Zhang - Modern CNNs

AlexNet (Krizhevsky, 2012)

↳ first large scale network to beat conventional CV methods on a large scale

VGG network (Simonyan & Zisserman, 2014)

↳ make use of a number of repeating blocks of elements

network-in-network (Lin et al 2013)

↳ convolves whole NNs patchwise over inputs

GoogleNet (Szegedy et al 2015) multi-branch convos

ResNet (He et al., 2016) one of most pop off their architectures in CV

ResNext blocks (Xie et al 2017) for sparser connects

Densenet (Huang et al 2017) generalization of the residual architecture

8.1. Deep Convolutional Neural Networks (Alexnet)

CNNs did not dominate immediately after LeCun (1995) LeNet

LeNet was only established on small datasets

Pre-2012 neural networks were generally over shadowed by other ML methods

Historically, CV practioners would craft feature extraction & pass this as inputs into the models

↳ rather than learning

feature extraction was the main goal
→ the learning Algo was an after thought

Historicall pipeline:

(1) Data collect

(2) Preprocess data

(3) Feed into feature extractors

(4) input into classifer model

8.1.1. Representation learning

up until 2012 the most important part of the pipeline was representation calculated / cracked by hand

- LeCun, Hinton, Bengio, Ng, amari, schmidhuber
  - believed features ought to be learned

First modern CNN is Alexnet

evolutionary improvement over LeNet

Main diff between LeCun & Alexnet is increase in computational power

8.1.1.1 Missing ingred: Data

trad Methods are based on convex optims (linear & kernel)

Deep models need vast amounts of comper data to out perform this

ImageNet (Deng et al 2009) = 1 mil examples
1000 from 1000 examples

8.1.1.2 Miss ingred: Hardware

8.1.2 AlexNet

8 layer CNN

Showed that features obtained by learning can be Manual features

① Image

② Conv w/ 96 filters

③ Max Pool

④ Conv

⑤ Max Pool

⑥ Conv → Conv → Conv

⑦ Max Pool

⑧ FC → FC → FC

AlexNet changed sigmoid activation to a simpler Relu activation

8.1.4 Discussion

AlexNets improve on LeNets using dropout for accuracy & for ReLu for ease of training

AlexNets downfall is its MLP layers which are very comp heavy

---

8.2. Networks using Blocks (VGG)

Alex = evidence that Deep CNNs can achieve good results

but did not provide a general template to guide researchers in designing new networks

Following section introduces common concepts used in designing deep networks

Researchers used to think in terms of individ NNs

then whole layers

now to Blocks
↳ repeating patterns of layers

Super modern approach is to repurpose pre-trained models called foundational models

VGG = Visual geometry group

Idea of using Blocks first emerged from oxford group

easy to implement repeated code structures using loops & subroutines

8.2.1. VGG Blocks

Historically, the Basic CNN sequence is:
1. conv w/ Padding
2. ReLU for Non-lin
3. Max pool to reduce reso

issue here is that the reso reduce has a limit before all pixels are used up

Simonyan & Zisserman (2014) idea was to use multi convs between pooling

they wanted to know whether deep or wide networks were better

Deep & narrow = better

Stacking 3x3 convs has been gold standard
- Liu (2022) convnets for the 2020s revisits this

8.2.2 VGG Network

Vgg can be thought of in two ways

- Blocks of Convo & Pooling sequences
- A FC layer

Conv layers do not change Dimensions

example vgg:

conv → conv → Conv → Pool

VGG is more computationally expensive than AlexNet

Summary

VGG Might be the First Modern CNN introduced

① Blocks of multiple convolutions

② Preference for deep & narrow

8.3 Network-in-network (NiN)

LeNet, AlexNet & VGG all share common

extract features exploiting spatial structure via a sequence of convs & pool layers

↳ finished w/ fully connected layers

- these FC layers are a problem & require alot of computation

NiN offer

① use 1x1 convs to add local nonlins across channel activations

② use global average pool to integrate across layers

- 1x1 can be considered a fully connected layer @ each pixel

NiN has no FC layer @ the end

reduces params but may increase run time

8.4 Multi-Branch Networks (GoogLeNet)

Mix of NiN, blocks (vgg) & Conv kernels

1st to have clear distinct between:

① Stem (ingest)

② body (data process)

③ head (Pred)

Stem = 2-3 convs to learn low level

bodys = Conv Blocks

head = maps obtained features to task (Classif, Reg, Det etc)

Key Contribution

Problem = Selecting convolution kernels right

Prob is to select correct kernel: 1×1 or 11×11

Solution: to concat multibranch

Block in goog le net = inception Block

4 (n) parrel branches each w/ diff spatial sizes

Also implement 1x1 layers to reduce channels

& the branches are coordinated in the padding to give the same dimensional output

finally all Branches are concatenated

google net is computationally complex

large num of relatively arbitary hyperparams:

1. number channels chosen
2. num blocks prior to Dim reduct

Reason for this is that tools for Auto network Def or design Explor had not been ~~implemented~~ yet introduced

8.5 Batch normilizaton

training Deep NNs is DIFFICULT

Specifically the time to converge

Batch normilizaton helps of this

Batch combined w/ residual Blocks makes it possible to routinely train networks w/ over 100 layers

Also acts as regulerizaton

Generally we preprocess data before training

Standardizaton plays nicely w/ optimizers as it puts params on a similar scale

Batch is applied to individual layers

in each training iteration

① normalize the inputs by subtracting their mean & dividing by their standard deviation (based on current mini batch

② Apply scale coeff & an offset to recover the lost Degrees of freedom

Normalization + Batch statistics =
Batch Normilization

this makes the size of batch chosen very important

↳ cant work w/ mini batches of size 1

After standardizing, the resulting mini batch has zero mean & unit variance

A small constant is added to avoid ever dividing by 0

throughout Deep learning, noise leads to faster training & less overfitting
- regularization

Batch norm layers are slightly diff for MLP vs convolution

fully connected

before activation & After Affine transform

Convolution

Also After conv but before activation

Diff is application on a per-channel basis across all locations

each channel has its own Scale & Shift params

in the context of convolutions, BN is well defined even w/ mini batch = 1

have locations across an image to average

leads to concept of layer normilization

like batch norm but one observation @ a time

8.6 Residual Networks: Resnet & Resnext

As we build deeper networks it is important to understand how adding layers can increase the complexity

More important

- More layers should make the network more expressive not just. Different

only if larger functions contain the smaller ones are we guaranteed that increasing them will increase the expressive power

At the heart of resnet is idea is that every additional layer should more easily contain the identity function as one of the elements

leads to residual block

8.6.2 Residual Blocks

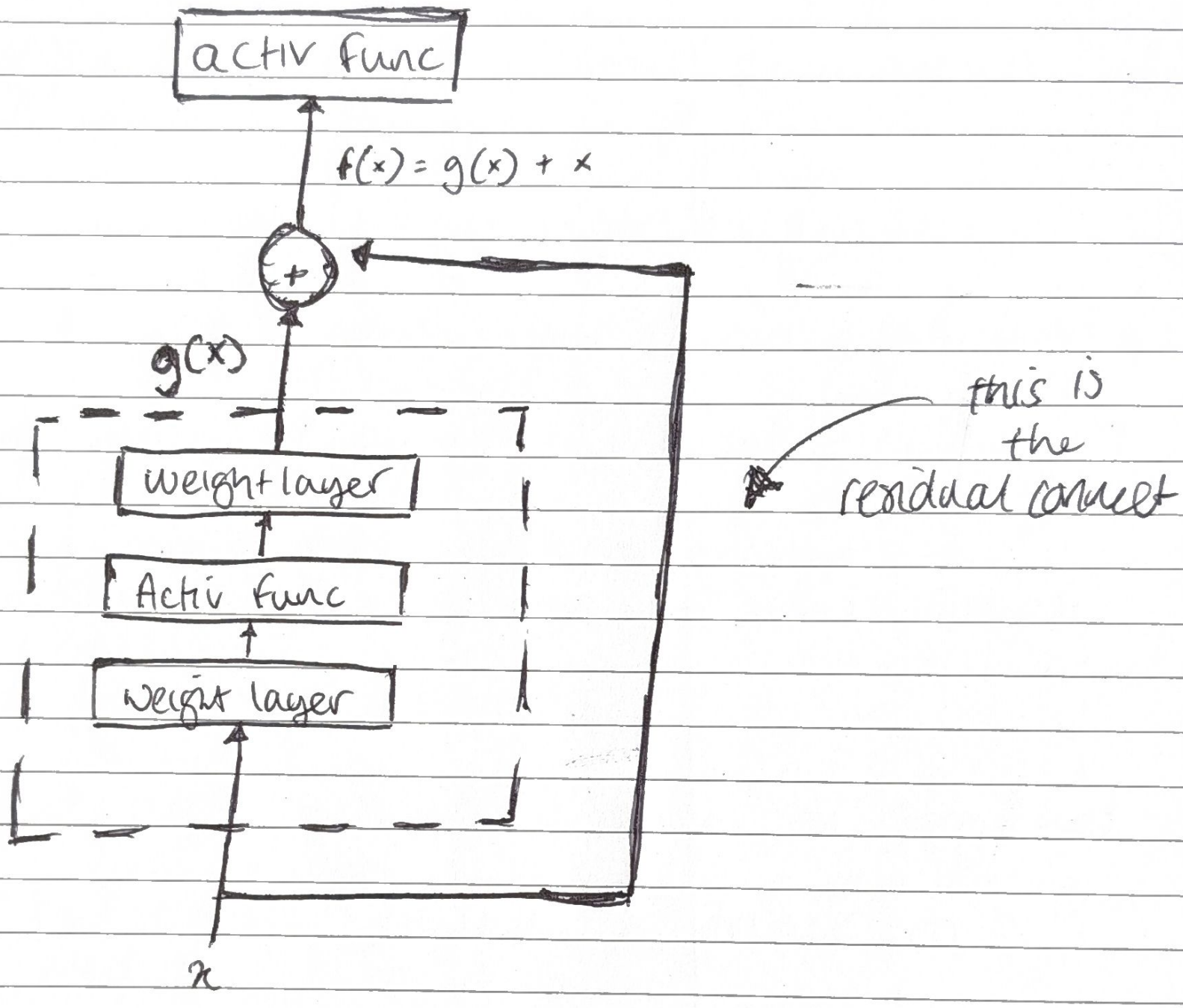
in A Normal Blocks, the $F(x)$ needs to be learned & passed into the activ func

Residual Block learns the residual mapping

$$g(x) = f(x) - x$$

if the ident mapping $f(x) = x$ is the desire mappin

then the residual map = $g(x) = 0$



8.6.3 ResNet Model

first two layers of resnet are the same as googlenet

After:

googlenet uses four modules made up of inception Blocks

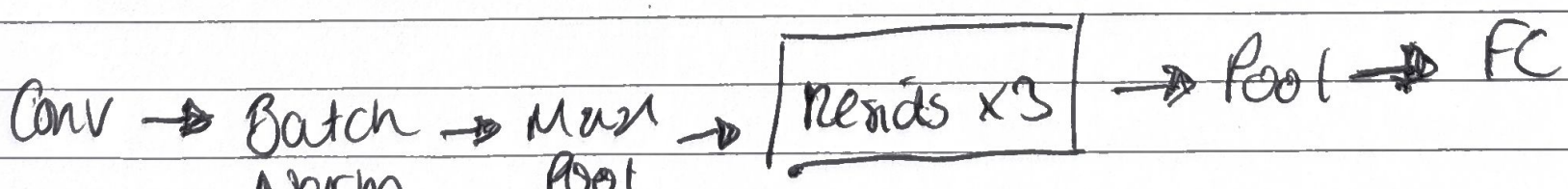
ResNet uses four modules made up of residual Blocks
↳ of which uses several residual Blks

it has 18 layers in total - ResNet-18

By config DIFF nums of channels & residual Blocks you get DIFF Resnet models
i.e. ResNet-152

Resnet architecture is similar to googlenet
↳ but simpler structure & easier to modify

Resnet 18



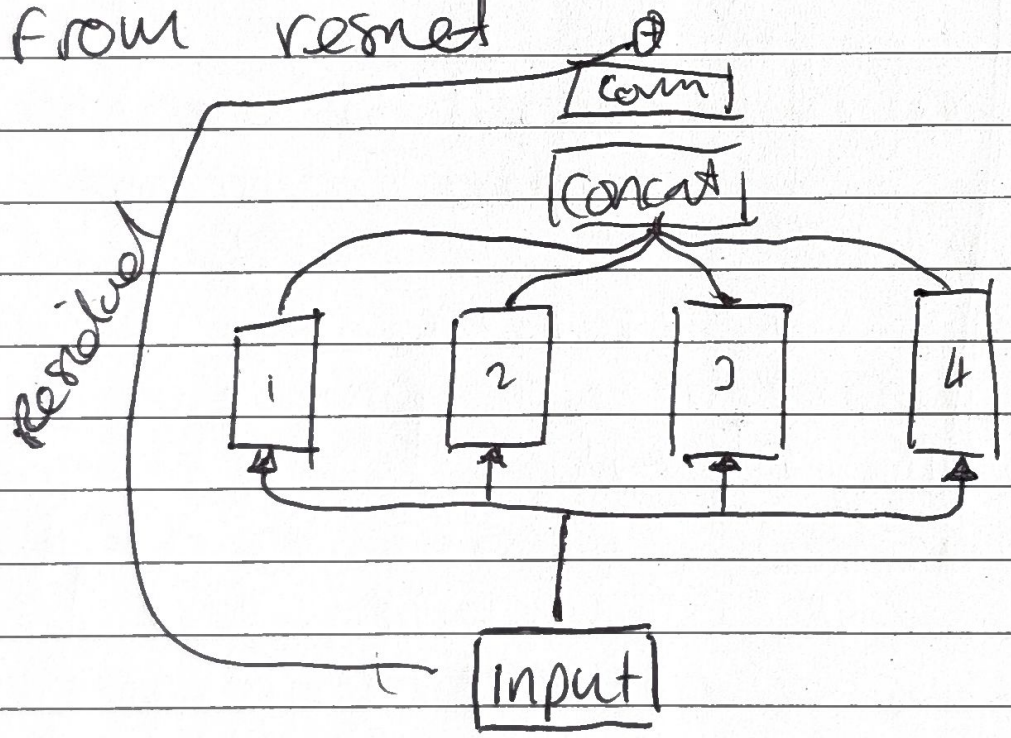
8.6.5 ResNeXt

One of the challenges Resnet faces is the trade-off between non-linearty & Dimensionality within of given Block

Could add more non-linearity by increasing the number of layers

We can take inspo from resnets which has information flowing through the Block in seperate groups

this idea is what lead to ResNext from resnet



this scheme reduces the number of params needed

Initially the perceived challenge is that no info is shared between channels

Resnext amends this in two ways:

① grouped convolution

②

Summary & Discussion

Nested functions/classes are desirable as they allow us to obtain strictly more powerful classes when adding capacity

↳ rather than just being DIFF

Achieve by allowing layers to flow from input to output (residual connections)

this changes the inductive bias from

$f(x) = 0$ to $f(x) = x$

Residual Mapping can learn the identity function more easily

such as pushing params to zero

NNs w/ resids → inputs can forward prop faster through the next works

Allows for training deeper networks

orginal Resnet paper allowed for upto 152 layers

8.7 Densely Connected Networks (DenseNet)

ResNet significantly changed the view of how to parametrize functions in deep nets

DenseNets is a logical extension

Skipped

8.8 Designing Convolutional Network Arch

Ever since AlexNet it has become popular to construct very deep networks by stacking Blocks of convolutions

3×3 convs were popularised by VGG

NiN showed 1×1 convs help add local non-linearities

NiN also solved prob of aggregating info @ the head of a network by aggs across all locations

GoogleNet added multiple branches of diff conv width

ResNets changes the inductive bias towards the identity mapping (from $f(x)=0$) Allowing for very deep networks

upto now we have omitted networks obtained via neural architecture search (NAS)

Cost of doing so is very large