Computer Vision : Kaggle Learn

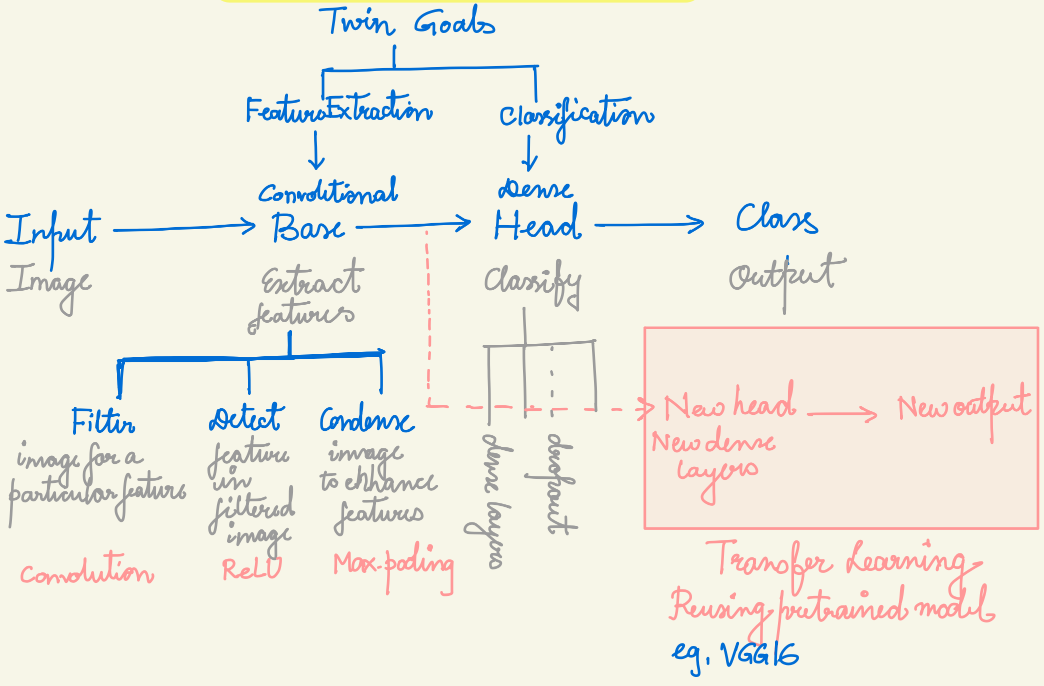
Overview

- → Image Classifier using DL w/ Keras
- → Custom convnet w/ reusable blocks
- → Feature extraction: fundamental ideas
- → Transfer Learning to boost models
- → Data augmentation: extend dataset

Neural Network for CV : Convolutional Neural Networks/CNN/Convnet

- → Image classifier
- → Generative adversarial networks
- → Image segmentation

CONVOLUTIONAL CLASSIFIER



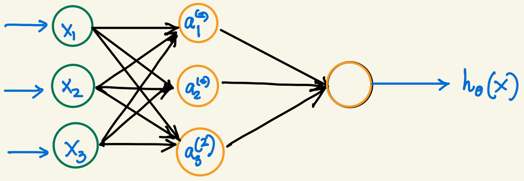
Feature Extraction / Convolutional Base

Filter w/ Convolution

Weights/ Kernels

Activation/ Feature Maps

Training using N.N:



$[x_1 x_2 \dots x_n] \longrightarrow [\quad] \longrightarrow h_\theta(x)$

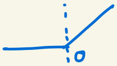
$a^{(1)} = x$
(add $a_0^{(1)}$)

$z^{(2)} = \Theta^{(1)} a^{(1)}$
$a^{(2)} = \boxed{g(z^{(2)})}$
(add $a_0^{(2)}$)

$z^{(3)} = \Theta^{(2)} a^{(2)}$
$a^{(3)} = g(z^{(3)}) = h_\theta(x)$

Detect w/ ReLU

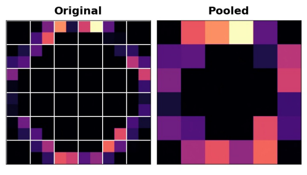
ReLU activation.



→ creates a lot of dead space

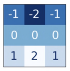
Condense w/ Max Pooling

"Intensifying" features



Translational Invariance
→ not distinguish features by location in image
unless separated by large distances

INPUT

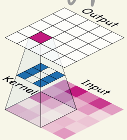


-1	-2	-1
0	0	0
1	2	1

kernel

FILTER

weighted sum of pixels



ReLU

DETECT

pixels scored by measure of importance, eg, ReLU

CONDENSE

Intensifying features while losing some positional information

Con2D: filters, kernel size — activation — maxPool2D

sliding window: kernel_size, pool_size

strides
how far the window moves in each step

padding
how we handle pixel at input edge

>1 : skip over some pixels (max pooling)

=1 : highest quality (conv)

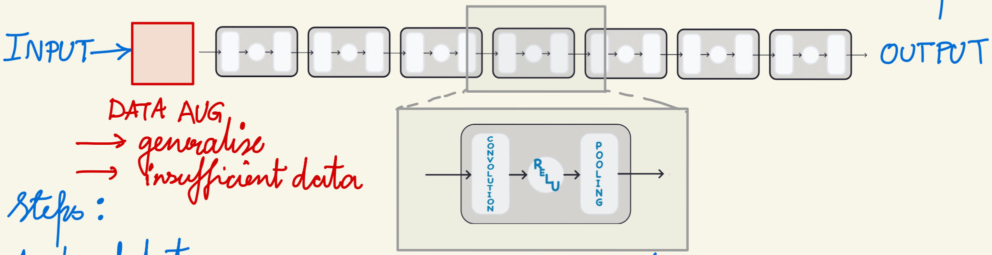
visiontools

valid : conv window inside input → pixel loss

same : pad w/ 0's → dilute effect of border pixels

CUSTOM CONVNET

Convolutional Blocks : Filter → Detect → Condense → Repeat



DATA AUG
→ generalise
→ insufficient data

Steps :

1. Load data
2. Define model
   - → data augmentation
   - → filter
   - → detect
   - → condense
3. Train
4. Evaluate performance
5. Repeat