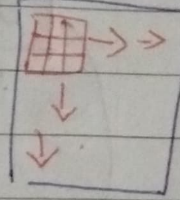
Unet -> Image Segmentation

pooling operator ↓ upsampling operators

Initial Approach -> sliding window.



Drawbacks:

↳ ① slow (network run seperately for each patch) & a lot of redudancy due to overlapping patches.

inclusion of extra non-necessary components.

↳ ② trade off betn localization accuracy & use of content

① large patches (images) require more mom pooling layers.

⇑ reduce localization accuracy.

② small patches allow network to see only little content.

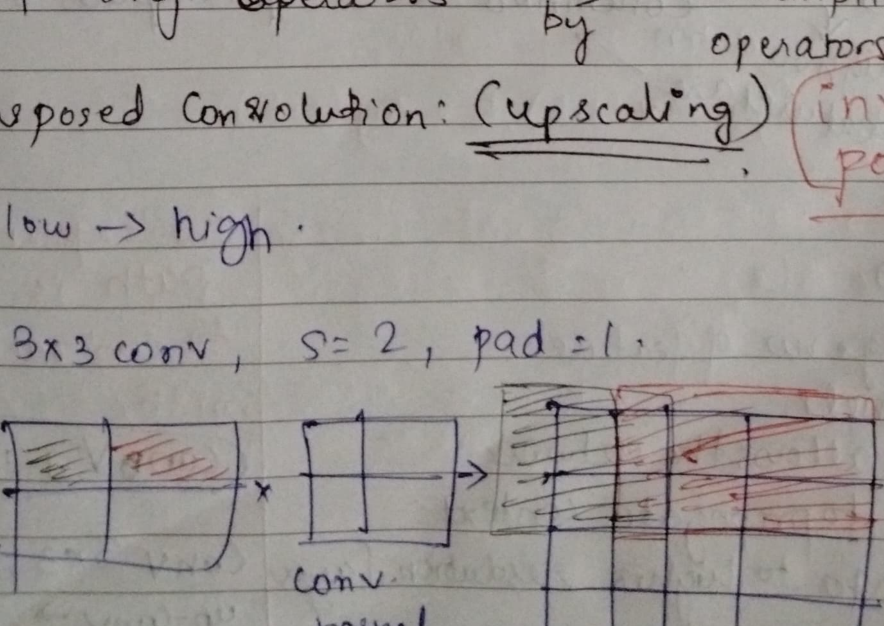
unet architecture,

pooling operators $\xrightarrow[\text{by}]{\text{replaced}}$ unsampling operators.

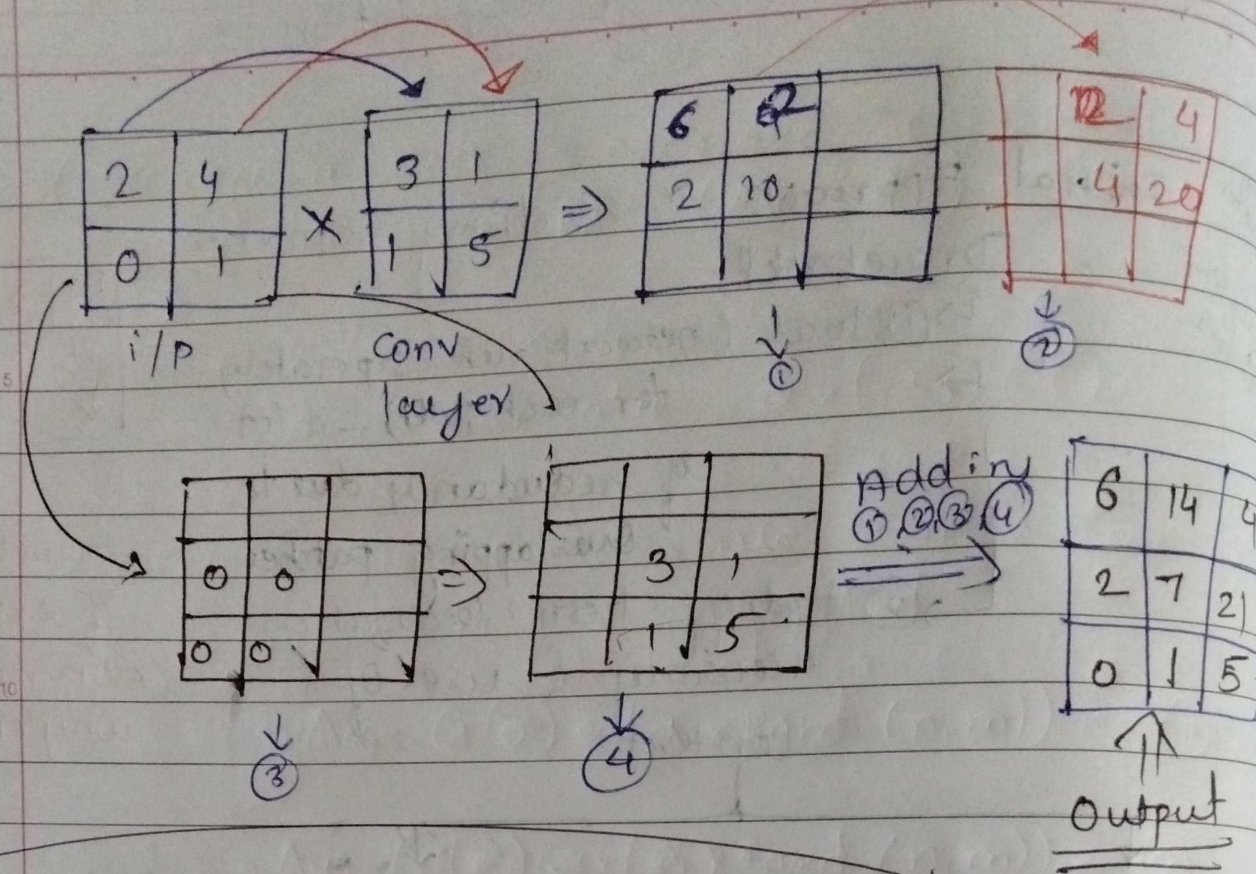
Transposed Convolution: (upscaling) inverse of pooling

low -> high.

3x3 conv, s = 2, pad = 1.



conv. kernel



Simpleupsampling ⇒ max unpooling

—— x —— x —— x ——

⇒ ~~To~~ Add upscaling operations to to contracting network, to help increase the resolution of the O/P.

⇒ To localize.
↳ high resolution features are combined with upsampled o/ps

modification in upsampling
↳ large no of feature channels
↳ allows the network to to propogate context into to higher resolution layer.

The expanison path is similar to contraction with:

Conv + upConv

Conv 3x3, ReLU.
up-Conv → 2x2.

⇒ No fully connected layer. & only uses valid conv.

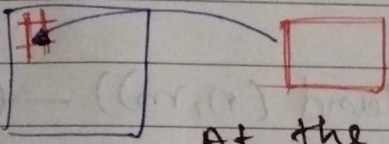
segmentation map only contains the pixels, for which full content is available in i/p img.

overlap tile strategy

extracting patches from large images and masks for segmentation.

Due to unpaded conv → o/p size ↓ than i/p size.

The whole image is predicted part by part



At the image boundary the image is ~~extrapolated~~ ~~thing~~ by mirroring.

extending by infering unknown values from trends in the known data.

↳ forming a mirror image.
↳ closest point
↳ wraping

similar to predicting the stock movement from recent movements

elastic Deformation for Data Augmentation

⇒ deforming the images → giving them elastic effects during data Augmentation.

⇒ Also used in handwritten digit recog.

deformation on Image

① random stress field is generated

② stress generated is applied to images.

In a cont body, deformation results from a stress field induced by applied forces. If the deformation recovers it is called elastic.

⇒ ① random stress is generated in horizontal & verticle directions. (Δx & Δy)

⇒ for each pixel & direction.
↳ random value $\alpha \times (-0.5, 0.5)$

⇒ to ensure close pixels have similar displacement
↳ gaussian filters are applied

$$\Delta x = G(\sigma) * (\alpha \times Rand(n, m)) \quad -① $$

$$\Delta y = G(\sigma) \times (\alpha \times Rand(n, m)) \quad -② $$

α → max^m^ value for random initial ~~initialization~~ disp.
σ ⇒ strength of smoothing factor.
↳ S.D of Gaussian filter (σ)

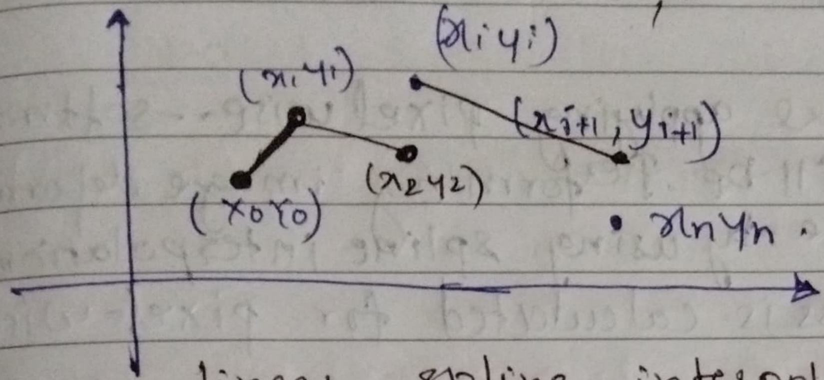
⇒ ② Stress field is applied to the image.
↳ done by moving each pixel to a new position & using spline interpolation of order one.
↳ to obtain intensities.

$$I_{trans}(j + \Delta x(j,k), k + \Delta y(j,k)) = I(j,k) \quad -③ $$

↳ I → original Image.
I_{tran} → transformed image.

→ Given $(x_0, y_0) \ldots\ldots, (x_n, y_n)$ → do interpolation of data to linear spline.

$x_0 < x_1 < x_2, \ldots\ldots < x_n$ (~~ascending or descending form~~) (ascending pattern)



linear spline interpolation draw straight lines betn two consecutive points.

$$f_i(x) = y_i + \frac{y_{i+1} - y_i}{x_{i+1} - x_i}(x - x_i), \quad x_i \le x \le x_{i+1}$$

Spline Interpolation

↳ uses low degree polynomials in each interval & chooses the polynomial piece such that they fit smoothly together.

Seperation of touching objects of same classes.

↳ use of weighted loss.

↳ seperating background labels betn touching cells obtain a large weight in the loss func.

pixel-wise softmax over the

feature map combined with cross entropy loss func.

↳ normally softmax predicts lossfunc for whole result.

↳ but we are applying pixel wise - softmax as we will be performing image deformation pixel-wise by using splide interpolation

↳ so the loss is calculated for pixel-wise prediction.

Softmax ⇒ $p_k(x) = \dfrac{\exp(a_k(x))}{\left(\sum_{k'=1}^{K} \exp(a_{k'}(x))\right)}$

$a_k(x)$ → activation in feature channel k at pixel position $x \in \Omega$ with $\Omega \subset \mathbb{Z}^2$

K → no. of classes,

$p_k(x)$ → approximated maxm func. ≈ 1

$$p_k(x) \begin{cases} \approx 1 & \text{for } k \text{ with maxm activation } a_k(x) \\ \approx 0 & \text{for all other } k. \end{cases}$$

Cross entropy penealised deviation of $p_{l(x)}(x)$ from 1 using

$$E = \sum_{x \in \Omega} w(x) \log\left(p_{l(x)}(x)\right)$$

$l : \Omega \to \{1, \dots, K\}$ → true label of each pixel.

$w : \Omega \to \mathbb{R}$ → weighted map, we have introduced to give some pixels more importance

⇒ Pre-compute the weight map. to compensate the diff freq. of pixels ~~at~~ ~~to~~ from certain class.

& force network to learn small seperation borders betn the touching cells.

↳ computed using morphological operations.

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

$w_c \Rightarrow \mathbb{R}$ is the weight map to balance freq.

$d_1 \to \mathbb{R}$ denotes distance to the border of the nearest cell.

$d_2 \to \mathbb{R}$ distance to border of second nearest cell.

Weight initialization

↳ drawbacks.

↳ give exessive activations.

↳ some parts might not contribute.

↳ should be initialised in such a way the feature maps in the network have unit variance.

In our case.

initial weights → Gaussian Dist.

↳ $S.D = \sqrt{2/N}$ $N \to$ no of incom nodes, of one neuron

loss function in binary case.

Probability of class 1 → $\sigma(x)$ of o/p x.
—"— —"— 0 → 1 → $-\sigma(x)$

$$-\sum_i (l_i \log \sigma(x) + (1-l_i) \log (1-\sigma(x)))$$

⇒ Sum over all samples, l_i → one of two values is zero.

⇒ G.D forces the model to predict true label with more conf.

Data Augmentation.

↳ shift & rotation invariance.
↳ robust to deformation.
↳ gray value variations

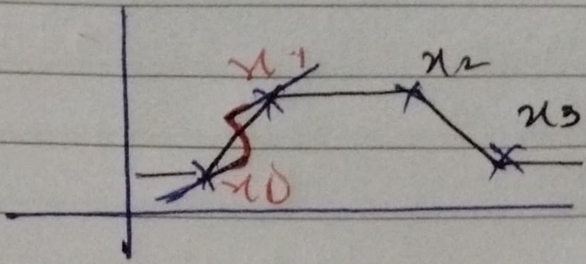
Per-pixel displacement is calculated using bicubic interpolation

Bicubic Interpolation

↳ performed by
↳ Lagrange polynomial.
↳ Cubic splines

spline - small section of curve betn two end points.

Does ~~joing~~ joining points more smoothly.



— ← linear interpolation

— ← gradient of point

S → cubic interpolation curve

$$\boxed{y = ax^3 + bx^2 + cx + d}$$

find a, b, c, d.

↳ based on the positions of x & derivatives of x.

the derivatives of gradient of the x_0 & x_1 are going to depend on the position of next two pixels.

bi-linear → considers 4 pixels around each pixel value.

bi-cubic → 16 pixels around each pixel value

↳ use interpolation in x & y-axis.

— x ——— x Conclude ——— x ———