lec 12

Deep learning in depth

① AlexNet

227 X 227 iPut

8 layers

Image net
1 Million size
1000 Class

(5) Convolutional layers

Max Pooling

3 fully Connected layers

Relu

uses local response normalization

Conv 1 — output = number of filters

$W' = (W - K + 2P)/S + 1$

low Compute, lots of Params

error → 16.4

Pool 1 — output Channel = input Channels
have (no) learnable params

Most of memory used in early Convolutional layer	all parameters are Concentrated in fully Connected layers	Floating Points operation in Convolution layer

② ZFNet

extended from alexnet

Conv₁ → 7X7 Stride 2 (بدل) 11X11 Stride 4

More trial and error

error → 11.7%

8 layers

highest memory
more operations

③ VGG

error ↓ 7.3

Same design of AlexNet

19 layers

Conv 3×3 stride 1 pad 1

Pool 2×2 stride 2

⇒ Conv. layers at each spatial resolution take the same amount of Computation

④ GoogleNet

22 layer

→ Focus on efficency

↳ reduce Parameter Count, memory usage Computation

Very efficient

Main Features

① Aggressive Stem ✓

② inception Module ✓

③ Global average Pooling ✓

④ Auxiliary classifiers

① aggressive stem

⇒ Down Sample input

② inception module

⇒ local unit with Parallel branches

⇒ local structure repeated many times through Net.

③ Global average Pooling

⇒ No large FC layers at end

* Collabse spatial dimensions 1 one linear layer

Inception V4 → Resnet + inception

④ Auxiliary classifier

training using loss at end

BC Network is so deep, gradients dont propagate clearly

(3.1)

attach auxiliary classifier at several intermediate points

classify img, recieve loss

Google net was before batch normalization

with BatchNorm no longer need to use this

⑤ Res Net — Residual — 152 layer

emulate shallow model using deeper layer model

hypothesis → optimization problem

Solution → learning identity function with extra layers

residual netrok → stack of many residual blocks

Regular design like Vgg → (2) 3X3 Conv

Network is devided into stages

1st block → 1/2 resolution, double num of channels

* use aggressive stem

Global averge Pooling like google net instead of fully Connected layers

* more accurate
* more Computational

train very deep network

high accuracy / simple design / Moderate efficiency