Convolution:

$$g(x) = f(x) * h(x) = \int_{-\infty}^{\infty} f(\tau)\, h(x - \tau)\, d\tau.$$

'Convolutional Neural Nets'

- also called CNN convnets.
- CNN chop input image into little patches and then process each patch independently and identically.

Convolutional layer: convolution.

- way of combining two functions (image + filter) to get a new filter
- includes flipping a filter before sliding across the image.

Cross-correlation:

- similar to convolution.
- filter not flipped.

Separable filters in CNN:

- instead of using a full square filter (3x3) we can break it into 2 smaller filters
  - → vertical filter (3x1) goes up-down (kx1)
  - → Horizontal filter (1x3) goes left-right (1xk)

- fast and cheap
- less memory

Scanned with CamScanner

Downsampling and Upsampling layers:

smaller → shrinking

bigger → growing

- → strided convolution
  - skips pixels eg. every 2nd
- → pooling (max pooling).
  - biggest value choosen from small patches

- → Dilated convolution (transposed convolution)
  - inserts 0's zeros b/w pixels
  - apply convolution
- → Simply

Nonlinear filtering layers - Pooling layers:

- reduce size of the data while keeping the important features
- removes extra details (small, fast changing one's) high frequency info

Invariance:

- no matter where the object is in the image, the network identifies it.
- CNN does not care about the position

Equivariance:

- if object moves in the image the output also shifts.

* pooling shrinks the image, keeps important info
  it helps the network become less sensitive to invariance.
  I know it moved → I still know what it is

Scanned with CamScanner

Normalization layers.

- adjusting the values so they are consistant.
  no too big or too small values.

Local normalization:

- looks at the small neighborhood of values and adjusts each value based on the values around it.

Identifying vulnerabilities:

- by finding weak spots.
- identify weaknesses in the learned representation and in our training set.

modulation: change frequency content of an image so it fools the network.

Feature Maps.

- snapshot of what network sees.
- focus on each layer
- image transformed into more meaningful info.

1) Image becomes smaller (downsampling)
2) No. of feature maps increased

Image size $32 \times 32 \rightarrow 16 \times 16 \rightarrow 8 \times 8$

maps $3 \rightarrow 32 \rightarrow 64 \rightarrow 128$.

→ Early layers
first layer
detect → lines, edges, colors.

→ Deeper layers
detect abstract things - shapes, eyes/wheels, object.

Scanned with CamScanner

Sliding filter.

- remove nonlinear parts (like ReLU).
- stack multiple convolution layer to make one big convolution.
- nonlinearities affect each pixel individually
- same procss applied to every patch

Treating patches as independent.

1) - easier to solve
   - faster
   - flexible size
   - locality of visual content.

2) - Translation invariance in Images.
   objects do not change when they move.
   - convolution works the same everywhere.
   does not care where the obj is

Encoder	Decoder.
- shrinks the image (downsampling)	- rebuilds the full image (upsampling)
- keep only imp feature	- recover details
- uses less memory + faster	

Information Bottleneck.

- tiny connection narrow space blw encoder and decoder
- only keep essential info.
- forced to learn abstract concepts.

Scanned with CamScanner

U-nets: Encoder + Decoder + skip connections

- encoder when compresses an image some details are lost.
- skip connections pass some of that lost information directly from early layer to later layer.

'Image Pyramids'

Introduction.

→ Translation invariant linear filters.

- works the same way no matter where the obj is in the image

→ Scale invariant processing.

- want our system to recognize obj even if it appear bigger or smaller.

→ To handle different sizes, we need a way to process images at multiple scales (zoomed in and zoomed out).

→ Template matching works if the object size is fixed.

Image pyramids:

- set of copies of same image but at different sizes/resolution
- original image at the bottom (full size)
- as we go up the pyramid, each level smaller and blurrier.
- image is analyzed at different zoom levels

Scanned with CamScanner

Gaussian Pyramid:

1) Low pass filtering:
   blur the images to remove small details
2) Subsampling
   shrinking the image to half its size (subsampling by 2).
3) Repeat the process.
   previous image → blurring → shrinking.

Laplacian Pyramid:

- shows what's different b/w two levels.
- focuses on details (edges and textures).

1) Difference b/w two levels.
   take small blur image resize to the size of larger one and substract it.
   shows what's lost b/w the two images/levels.
2) Repeat for each level
   Do the same process for every pair and form a pyramid of detailed images called laplacian pyramid
3) Low-pass residual.
   the smallest gaussian image is kept at the top of the pyramid

Image Blending:

- Blending in gradual and natural way can be done by using laplacian and gaussian pyramids

1) Build laplacian pyramid for both images.
   Image A → build laplacian pyramid → capture details of A.
   Image B → " → "

Scanned with CamScanner

3) Create a mask and its Gaussian Pyramid.

- make a mask and decide where to blend
  eg left half = 1 (image A) Right half = 0 (Image B)
- create gaussian pyramid of the mask.

4) Blend the laplacian pyramids

Image A laplacian * mask.
+
Image B laplacian * 1 - mask.
Add.

$l_{out} = l^A * m + l^B * (1-m)$

4) Reconstruct the final image

take blended laplacian pyramid and collapse it.
final image found

'Sampling and Aliasing'

Intro to Sampling.

- converting something smooth and continuous into a set of points or pixels that a computer can understand

Two goals of sampling.

→ More samples = more details ↑ memory + time to process
→ Fewer samples = faster and lighter lose details.

Aliasing.

- sample slowly and computer misunderstands the original signal fast wave cos(18πt) but sample it slowly and looks like a slow wave, this is wrong and is called aliasing (error).

Scanned with CamScanner

Sampling Theorem: Nyquist Theorem.

- states how fast we need to sample
- sample at twice the highest frequency
- sampling frequency $\omega_s = 2\pi / T_s \geq 2\omega_{max}$
- sampling period $T_s < T_{min}/2$, $T_{min} = 2\pi/\omega_{max}$

Anti-Aliasing:

- shrink an image ($412 \times 512 \rightarrow 26 \times 32$) and make it big again
- image looks blurry details lost.
- anti-aliasing smoothes out these problems.

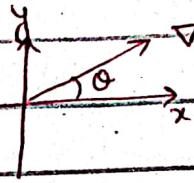
Image Derivatives

- Image derivative tells us how intensity changes in an image.
- detects edges and transitions.

Derivative - discrete approximation

~~$\frac{dI}{dx} = \lim_{\varepsilon \to 0} \frac{I(x+\varepsilon) - I(x)}{\varepsilon}$~~

$\nabla I = \left(\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}\right)$. $\frac{\partial I}{\partial x}$ = change in intensity horizontally $= I(x,y) - I(x-1,y)$



$\frac{\partial I}{\partial y}$ = change in intensity vertically $= I(x,y) - I(x,y-1)$

★ derivative can't be used on pixels. difference used to approximate it

Scanned with CamScanner

Derivative as a convolution:

The derivative operator is linear and translation invariant. Therefore, it can be written as convolution.

Filter is slidded across the image to compute derivative

$I(x,y) \longrightarrow \boxed{\frac{\partial}{\partial x}, \frac{\partial}{\partial y}} \longrightarrow (I_x(x,y), I_y(x,y))$.

Derivative - discrete approximation

- 2 filters to compute 1D derivative.

1) $d_0 = [1, -1]$

$I * d_0 = I[n] - I[n-1]$

subtracts previous value from current value.

fast but not smooth

2) $d_1 = [1, 0, -1]/2$

called central difference

$I * d_1 = \frac{I[n+1] - I[n-1]}{2}$

neighbor on left + right / 2.

balanced + smoother.

Scanned with CamScanner