Suppose we want to classify an image as a bird, sunset, dog, cat, etc. If we can identify features such as feather, eye, or beak which provide useful information in one part of the image, then those features are likely to also be relevant in another part of the image. We can exploit this regularity by using a convolution layer which applies the same weights to different parts of the image.
**Convolutional Network Components**

- **convolution layers**: extract shift-invariant features from the previous layer
- **subsampling or pooling layers**: combine the activations of multiple units from the previous layer into one unit
- **fully connected layers**: collect spatially diffuse information
- **output layer**: choose between classes

**Convolutional Network Architecture**

There can be multiple steps of convolution followed by pooling, before reaching the fully connected layers.

Note how pooling reduces the size of the feature map (usually, by half in each direction).
**Softmax (6.2.2)**

Consider a classification task with \( N \) classes, and assume \( z_j \) is the output of the unit corresponding to class \( j \).

We assume the network’s estimate of the probability of each class \( j \) is proportional to \( \exp(z_j) \). Because the probabilities must add up to 1, we need to normalize by dividing by their sum:

\[
\text{Prob}(i) = \frac{\exp(z_i)}{\sum_{j=1}^{N} \exp(z_j)}
\]

\[
\log \text{Prob}(i) = z_i - \log \sum_{j=1}^{N} \exp(z_j)
\]

If the correct class is \( i \), we can treat \( -\log \text{Prob}(i) \) as our cost function. The first term pushes up the correct class \( i \), while the second term mainly pushes down the incorrect class \( j \) with the highest activation (if \( j \neq i \)).

**Convolution Operator**

**Continuous convolution**

\[
s(t) = (x * w)(t) = \int x(a)w(t-a)da
\]

**Discrete convolution**

\[
s(t) = (x * w)(t) = \sum_{a=-\infty}^{\infty} x(a)w(t-a)
\]

**Two-dimensional convolution**

\[
S(j,k) = (K * I)(j,k) = \sum_{m} \sum_{n} K(m,n)I(j+m,k+n)
\]

Note: Theoreticians sometimes write \( I(j-m,k-n) \) so that the operator is commutative. But, computationally, it is easier to write it with a plus sign.

**Convolutional Neural Networks**

Assume the original image is \( J \times K \), with \( L \) channels.

We apply an \( M \times N \) “filter” to these inputs to compute one hidden unit in the convolution layer. In this example \( J = 6, K = 7, L = 3, M = 3, N = 3 \).

\[
Z_{j,k}^l = g(b^l + \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} K_{l,m,n}^i V_{j+m,k+n}^l)
\]

The same weights are applied to the next \( M \times N \) block of inputs, to compute the next hidden unit in the convolution layer (“weight sharing”).
Convolutional Neural Networks

If the original image size is $J \times K$ and the filter is size $M \times N$, the convolution layer will be $(J + 1 - M) \times (K + 1 - N)$

Example: LeNet

For example, in the first convolutional layer of LeNet, $J = K = 32, M = N = 5$.

The width of the next layer is

$$J + 1 - M = 32 + 1 - 5 = 28$$

Question: If there are 6 filters in this layer, compute the number of:
- weights per neuron?
- neurons?
- connections?
- independent parameters?

Max Pooling (9.3-9.4)

The $5 \times 5$ window of the first convolution layer extracts from the original $32 \times 32$ image a $28 \times 28$ array of features. Subsampling then halves this size to $14 \times 14$. The second Convolution layer uses another $5 \times 5$ window to extract a $10 \times 10$ array of features, which the second subsampling layer reduces to $5 \times 5$. These activations then pass through two fully connected layers into the 10 output units corresponding to the digits '0' to '9'.

Example: LeNet trained on MNIST

The $5 \times 5$ window of the first convolution layer extracts from the original $32 \times 32$ image a $28 \times 28$ array of features. Subsampling then halves this size to $14 \times 14$. The second Convolution layer uses another $5 \times 5$ window to extract a $10 \times 10$ array of features, which the second subsampling layer reduces to $5 \times 5$. These activations then pass through two fully connected layers into the 10 output units corresponding to the digits '0' to '9'.
Convolution with Zero Padding

Sometimes, we treat the off-edge inputs as zero (or some other value).

This is known as “Zero-Padding”.

Example: AlexNet (2012)

- 5 convolutional layers + 3 fully connected layers
- max pooling with overlapping stride
- softmax with 1000 classes
- 2 parallel GPUs which interact only at certain layers

With Zero Padding, the convolution layer is the same size as the original image (or the previous layer).
Assume the original image is $J \times K$, with $L$ channels. We again apply an $M \times N$ filter, but this time with a “stride” of $s > 1$. In this example $J = 7, K = 9, L = 3, M = 3, N = 3, s = 2$.

$j$ takes on the values $0, s, 2s, \ldots, (J - M)$

$k$ takes on the values $0, s, 2s, \ldots, (K - N)$

The next layer is $(1 + (J - M)/s) \times (1 + (K - N)/s)$

When combined with zero padding of width $P$,

$j$ takes on the values $0, s, 2s, \ldots, (J + 2P - M)$

$k$ takes on the values $0, s, 2s, \ldots, (K + 2P - N)$

The next layer is $(1 + (J + 2P - M)/s) \times (1 + (K + 2P - N)/s)$
Example: AlexNet Conv Layer 1

For example, in the first convolutional layer of AlexNet, $J = K = 224$, $P = 2$, $M = N = 11$, $s = 4$.

The width of the next layer is

$$1 + (J + 2P - M)/s = 1 + (224 + 2 \times 2 - 11)/4 = 55$$

Question: If there are 96 filters in this layer, compute the number of:

- weights per neuron?
- neurons?
- connections?
- independent parameters?

Overlapping Pooling

If the previous layer is $J \times K$, and max pooling is applied with width $F$ and stride $s$, the size of the next layer will be

$$(1 + (J - F)/s) \times (1 + (K - F)/s)$$

Question: If max pooling with width 3 and stride 2 is applied to the features of size $55 \times 55$ in the first convolutional layer of AlexNet, what is the size of the next layer?

Answer:

Question: How many independent parameters does this add to the model?

Answer: