EECS 182/282A

Today: convolutional neural netw (continued)

Reading: Prince through Ch 9 (so far)
Ch 10 & 11 now.

Reminder: TA OH Today
Discussion Tomorrow

Architecture Order In Class:

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

we are here

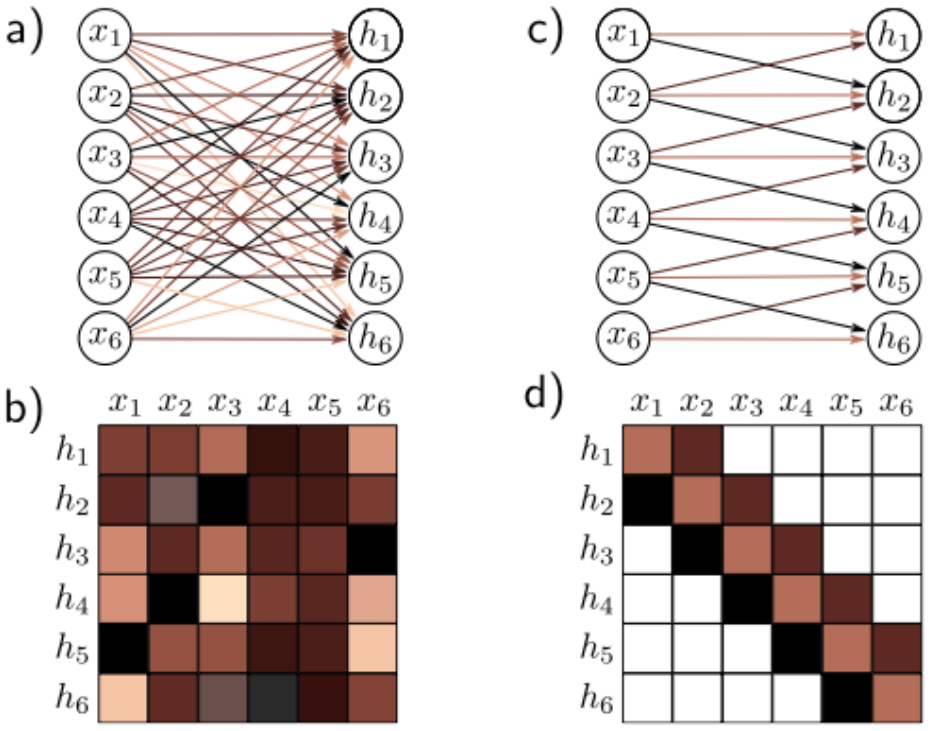
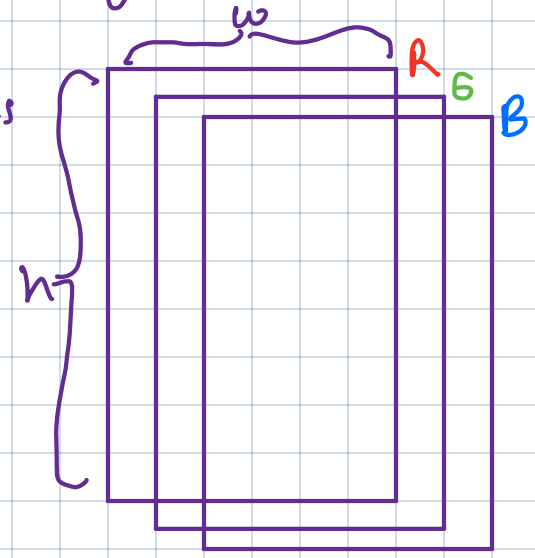


Fig 10.4 in Prince

MLP CNN

Inspired by computer vision problems: classification,



Images

$h \times w \times 3$

channels c

semantic segmentation, etc.

Key Ideas (Expressivity)

1) Respect locality → convolutional structure with small filters.
2) Respect certain invariances / equivariances / symmetries → weight sharing
   → data augmentations (training)
3) Hierarchical Structure (Parts make wholes) → Depth
   → Multiresolution & Filterbanks

Key Ideas (Getting it to work)

1) Normalization Layers
2) Dropout
3) Residual / Skip Connections

Recall 1D convolution (LTI systems ... e.g. from EECS 120, 123)

$$y(t) = \int_{-\infty}^{+\infty} x(\tau)\, h(t-\tau)\, d\tau = \int_{-\infty}^{+\infty} h(\tau)\, x(t-\tau)\, d\tau$$

$$y[t] = \sum_{\tau=-\infty}^{+\infty} x[\tau]\, h[t-\tau] = \sum_{\tau=-\infty}^{+\infty} h[\tau]\, x[t-\tau] \xrightarrow{\text{FIR}} \sum_{\tau=-k}^{+k} h[\tau]\, x[t-\tau]$$

In Deep Learning, we don't "flip"

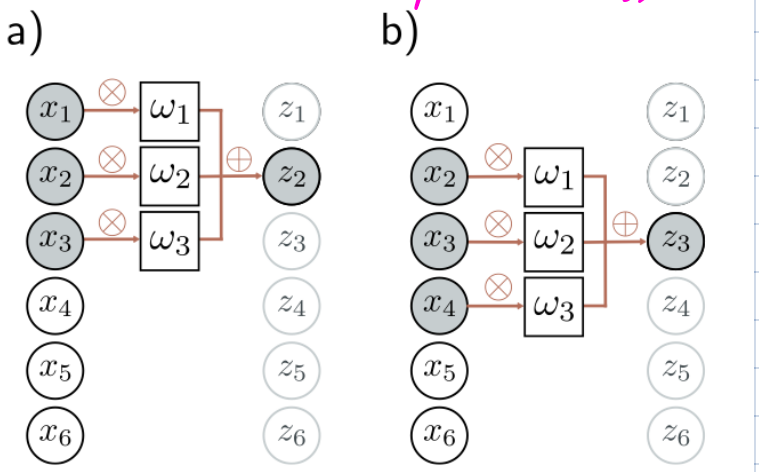


Figure 10.2 in Prince.

Notice: No flip

Can also have a bias

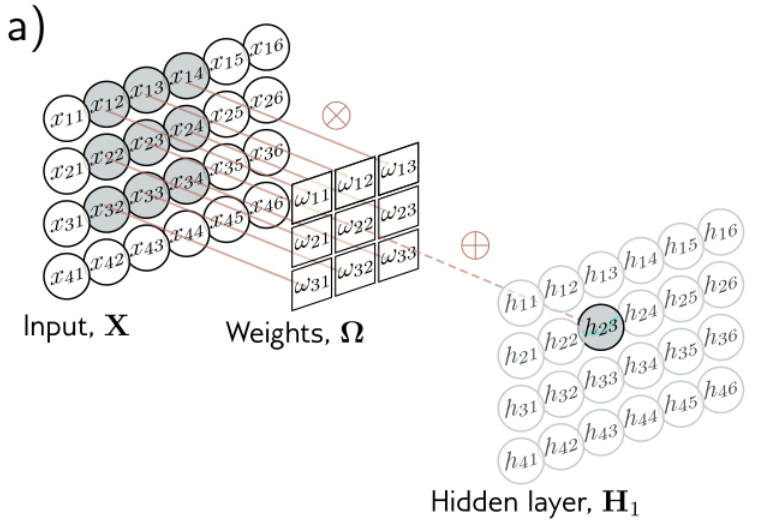


Figure 10.9 in Prince

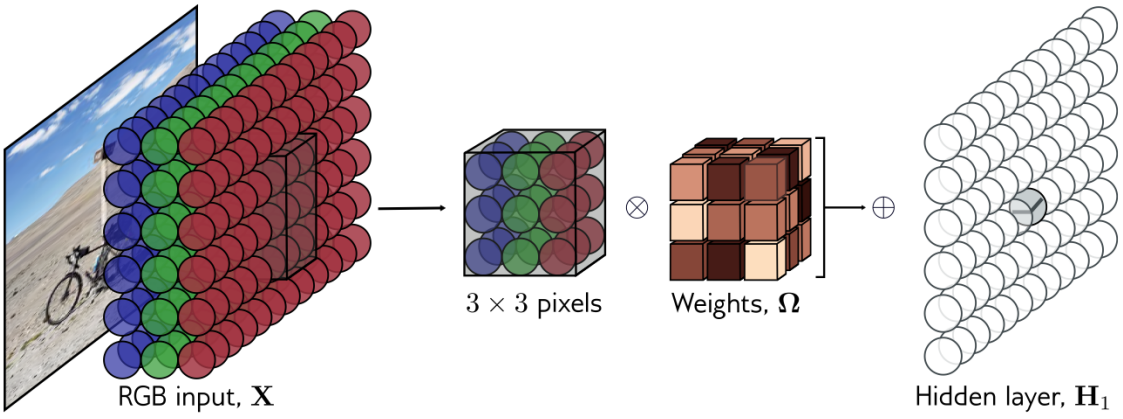


Figure 10.10 in Prince

Note: weights NOT shared across channels.

New details

Notation note: convolutional filters called "kernels"

Examples:

1x1 conv 1 channel input 1 channel output

For every pixel, scale by a constant and add a bias

learnable weights

How many weights: 1

" " biases: 1

3x3 conv with 1 channel input 1 channel output

For every pixel, scale the value at this pixel and immediate neighbors by individual weights, add them up, and add bias.

How many weights: 9

" " biases: 1

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

Filter

3x3 conv with c_{in}-channel input and 1 output

Repeat for each channel (with distinct individual weights per channel) and fold together by adding up.

How many weights: $9 c_{in}$

" " biases: 1

kxk conv with c_{in}-channel input and 1 output

How many weights: $k^2 c_{in}$

" " biases: 1

1×1 conv with C_{in}-channel input and C_{out} outputs

Input (from one pixel) is a vector $\vec{x}\}C_{in}$

Output (for this pixel) is a vector $\vec{y}\}C_{out}$

$$\vec{y} = \text{conv}\{W\vec{x} + \vec{b}\}$$ (W: C_{in})

How many weights: $C_{in}C_{out}$

" " biases: C_{out}

k×k conv with C_{in}-channel input and C_{out} outputs

How many weights: $k^2 C_{in} C_{out}$

" " biases: C_{out}

Backprop?? We share parameter across different pixels

How can we compute $\frac{\partial L}{\partial w}$ when the same parameter w appears in different places in computation graph?

Easy Answer: Don't worry, PyTorch will handle it. ☺

Underlying Answer: 1) Pretend the weight w was actually $\tilde{w}_1, \tilde{w}_2, \ldots$ in all the different places and just happened to be that $\tilde{w}_1 = \tilde{w}_2 = \tilde{w}_3 = \cdots = w$

2) Now to get $\frac{\partial L}{\partial w} = \sum_i \frac{\partial L}{\partial \tilde{w}_i}$

i.e. Add them up

Question: How does the scale of $\frac{\partial L}{\partial w}$ change with the number of terms in Σ?

Pooling

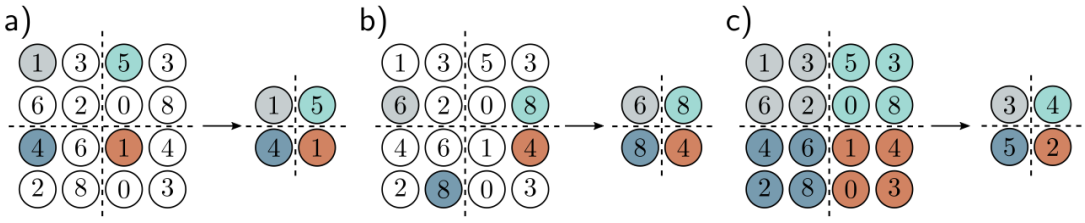


Fig 10.11 in Prince

stride / subsample

Max-Pool

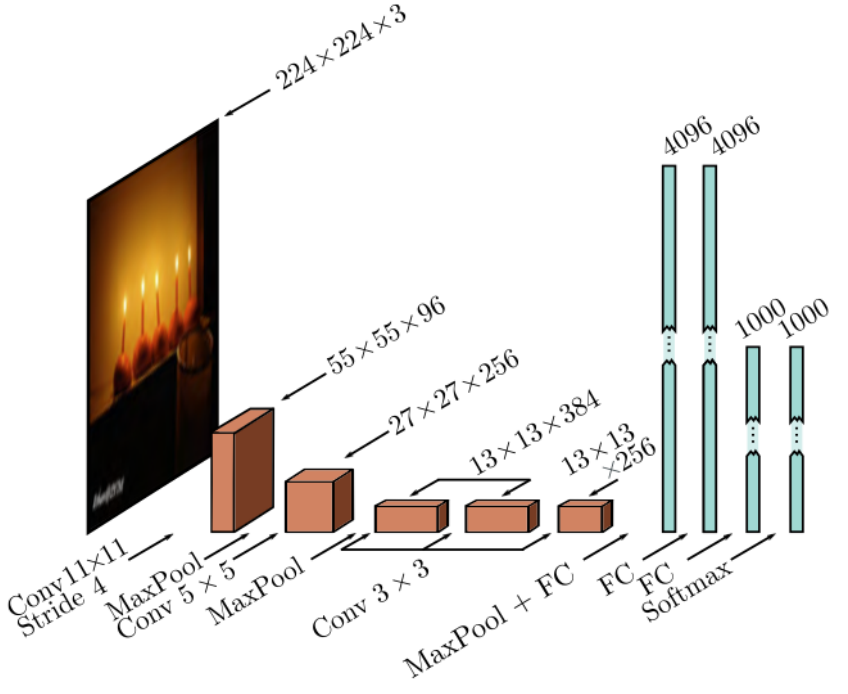
$$\text{Output} = \max_{i \text{ in pooling domain}} h_i$$

Mean-Pool

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

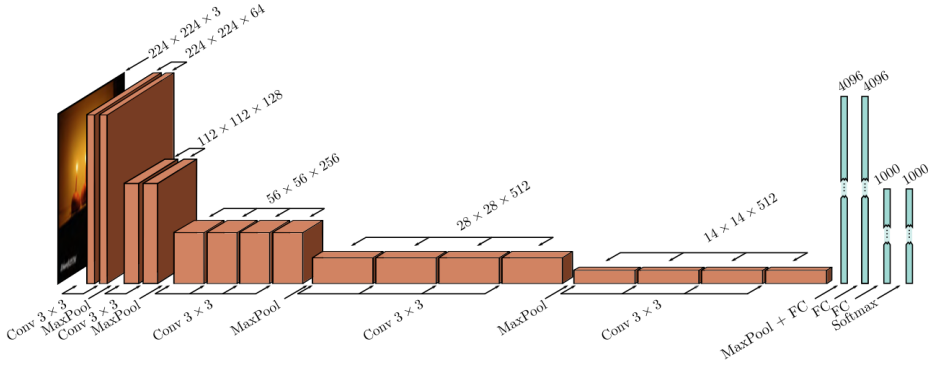
Backprop? → Follows argmax (Router Gradients) MaxPool

Putting things together



AlexNet (Krizhevsky et al 2012)

Fig 10.16 in Prince



VGG from 2014

Fig 10.17 in Prince

Data Augmentations

Basic:

```python
augmentations = [
    autocontrast, equalize, posterize, rotate, solarize, shear_x, shear_y,
    translate_x, translate_y
]

augmentations_all = [
    autocontrast, equalize, posterize, rotate, solarize, shear_x, shear_y,
    translate_x, translate_y, color, contrast, brightness, sharpness
]
```

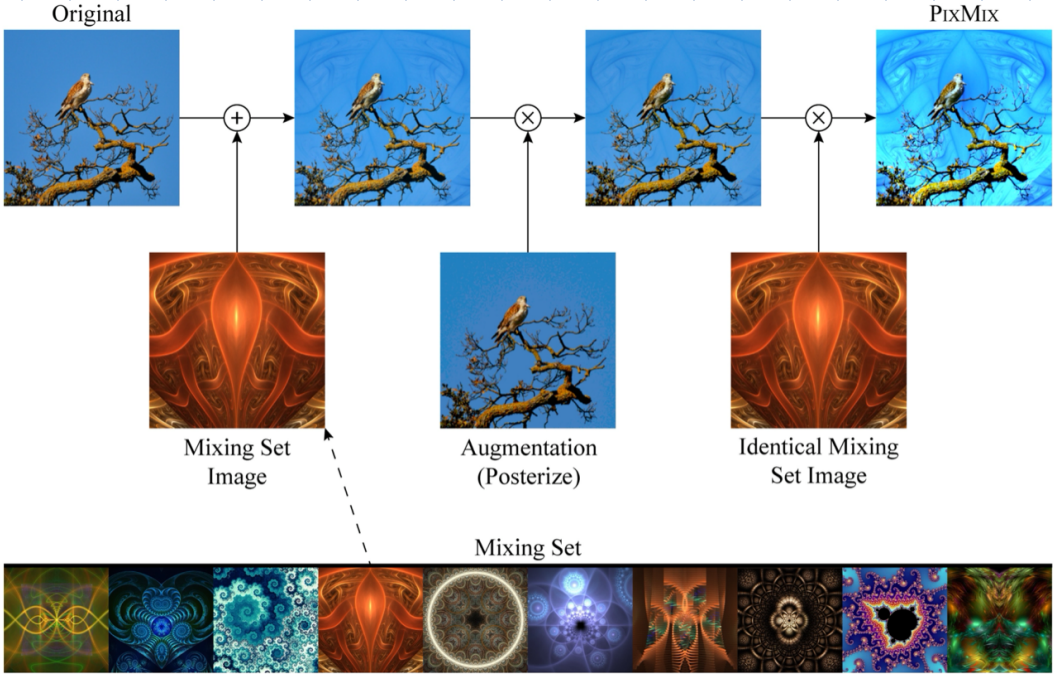
Code in PixMix Repo

Hendrycks, et. al. CVPR 2022

More aggressive: gin robustness

PixMix Paper

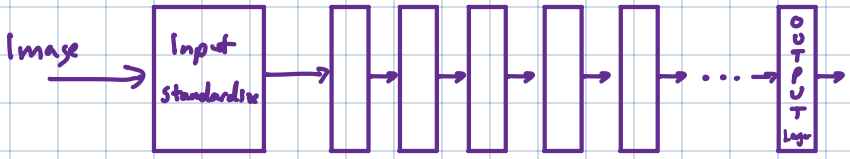
Psychedelic



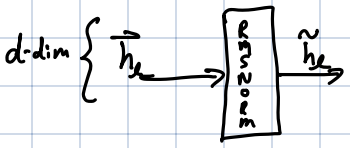
PixMix Paper

Training: Stability and Effectiveness

Normalization Layers



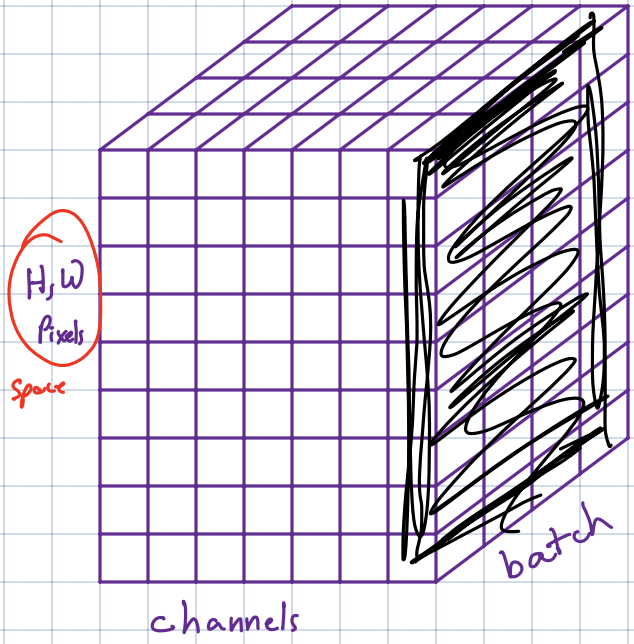
Idea: RMS Norm Layer



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

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

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



Batch Norm

Average over space & batch

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

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

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

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

(γ, δ: learnable)