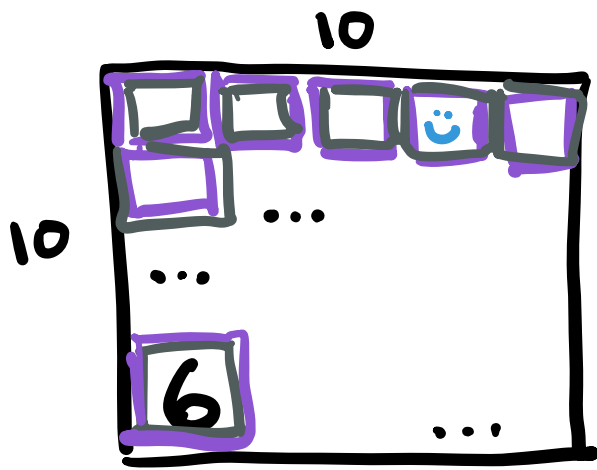


DS 4440 Conv Nets (2)



Two 2x2 filters

$$O_1 = [\dots \quad \text{[shaded box]} \quad \dots]$$

$$O_2 = [\dots \quad \text{[shaded box]} \quad \dots]$$

Pooling



Design decisions

- filter sizes
- padding
- # filters
- Stride

In practice: Deep models dominate.

Some history: LeNet (LeCun, '98)

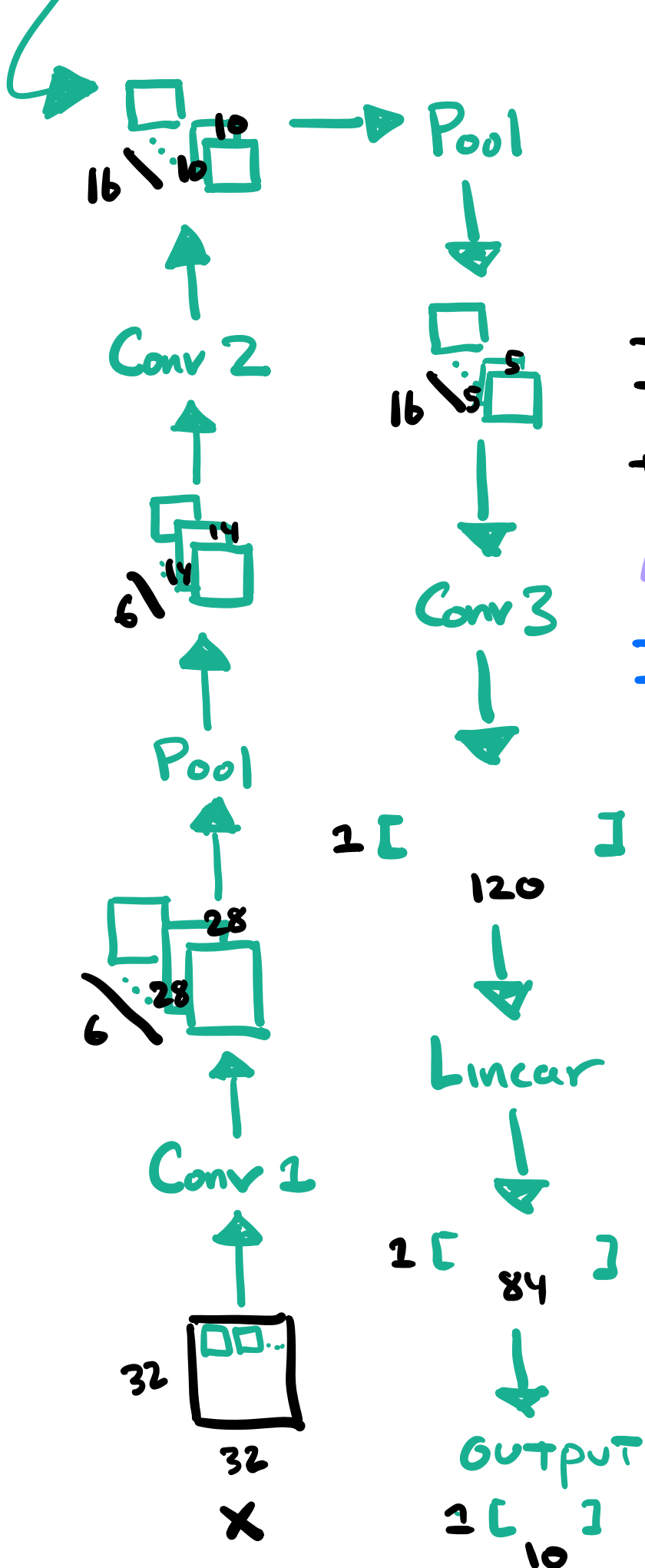
But hardware & data had to catch up.

LeNet (98!)

In 2012, The
time came —
AlexNet won The
ImageNet Challenge.

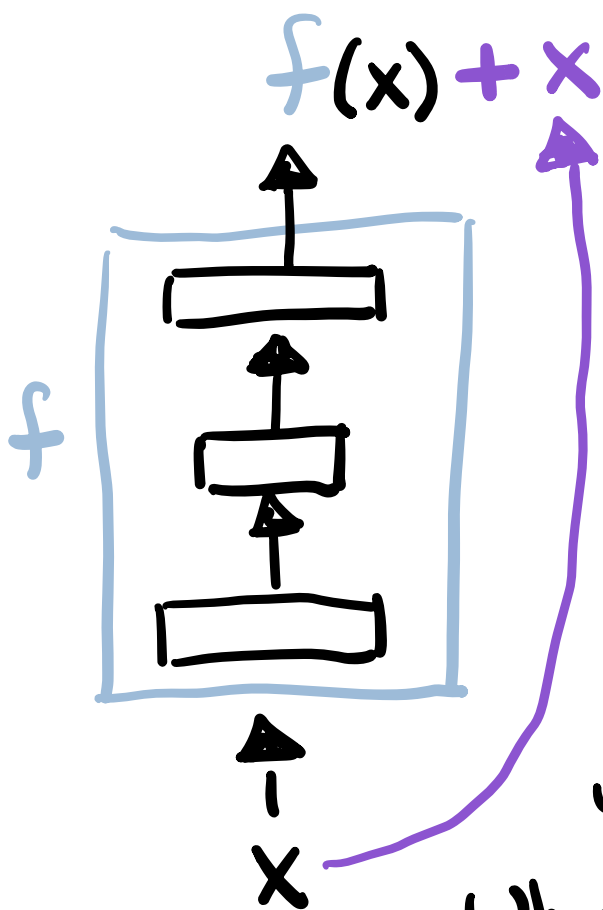
Was deeper and
used ReLU.

(See CoLab
Notebook.)



Residual Connections

Training gets tricky with deeper nets. Adding paths that skip layers can help.



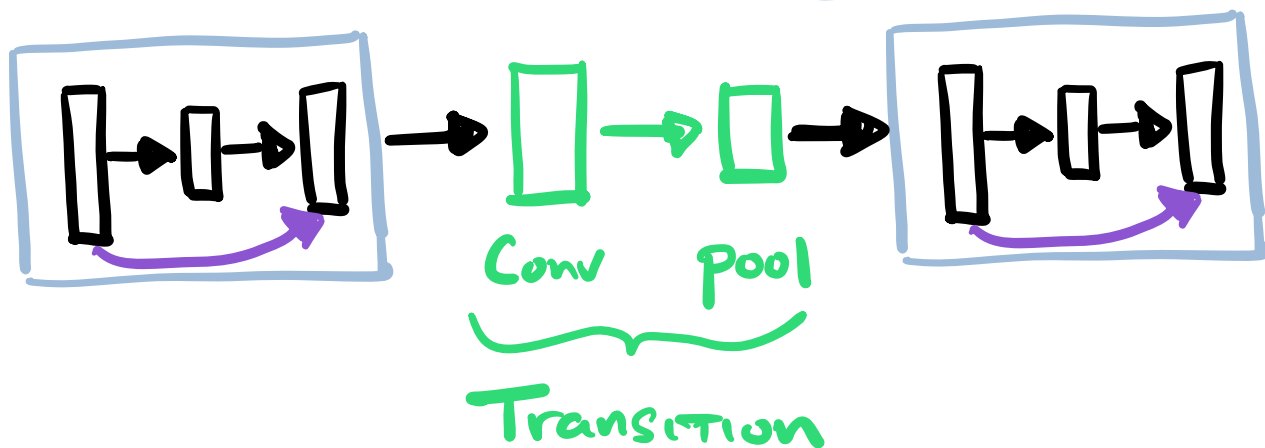
Can add or concat.
 $+$ $\oplus$

If we $\oplus$, dimensions may get out of hand.

DenseNet and similar

use transition layers

which downsample feature maps (e.g., via pooling)

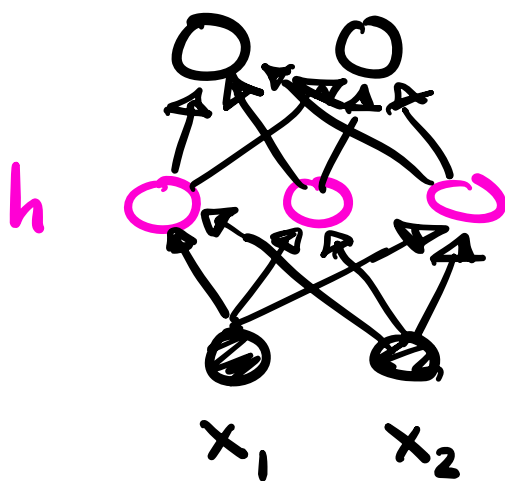


Dropout!

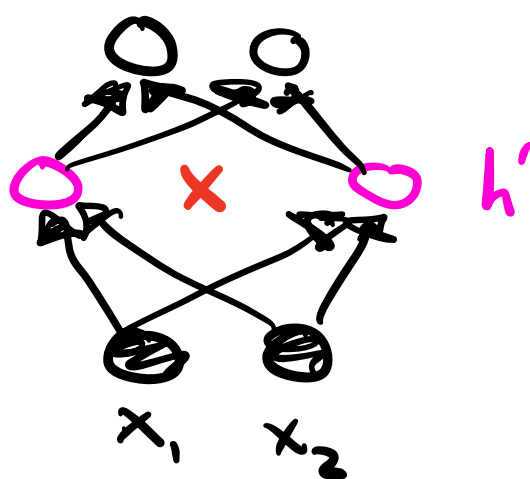
Another standard Trick used while Training (deep) nets:

Inject **noise** during Training as a regularization Technique.

$$h' = \begin{cases} 0 & \text{with prob. } p \\ \frac{h}{1-p} & \text{otherwise} \end{cases}$$



Standard

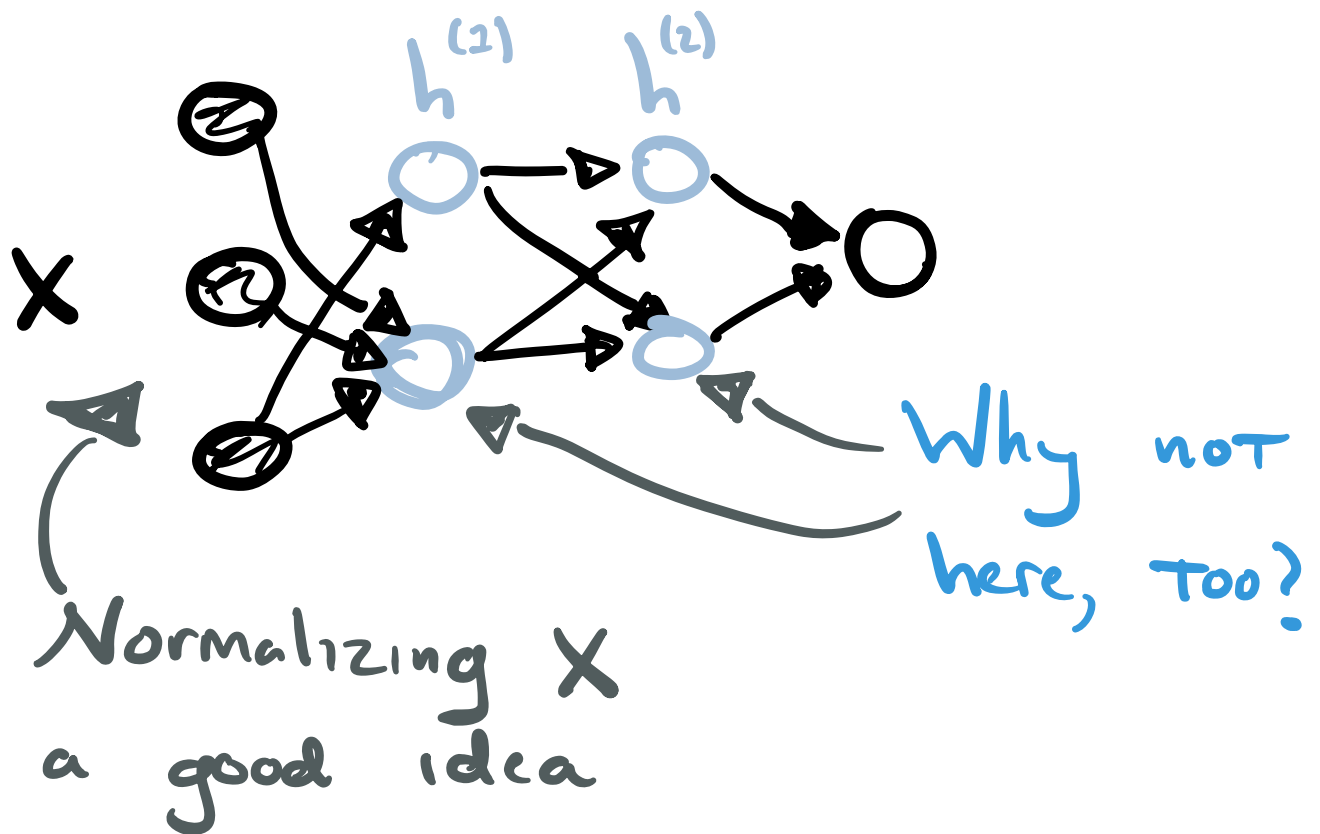


Dropout

Why The $\frac{1}{(1-p)}$ Term? Leaves expectation unchanged. $E[h'] = p \cdot 0 + (1-p) \frac{h}{(1-p)} = h$

Batch Norm

When we covered optimizers we discussed Normalizing inputs for gradient descent.



With batch norm we do this
Per batch.

Assume pre-activation outputs
 $\tilde{h}^l$ at layer l - So

$$h^l = a(\tilde{h}^l).$$

Then

$$\mu^l = \frac{1}{m} \sum_{j=1}^m \tilde{h}_j^l \quad \sigma^l = \frac{1}{m} \sum_{j=1}^m (\tilde{h}_j^l - \mu^l)^2$$

and

$$\tilde{h}_{\text{Norm}}^l = \frac{\tilde{h}^l - \mu^l}{\sqrt{\sigma^l + \epsilon}} \quad \text{For numerical reasons}$$

0 mean & Unit Variance; but
BN provides additional flexibility
by allowing This to be shifted

Per layer l

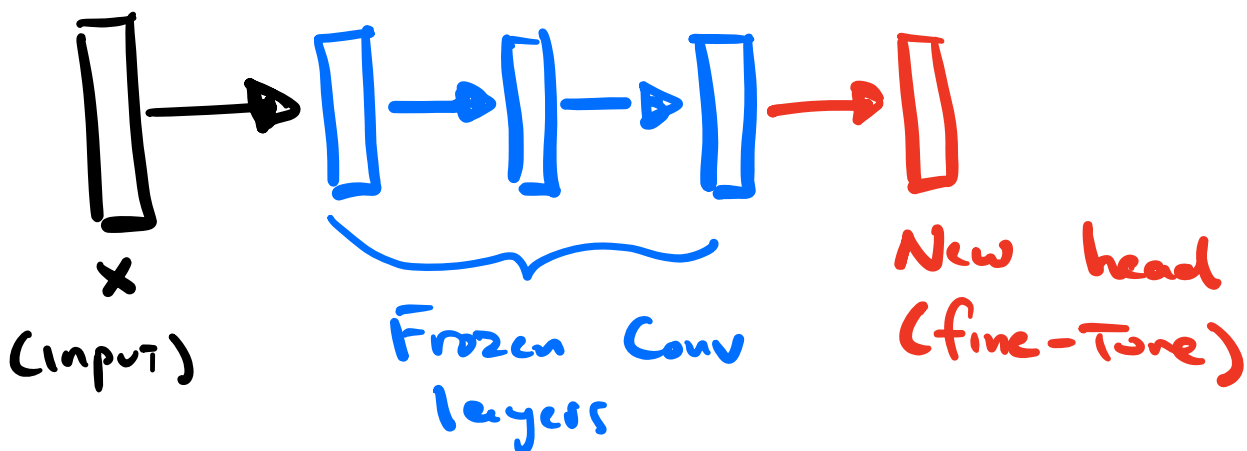
$$Z_{\text{Norm}}^l = \boxed{\hat{\gamma}^l} \odot \frac{\tilde{h}^l - \mu^l}{\sqrt{\sigma^l + \epsilon}} + \boxed{\hat{\beta}^l}$$

To be learned.

Transfer Learning

We wouldn't typically train from scratch — too data intensive.

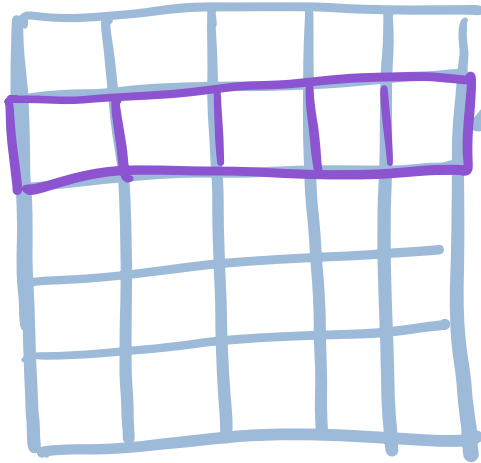
Instead, freeze lower layers and fine-tune upper layer(s).



CNNs for Text!

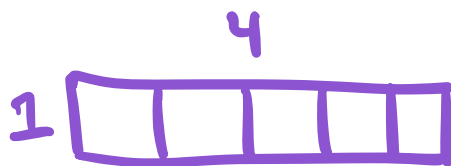
ConvNets mostly used for images

This
Movie
is
great!

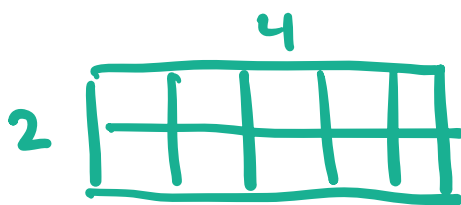


Embeddings
(10/5 lecture)!

Q. What should
our filter size be?

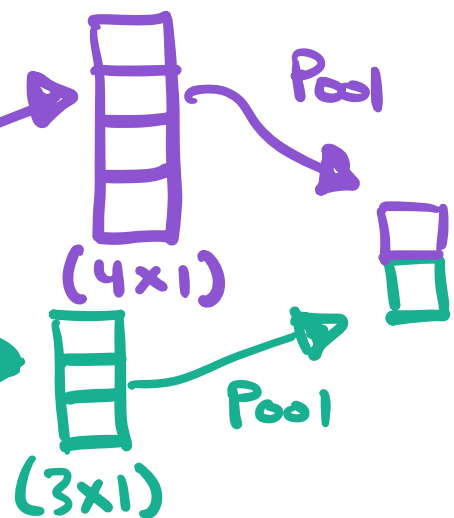
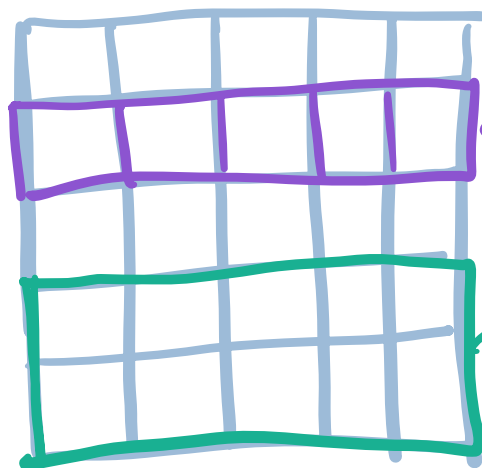


Unigrams



Bigrams

This
Movie
is
great!



(See Cohab)

Adversarial Examples

Earlier this semester, we took

$\frac{\partial \mathcal{L}}{\partial x}$ - grad w.r.t. **input**.

A famous example from Goodfellow et al.,

$$\begin{array}{c} \text{X} \\ \text{(panda)} \end{array} + \epsilon \cdot \begin{array}{c} \text{X} \\ \text{Sign}(\nabla_{\text{X}} \mathcal{L}) \end{array} = \begin{array}{c} \text{X} \\ \text{"gibbon"} \text{ 99\% Conf} \end{array}$$

The diagram illustrates the creation of an adversarial example. It shows a panda image (labeled 'X (panda)') being added to a small perturbation matrix (labeled 'X Sign(∇_X L)') to produce a new image (labeled 'X "gibbon" 99% Conf'). The perturbation matrix contains small black squares, representing the adversarial perturbation. The resulting image is visually indistinguishable from the original panda image but is classified as a gibbon with 99% confidence by the model.

Visually indistinguishable, but very different to the model.