U-net 2015

Monday, July 7, 2025 7:23 AM

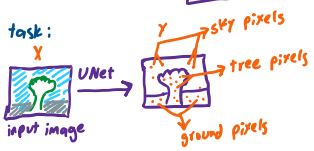
Unet: Convolutional Neural Network for Biomedical Image Segmentation

Abstract

- Presents a CNN architecture & training strategy using data augmentation to train more efficiently

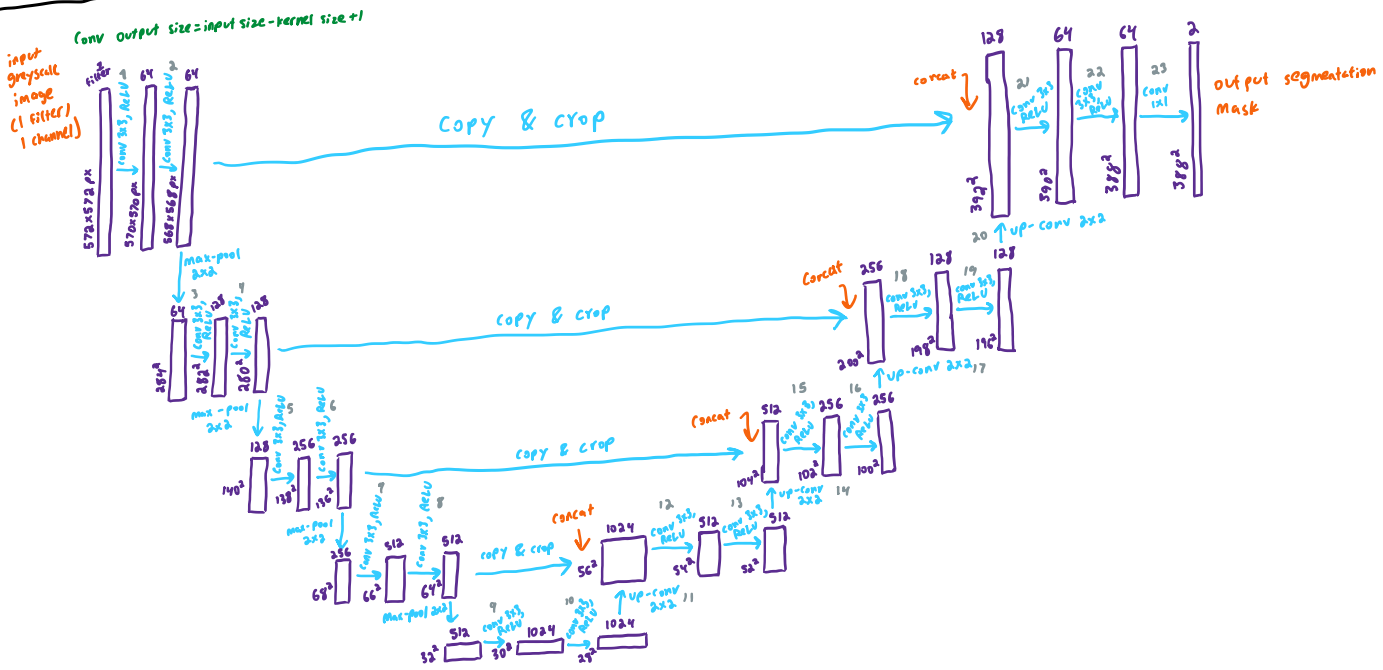
Introduction

- Traditional CNN task: image → CNN → label (X → Y)
- UNet task:



Network Architecture

U-net for 572x572px input:



- Works with very few training images
- Yields more precise segmentations
- Down-sampling (Shrinking the image):
  - captures global context
  - Reduces computation
  - extracts abstract features
- Up-sampling (expanding the image):
  - Reconstruct spatial details
  - Generate dense predictions (e.g., segmentation)

Parameter Count

Learnable Params = output channels x (input channels x kernel height x kernel width + 1) ← bias

for conv & up-conv

max-pool learnable params = 0 (since it takes a maximum value)

Learnable Params for U-net (above):

Conv	out ch.	in ch.	rec.	# Params
1	64	1	9	64(1·9+1) = 640
2	64	64	9	36928
3	128	64	9	73856
..	128	128	9	147584

Training

X: input image

y: segmentation mask

- Favors large input tiles over large batch sizes to reduce the overhead and reduce batch size to single image
- use high momentum (0.99) so a large # of previously seen training samples determines the update in optimization

Momentum meaning in SGD:

Vanilla SGD	SGD w momentum
$\theta_{t+1} = \theta_t - \eta \cdot \nabla L(\theta_t)$	$\theta_{t+1} = \theta_t - v_t$
• Takes step in direction of negative gradient	$v_t = \gamma \cdot v_{t-1} + \eta \cdot \nabla L(\theta_t)$
• η: learn rate, based on mini-batch	• start with $v_0 = 0$ and init. weights θ_0
• Problems: -slow -noisy -unstable	• $\nabla L(\theta_t)$: gradient of loss
	• γ: momentum coefficient
	• v_t: weighted sum of recent gradients
	• Intuition: pushing a ball down hill: -if ball picks up speed, it keeps moving -ball isn't stopped by small bumps – ignores short-term noise -smooths noisy gradients -accelerates when constantly moving in one direction -slows down when switching directions frequently

5	256	128	9	295168
6	256	256	9	590080
7	512	256	9	1180160
8	512	512	9	2359808
9	1024	512	9	4719616
10	1024	1024	9	9438208
11	1024	1024	4	4195328
12	512	1024	9	4719104
13	512	512	9	2359808
14	512	512	4	1049088
15	256	512	9	1179904
16	256	256	9	590080
17	256	256	4	262400
18	128	256	9	295040
19	128	128	9	147584
20	128	128	4	65664
21	64	128	9	73792
22	64	64	9	36928
23	2	64	1	130
				33,816,898

Learnable Params

Calculating Loss:

$p_k(x)$: model's predicted probability for class k at pixel x

$p_k(x)$: Softmax $= \frac{e^{a_k(x)}}{\sum_{k'=1}^{K} e^{a_{k'}(x)}}$

$p_{\ell(x)}(x)$: predicted probability at pixel x for correct class, $\ell(x)$

$$\text{Loss: } E = -\sum_{x \in \Omega} w(x) \cdot \ln\left(p_{\ell(x)}(x)\right)$$

- $w(x)$: pixel-wise weight
- $\ell(x)$: true class label at pixel x
- $x \in \mathbb{Z}^2$: pixel location
- $\Omega \subset \mathbb{Z}^2$: image domain

Loss/Energy is the negative log-likelihood of the sum of all pixels, weighted by how important each pixel is and by the predicted probability of the correct class at that pixel.

- Problem: Foreground/background distribution is uneven, which incentivizes the model to predict background (since it occurs more frequently).

Solution: Gives more weight to underrepresented classes, so they fairly impact the total loss
- IF $\ell(x)$ is foreground, set $w(x)$ to higher value
- IF $\ell(x)$ is background, set $w(x)$ to lower value

$d_1(x)$: distance from border to nearest cell

$d_2(x)$: distance from border to second nearest cell

$$w(x) = w_c(x) + w_0 \cdot \exp\left(-\frac{(d_1(x)+d_2(x))^2}{2\sigma^2}\right)$$

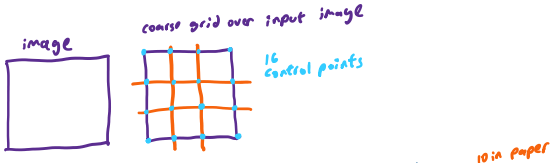
- $w(x)$: pixel weight
- $w_c(x)$: pixel class weight
- w_0: 10 from paper
- σ: 5 from paper

the pixel weight is the sum of its class weight and an exponential boost based on how close the pixel is to neighboring cells.

Data Augmentation

- Data is augmented to teach network desired invariance and robustness properties when there's limited samples
- Augmentations:

1) Create coarse grid (3x3) over image



2) Apply random transformation at each control point according to displacement vector: $\Delta(x,y) = (\Delta_x, \Delta_y) \sim N(0, \sigma^2)$ (σ: 10 in paper)

U-net Summary:

Architecture:
- Downsamples the image while increasing # of filters
  - this captures large features and context
  - extracts hierarchical features
  - reduces computation
- Upsamples the image while reducing # of filters
  - restores spatial resolution
  - enables pixel-wise prediction
  - recovers fine details
- Downsampling understands what is in the image
- Upsampling figures out where each thing is in the image
- Output is smaller due to pooling & upsample operations

Training:
- Loss considers the frequency of the class each pixel is in and the predicted probability at that pixel of the correct class.
- Augments trainig data to increase training samples