Data Augmentation (overfitting)

CNNs automatically deal well w/ translational invariance because of their architecture design:

- shared parameters (using the same filter at each pixel location)
- pooling

help to detect the same features in different regions

They are not very good at rotational invariance or scale invariance, so we build these in by augmenting the training dataset: add randomly rotated & cropped images to the dataset.
Inception architecture - state-of-the-art perf. on ILSVRC 2014

22 layers deep initially

Idea: max pooling loses spatial details

CNNs as we've seen them have issues w/ detecting the same feature at different scales

Trend: increase # layers & # filters per layer, using dropout to prevent overfitting

Problem w/ this approach is you see a quadratic increase in the # of parameters in our model:

if \( n \) and \( n_{2-1} \) (# of filters in each layer) increase by a factor of \( C \)
initially for each of the output channels on layer $n$, you had to learn $n_{l-1}$ kernels
so now you have to learn

$$(c_{n_{l}})(c_{n_{l-1}}) = c^2 n_{l} n_{l-1} \text{ kernels}$$

This is computationally expensive
Recall convolutional layers w/ multi-channel inputs:

layer \( l \)

\( \mathcal{I}_{l-1} \)

\( \mathcal{O}_l \)

input channels

output channel \( 1 \)

output channel \( \ell \)
Ex: passing a single image through my CNN
\( x \in \mathbb{R}^{28 \times 28 \times 3} \)
layer 1 has 2 output channels

Learn the green & red filters via backprop
GoogleNet (22 layers) Inception V1

Depth Concat (256)

28 x 28 x 256

1 x 1 conv (64)

3 x 3 conv (128)

5 x 5 conv (32)

28 x 28 x 192

For example: for 3 x 3 conv block we have

192 x 128 x 3 x 3 + 128 = 221,312 parameters

"Naive"

Inception Block
Issue: increased number of parameters
Soln: use some dim reduction via 1x1 convs

Inception Block

Depth Concat (256)

3x3 conv (128)
1x1 conv (96)
28 x 28 x 192

5x5 conv (32)
1x1 conv (16)
1x1 conv (32)
3x3 maxpool
With the $1 \times 1$ conv for dim reduction, the # of parameters for the $3 \times 3$ conv features now becomes

\[
(192 \times 96 \times 1 \times 1 + 96) + (96 \times 128 \times 3 \times 3 + 128)
\]

for the $1 \times 1$ conv

for the $3 \times 3$ conv

dim red.

\[
= 129,248 \text{ parameters}
\]

significantly less
Original Inception (v1) architecture

CNN $\rightarrow$ Inception Block $\times 2$
  $\rightarrow$ Inception Block $\times 5$
  $\rightarrow$ Inception Block $\times 2$
  $\rightarrow$ AvgPool $\rightarrow$ Dropout $\rightarrow$ Linear (1000)
  $\rightarrow$ Softmax
  $\rightarrow$ Classification loss

Train this architecture to minimize the sum of the classification losses (weighted)
Lower versions of Inception:
- replace each 5x5 conv layer w/ two sequential 3x3 conv layers
- use Batch Normalization so can go deeper & train faster (and remove dropout)
Skip Connections

Introduce skip connections b/w non-consecutive layers. This helps w/ gradient propagation.

\[ O_3 = \sigma(\omega^3 o_2 + b^3) + \omega^{1 \rightarrow 3}_{o_1} \]

\[ O_3 = \sigma(\omega^3 o_2 + \omega^{1 \rightarrow 3}_{o_1} + b^3) \]

- Skip connect from output of layer 1 to input of layer 3
- Skip connect from output of layer 1 to output of layer 3
Residual Network

Idea: it would be easy to learn arbitrarily deep networks if all the last layers could learn the identity function.

Residual block

\[ o^l = \sigma(\omega^l o^{l-1} + b^l) + o^{l-1} \]

This means \( \nabla o^l = \nabla o^{l-1} \)

For example, if \( \sigma = \text{ReLU} \) then learning the identity is easy: \( \omega^l = 0 = b^l \) \( \Rightarrow \) \( o^l = o^{l-1} \)
Recall vanishing & exploding gradients caused by the deriv of the activation and the norm of the weight matrix.
With residual blocks we can get very simple, very deep architectures

ResNet-152