EECS 182/282A

Today: Finish CNN
Graph Neural Nets

Reading: Prince through Ch 11
start Ch 13

Discussion Tomorrow
Fri TA OH 531 Cory 2-3PM

Architecture Order In Class:

MLPs → CNNs → Graph NN → RNN/state-space → Transformers

Goal

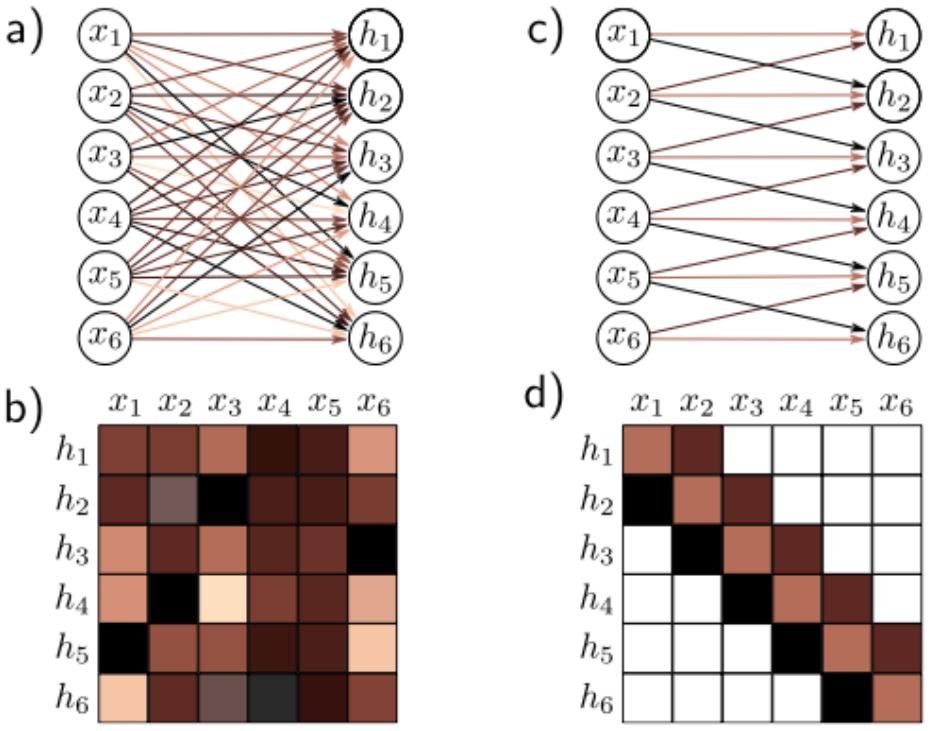
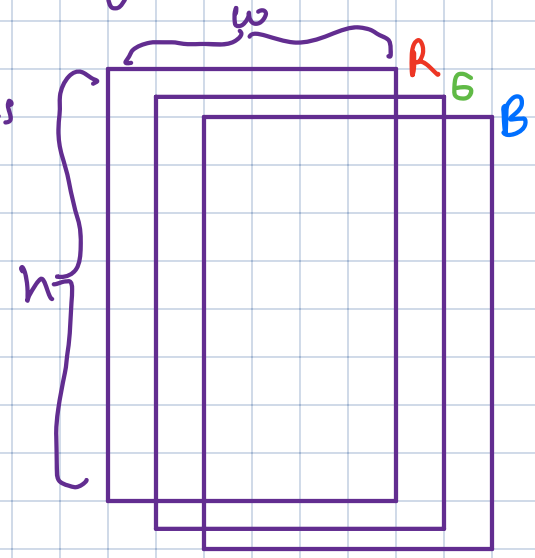


Fig 10.4 in Prince

MLP CNN

Inspired by computer vision problems: classification,



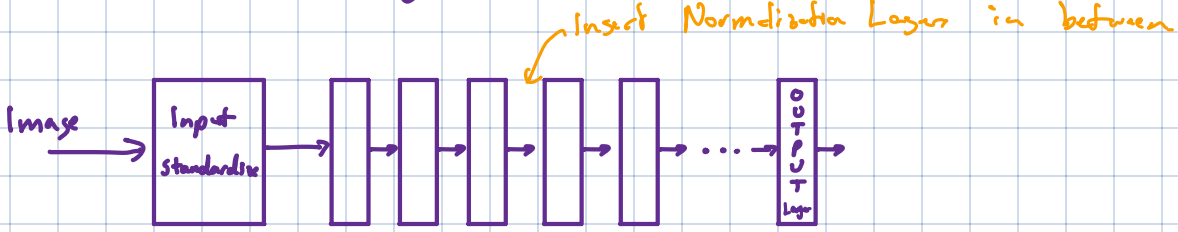
$h \times w \times 3$

channels c

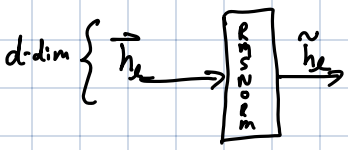
semantic segmentation, etc.

Training: Stability and Effectiveness

Normalization Layers



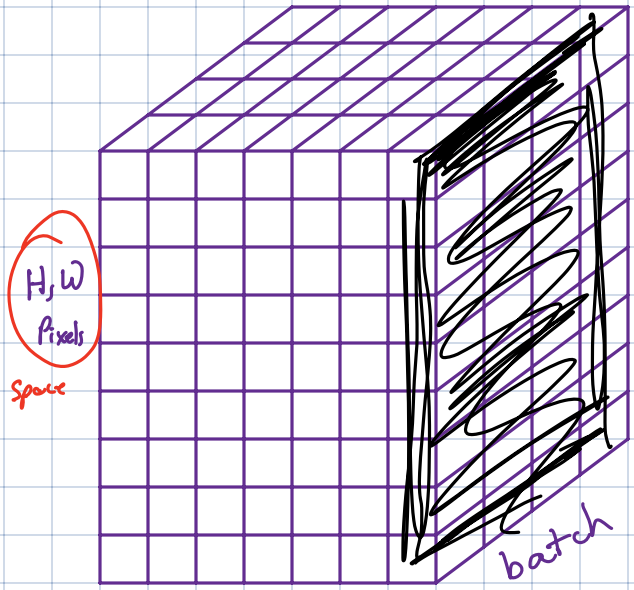
Idea: RMS Norm Layer



has $\|\tilde{h}_\ell\|_{RMS} = 1$

$$\tilde{h}_\ell = \frac{\vec{h}_\ell}{s} \quad \text{where } s = \sqrt{\frac{1}{d}\sum_i h_\ell[i]^2 + \epsilon}$$

$$= \sqrt{d}\,\frac{\vec{h}_\ell}{\|\vec{h}_\ell\| + \epsilon} = \frac{\vec{h}_\ell}{\|\vec{h}_\ell\|_{RMS} + \epsilon}$$



BatchNorm

Average over space & batch.

Let $\mathcal{B}$ be what is averaged over

$$m = \frac{1}{|\mathcal{B}|}\sum_{i\in\mathcal{B}} h_{in}$$

$$s = \sqrt{\frac{1}{|\mathcal{B}|}\sum_{i\in\mathcal{B}} (h_{in} - m)^2}$$

$$h_{out}[i,j] = \gamma\left(\frac{h_{in}[i,j] - m}{s + \epsilon}\right) + \delta$$

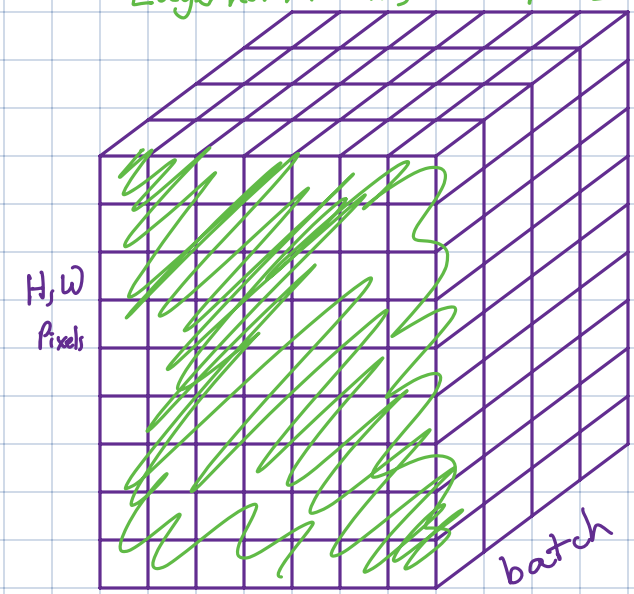
(γ, δ: learnable)

Learnable center δ, scale γ

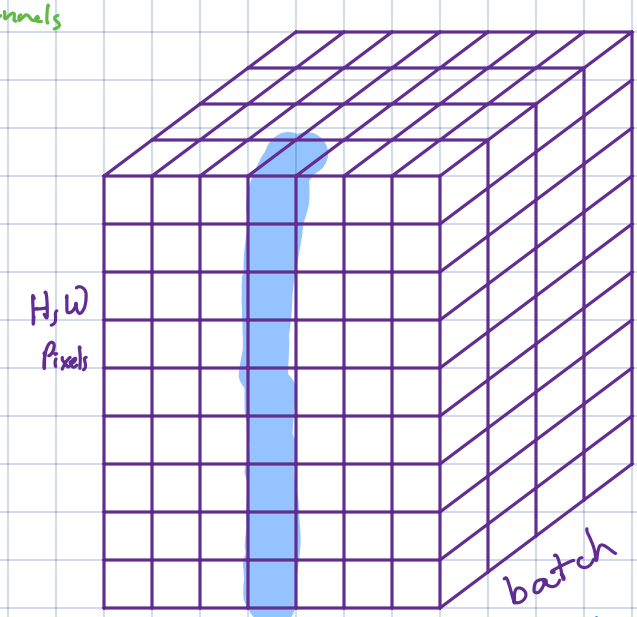
Initialize to $\delta = 0$, $\gamma = 1$

Typically: Don't use weight decay on δ, γ.

Layernorm: Avg over space & channels



In Layernorm can have different γ and δ per channel.



InstanceNorm: Avg across space only

Residual / Skip-connections

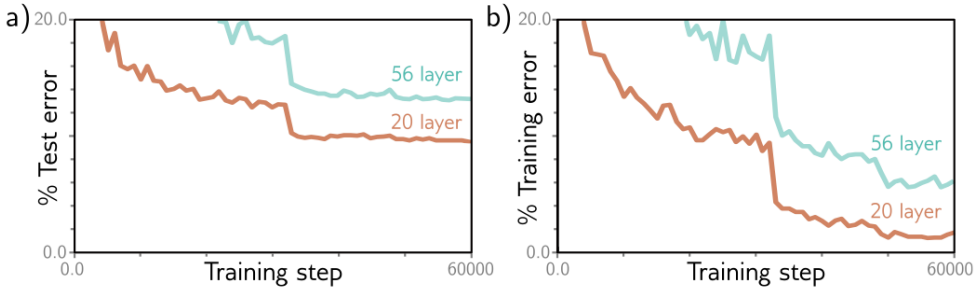
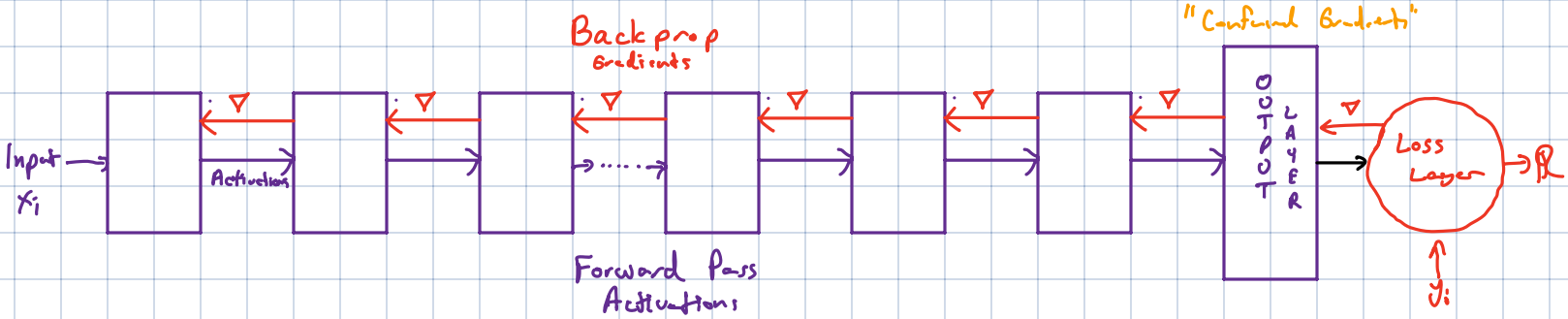


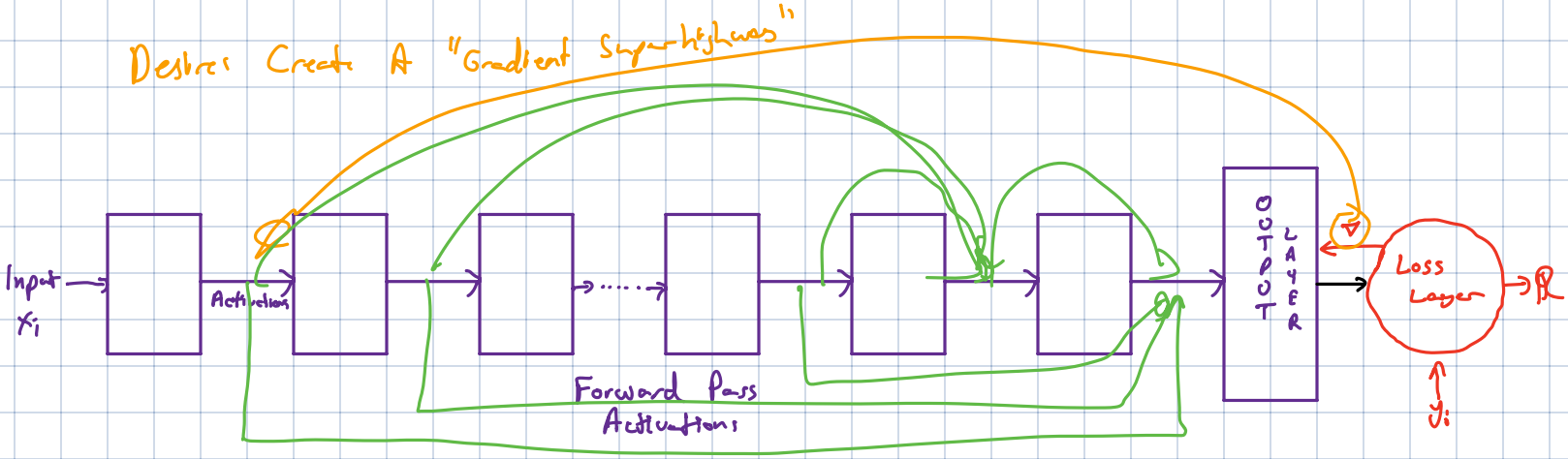
Fig 11.2 in Prince

← Feels like Underfitting.

Problem could be Dying Gradients
"Confused Gradients"



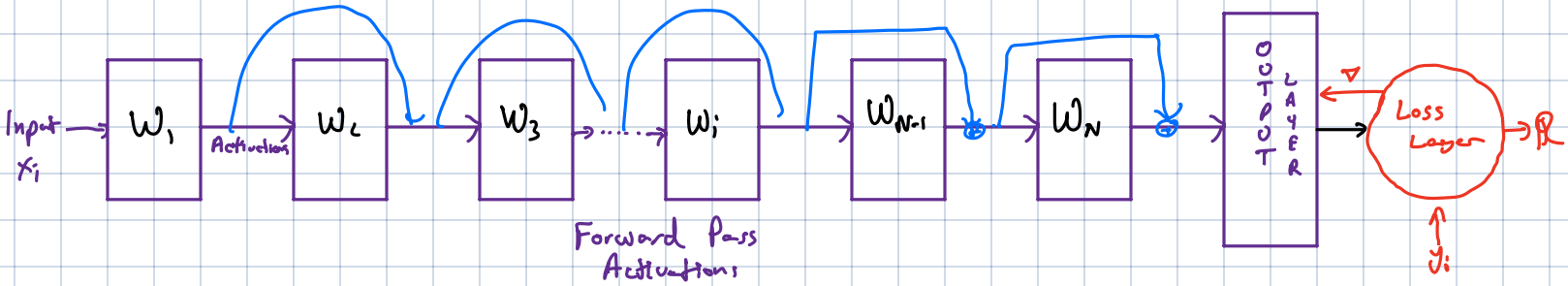
Desire: Create A "Gradient Superhighway"



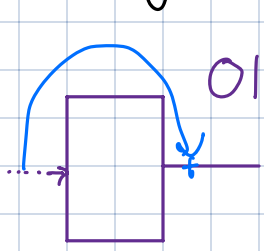
Idea 1: Add links from outputs of all earlier layers to the input of a given layer.

Blows up activations ☹

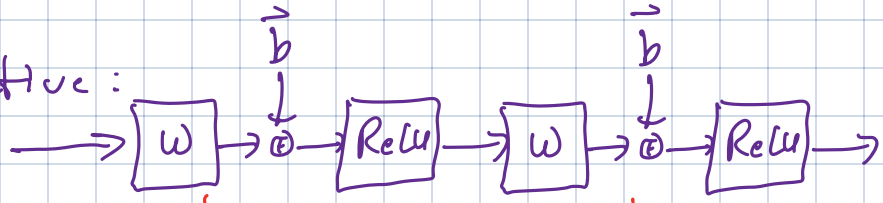
Skip Connections: Add input to the output of a layer



What goes inside a block that's skipped over?



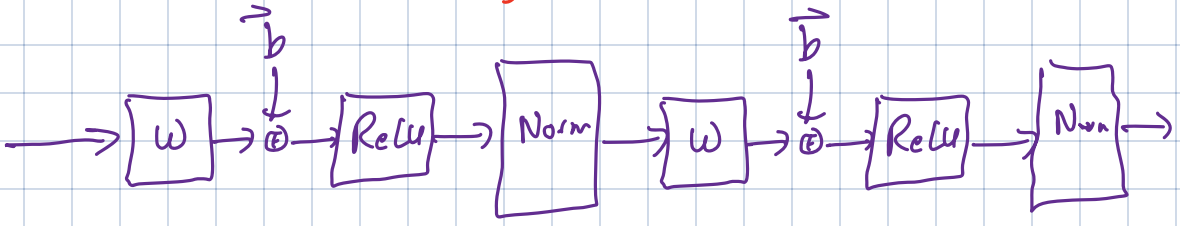
Old Perspective:



conv layer or MLP-style

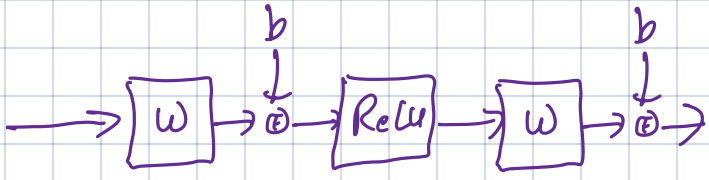
conv layer or MLP style

or



What matters most?

CORE



INTUITION: Pre-ResNet Perspective: $f_1(f_2(f_3(\cdots f_L(x))\cdots)$

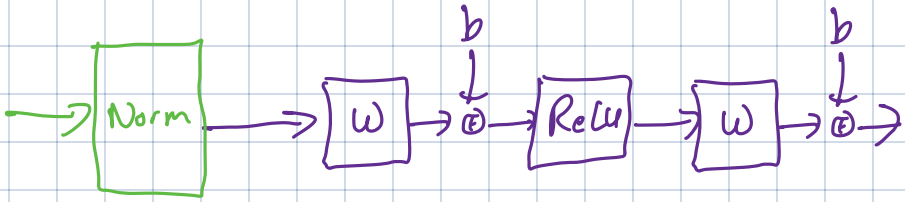
ResNet Perspective $f_{i+1}(x) = f_i(x) + g_i(f_i(x))$

Feels like $\frac{d}{dt}x(t) = g(x(t))$

Where should normalizations go?

Modern Default: Inside the block near the start

e.g.



Called "Pre-Norm" convention

Associated with more stable training

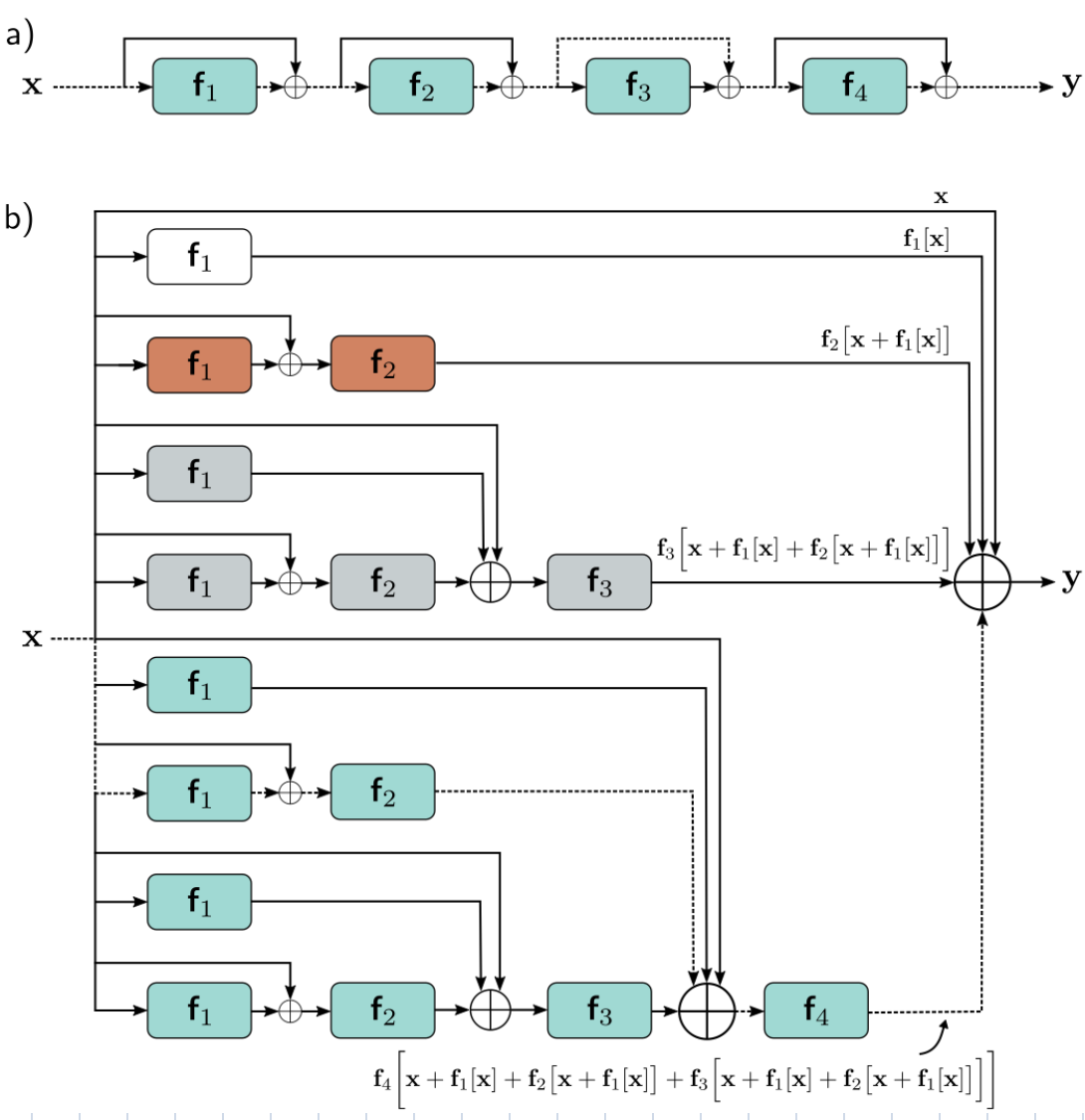


Fig 11.4 in Prince

One challenge...

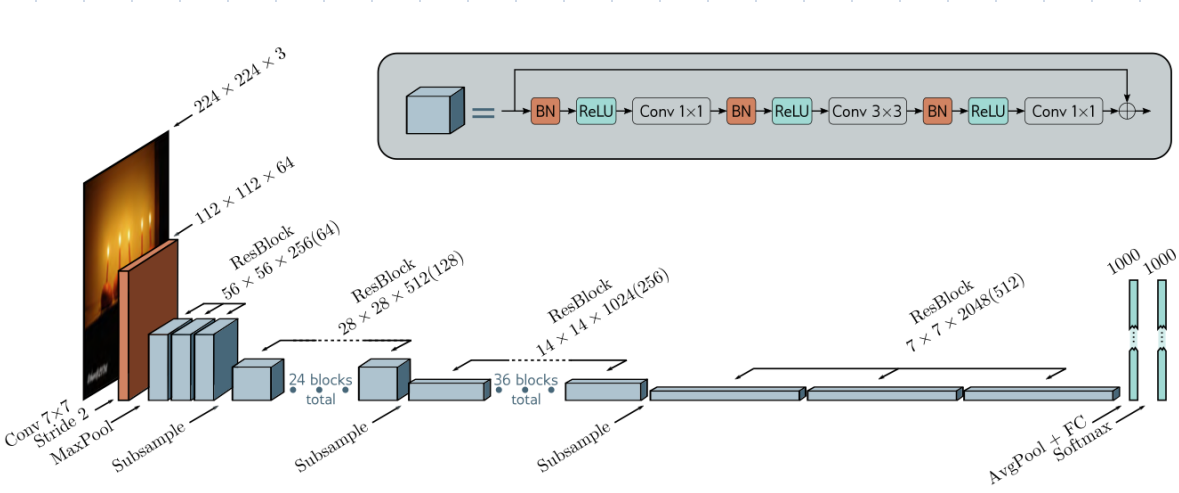


Fig 11.8 in Prince

ResNet-200

What do we do for subsampling? (e.g. a block that has a maxpool, stride, etc) along with other stuff

How to skip?? Need something that changes shape.

e.g. Linear Layer of some kind. (Matrix with right shape)

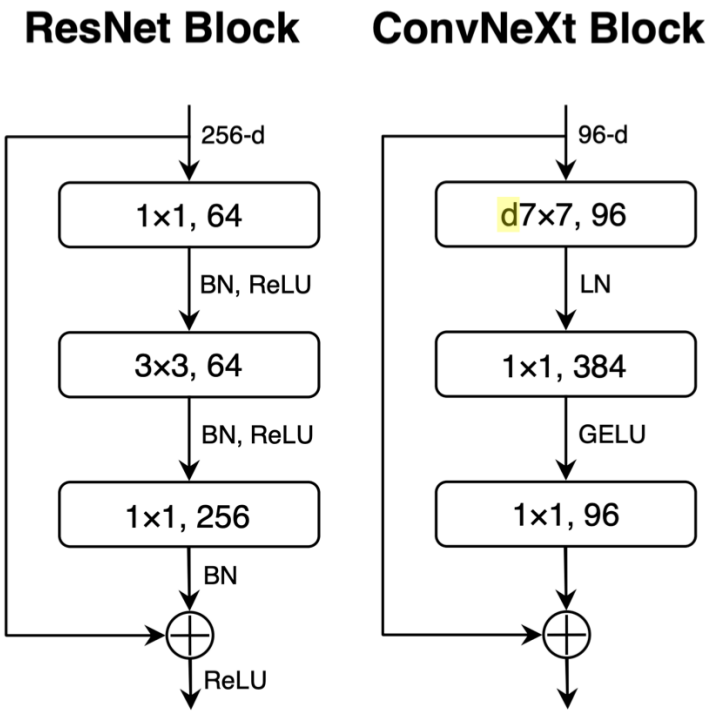
P

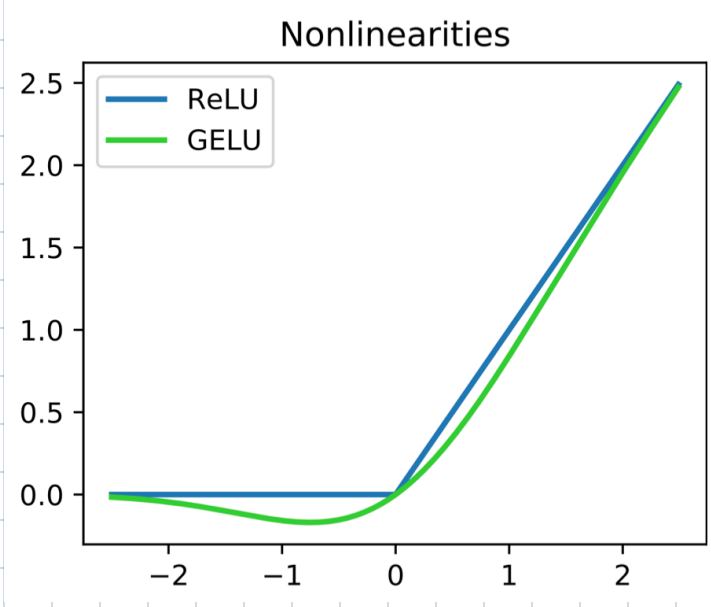
Note: If the entire block is just a maxpool downsampling or a subsampling, then that particular block doesn't need a skip connection because gradients pass through cleanly.

↖ Added in Office Hours.

A more modern perspective:

"A ConvNet for the 2020s"
by Liu, et. al. in 2022
(Meta FAIR & UC Berkeley)

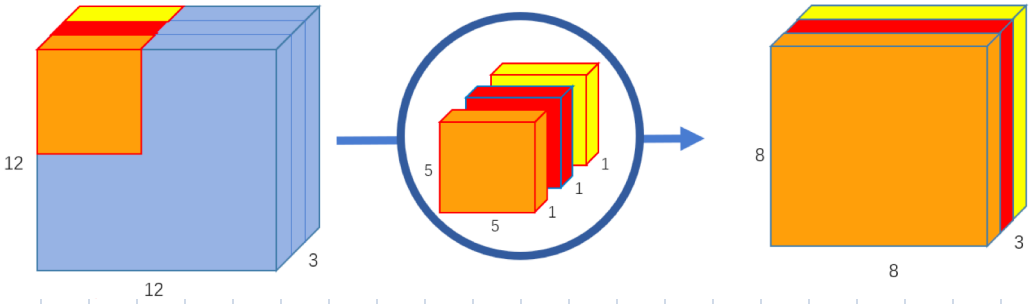




Pic from Wikipedia

$x\Phi(x)$

Depthwise – convolution

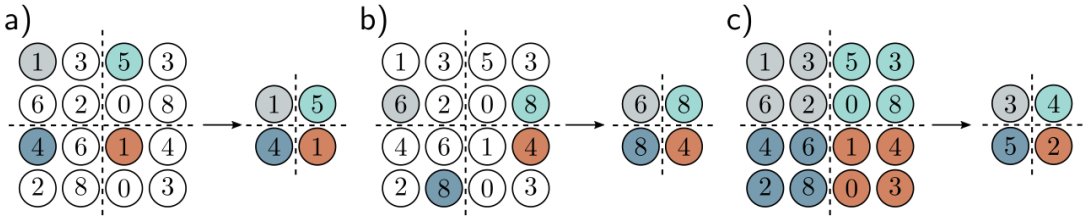


Just treat each channel separately as though it was a one-input-channel & one-output-channel conv.

No weight-sharing across channels.

Recall Pooling (Downsampling)

Fig 16.11 in Prince



stride/subsample

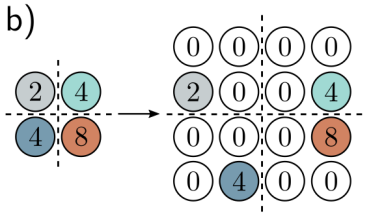
Max-Pool

$$\text{Output} = \max_{i \text{ is pooling domain}} h_i$$

Mean-Pool

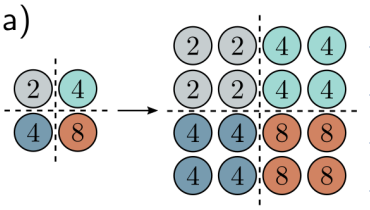
$$\text{Output} = \frac{1}{|\text{pooling domain}|} \sum_{i \text{ is pooling domain}} h_i$$

What if we wanted the reverse? Go from low-res to high res? Upsampling

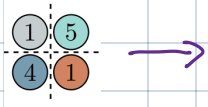


Max-unpool

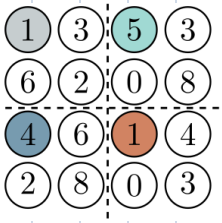
Must remember which position was the max.

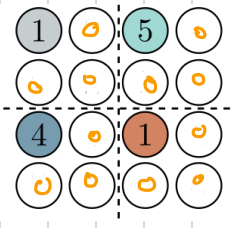


Duplication — like opposite of mean



What is counterpart upsampling for sub-sampling?



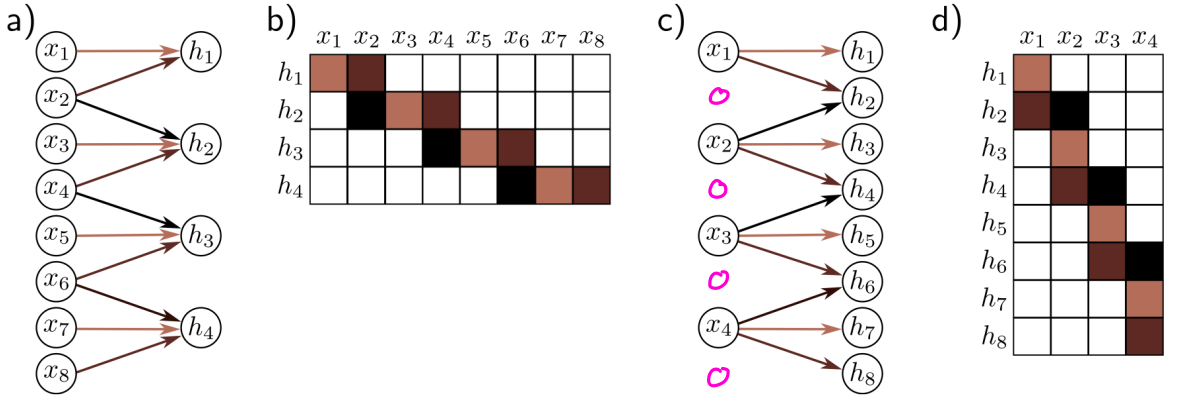


Ideas: a) Interpolate in a smooth way
b) Randomly sample under a distribution.
c) Make it learnable (version of a)
d) Fill with zeros

Idea: Use (d) Followed by a learnable conv.

Behaves like learnable interpolation

Transpose Convolution: Downsample has: Conv + Subsample ⇒ Strided Conv
Upsample has: Zero-Fill Upsample + Conv ⇒ Transpose Conv



Strided Conv

Transpose Conv

3×3 example:

w_{NW}	w_N	w_{NE}
w_W	w_{mc}	w_E
w_{SW}	w_S	w_{SE}

$\overbrace{\quad}^{c_{in}}$

c_{out} { W_N } $\vec{b}$ } c_{out}

$$\vec{h}_{out} = \vec{b} + \sum_i W_i \vec{h}_{in,i}$$

Standard Conv

i ranges over the $k \times k$ positions in the conv filter.

Transpose Conv: i ranges only over those positions that are neighbors given stride

The code perspective:

Standard conv

PYTORCH | MXNET | JAX | TENSORFLOW

From D2L.ai Ch 7.2

```python
def corr2d(X, K):  #@save
    """Compute 2D cross-correlation."""
    h, w = K.shape
    Y = torch.zeros((X.shape[0] - h + 1, X.shape[1] - w + 1))
    for i in range(Y.shape[0]):
        for j in range(Y.shape[1]):
            Y[i, j] = (X[i:i + h, j:j + w] * K).sum()
    return Y
```

Transpose conv

PYTORCH | MXNET

Ch 14.10

```python
def trans_conv(X, K):
    h, w = K.shape
    Y = torch.zeros((X.shape[0] + h - 1, X.shape[1] + w - 1))
    for i in range(X.shape[0]):
        for j in range(X.shape[1]):
            Y[i: i + h, j: j + w] += X[i, j] * K
    return Y
```

How to do image-level tasks?

Semantic segmentation

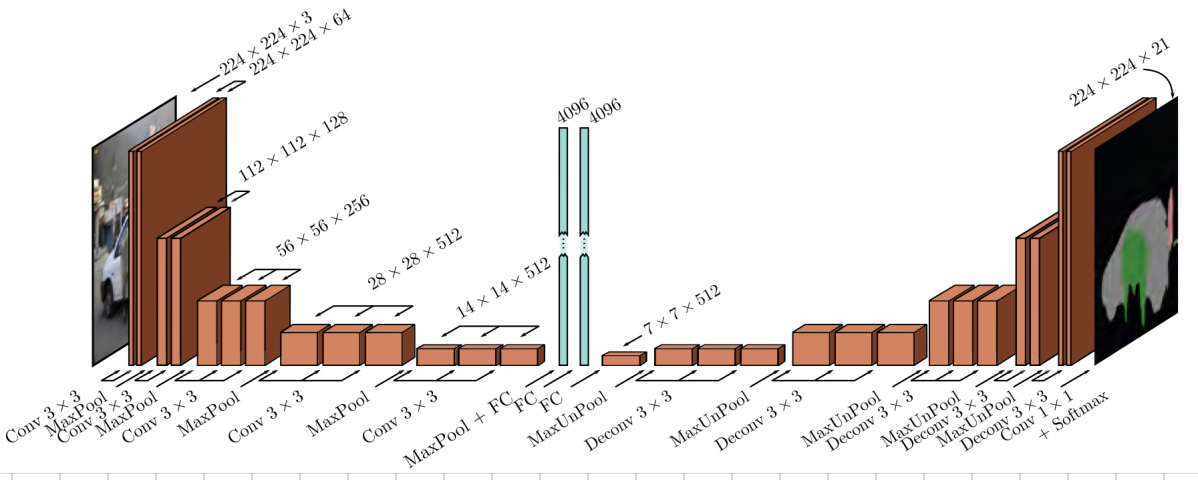


Fig 10.19 in Prince

Transpose-Convs / Upsampling kind of works, but there's a tension between getting details and big picture. Later layers doing upscaling have no local context.

U-nets

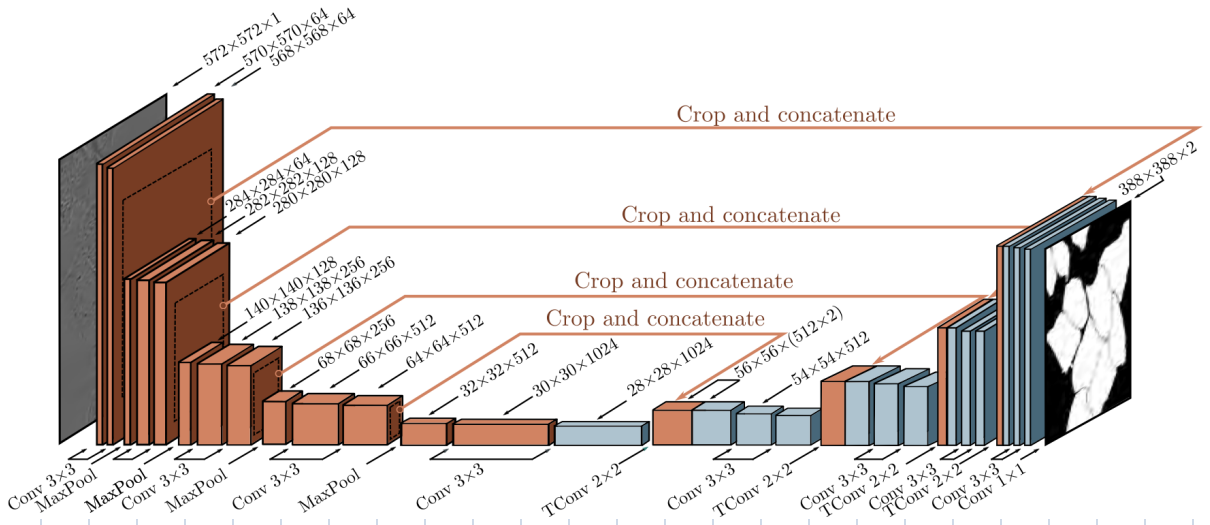


Fig 11.10 in Prince

Brings information for this pixel from earlier processing.

Like skip connections vis-a-vis gradient flow, but do not replace the block-by-block residual connections.