DATA 586: Advanced Machine Learning

2023W2

Shan Du

Convolutional Neural Networks

- Neural networks rebounded around 2010 with big successes in image classification.
- Around that time, massive databases of labeled images were being accumulated, with ever-increasing numbers of classes.

Convolutional Neural Networks

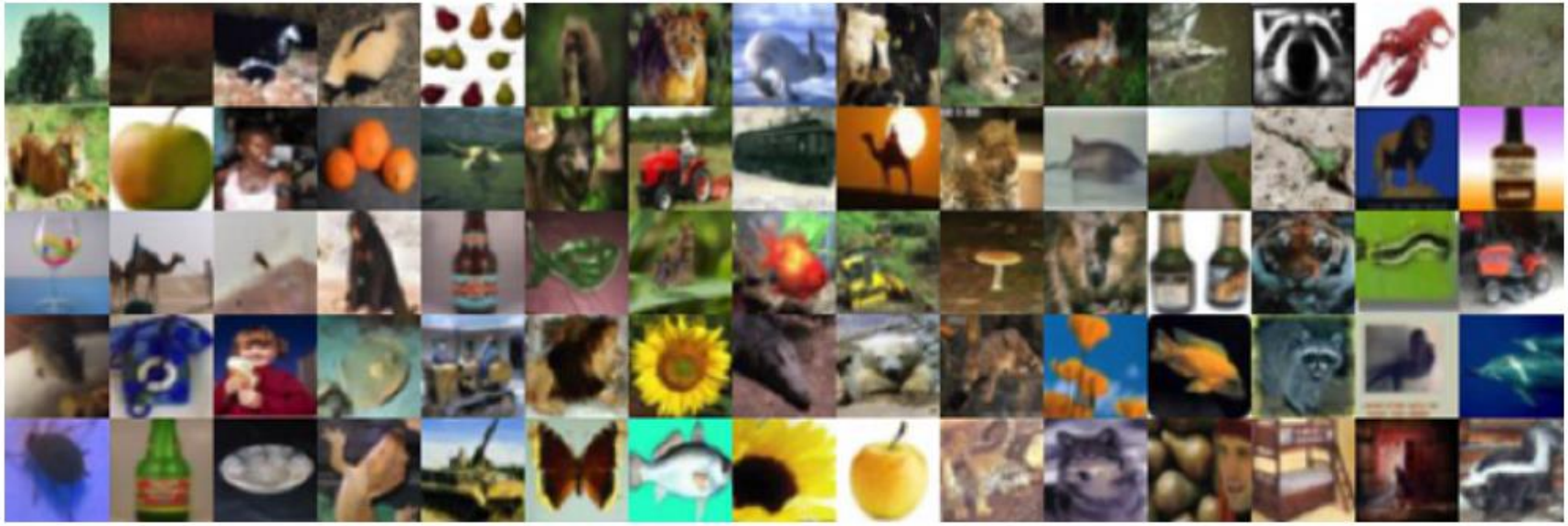


FIGURE 10.5. *A sample of images from the* CIFAR100 *database: a collection of natural images from everyday life, with 100 different classes represented.*

Convolutional Neural Networks

- A special family of convolutional neural networks (CNNs) has evolved for classifying images such as these, and has shown spectacular success on a wide range of problems.
- CNNs mimic to some degree how humans classify images, by recognizing specific features or patterns anywhere in the image that distinguish each particular object class.

Convolutional Neural Networks

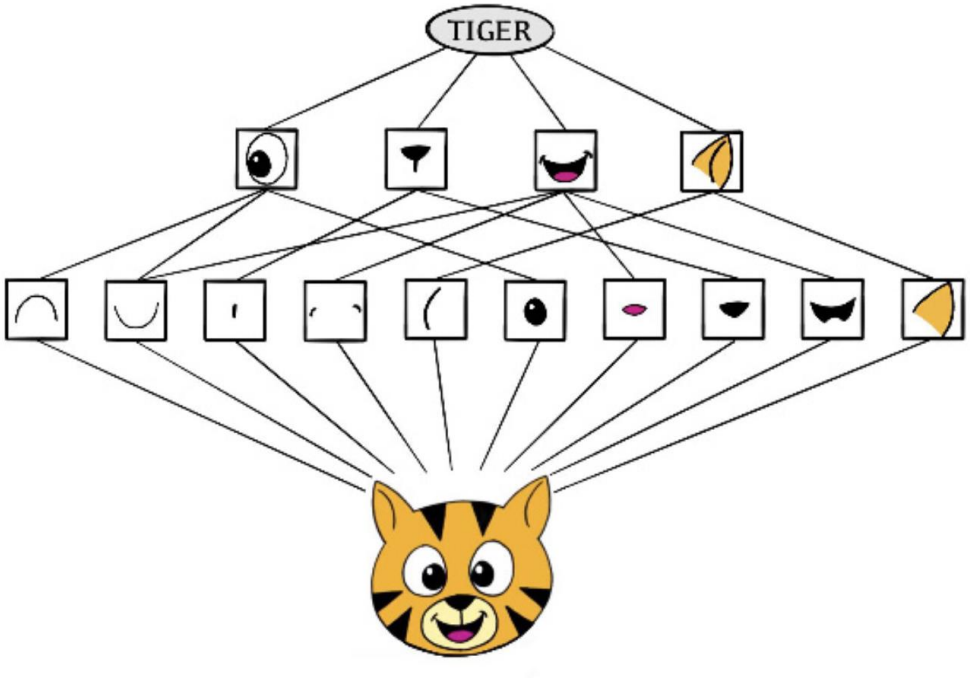


FIGURE 10.6. *Schematic showing how a convolutional neural network classifies an image of a tiger. The network takes in the image and identifies local features. It then combines the local features in order to create compound features, which in this example include eyes and ears. These compound features are used to output the label "tiger".*

Convolutional Neural Networks

- How does a convolutional neural network build up this hierarchy? It combines two specialized types of hidden layers, called *convolution* layers and *pooling* layers.
- Convolution layers search for instances of small patterns in the image, whereas pooling layers downsample these to select a prominent subset.
- In order to achieve state-of-the-art results, contemporary neural network architectures make use of many convolution and pooling layers.

Convolution Layers

- A *convolution layer* is made up of a large number of *convolution filters*, each of which is a template that determines whether a particular local feature is present in an image.
- A convolution filter relies on a very simple operation, called a *convolution*, which basically amounts to repeatedly multiplying matrix elements and then adding the results.

Convolution

- Original Image = $\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \\ j & k & l \end{bmatrix}$
- Convolution Filter = $\begin{bmatrix} \alpha & \beta \\ \gamma & \delta \end{bmatrix}$
- Convolved Image = $\begin{bmatrix} a\alpha + b\beta + d\gamma + e\delta & b\alpha + c\beta + e\gamma + f\delta \\ d\alpha + e\beta + g\gamma + h\delta & e\alpha + f\beta + h\gamma + i\delta \\ g\alpha + h\beta + j\gamma + k\delta & h\alpha + i\beta + k\gamma + l\delta \end{bmatrix}$

Convolution

- Original Image = $\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \\ j & k & l \end{bmatrix}$
- Convolution Filter = $\begin{bmatrix} \alpha & \beta \\ \gamma & \delta \end{bmatrix}$
- Convolved Image =

$$\begin{bmatrix} a\alpha + b\beta + d\gamma + e\delta & b\alpha + c\beta + e\gamma + f\delta \\ d\alpha + e\beta + g\gamma + h\delta & e\alpha + f\beta + h\gamma + i\delta \\ g\alpha + h\beta + j\gamma + k\delta & h\alpha + i\beta + k\gamma + l\delta \end{bmatrix}$$

Attn: Correct only if the filter is symmetric!

Convolution and Correlation

- Convolution is a widely used mathematical operator that processes an image by computing—for each pixel—a weighted sum of the values of that pixel and its neighbors. Depending on the choice of weights, a wide variety of image processing operations can be implemented.
- Convolution and correlation are the two fundamental mathematical operations involved in linear neighborhood-oriented image processing algorithms. The two operations differ in a very subtle way.

Convolution in the 1-D Domain

- The convolution between two discrete one-dimensional (1D) arrays $A(x)$ and $B(x)$, denoted by $A * B$, is mathematically described by the equation

$$A * B = \sum_{j=-\infty}^{\infty} A(j) \cdot B(x - j)$$

Convolution in the 1-D Domain

- In this example, we show how the result of a 1D convolution operation can be obtained step-by-step. Let $A = \{0, 1, 2, 3, 2, 1, 0\}$ and *B = {1, 3,−1}*. The partial results of multiplying elements in *A* with corresponding elements in *B*, as *B* shifts from $-\infty$ to ∞, are displayed below.

Convolution in the 1-D Domain

1. Initially, we mirror array *B* and align its center (reference) value with the first (leftmost) value of array *A*. The partial result of the convolution calculation $(0 \times (-1)) + (0 \times 3) + (1 \times 1) = 1$ (where empty spots are assumed as zero) is stored in the resulting array ($A * B$).

A		0	1	2	3	2	1	0
B	-1	3	1					
$A * B$		1						

Convolution in the 1-D Domain

2. Array *B* is shifted one position to the right. The partial result of the convolution calculation $(0 \times (-1)) + (1 \times 3) + (2 \times 1) = 5$ is stored in the resulting array $(A * B)$.

A	0	1	2	3	2	1	0
B	-1	3	1				
$A * B$	1	5					

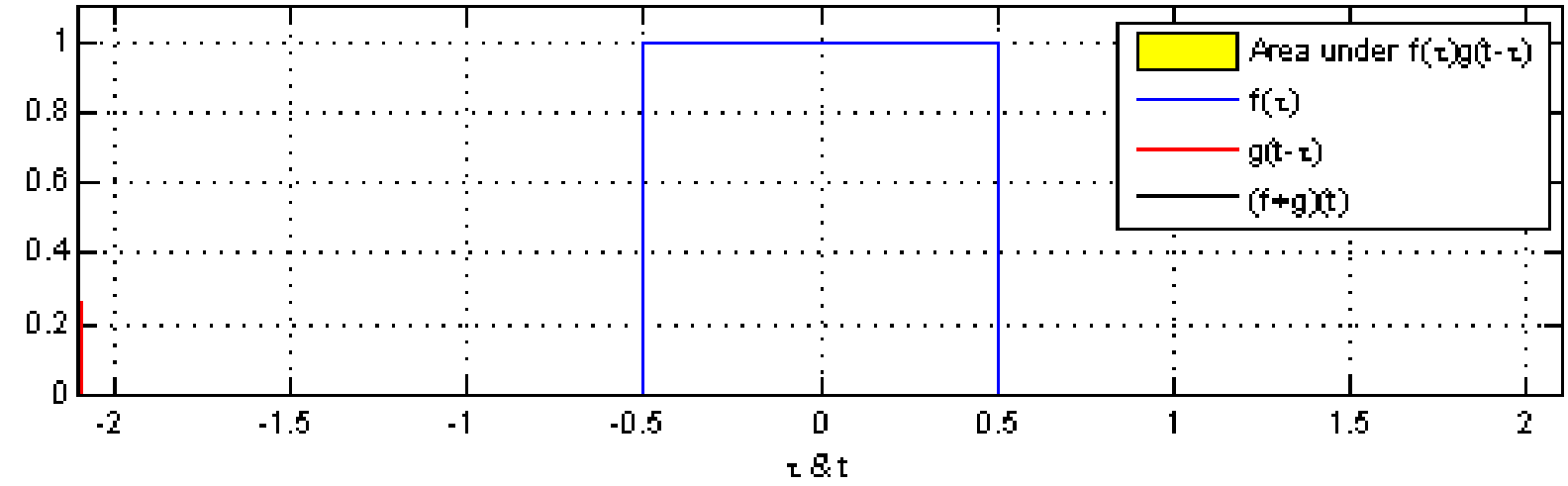
Convolution in the 1-D Domain

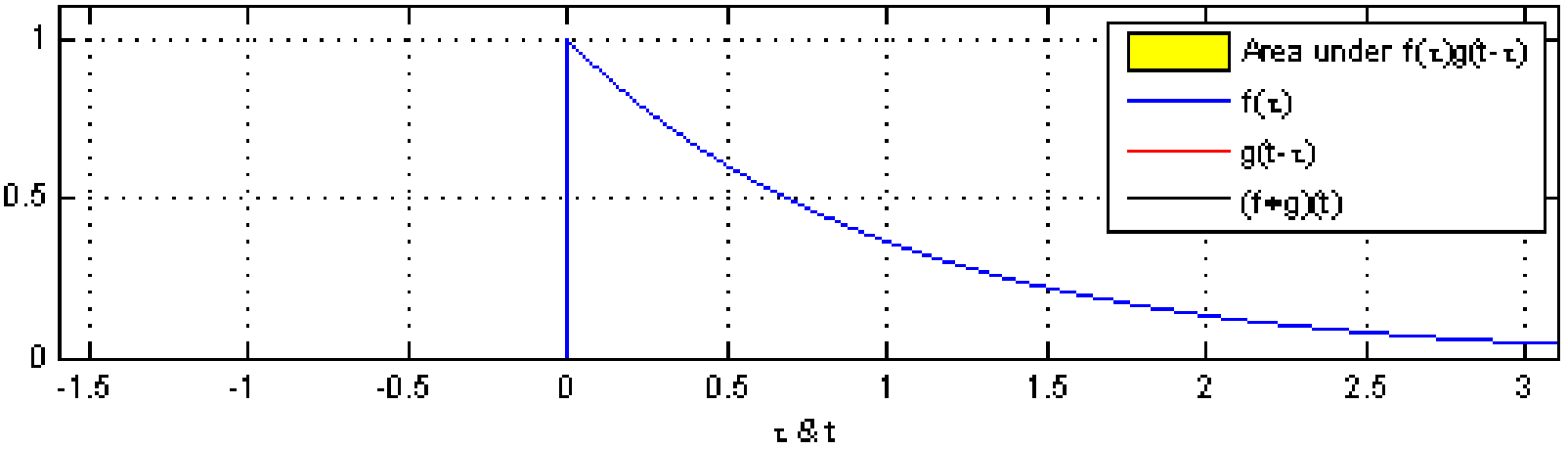
- Array *B* is shifted another position to the right. The partial result of the convolution calculation $(1 \times (-1)) + (2 \times 3) + (3 \times 1) = 8$ is stored in the resulting array $(A * B)$.

A	0	1	2	3	2	1	0
B		-1	3	1			
$A * B$	1	5	8				

- The final result of the convolution operation is the array {1, 5, 8, 8, 4, 1,−1}.

Convolution in the 1-D Domain (wikipedia)





Convolution in the 2-D Domain

- The mathematical definition for 2D convolution is

$$g(x,y) = \sum_{k=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} h(j,k) \cdot f(x-j, y-k)$$

In practice, this is rewritten as

$$g(x,y) = \sum_{k=-n_2}^{n_2} \sum_{j=-m_2}^{m_2} h(j,k) \cdot f(x-j, y-k)$$

Convolution in the 2-D Domain

where m_2 is equal to half of the mask's width and n_2 is equal to half of the mask's height,

$$m_2 = \lfloor m/2 \rfloor$$
$$n_2 = \lfloor n/2 \rfloor$$

- The basic mechanism used to understand 1D convolution can be expanded to the 2D domain. In such cases, the 2D array *A* is usually the input image and *B* is a small (usually 3×3) mask. The idea of mirroring *B* and shifting it across *A* can be adapted to the 2D case as well: mirroring will now take place in both *x* and *y* dimensions and shifting will be done starting from the top left point in the image, moving along each line, until the bottom right pixel in *A* has been processed.

Convolution in the 2-D Domain

$$A = \begin{bmatrix} 5 & 8 & 3 & 4 & 6 & 2 & 3 & 7 \\ 3 & 2 & 1 & 1 & 9 & 5 & 1 & 0 \\ 0 & 9 & 5 & 3 & 0 & 4 & 8 & 3 \\ 4 & 2 & 7 & 2 & 1 & 9 & 0 & 6 \\ 9 & 7 & 9 & 8 & 0 & 4 & 2 & 4 \\ 5 & 2 & 1 & 8 & 4 & 1 & 0 & 9 \\ 1 & 8 & 5 & 4 & 9 & 2 & 3 & 8 \\ 3 & 7 & 1 & 2 & 3 & 4 & 4 & 6 \end{bmatrix}$$

$$B = \begin{bmatrix} 2 & 1 & 0 \\ 1 & 1 & -1 \\ 0 & -1 & -2 \end{bmatrix}$$

Mirror of B is

$$\begin{bmatrix} -2 & -1 & 0 \\ -1 & 1 & 1 \\ 0 & 1 & 2 \end{bmatrix}$$

Convolution in the 2-D Domain

- The result of convolution will be

$$A * B = \begin{bmatrix} 20 & 10 & 2 & 26 & 23 & 6 & 9 & 4 \\ 18 & 1 & -8 & 2 & 7 & 3 & 3 & -11 \\ 14 & 22 & 5 & -1 & 9 & -2 & 8 & -1 \\ 29 & 21 & 9 & -9 & 10 & 12 & -9 & -9 \\ 21 & 1 & 16 & -1 & -3 & -4 & 2 & 5 \\ 15 & -9 & -3 & 7 & -6 & 1 & 17 & 9 \\ 21 & 9 & 1 & 6 & -2 & -1 & 23 & 2 \\ 9 & -5 & -25 & -10 & -12 & -15 & -1 & -12 \end{bmatrix}$$

Correlation

- 1D correlation can be mathematically expressed as

$$A \odot B = \sum_{j=-\infty}^{\infty} A(j) \cdot B(x+j)$$

whereas the 2D equivalent is given by

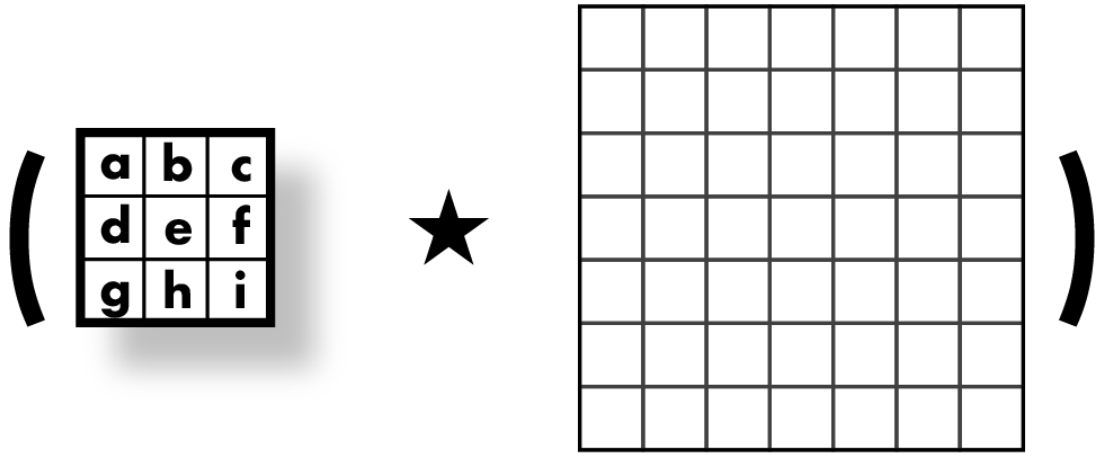
$$g(x,y) = \sum_{k=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} h(j,k) \cdot f(x+j, y+k)$$

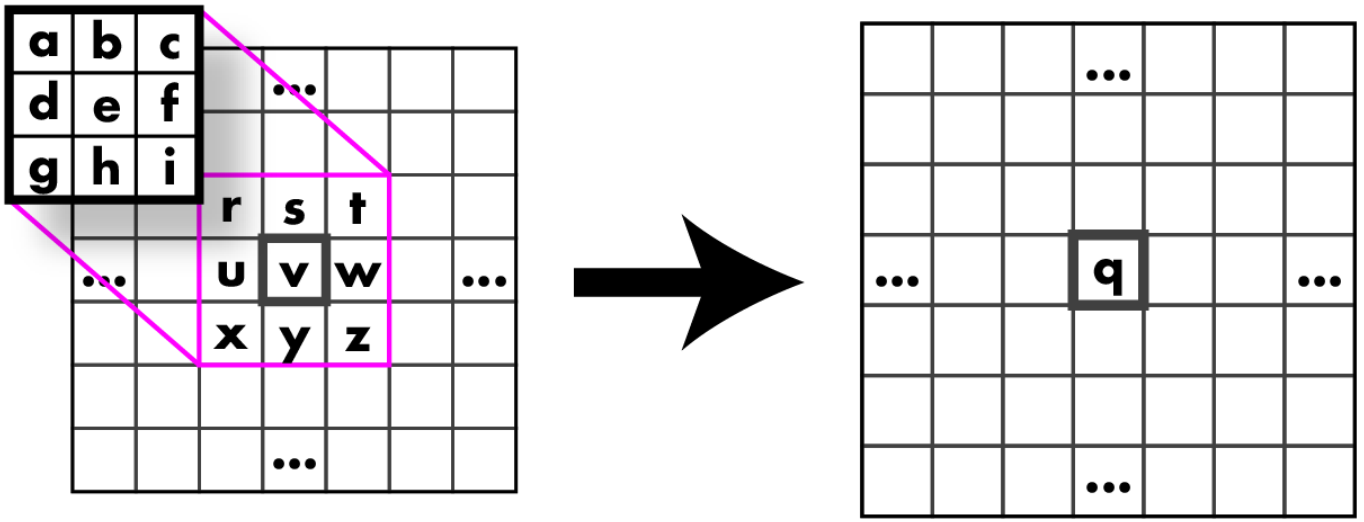
Correlation

In practice, this is rewritten as

$$g(x,y) = \sum_{k=-n_2}^{n_2} \sum_{j=-m_2}^{m_2} h(j,k) \cdot f(x+j, y+k)$$

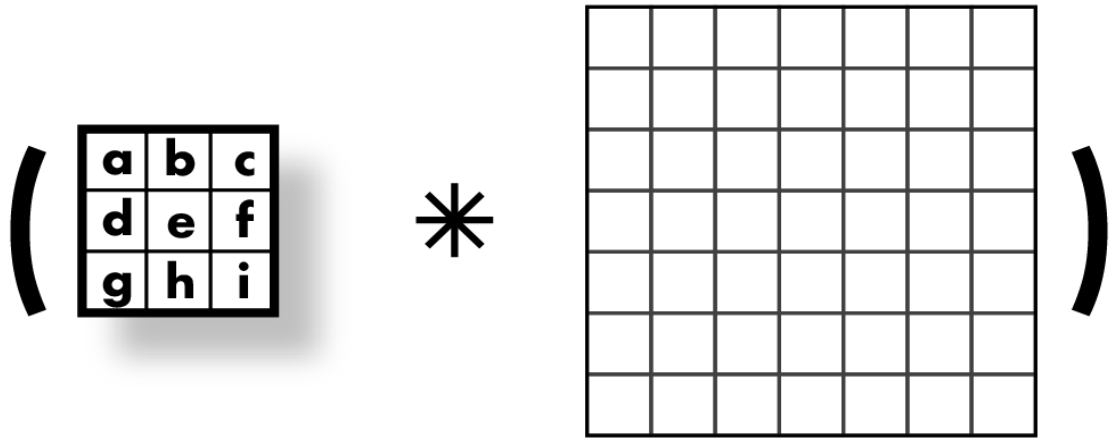
Cross-Correlation

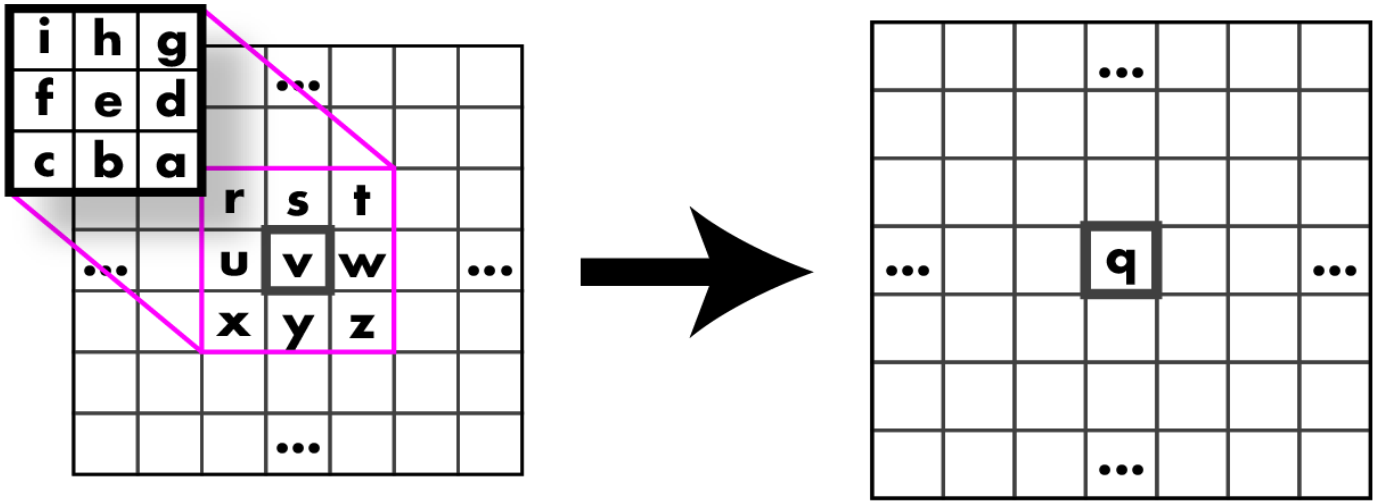




$$q = a \times r + b \times s + c \times t + d \times u + e \times v + f \times w + g \times x + h \times y + i \times z$$

Convolution





$$q = i \times r + h \times s + g \times t + f \times u + e \times v + d \times w + c \times x + b \times y + a \times z$$

Convolution

- If some region of the original image resembles the convolution filter, then it will have a *large* value in the convolved image; otherwise, it will have a *small* value.
- Thus, the *convolved image highlights regions of the original image that resemble the convolution filter*.

Convolution

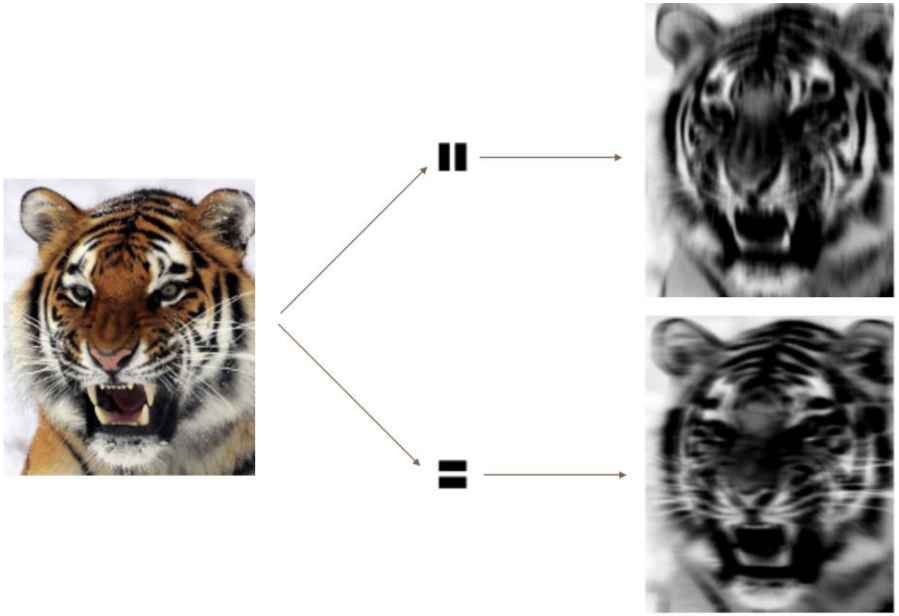
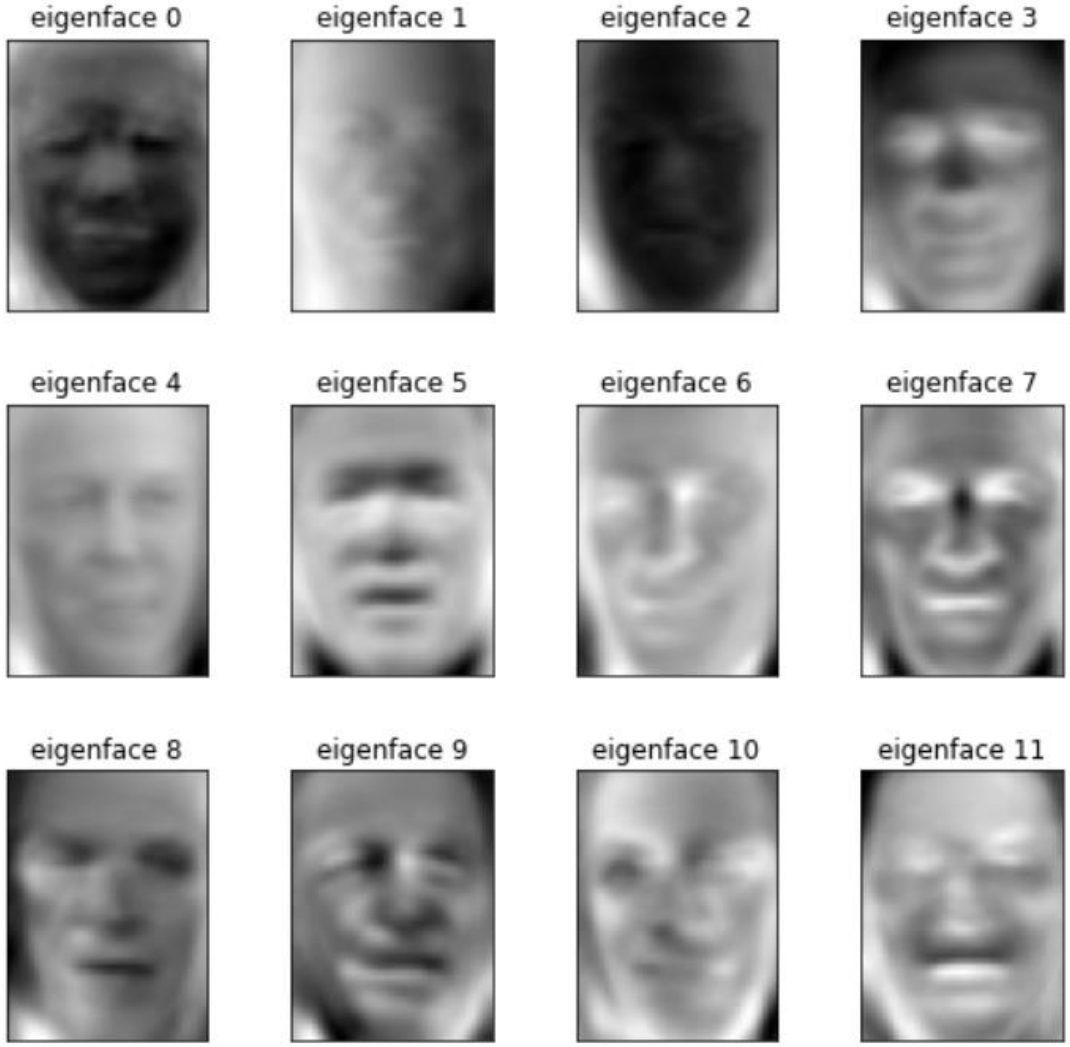
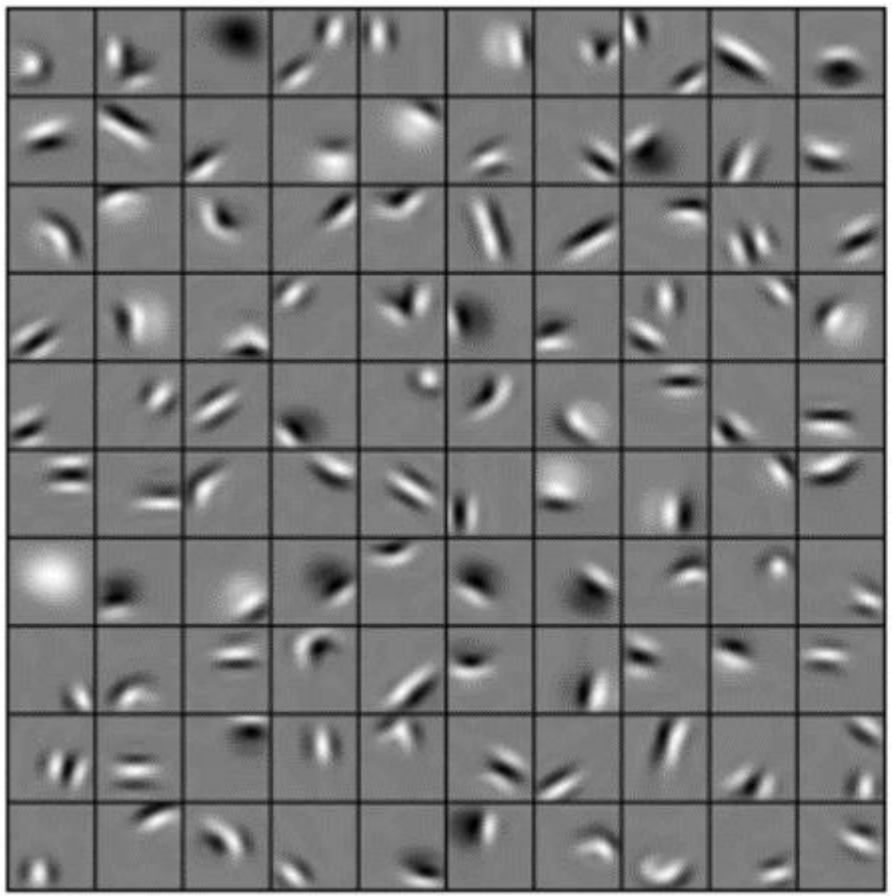


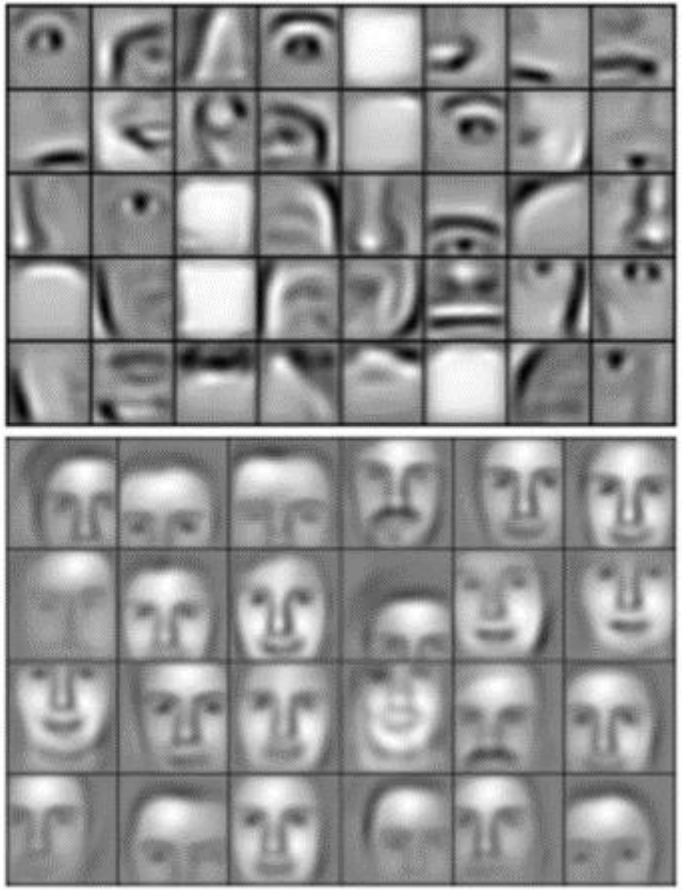
FIGURE 10.7. *Convolution filters find local features in an image, such as edges and small shapes. We begin with the image of the tiger shown on the left, and apply the two small convolution filters in the middle. The convolved images highlight areas in the original image where details similar to the filters are found. Specifically, the top convolved image highlights the tiger's vertical stripes, whereas the bottom convolved image highlights the tiger's horizontal stripes. We can think of the original image as the input layer in a convolutional neural network, and the convolved images as the units in the first hidden layer.*

Features of Faces

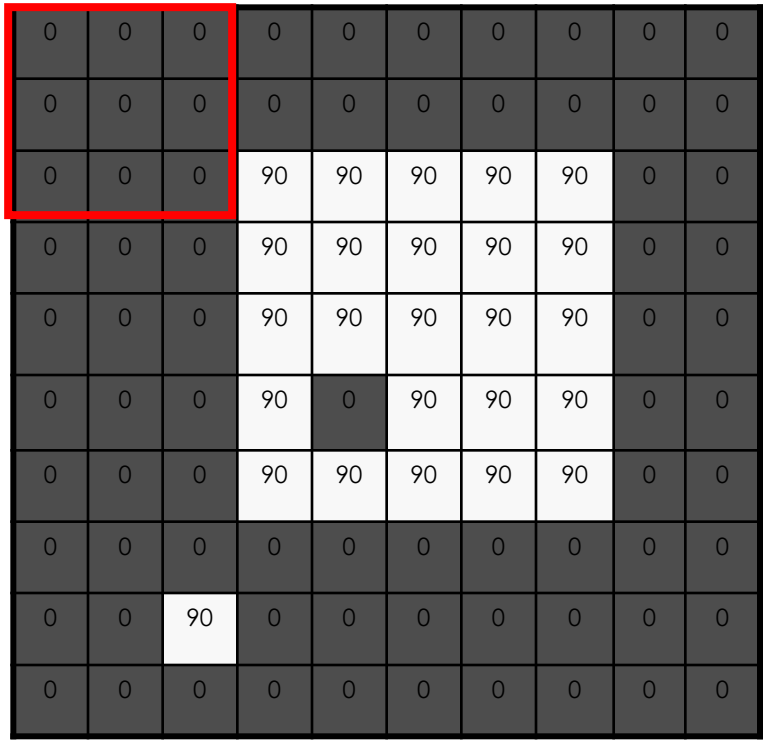


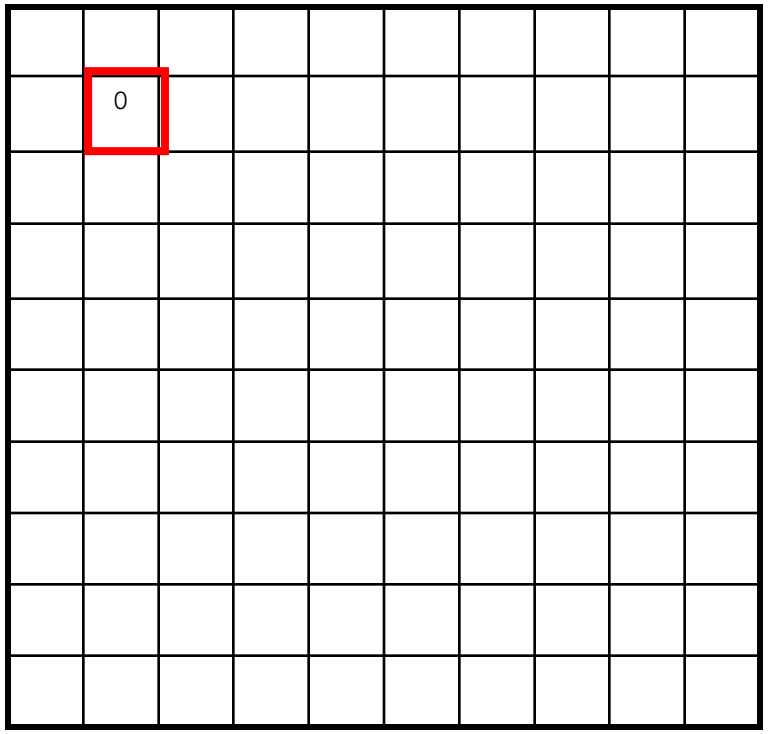
Features of Faces and Kernels



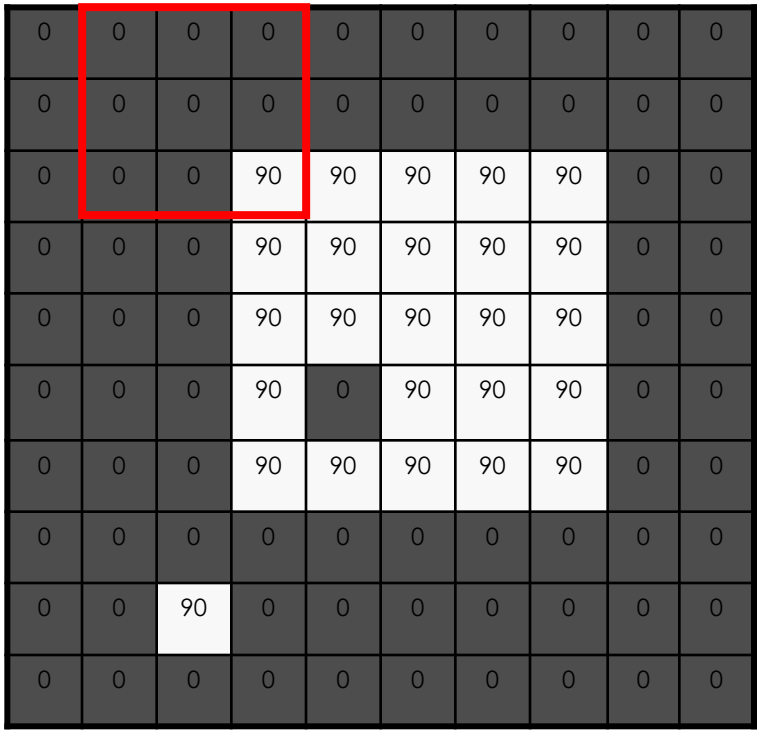


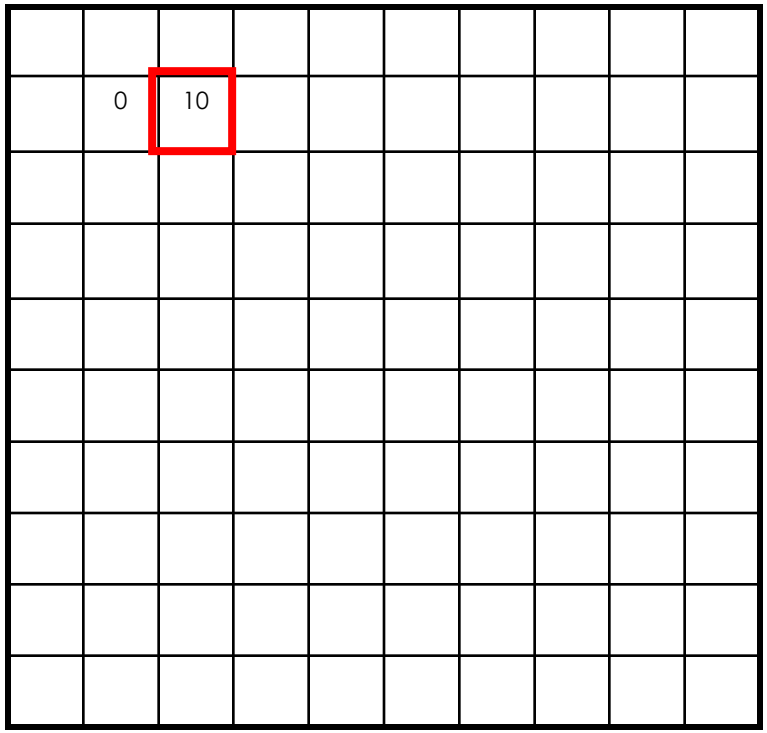
Sliding Averaging Kernel for Convolution



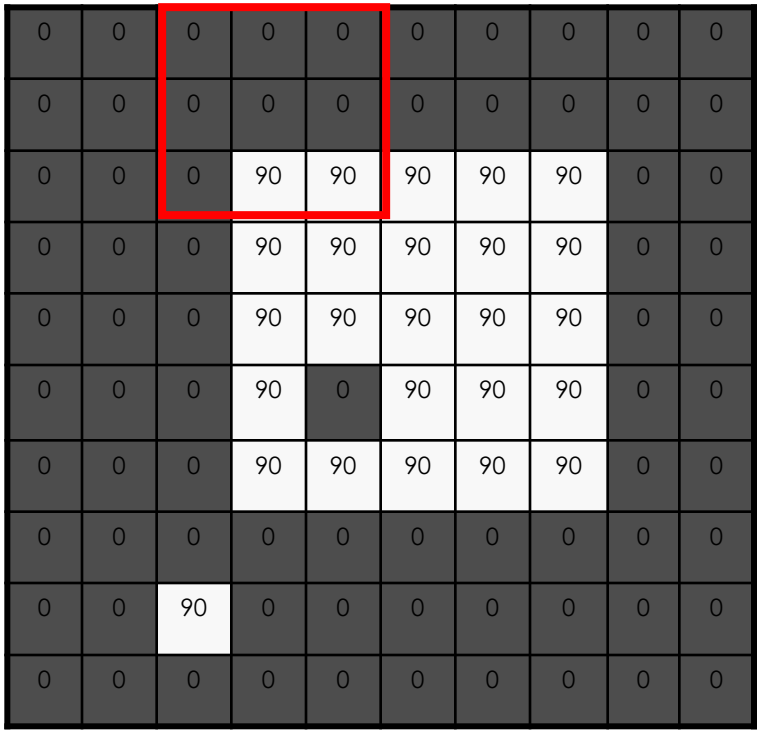


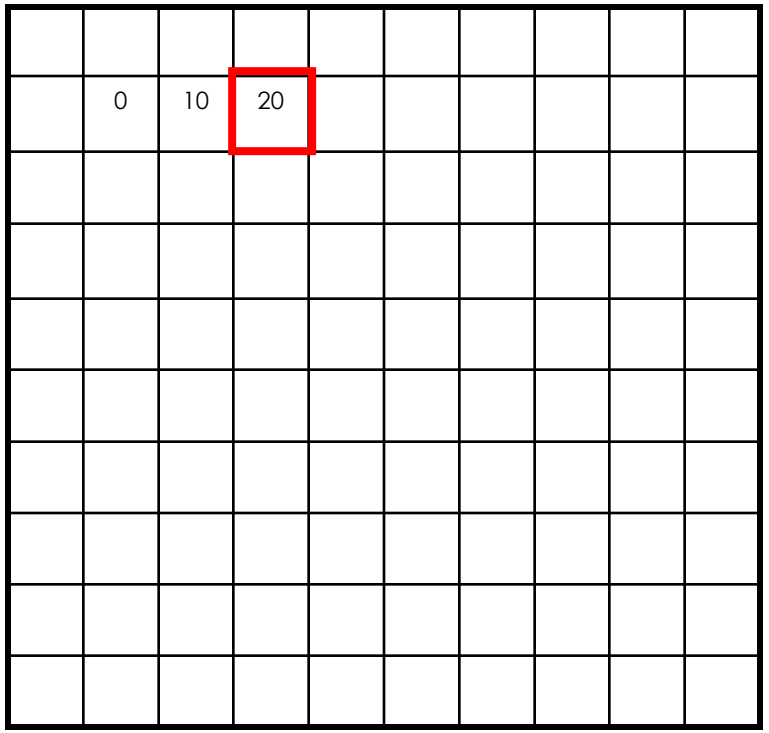
Sliding Averaging Kernel for Convolution



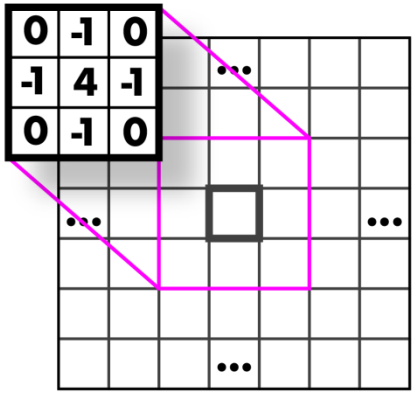


Sliding Averaging Kernel for Convolution



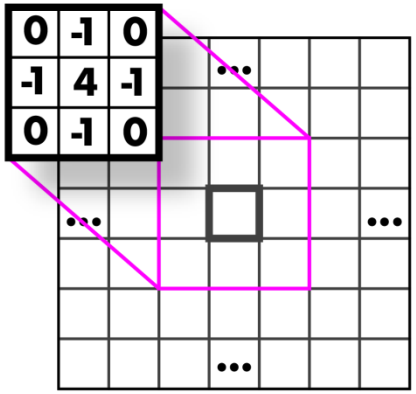


Convolution Effect on An Image



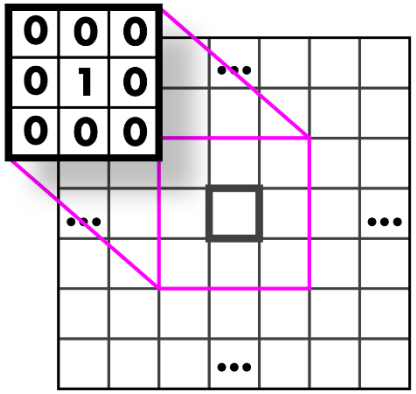
https://docs.google.com/presentation/d/18f0cWwS40jwbP5u37LKvM9ZaHEtRQFUKQ7MCtMdGLmA/edit#slide=id.g3756a210c2_0_1389

High-Pass Kernel: Finds edges



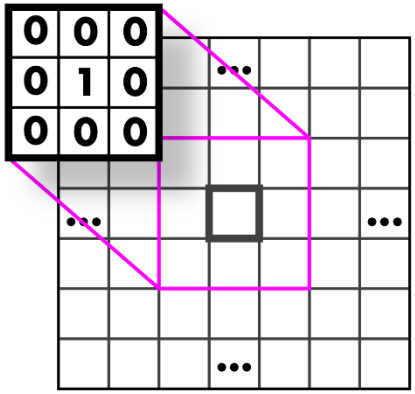
https://docs.google.com/presentation/d/18f0cWwS40jwbP5u37LKvM9ZaHEtRQFUKQ7MCtMdGLmA/edit#slide=id.g3756a210c2_0_1389

Convolution Effect on An Image



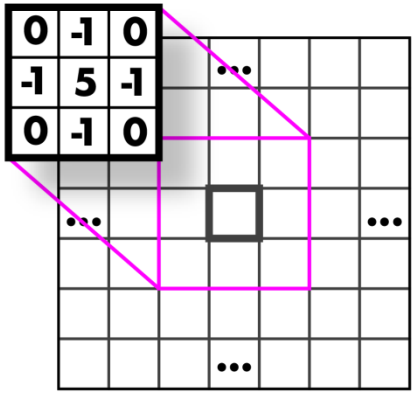
https://docs.google.com/presentation/d/18f0cWwS40jwbP5u37LKvM9ZaHEtRQFUKQ7MCtMdGLmA/edit#slide=id.g3756a210c2_0_1389

Identity Kernel: Does nothing!



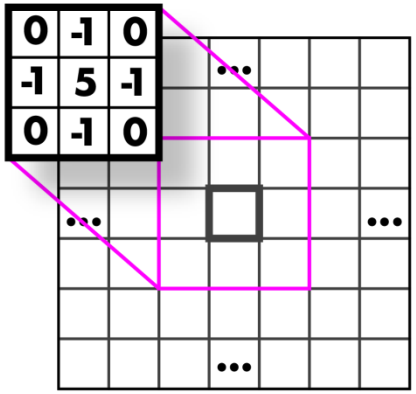
https://docs.google.com/presentation/d/18f0cWwS40jwbP5u37LKvM9ZaHEtRQFUKQ7MCtMdGLmA/edit#slide=id.g3756a210c2_0_1389

Convolution Effect on An Image



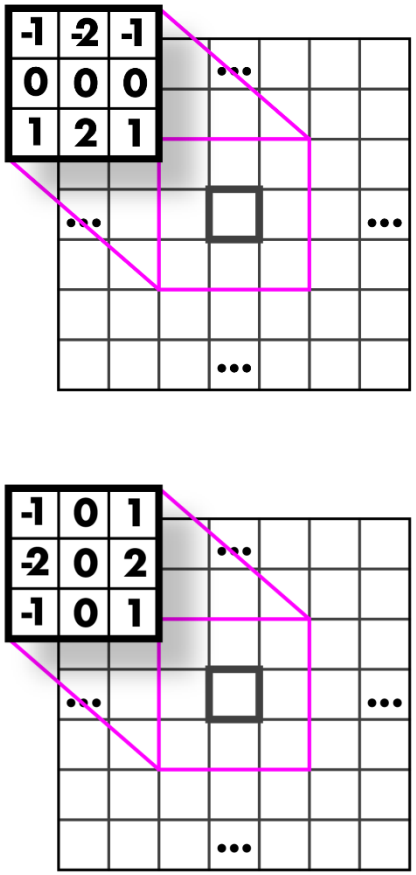
https://docs.google.com/presentation/d/18f0cWwS40jwbP5u37LKvM9ZaHEtRQFUKQ7MCtMdGLmA/edit#slide=id.g3756a210c2_0_1389

Sharpen Kernel: Sharpens!



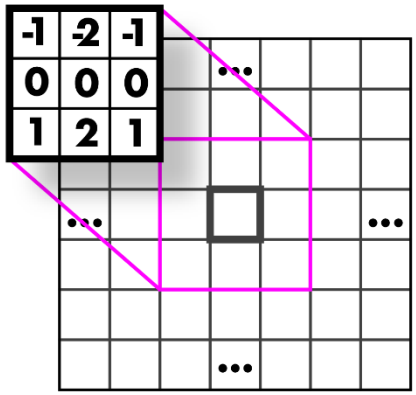
https://docs.google.com/presentation/d/18f0cWwS40jwbP5u37LKvM9ZaHEtRQFUKQ7MCtMdGLmA/edit#slide=id.g3756a210c2_0_1389

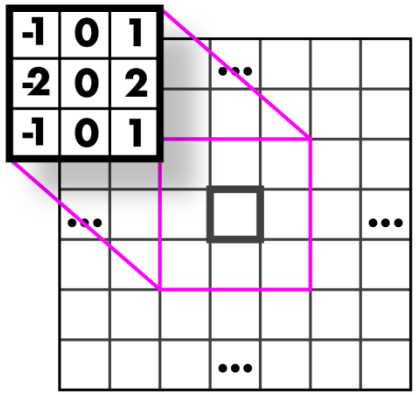
Convolution Effect on An Image



https://docs.google.com/presentation/d/18f0cWwS40jwbP5u37LKvM9ZaHEtRQFUKQ7MCtMdGLmA/edit#slide=id.g3756a210c2_0_1389

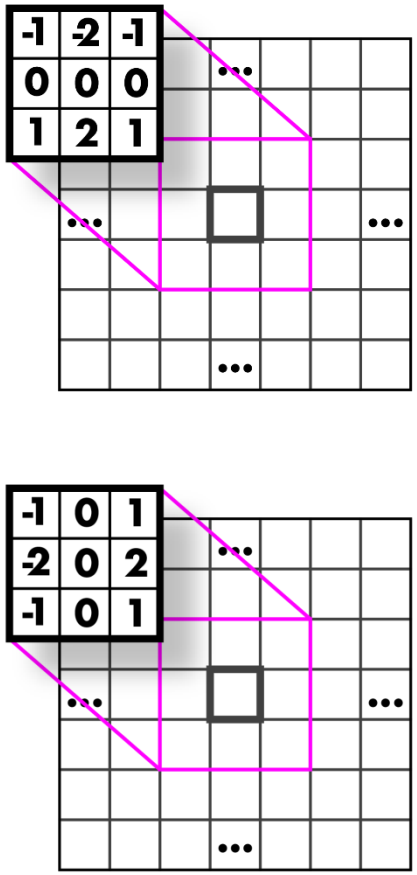
Sobel Kernels: Edges and…

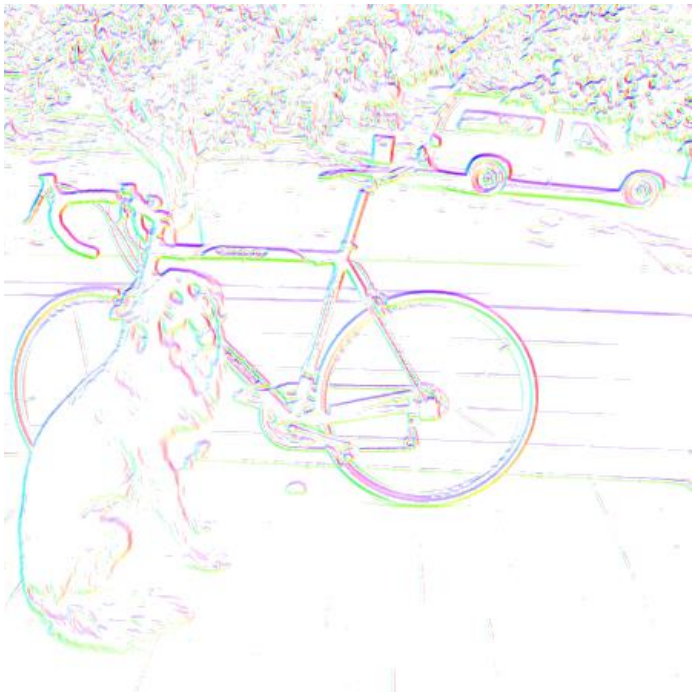




https://docs.google.com/presentation/d/18f0cWwS40jwbP5u37LKvM9ZaHEtRQFUKQ7MCtMdGLmA/edit#slide=id.g3756a210c2_0_1389

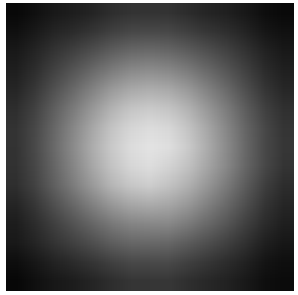
Sobel Kernels: Edges and gradient!



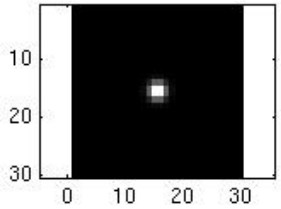


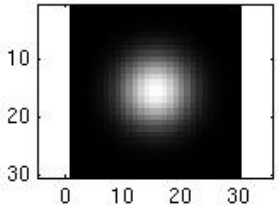
https://docs.google.com/presentation/d/18f0cWwS40jwbP5u37LKvM9ZaHEtRQFUKQ7MCtMdGLmA/edit#slide=id.g3756a210c2_0_1389

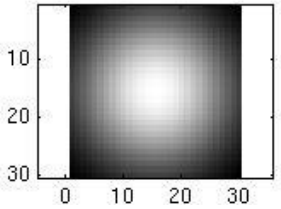
Convolution Effect on an Image



$$\frac{1}{256}\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$$







Convolution Layers

- We can think of the original image as the input layer in a convolutional neural network, and the convolved images as the units in the first hidden layer.
- In a convolution layer, we use a whole bank of filters to pick out a variety of differently-oriented edges and shapes in the image.

Convolution Layers

- Using predefined filters in this way is standard practice in image processing.
- By contrast, with CNNs the filters are *learned* for the specific classification task.
- We can think of the filter weights as the parameters going from an input layer to a hidden layer, with one hidden unit for each pixel in the convolved image.

Convolution Layers

- Since the input image is in color, it has three channels represented by a three-dimensional feature map (array).
- Each channel is a two-dimensional feature map — one for red, one for green, and one for blue.
- A single convolution filter will also have three channels, one per color, each of dimension, say 3×3, with potentially different filter weights.
- The results of the three convolutions are summed to form a two-dimensional output feature map. Note that at this point the color information has been used, and is not passed on to subsequent layers except through its role in the convolution.

Convolution Layers

- If we use K different convolution filters at this first hidden layer, we get K two-dimensional output feature maps, which together are treated as a single three-dimensional feature map.
- We view each of the K output feature maps as a separate channel of information, so now we have K channels in contrast to the three color channels of the original input feature map. The three-dimensional feature map is just like the activations in a hidden layer of a simple neural network, except organized and produced in a spatially structured way.

Convolution Layers

- We typically apply the ReLU activation function to the convolved image.
- This step is sometimes viewed as a separate layer in the convolutional neural network, in which case it is referred to as a *detector layer*.

Pooling Layers

- A *pooling* layer provides a way to condense a large image into a smaller summary image.
- While there are a number of possible ways to perform pooling, the *max pooling* operation summarizes each non-overlapping 2×2 block of pixels in an image using the maximum value in the block.
- This reduces the size of the image by a factor of two in each direction, and it also provides some *location invariance*: i.e., as long as there is a large value in one of the four pixels in the block, the whole block registers as a large value in the reduced image.

Pooling Layers

- Here is a simple example of max pooling:

$$Max\ pool \begin{bmatrix} 1 & 2 & 5 & 3 \\ 3 & 0 & 1 & 2 \\ 2 & 1 & 3 & 4 \\ 1 & 1 & 2 & 0 \end{bmatrix} \rightarrow \begin{bmatrix} 3 & 5 \\ 2 & 4 \end{bmatrix}$$

Architecture of a Convolutional Neural Network

- The number of convolution filters in a convolution layer is akin to the number of units at a particular hidden layer in a fully-connected neural network.
- This number also defines the number of channels in the resulting three-dimensional feature map.
- A pooling layer reduces the first two dimensions of each three-dimensional feature map.
- Deep CNNs have many such layers.

Architecture of a Convolutional Neural Network

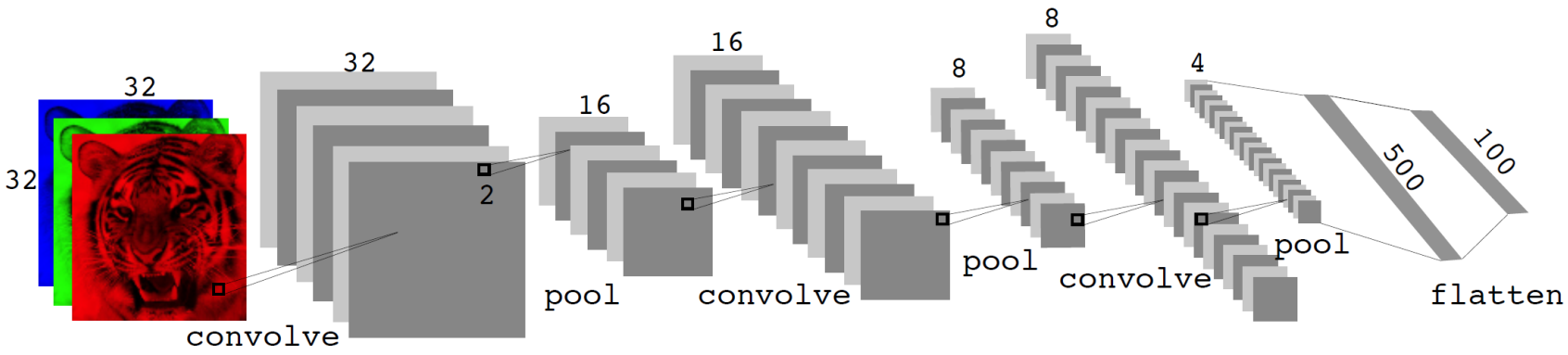


FIGURE 10.8. *Architecture of a deep CNN for the* CIFAR100 *classification task. Convolution layers are interspersed with* 2×2 *max-pool layers, which reduce the size by a factor of 2 in both dimensions.*

Architecture of a Convolutional Neural Network

- Since the channel feature maps are reduced in size after each pool layer, we usually increase the number of filters in the next convolve layer to compensate.
- These convolve-then-pool operations are repeated until the pooling has reduced each channel feature map down to just a few pixels in each dimension.

Architecture of a Convolutional Neural Network

- At this point the three-dimensional feature maps are *flattened* — the pixels are treated as separate units — and fed into one or more fully-connected layers before reaching the output layer, which is a *softmax* activation for the 100 classes.
- There are many tuning parameters to be selected in constructing such a network, apart from the number, nature, and sizes of each layer.

Data Augmentation

- An additional important trick used with image modeling is *data augmentation*.
- Essentially, each training image is replicated many times, with each replicate randomly distorted in a natural way such that human recognition is unaffected.
- Typical distortions are zoom, horizontal and vertical shift, shear, small rotations, and horizontal flips.

Data Augmentation

FIGURE 10.9. *Data augmentation. The original image (leftmost) is distorted in natural ways to produce different images with the same class label. These distortions do not fool humans, and act as a form of regularization when fitting the CNN.*

Data Augmentation

- At face value this is a way of increasing the training set considerably with somewhat different examples, and thus protects against overfitting.
- In fact, we can see this as a form of regularization: we build a cloud of images around each original image, all with the same label. This kind of fattening of the data is similar in spirit to ridge regularization.

Using a Pretrained Classifier

- We use an industry-level pretrained classifier to predict the class of some new images.
- For example, the *resnet50* classifier is a convolutional neural network that was trained using the *imagenet* data set, which consists of millions of images that belong to an ever-growing number of categories.






flamingo		Cooper's hawk		Cooper's hawk	
flamingo	0.83	kite	0.60	fountain	0.35
spoonbill	0.17	great grey owl	0.09	nail	0.12
white stork	0.00	robin	0.06	hook	0.07
Lhasa Apso		**cat**		**Cape weaver**	
Tibetan terrier	0.56	Old English sheepdog	0.82	jacamar	0.28
Lhasa	0.32	Shih-Tzu	0.04	macaw	0.12
cocker spaniel	0.03	Persian cat	0.04	robin	0.12

FIGURE 10.10. *Classification of six photographs using the* `resnet50` *CNN trained on the* `imagenet` *corpus. The table below the images displays the true (intended) label at the top of each panel, and the top three choices of the classifier (out of 100). The numbers are the estimated probabilities for each choice. (A kite is a raptor, but not a hawk.)*