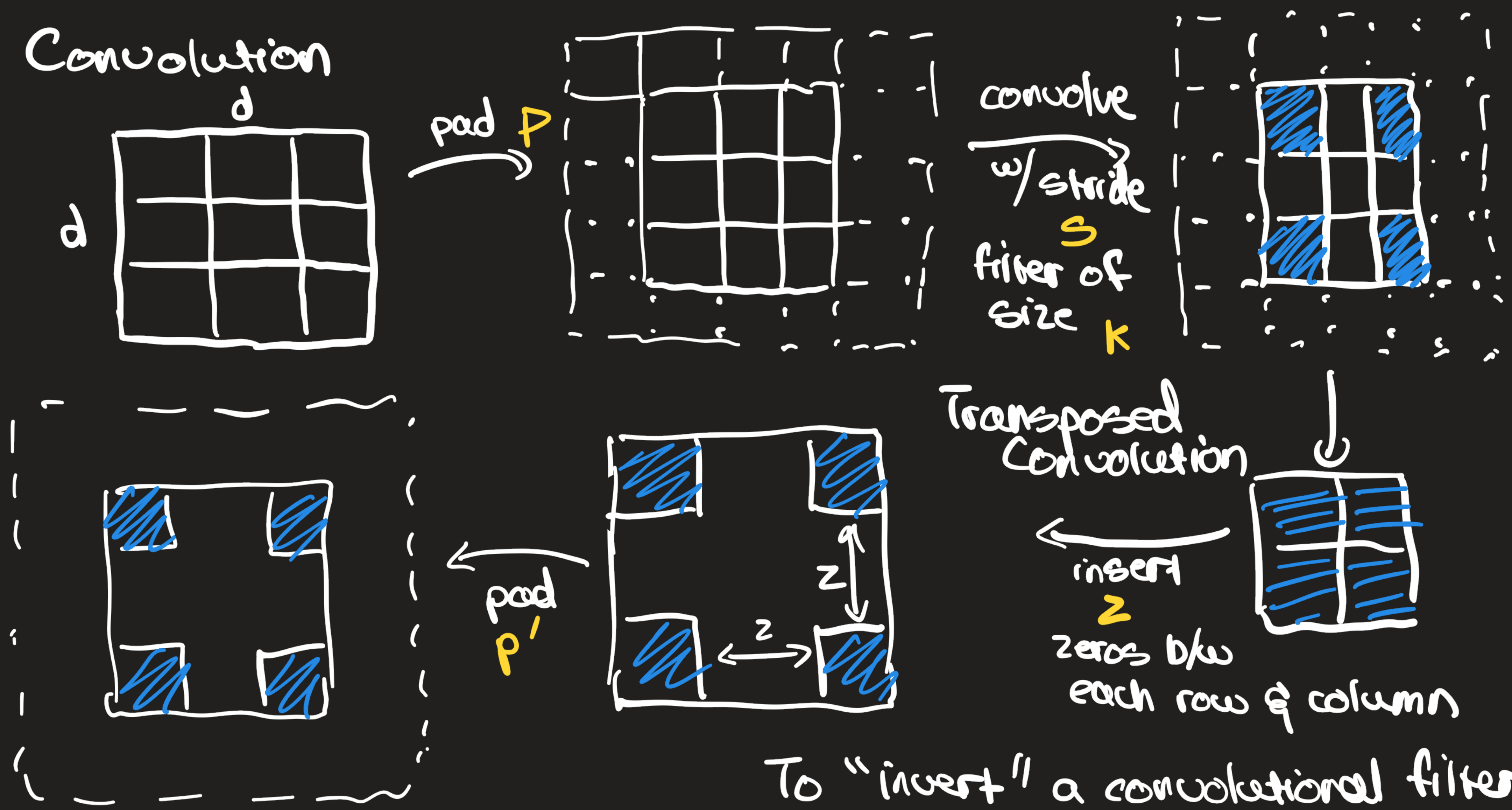
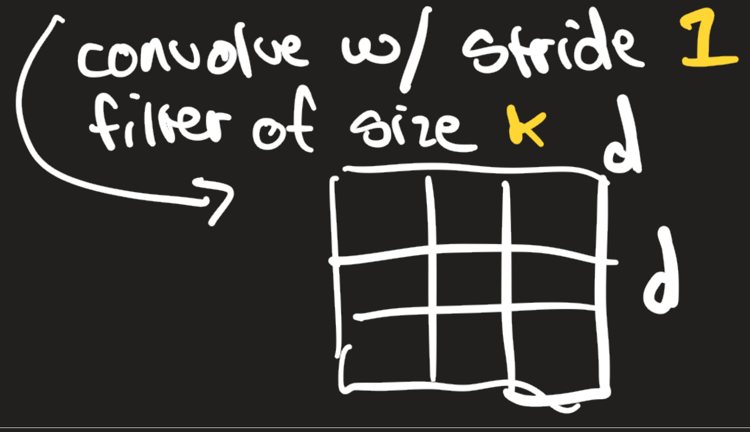
ML and Opt Lecture 17

- Transposed Convolutions
- Backprop for Convolutional Layers (sketch)
- Vanishing and Exploding Gradients
- Google Inception architecture (v1); auxiliary losses

Transposed Convolutions





To "invert" a convolutional filter K, we learn via BP, a filter $\bar{K}$. We need choose z, p' so that the images from K when passed through $\bar{K}$ have dimension d

Let o be the size of the image from the forward convolution

$$o = \frac{d + 2p - k}{s} + 1$$

pytorch:
Conv2D

Then d' be the size of the image from the transposed convolution when the input is size o

$$d' = o + (o-1)\cdot z + 2p' - k + 1$$

Want to choose z and p' so $d' = d$

pytorch:
Conv2DTranspose

Claim is: $z = s - 1$

$p' = k - p - 1$

$$d' = \left(\frac{d+2p-k}{s} + 1\right) + \left(\frac{d+2p-k}{s}\right)\cdot(s-1) + 2(k-p-1) - k + 1$$

$$= 1 + d + 2p - k + 2k - 2p - 2 - k + 1$$

$$= d \quad ✓$$

Conv2d

Input: $d \times d$

Kernel: $k \times k$

Padding: p

Stride: s

Output: $o \times o$

where $o = \frac{d+2p-k}{s} + 1$

Pads, then convolves with stride

Conv2D Transpose

Input: $o \times o$

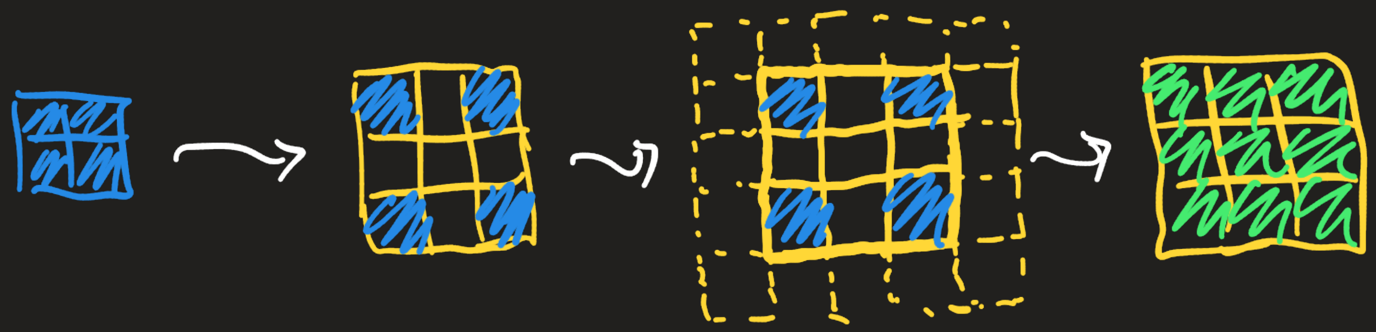
Kernel: $k \times k$

Padding: $k-p-1$

Spacing: $s-1$

Output: $d \times d$

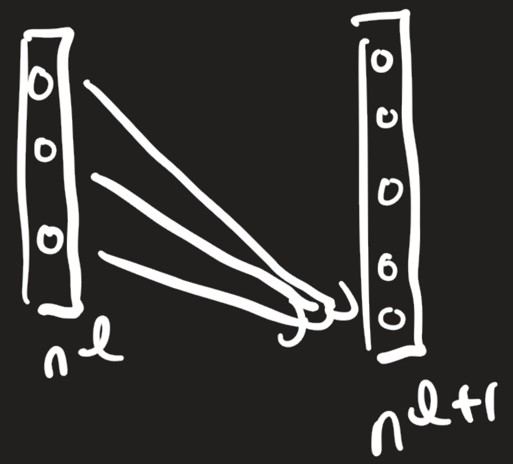
Inserts spaces, pads, then convolves with stride 1



Efficient Convolutions & Backprop

Parallels b/w MLPs and ConvNets

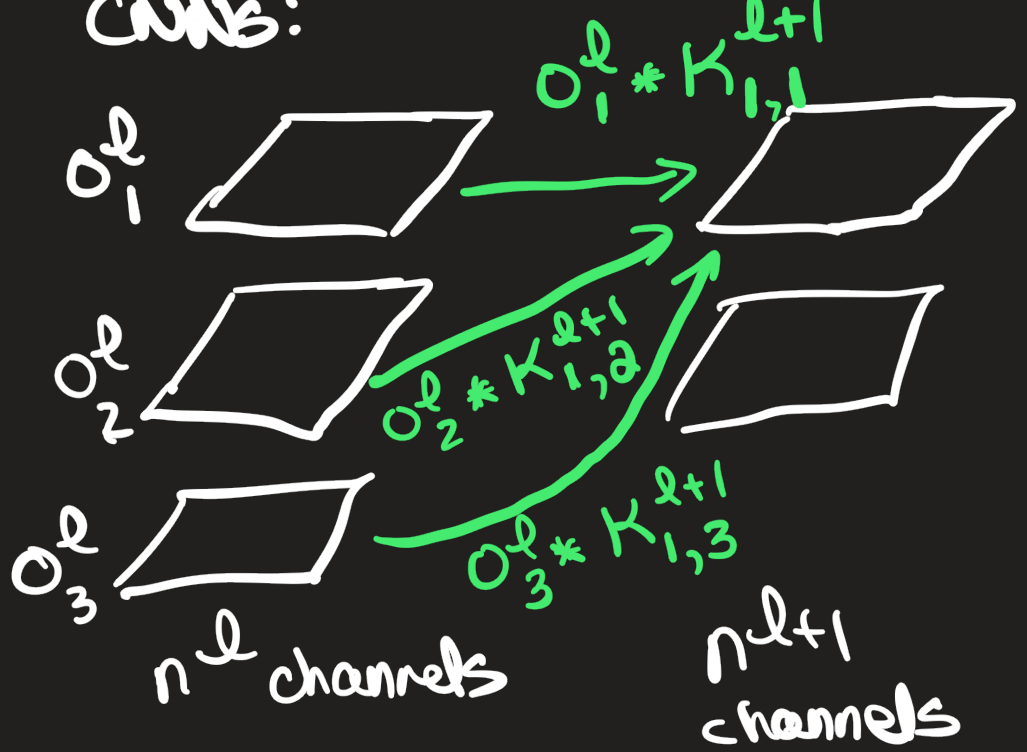
MLP:



$$a^{l+1} = W^{l+1} O^{l} + b^{l+1}$$

$$O^{l+1} = \sigma(a^{l+1})$$

CNNs:



$$A^{l+1}_{1} = O^{l}_{1} * K^{l+1}_{1,1} + O^{l}_{2} * K^{l+1}_{1,2} + O^{l}_{3} * K^{l+1}_{1,3} + b^{l+1}_{1}$$

a scalar that is the same for each pixel

$$O^{l+1}_{1} = \sigma(A^{l+1}_{1})$$

If layer ℓ has n_ℓ channels, then

$$A_i^{\ell+1} = \left(\sum_{j=1}^{n_\ell} O_j^\ell * K_{i,j}^{\ell+1} \right) + b_i^{\ell+1}$$

$$O_i^{\ell+1} = \sigma\left(A_i^{\ell+1}\right)$$

The number of parameters for layer $\ell+1$ is

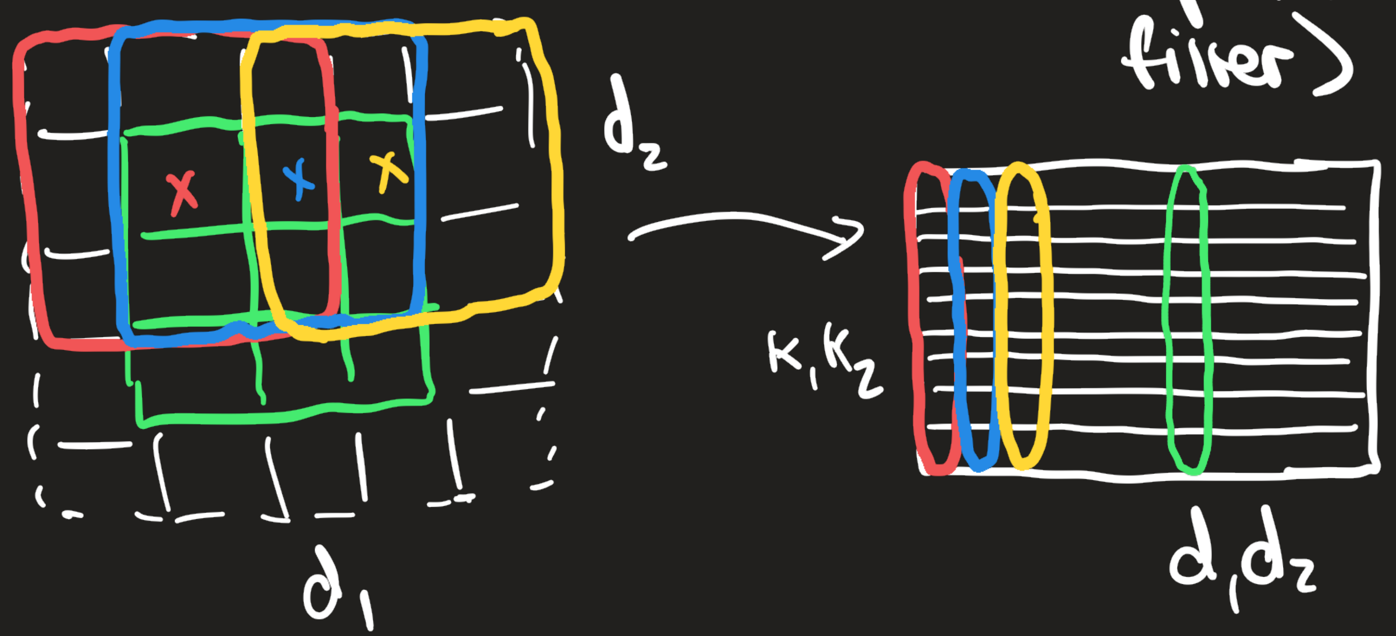
$n_\ell n_{\ell+1} K_1 K_2 + n_{\ell+1}$ (assuming all the filters connecting layer ℓ to layer $\ell+1$ are $K_1 \times K_2$)

(Sketch) Efficient Computations

cf Manas Sahni "Anatomy of a High-Speed Convolution"

idea: reduce convolution to the im2col operation and GEMMs

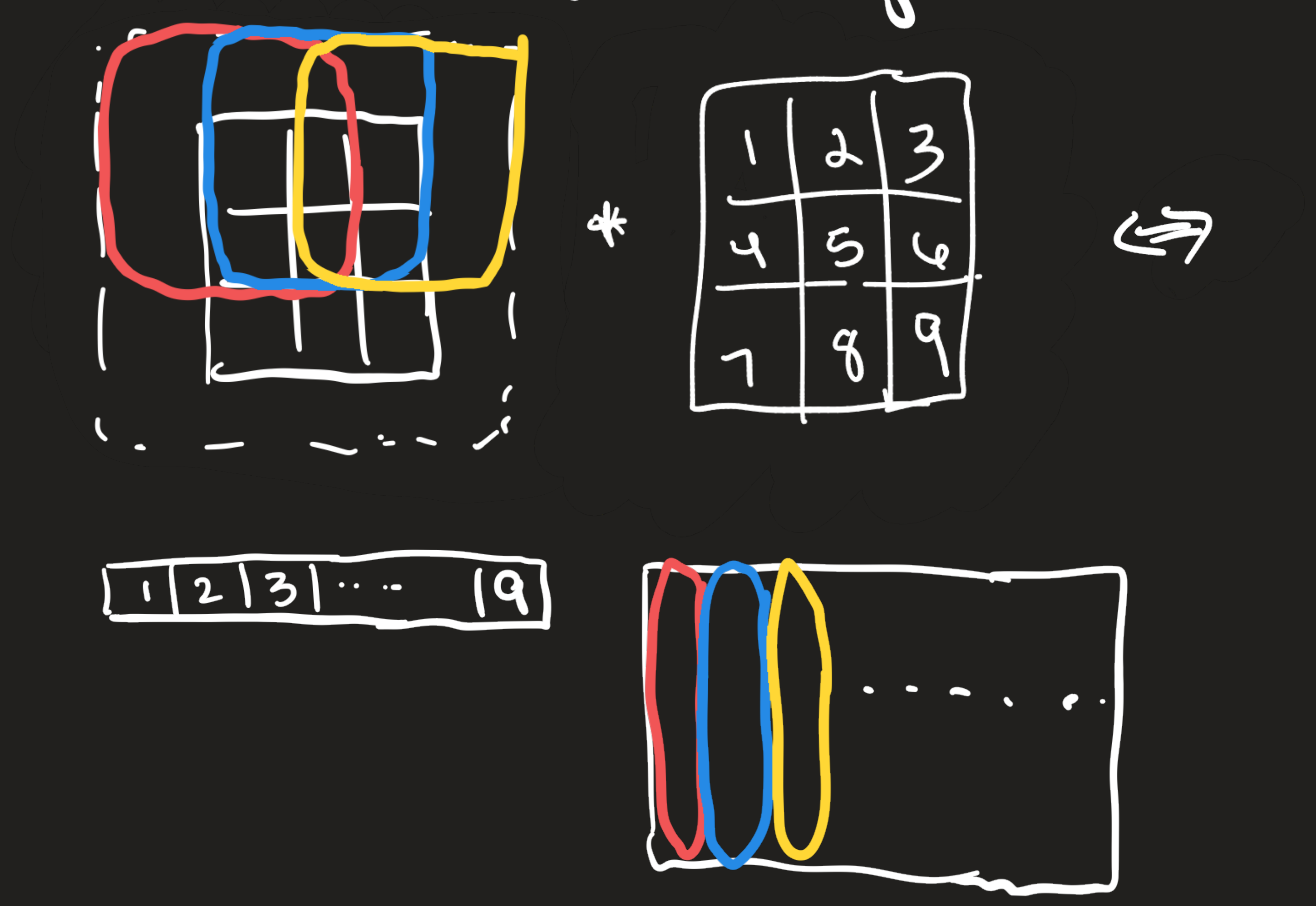
- im2col takes an image in $\mathbb{R}^{d_1 \times d_2}$ and maps to a matrix $\mathbb{R}^{k_1 k_2 \times d_1 d_2}$ (corresponding to zero-padding the input and convolving by a $k_1 \times k_2$ filter)



note that we want to compute $O_j^l * K_{i,j}^{l+1}$

and this implies

$$\mathrm{vec}\left(O_j^l * K_{i,j}^{l+1}\right) = \mathrm{vec}\left(K_{i,j}^{l+1}\right) \cdot \mathrm{im2col}\left(O_j^l\right)$$



This means we can write our CNN layers in the form

$\tilde{o}^{l}_{i} = \text{vec}(O^{l}_{i})$ — the vectorized channel i on convolutional layer l

$\tilde{a}^{l}_{i} = \text{vec}(A^{l}_{i})$ — the vectorized preactivation of channel i on layer l

$\tilde{K}^{l}_{i,j} = \text{vec}(K^{l}_{i,j})$ — the vectorized kernel connecting channel j from layer $l-1$ to channel i from layer l

$m^{l}_{i} = \text{im2col}(O^{l}_{i})$ — the matrix coming from applying im2col to the ith channel on layer l

$$\tilde{a}^{l+1}_{i} = \sum_{j=1}^{n_{l-1}} \tilde{K}^{l+1}_{j} m^{l}_{j} + b^{l+1}_{i}$$

$$\tilde{o}^{l+1}_{i} = \sigma(\tilde{a}^{l+1}_{i})$$

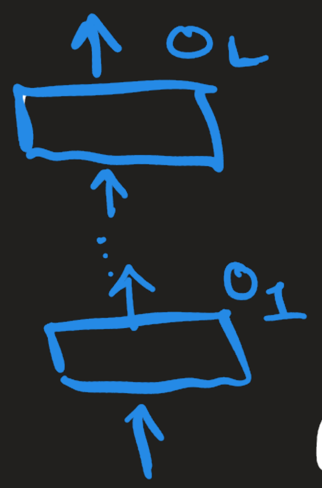
very similar to MLP forward pass, except uses im2col

Issues with deep NNs (not just CNNs):

- overfitting (too much capacity for the amount of training data)
- vanishing & exploding gradients $\Rightarrow$ slow learning!
- hyperparameter selection, e.g.
  - kernel sizes?
  - # channels per layer?
  - # layers?
  - type of pooling & locations?
  - stride & dilation & padding?
  - learning rates? algorithm? minibatch size?
  - weight decay?
  - ....

Vanishing & Exploding Gradients

Phenomenon that, as the number of layers increases,



as $l \to 1$ in the backpropagation algorithm,

$$\|\nabla_{W^l} f\|_2 \longrightarrow \begin{cases} 0 & \text{vanishing} \\ \infty & \text{exploding} \end{cases}$$

Occurs because of the chain rule.

Recall $o^{l+1} = \sigma(W^{l+1} o^l + b^{l+1})$ so by the multivariate chain rule,

$$\nabla_{o^l} f = J_{o^{l+1}}(o^l)^T \nabla_{o^{l+1}} f$$

and

$$\left[J_{o^{l+1}}(o^l)\right]_{ij} = \left[\frac{\partial o^{l+1}_i}{\partial o^l_j}\right]_{ij} = \sigma'\left(\left(W^{l+1} o^l + b^{l+1}\right)_i\right) \cdot \left(W^{l+1}\right)_{ij}$$

so

$$J_{o^{\ell+1}}(o^\ell) = \mathrm{Diag}\left(\sigma'\left(W^{\ell+1}o^\ell + b^{\ell+1}\right)\right)W^{\ell+1}$$

$$= \mathrm{Diag}\left(\sigma'(a^{\ell+1})\right)W^{\ell+1}$$

and consequently,

$$\nabla_{o^\ell} f = (W^{\ell+1})^T \mathrm{Diag}\left(\sigma'(a^{\ell+1})\right) \cdot \nabla_{o^{\ell+1}} f$$

$$= \cdots$$

$$= \prod_{i=\ell+1}^{L} \left[(W^i)^T \mathrm{Diag}\left(\sigma'(a^i)\right)\right] \cdot \nabla_{o^L} f$$

Two considerations:

1) How $\mathrm{diag}(\sigma'(a^{l+1}))$ behaves

so if a^l far from 0, then this looks like a zero matrix, so

$$\|\nabla_{a^l} f\|_2 \ll \|\nabla_{a^{l+1}} f\|_2$$

2) If the norm of our weight matrix W^{l+1} is large, then $\|\nabla_{a^l} f\|_2 \gg \|\nabla_{a^{l+1}} f\|_2$

if it is small, then $\|\nabla_{a^l} f\|_2 \ll \|\nabla_{a^{l+1}} f\|_2^2$

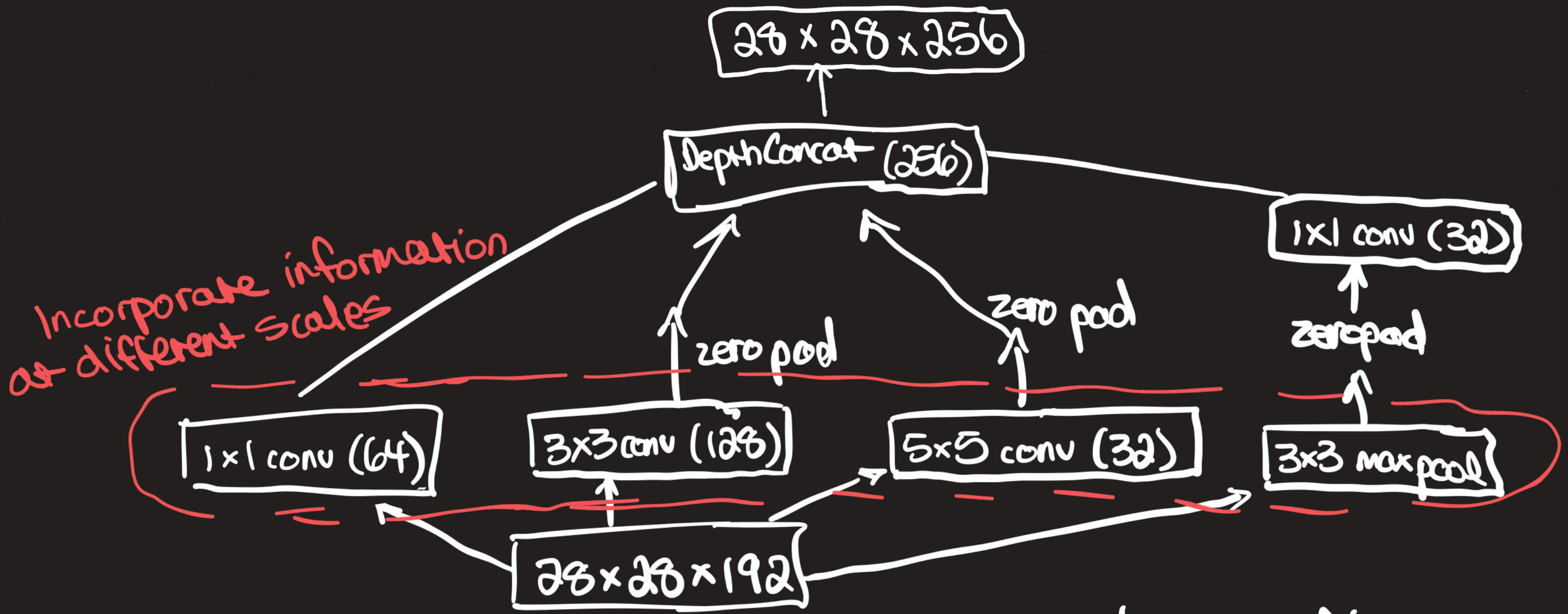
Consequence:

naively training deep NN architectures either fails or gives poor performance

Takeaways:

- we want to keep activations close to zero so the nonlinearities in the network do not saturate
  - suggests "normalization" layers to keep activations well-behaved.
- we want to keep weight matrices well-behaved:
  - suggests regularizing by norm of weight matrices
- we want to maintain short paths to the output to prevent attenuation/explosion to compound
  - suggests using "auxiliary" losses.

GoogLeNet (22 layer) CNN Inception v1 2014



Naive Inception Block: preserves input dimensions, uses information at multiple scales

Downside: lots of parameters

E.g. for the 3x3 conv features, we have

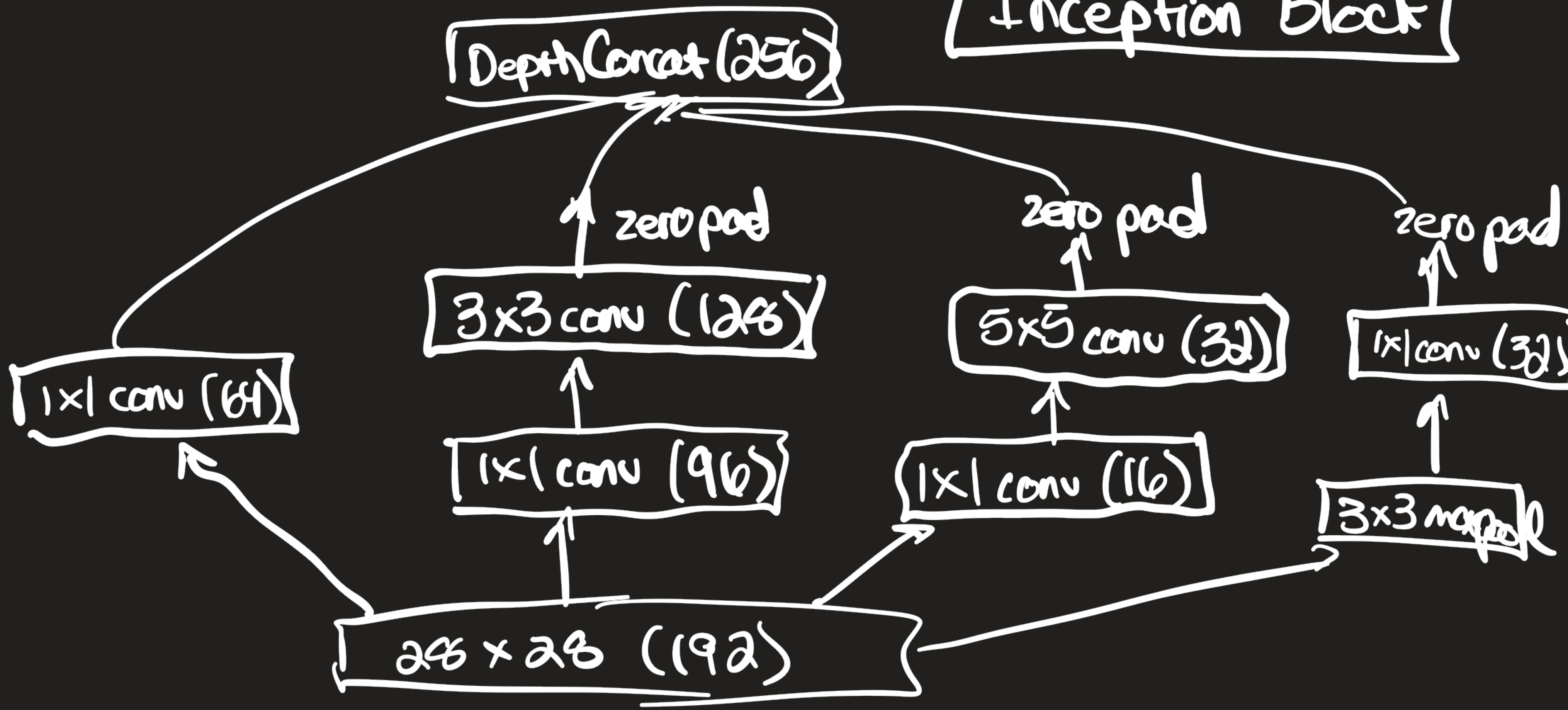
$$192 \times 128 \times 3 \times 3 + 128 = 221{,}312 \text{ parameters}$$

$96 \times 192 + 96$

Issue: too many parameters $\quad + 128 \times 96 \times 3 \times 3 + 128$

Soln: use 1×1 conv layers for dimensionality reduction

Inception Block



Check: # of 3x3 conv parameters (including the 96 1x1 conv)

= ? parameters

129,248

Original Inception (v1) architecture

CNN $\rightarrow$ Inception Block $\times 2$ $\rightarrow$ 3x3 max pool

$\hookrightarrow$ aux classification loss

$\rightarrow$ Inception Block $\times 5$ $\rightarrow$ 3x3 max pool

$\hookrightarrow$ aux classification loss

$\rightarrow$ Inception Block $\times 2$ $\rightarrow$ AvgPool $\rightarrow$ DropOut

$\rightarrow$ Linear $\rightarrow$ softmax $\rightarrow$ classification loss

Train this architecture to minimize the weighted sum of the three losses. Injects gradient information at intermediate layers to mitigate vanishing/exploding gradients.

Input: 224 x 224 x 3

type	patch size/ stride	output size	depth	$\#1\times1$	$\#3\times3$ reduce	$\#3\times3$	$\#5\times5$ reduce	$\#5\times5$	pool proj	params	ops
convolution	$7\times7/2$	$112\times112\times64$	1							2.7K	34M
max pool	$3\times3/2$	$56\times56\times64$	0								
convolution	$3\times3/1$	$56\times56\times192$	2		64	192				112K	360M
max pool	$3\times3/2$	$28\times28\times192$	0								
inception (3a)		$28\times28\times256$	2	64	96	128	16	32	32	159K	128M
inception (3b)		$28\times28\times480$	2	128	128	192	32	96	64	380K	304M
max pool	$3\times3/2$	$14\times14\times480$	0								
inception (4a)		$14\times14\times512$	2	192	96	208	16	48	64	364K	73M
inception (4b)		$14\times14\times512$	2	160	112	224	24	64	64	437K	88M
inception (4c)		$14\times14\times512$	2	128	128	256	24	64	64	463K	100M
inception (4d)		$14\times14\times528$	2	112	144	288	32	64	64	580K	119M
inception (4e)		$14\times14\times832$	2	256	160	320	32	128	128	840K	170M
max pool	$3\times3/2$	$7\times7\times832$	0								
inception (5a)		$7\times7\times832$	2	256	160	320	32	128	128	1072K	54M
inception (5b)		$7\times7\times1024$	2	384	192	384	48	128	128	1388K	71M
avg pool	$7\times7/1$	$1\times1\times1024$	0								
dropout (40%)		$1\times1\times1024$	0								
linear		$1\times1\times1000$	1							1000K	1M
softmax		$1\times1\times1000$	0								

Table 1: GoogLeNet incarnation of the Inception architecture