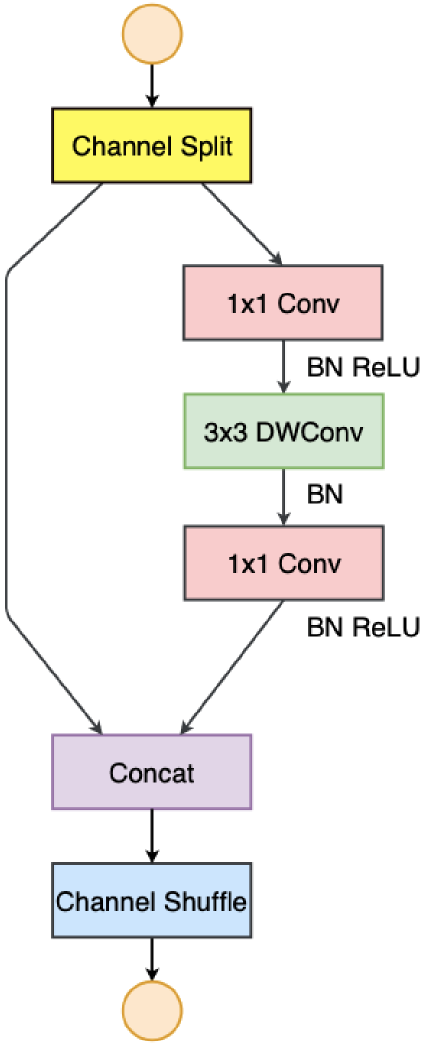
- **Light CNN**
  - MobileNet V2
  - Inverted Residual Block
    - Balance of the computation between 3x3 DS conv and 1x1 conv
  - Architecture design

Input	Operator	t	c	n	s
$224^2 \times 3$	conv2d	-	32	1	2
$112^2 \times 32$	bottleneck	1	16	1	1
$112^2 \times 16$	bottleneck	6	24	2	2
$56^2 \times 24$	bottleneck	6	32	3	2
$28^2 \times 32$	bottleneck	6	64	4	2
$14^2 \times 64$	bottleneck	6	96	3	1
$14^2 \times 96$	bottleneck	6	160	3	2
$7^2 \times 160$	bottleneck	6	320	1	1
$7^2 \times 320$	conv2d 1x1	-	1280	1	1
$7^2 \times 1280$	avgpool 7x7	-	-	1	-
$1 \times 1 \times 1280$	conv2d 1x1	-	k	-	



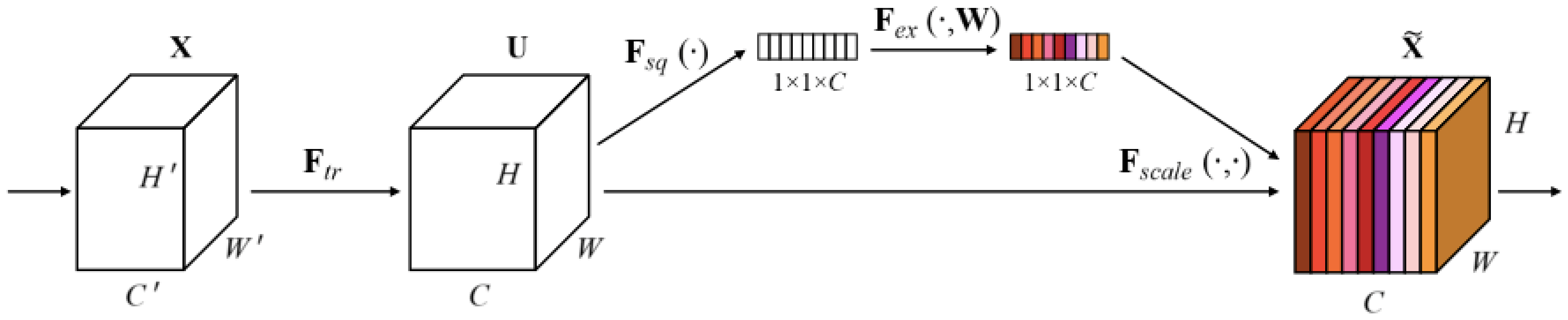
Sandler M, Howard A, Zhu M, et al. MobileNetV2: Inverted Residuals and Linear Bottlenecks. CVPR, 2018.

- **Light CNN**
  - Shufflenet V2
  - Practical Guidelines for Efficient Network Design
    - Equal channel width minimizes MAC（memory access cost）
    - Excessive group convolution increases MAC
    - Network fragmentation reduces degree of parallelism
    - Element-wise operations are non-negligible



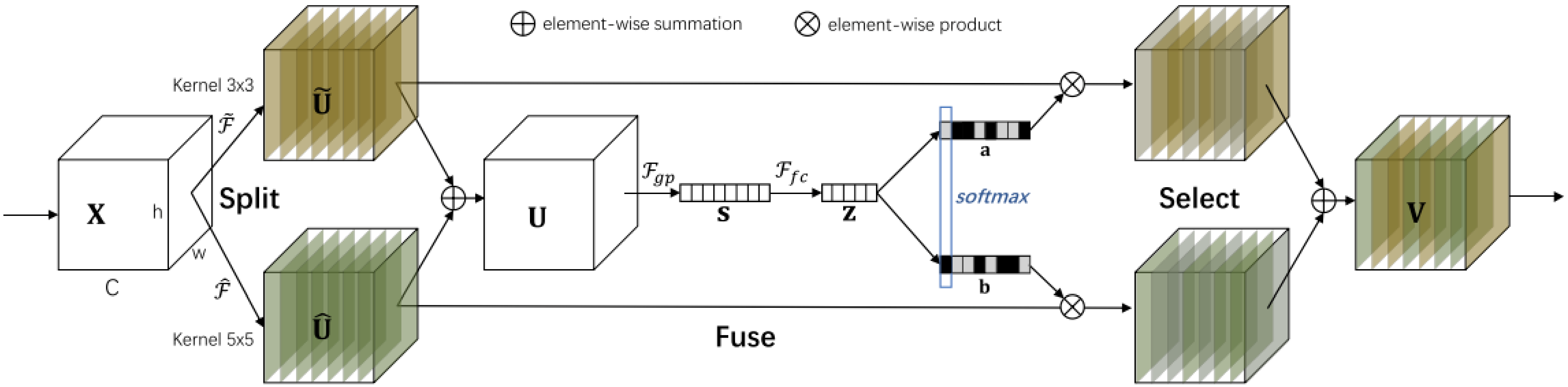
Ma N, Zhang X, Zheng H, et al. ShuffleNet V2: Practical Guidelines for Efficient CNN Architecture Design. ECCV, 2018.

- **Data Dependent Network**
  - SENet
    - Reweight channels according to the input, or Channel Attention
    - Capture global context information via GAP



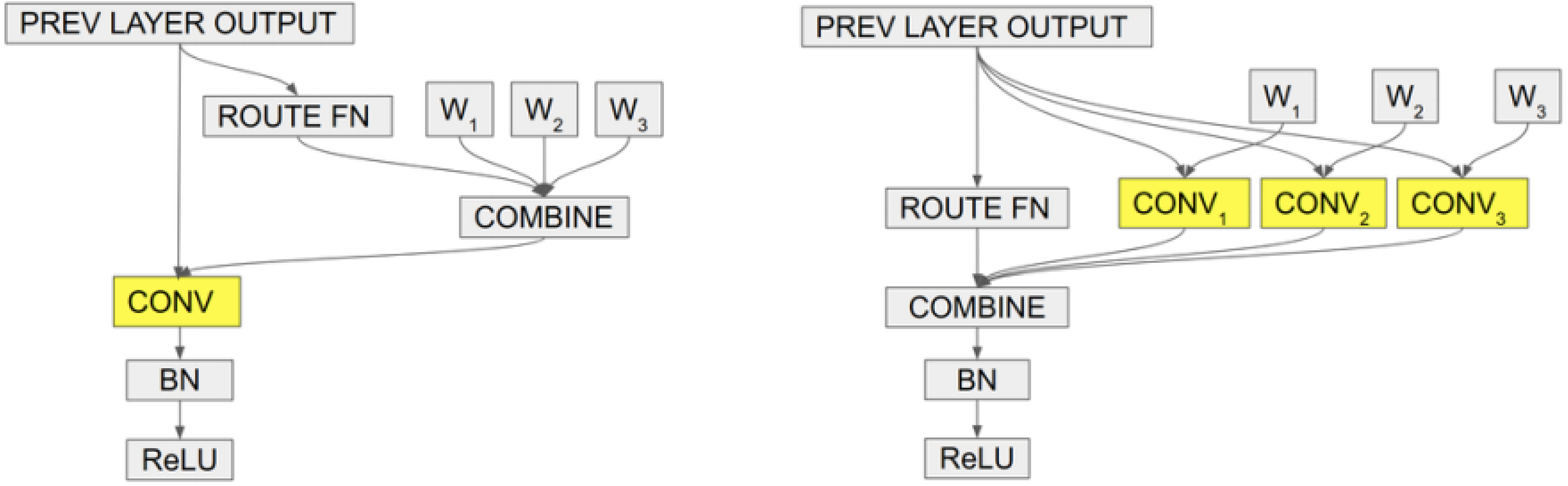
Hu J, Shen L, Sun G. Squeeze-and-Excitation Networks. CVPR, 2018.

- **Data Dependent Network**
  - SKNet
    - Switch between 5x5 conv and 3x3 conv according to the input



Li X, Wang W, Hu X, et al. Selective Kernel Networks. CVPR, 2019.

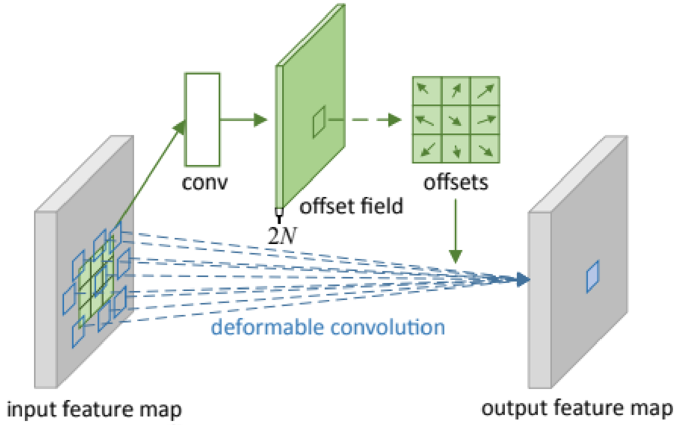
- **Data Dependent Network**
  - CondConv
    - Generate CNN weight according to the input

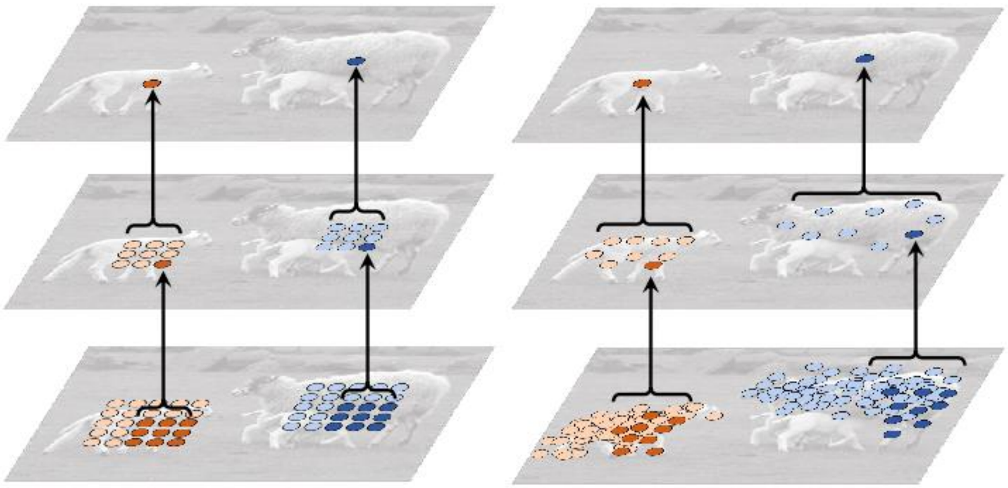


(a) CondConv: $(\alpha_1 W_1 + \ldots + \alpha_n W_n) * x$ (b) Mixture of Experts: $\alpha_1(W_1 * x) + \ldots + \alpha_n(W_n * x)$

Yang B, Bender G, Le Q, et al. CondConv: Conditionally Parameterized Convolutions for Efficient Inference. NIPS, 2019.

- **Data Dependent Network**
  - Deformable Convolution
    - Change the pixel-to-weight relation according to the input
    - Enlarge the receptive field
    - Helpful for the resistance of object deformation





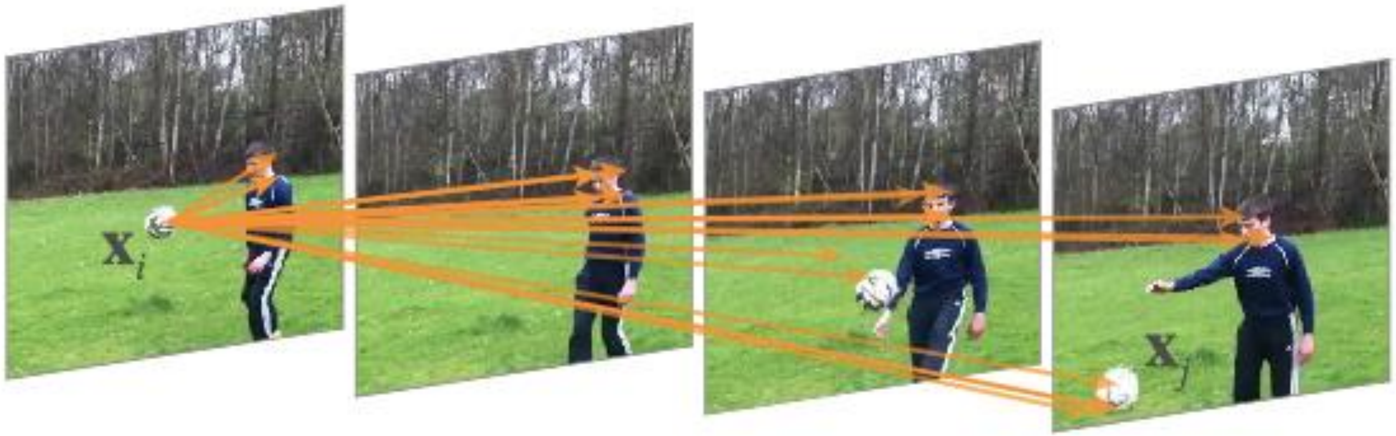
(a) standard convolution (b) deformable convolution

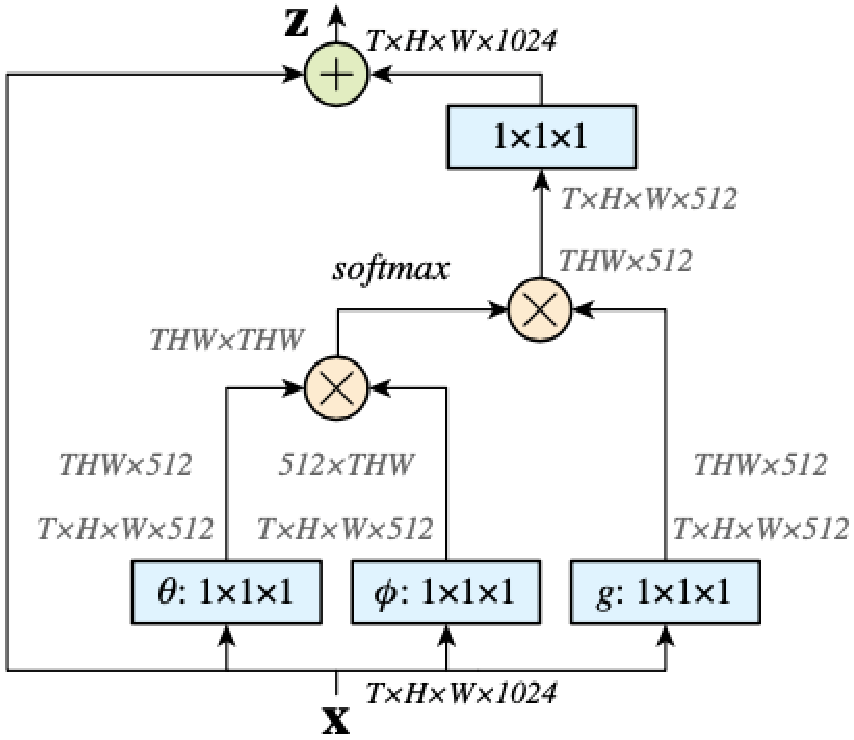
Dai J, Qi H, Xiong Y, et al. Deformable Convolutional Networks. ICCV, 2017.

- **Data Dependent Network**
  - Deformable Convolution

Dai J, Qi H, Xiong Y, et al. Deformable Convolutional Networks. ICCV, 2017.

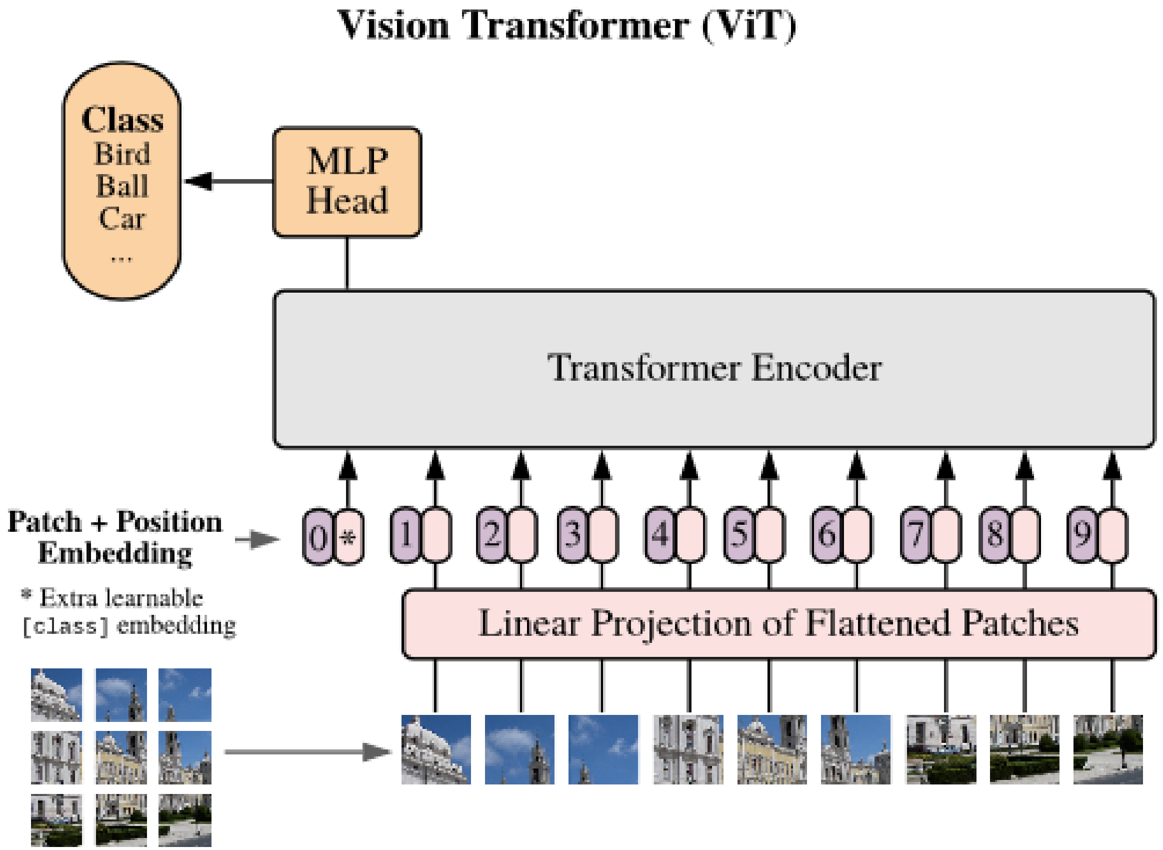
- **Attention Mechanism in CNN**
  - Non-local Neural Network
    - Capture pixel-to-pixel dependency information





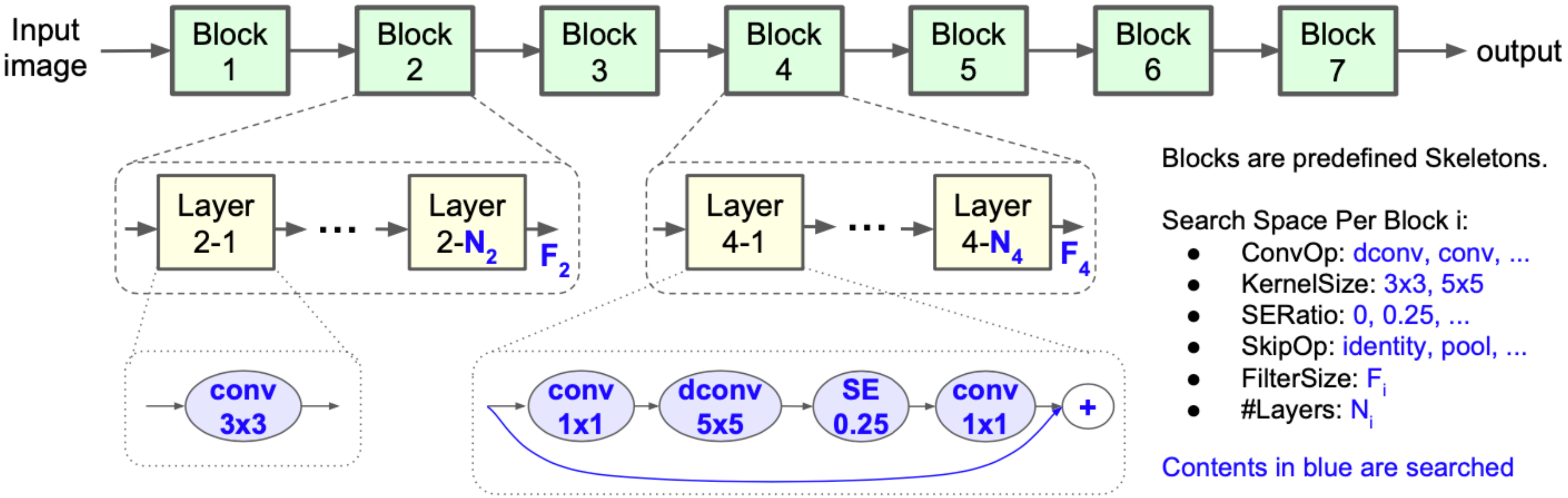
Wang X, Girshick R, Gupta A, et al. Non-local Neural Networks. CVPR, 2018.

- **Attention Mechanism in CNN**
  - Vision Transformer
    - Applying a standard Transformer directly to images
    - See the image as a sequence of patch
    - Require pretrain on large-scale dataset



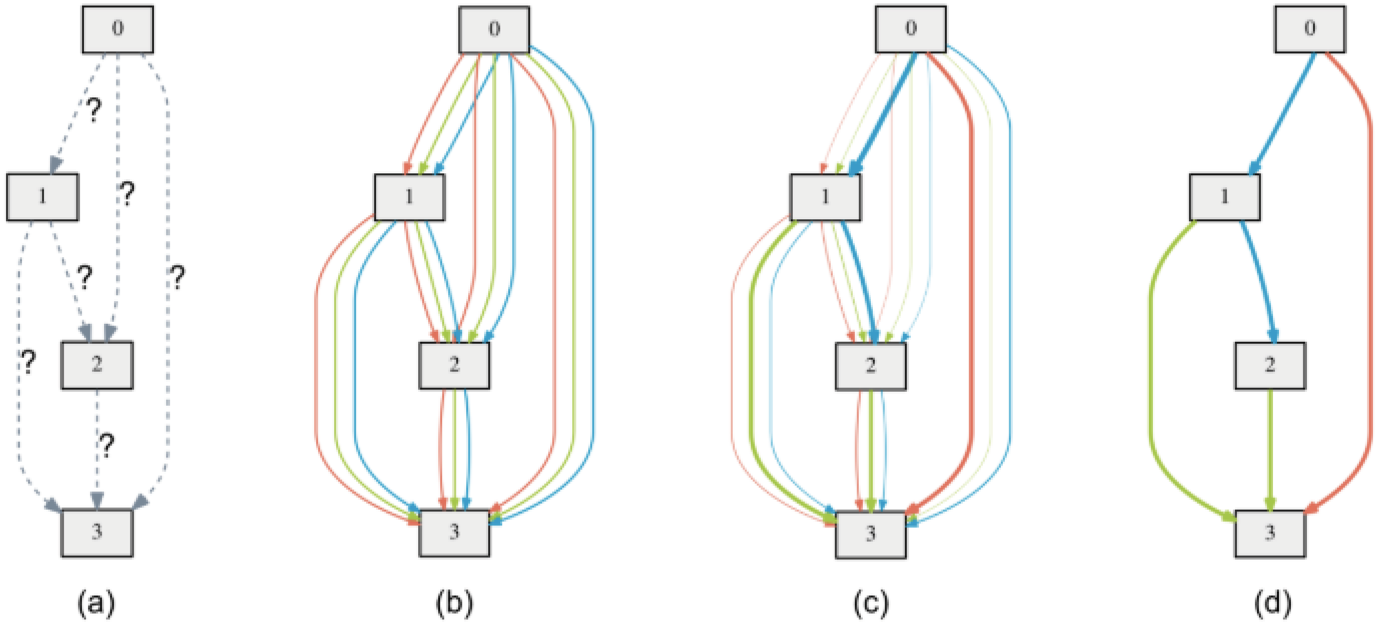
Dosovitskiy A, Beyer L, Kolesnikov A, et al. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. ICLR, 2021.

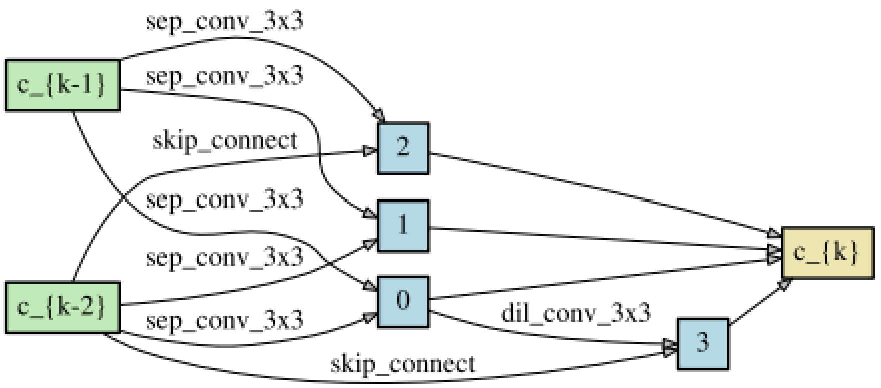
- **Network Architecture Search**
  - MNasNet
    - Search based on Reinforcement Learning
    - Show the superiority of NAS



Tan M, Chen B, Pang R, et al. MnasNet: Platform-Aware Neural Architecture Search for Mobile. CVPR 2019.

- **Network Architecture Search**
  - DARTS
    - General search space
    - Share weight between sub-networks
    - Differentiable search process





Liu H, Simonyan K, Yang Y. DARTS: Differentiable Architecture Search. ICLR, 2019.

sensetime

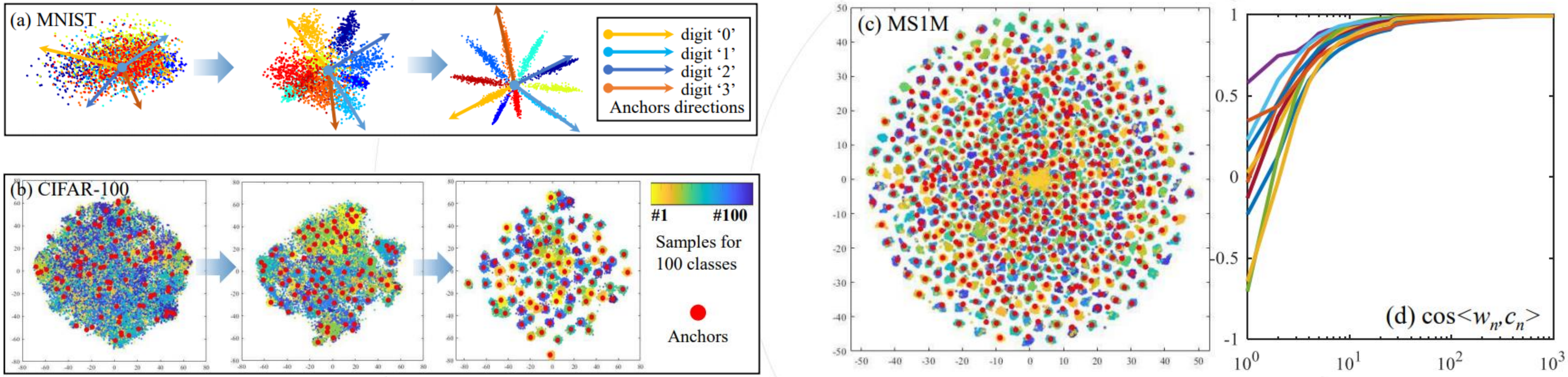
Tsinghua University

Outline

Part 1	Introduction to CNN
Part 2	The Progress of CNN
Part 3	Analysis

Analysis

- High-order feature learned by CNN
  - Understand the feature optimization at training stage



Liu Y , Song G , Shao J , et al. Transductive Centroid Projection for Semi-supervised Large-Scale Recognition. ECCV, 2018.

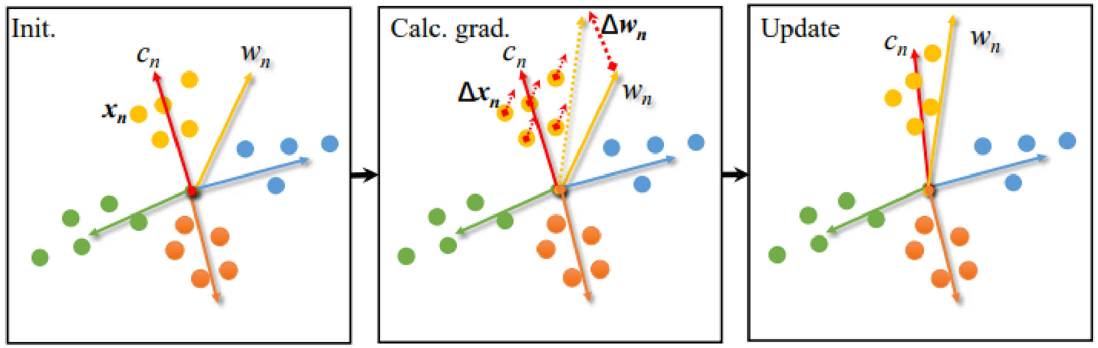
Analysis

- High-order feature learned by CNN
  - Observation inside the Softmax Classifier

$$y = W^T f + b,$$
$f \in \mathbb{R}^D$ denote the feature vector
$y \in \mathbb{R}^N$ denote the class labels
$W \in \mathbb{R}^{D\times N}, b \in \mathbb{R}^N$, denote the weight and bias

$$w_i = W_{[i]} \in \mathbb{R}^D$$
for class i points to the direction of the data centroid of class i, when the model has successfully converged

- Investigate in Gradients

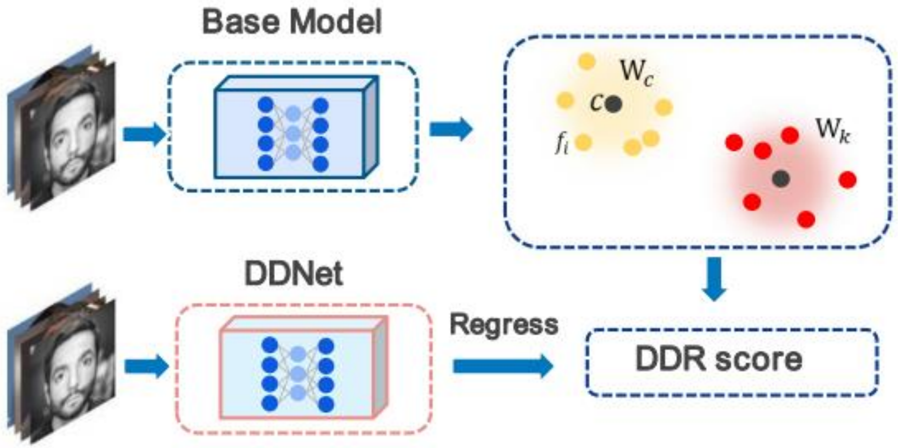


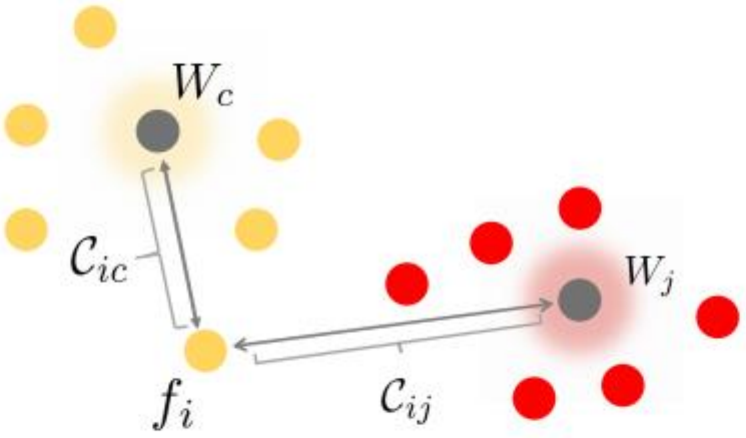
(a) Forward x, get c_n (b) BP, get Δx_n and Δw_n (c) Update w_n and x_n and re-calculate c_n

After this iteration, the directions between anchor w_n and centroid c_n get closer.

Analysis

- Explore information from high-order feature learned by CNN
  - Discriminability distillation regulation





the intra-class distance and inter-class

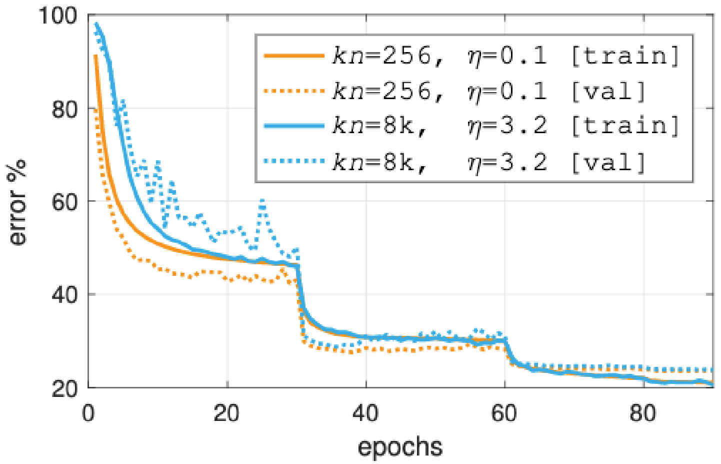
$$C_{ic} = \frac{f_i \cdot W_c}{||f_i||_2 ||W_c||_2} \qquad C_{ij} = \frac{f_i \cdot W_j}{||f_i||_2 ||W_j||_2}, j \neq c.$$

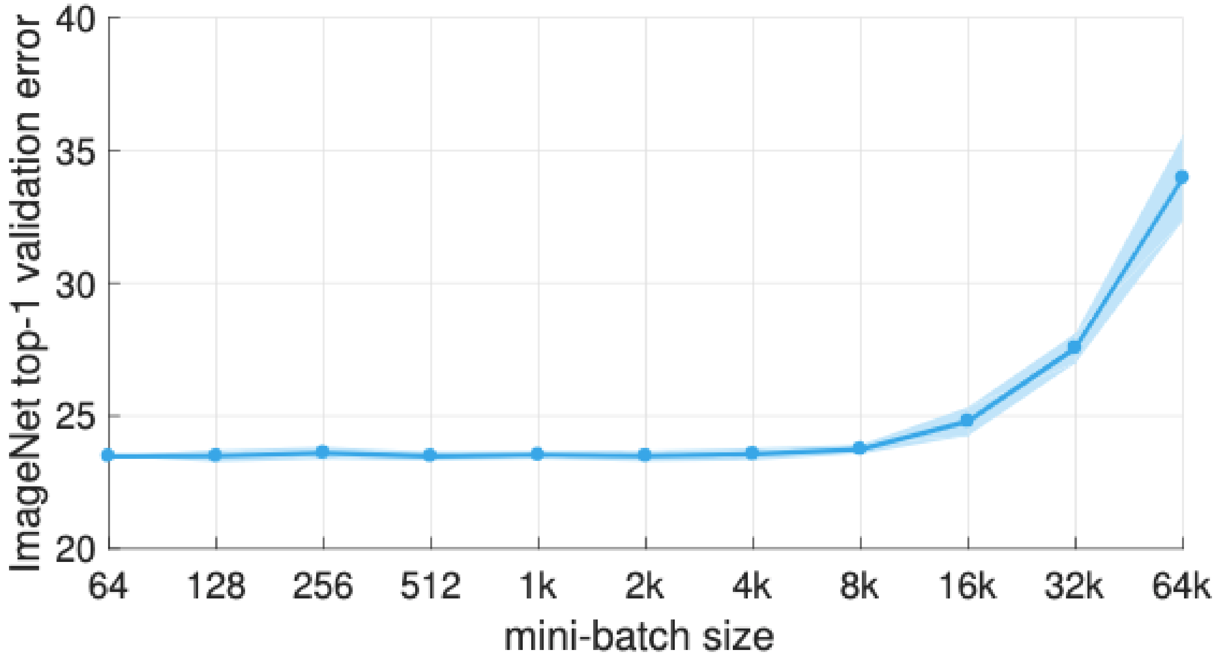
the discriminability Q_i of f_i :

$$Q_i = \frac{C_{ic}}{\max\{C_{ij} | j \in [1, C], j \neq c\}}$$

Images with lower Q have poor quality

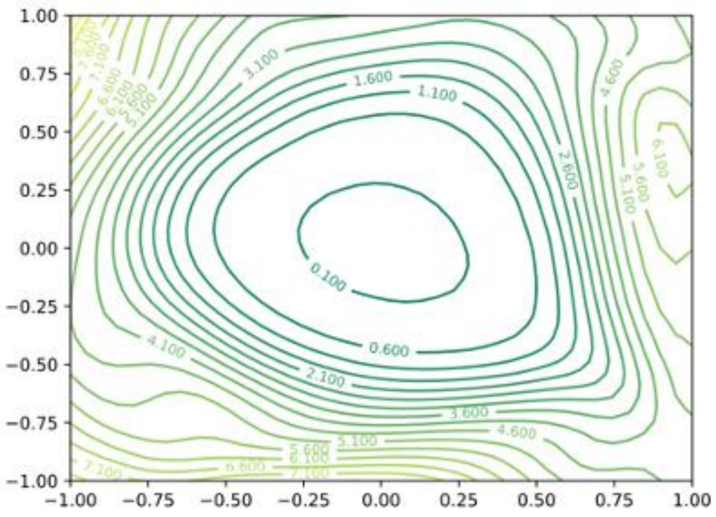
- Large batch training for CNN
  - Learning rate linear scale up
  - Learning rate warmup



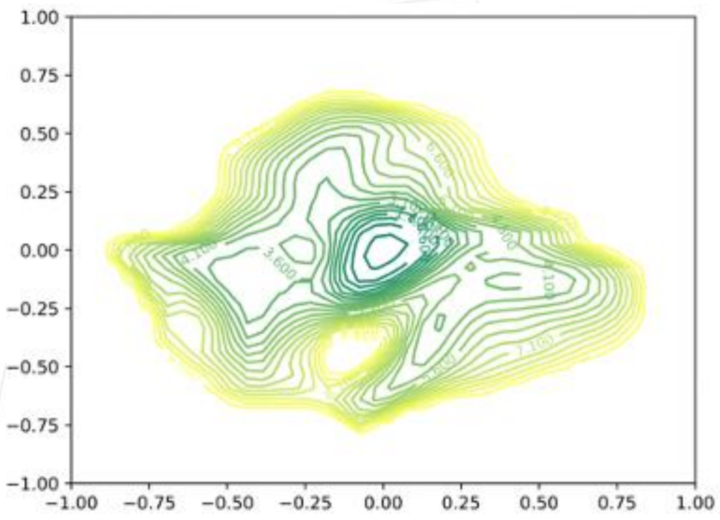


Priya Goyal, Piotr Dollar, Ross Girshick, Pieter Noordhuis. Accurate, Large Minibatch SGD:Training ImageNet in 1 Hour

- Optimization View
  - Large batch training tends to converge to a sharper minima



Loss landscape with batch size 128 on Cifar10



Loss landscape with batch size 50000 on Cifar10

Nitish Shirish Keskar, Dheevatsa Mudigere et al. On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima
Hao Li, Zheng Xu, Gavin Taylor et al. Visualizing the Loss Landscape of Neural Nets

- Gradient Analyze
  - Large norm of gradient leads to unstable training when applying linear learning rate scale up.

Algorithm 1 SGD with LARS. Example with weight decay, momentum and polynomial LR decay.

Parameters: base LR γ_0, momentum m, weight decay β, LARS coefficient η, number of steps T
Init: $t = 0, v = 0$. Init weight w_0^l for each layer l
while $t < T$ for each layer l **do**
  $g_t^l \leftarrow \nabla L(w_t^l)$ (obtain a stochastic gradient for the current mini-batch)
  $\gamma_t \leftarrow \gamma_0 * \left(1 - \frac{t}{T}\right)^2$ (compute the global learning rate)
  $\boxed{\lambda^l \leftarrow \frac{||w_t^l||}{||g_t^l|| + \beta ||w_t^l||} \text{ (compute the local LR } \lambda^l\text{)}}$
  $v_{t+1}^l \leftarrow m v_t^l + \gamma_{t+1} * \lambda^l * (g_t^l + \beta w_t^l)$ (update the momentum)
  $w_{t+1}^l \leftarrow w_t^l - v_{t+1}^l$ (update the weights)
end while

Yang You, Igor Gitman, Boris Ginsburg . Large-batch Training of Convolutional Neural Networks

Assignment Introduction

- **Assignment**
  - All assignments should be finished by one person
  - You can finish assignment on your local machines or on clusters provided by SenseTime
  - More details will be update on Course Homepage

Assignment	Released Date	Due Date	Topic
Assignment 1	Mar. 19	Apr. 2	Deep learning training framework and model optimization implementation
Assignment 2	Apr. 2	May. 7	Advanced Computer Vision Tasks
Assignment 3	May. 7	May. 30	Lightweight Model Quantization and Model Pruning

Assignment 1

实验目标：特定场景和限制下的模型设计与优化

实验描述：

在给定网络结构Flops的限制下，实现对于给定数据集（人脸识别子集）上的模型设计、训练以及验证调优。数据集中会有当前工业界和学术集通用的研究问题，如类别不均衡、数据噪声等因素存在。

实验内容

实验考察对于神经网络标准训练流程的设计与处理，包括数据分析处理、模型设计、训练算法设计、模型调优、结果分析等基本能力，需要基于给定的数据集实现对于评测集的性能调优，记录实验过程及调优方案并且完成一些问答题目，包括：

1.熟悉标准的神经网络训练代码搭建；

2.标准数据集的预处理流程；

3.分析数据集分布，确定Optimizer的配置、Augmentation的选取、训练优化的配置，并实践运用；

4.熟悉模型Flops和参数量计算，网络模型结构设计与调优；

5.掌握常用Evaluation Metrics的计算方式，了解基本的模型性能分析方法以及特征可视化方法

Description

Training data samples：

The number of images is on the order of 10^5.

The training data contains label noise:

intra-class noise: different identities with same label ID.

inter-class noise: same identity with different label IDs.

Test data samples:

The number of images is on the order of 10^4.

Evaluation protocol:

TPR @ FPR1e-5

TPR = TP(true positives) / (TP(true positives) + FN(false negatives))

FPR = FP(false positives) / (FP(false positives) + TN(true negatives))

Constraint:

1.The Flops of submitted model should be less than 500M madds （single image inference）.

2.External training data is not allowed.

Assignment 1

Timeline

报名分组截止	2021.3.18
数据下载开放	2021.3.15 [包含训练样例代码]
提交开放	2021.3.19
	每人每天有2次提交机会
提交截止	2021.4.2

实验资源使用说明

Tsinghua University
sensetime

使用原则

- 集群仅能通过清华校园网进行访问和使用
- 仅供完成实验作业及大作业使用，请勿用于其他用途
- 优先供已选课同学使用

集群资源分组

- 分为6组，每组不超过10人
- 每组设1名组长，由组长对本组服务器进行统一协调和管理
- 右侧扫码报名分组

集群使用

- 访问方式：跳板机+VM
- 每组一套账号及秘钥
- 请妥善保管，不可外泄
- 集群使用如有问题，请在群内及时反馈

其他说明

- 实验作业不会占用太多计算资源
- 如同学们个人或所在实验室有GPU服务器资源，建议可以把公共资源留给更需要的同学
- 校外已选课同学如无法访问清华校园网，且个人无法解决服务器资源，请在问卷中注明