ECE 310 Fall 2023

Lecture 12

Convolution as template matching

Corey Snyder

Learning Objectives

After this lecture, you should be able to:

- Explain how we may use convolution and matched filters to identify patterns in signals.
- Understand how we perform two-dimensional convolution.

Recap from previous lecture

***Note*: this lecture will not be tested on homeworks or exams. This lecture is meant to be a break before we begin Fourier analysis to show an important time-domain application of LTI system models**

1 Identifying patterns in signals

Thus far, we have used convolution and LTI systems to filter and modify signals. Our analysis of LTI systems will soon be guided heavily by the frequency-domain. For example, we will discuss frequency-selective filters and thus only look to the frequency-domain. Our goal here will always to remove undesirable frequencies and retain a subset of frequencies. However, this is only one possible perspective for how LTI systems may be applied to signals. We may instead interpret LTI systems primarily in the time-domain and use a different set of mathematical and intuitive tools. This lecture is meant to provide a highly-relevant example of how time-domain techniques are just as important in signal processing even though we will spend quite a bit more time in the frequency-domain in this course.

Consider the signal pictured in Fig. 1. This signal contains multiple copies of a common pattern given by a triangle or sawtooth shape of width $W = 8$. A common application in signal processing and data science is to identify patterns of interest in a collection of signals. Suppose we would like to identify every location of such a triangular pattern. How can we efficiently accomplish this?

If we know the pattern we are searching for, we can use it as a *template*! We can then drag this template across the signal and see where it best lines up. In other words, we can see at which locations the element-wise product between the template and signal is largest. This is exactly what convolution performs! The only small change we must make is to reverse the template filter since convolution will also flip one of the signals before shifting it along. Figure 2 shows the sawtooth template we should apply to the signal in Fig. 1. Time-reversing this template and convolving will give the result in Fig. 2b. Accounting for the width of the template, each of the peaks in Fig. 2b exactly identifies each of the triangle shapes in the original signal!

1.1 Matched filters

Beyond this course in other signal processing and communications applications, the template signal in the previous example is often referred to as a *matched filter*. Let $h[n]$ be the desired pattern we want to identify

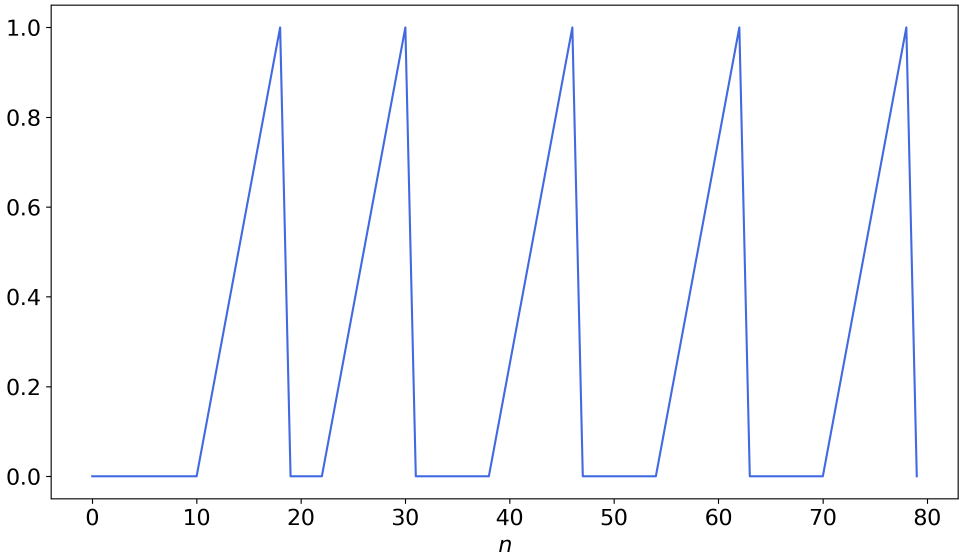
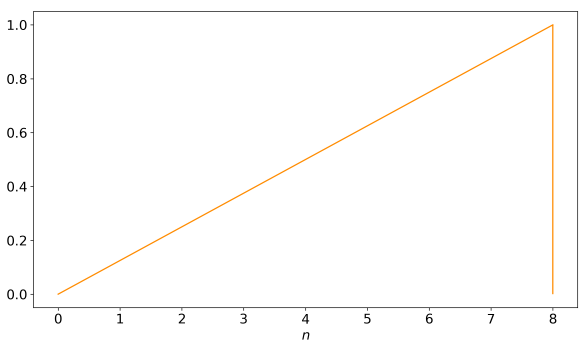
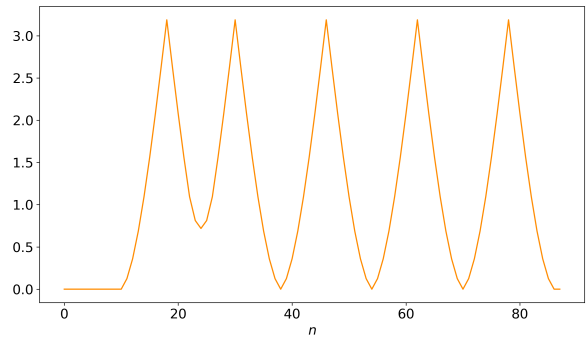


Figure 1: Example signal with replicas of a similar sawtooth/triangular pattern.



(a) Sawtooth template.



(b) Convolution result using reversed sawtooth template.

Figure 2: Example of template and convolution result for identifying sawtooth pattern from Fig. 1.

in a collection of signals. The matched filter $\tilde{h}[n]$ is then the time-reversed version of this pattern

$$\tilde{h}[n] = h[-n]. \tag{1}$$

We can then use the matched filter and convolution to measure the response $y[n]$ of each signal to the pattern of interest:

$$y[n] = x[n] * \tilde{h}[n] = \sum_{k=-\infty}^{\infty} x[k]\tilde{h}[n-k]. \tag{2}$$

Matched filters and the concept of template matching provide a new perspective for convolution in the time-domain. Convolution can be seen as checking the alignment between two signals as we shift one of them over the other. At each shift location, we check the alignment, or the *correlation*, between the signals. The strength of this alignment tells us the correlation between the signals at each location. A more detailed discussion of matched filters, cross-correlation, and auto-correlation are best left to another course on communications or statistical signal processing. We will move forward with this intuition of identifying patterns of interest using template filters. Rather than removing noise in the frequency-domain,

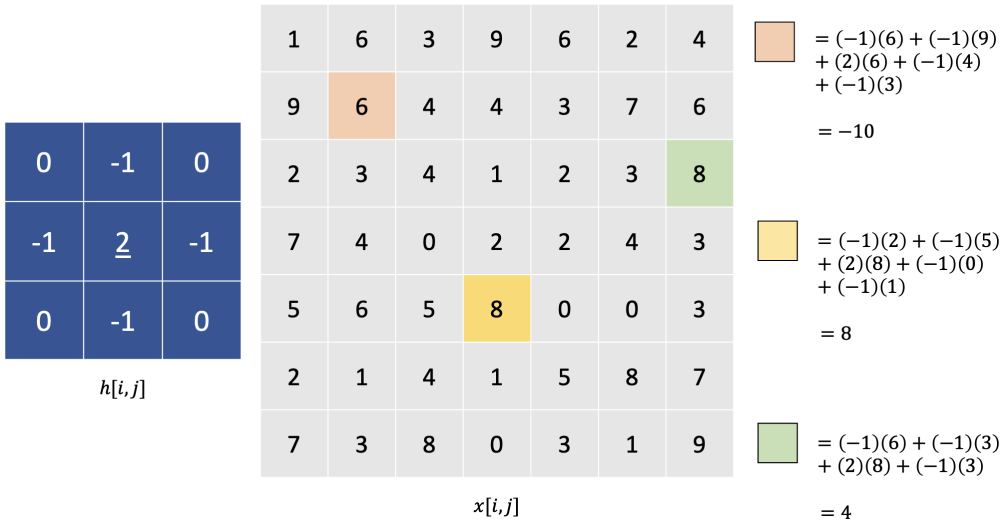


Figure 3: Example of two-dimensional convolution. The underlined sample in $h[i,j]$ is the center sample at $(i,j) = (0,0)$. We assume zero-extension of the image in the far-right example with the light-green sample.

these template filters remove portions of an input signal that do not match well with the intended pattern of the filter.

2 Two-dimensional convolution

Next, we would like to build up our understanding of matched filters and pattern recognition to images. This requires us to define signals and convolution in two dimensions. Let $x \in \mathbb{R}^{H \times W}$ be a two-dimensional signal of dimensions height H by width W, i.e. H rows and W columns. The signal value at row i and column j is then given by $x[i,j]$. We can convolve with a two-dimensional filter or *kernel* with impulse response $h \in \mathbb{R}^{K \times K}$ as follows:

$$y[i,j] = x[i,j] * h[i,j] = \sum_{k=-\infty}^{\infty} \sum_{m=-\infty}^{\infty} x[k,m]h[i-k, j-m]. \tag{3}$$

Note that we have assumed the kernel $h[i,j]$ is square for simplicity. This is often the case in image processing, though we may easily extend to rectangular filter shapes. Looking at Eqn. 3, we see that two-dimensional convolution is effectively the same as one-dimensional convolution since we perform the same flipping and shifting procedure. The only difference is that we flip in two dimensions and must compute shift locations in two dimensions as well. Figure 3 provides an example of two-dimensional convolution with a toy image and filter. We highlight three locations and show the result of computing the convolution at each of these indices. We choose the center sample in $h[i,j]$ to be the center location in the kernel, thus the filter is non-causal. However, this is not an issue in image processing since "future samples" are simply the next row or column in the image where we already know the signal values.

2.1 Two-dimensional pattern matching example

We conclude by extending the notion of matched filters to a two-dimensional example. Figure 4 shows an example image with several copies of three separate shapes: triangles, diamonds, and boxes. Our three template filters should then be examples of each of these shapes as shown in Fig. 5. Convolving each of these templates across the example image then provides the results of Fig. 6. We see in each case that the largest response values, i.e. brightest pixels, correspond to the center of each matching shape in each response image. For example, the diamond and box shape response images exactly pinpoint the center of each shape

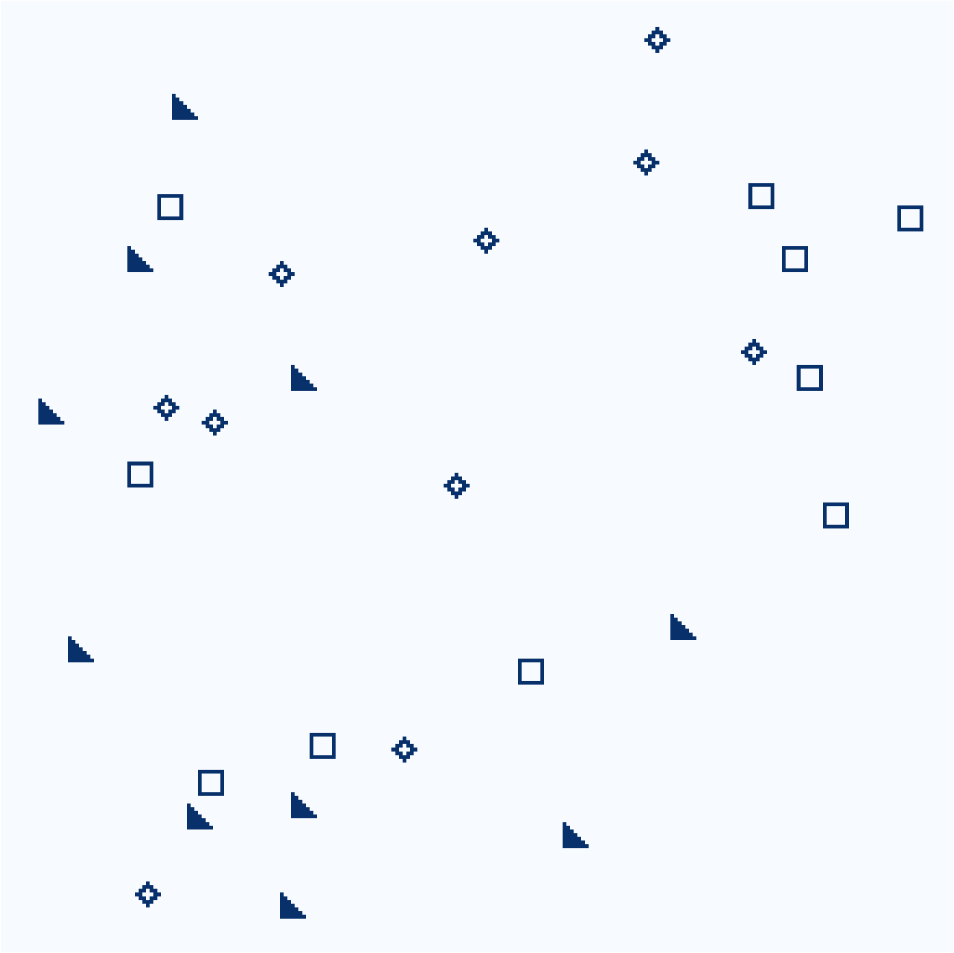
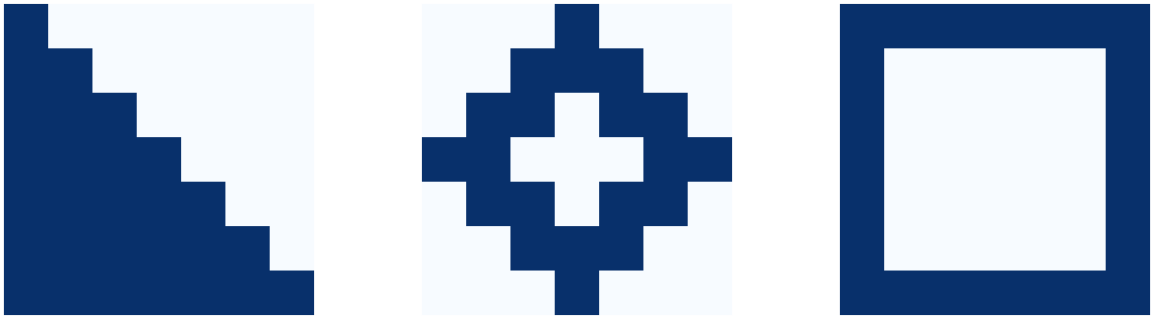


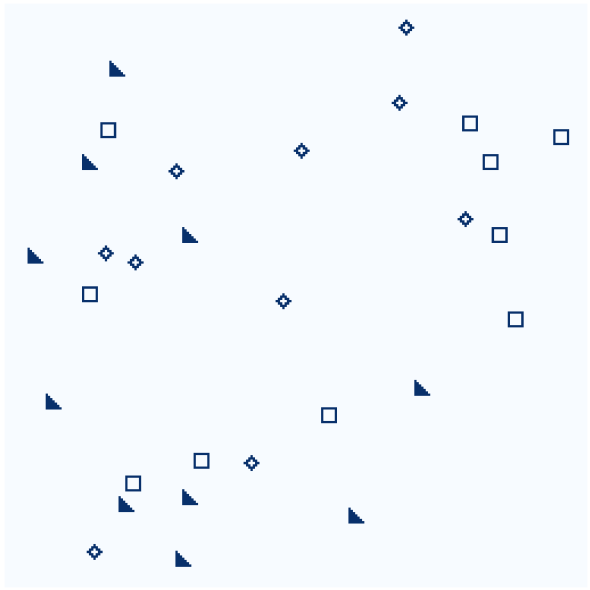
Figure 4: Example image with multiple copies of three distinct shapes.



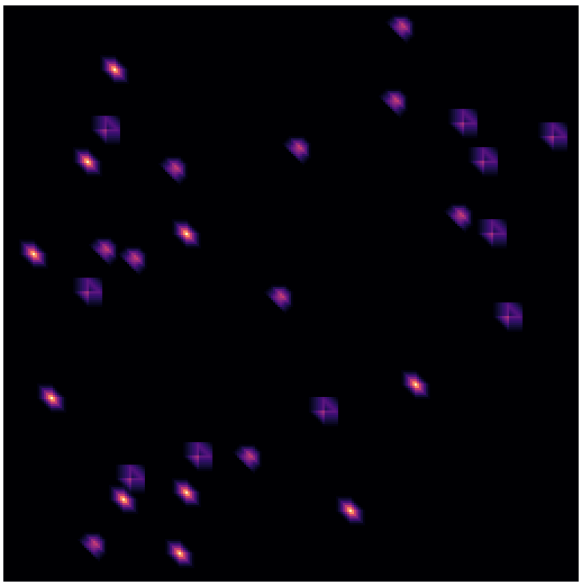
(a) Triangle shape template. (b) Diamond shape template. (c) Box shape template.

Figure 5: Shape templates for two-dimensional pattern matching.

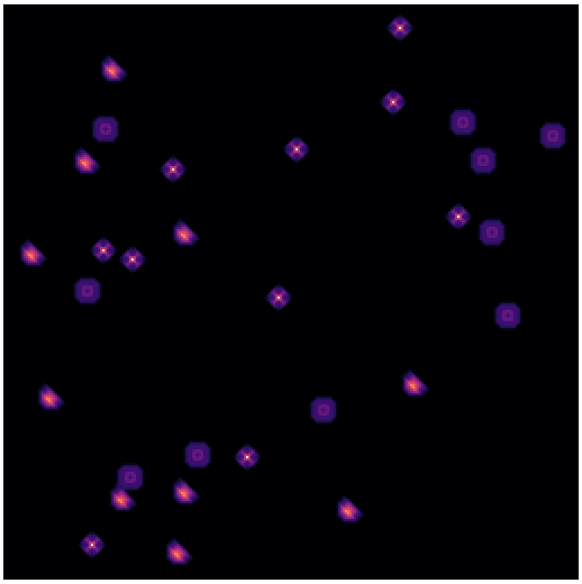
as there is one brightest pixel at each of their centers. The triangles in the diamond shape response are also fairly bright; however, the maximum values are all at the center of the correct diamond shapes.



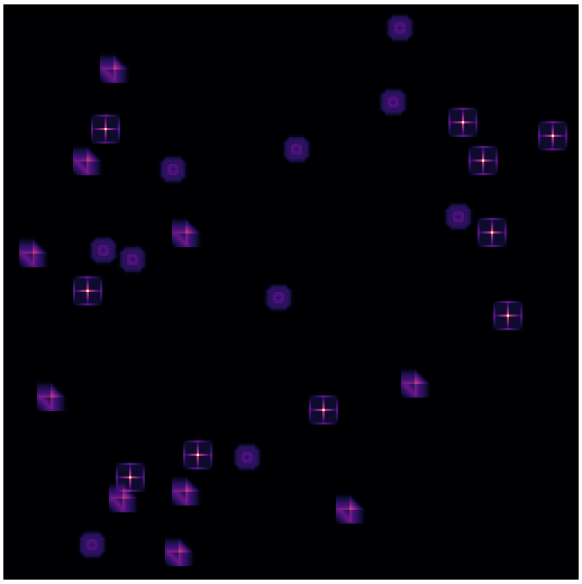
(a) Input image.



(b) Triangle shape response.



(c) Diamond shape response.



(d) Box shape response.

Figure 6: Example convolution results to each of the shape templates. Note how the largest response values, i.e. brightest pixels, correspond to the center of each appropriate shape.

2.2 Highlighting features in handwritten digits

This intuition of convolution as template matching is not restricted to exactly matching a template either. We can also view convolution as a way of highlighting the strength of simple features.

Suppose, for example, we would like to work on the task of handwritten digit identification. For this problem, we take an image of a hand-drawn number and our algorithm must predict which number – 0, 1, 2, etc. – this image is. The most popular dataset for this application is the MNIST handwritten digits dataset[1]. For MNIST, each image is 28×28 pixels and its digit or *class* simply labels the digit the image corresponds to. Figure 7 shows a few example images from the MNIST dataset.

One possible approach would be to take example images of each digit and simply take the inner product between the template image and the input image to "score" the similarity between the image and each possible digit. In other words, multiply the two images element-wise and sum up the result. This approach may work fairly well; however, consider how people have different handwriting and may have unique aspects to the way they write each digit. For example, some people cross their sevens while others do not; some people put a loop in their twos, others put a flat line; and some people have a triangle in their fours, while some people just have vertical lines like in the example from Fig. 7. Furthermore, what if someone writes the digit a little smaller or bigger, or what if they write at more of an angle?

Instead of matching templates of each digit, we can use templates of simple *features* and use the response of each feature in an image to predict the class it belongs to. This way, we can build up evidence of multiple aspects of an image and make a more intelligent decision. Two obvious feature templates would be horizontal and vertical lines. Consider the following two *filter kernels*:

$$h_v = \begin{bmatrix} -2 & 1 & 1 & 1 & -2 \\ -2 & 1 & 1 & 1 & -2 \\ -2 & 1 & \underline{\mathbf{1}} & 1 & -2 \\ -2 & 1 & 1 & 1 & -2 \\ -2 & 1 & 1 & 1 & -2 \end{bmatrix} \tag{4}$$

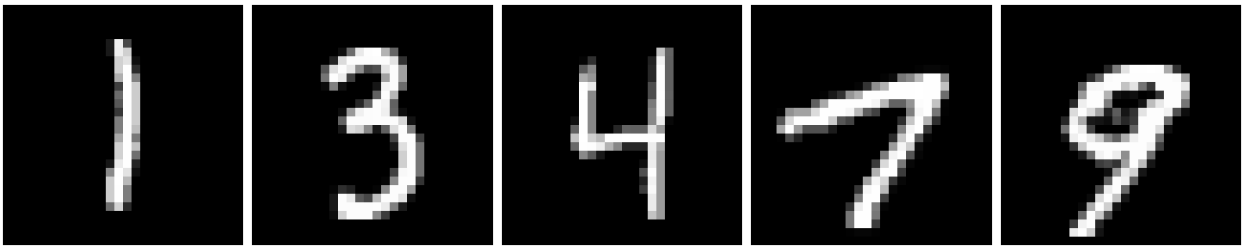
$$h_h = \begin{bmatrix} -2 & -2 & -2 & -2 & -2 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & \underline{\mathbf{1}} & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ -2 & -2 & -2 & -2 & -2 \end{bmatrix}. \tag{5}$$

If we plotted these two kernels as images, they would in fact look like vertical and horizontal lines, respectively. As a technical matter, we use negative numbers on the edges of each line to better define the boundaries of the line. We can think of positive numbers as encouraging higher energy at that location (brighter pixels), negative numbers as punishing higher energy (want darker pixels), and zeros would act like "don't cares" since bright or dark regions are treated the same.

Figures 8 and 9 show the response of each example image from Fig. 7 to the vertical and horizontal filter kernels, respectively. We see, for example, how the digit one example has a response that exclusively corresponds to the vertical kernel, as we should expect. The example of a four also has a strong output to the vertical kernel; however, we also get a nice response to the horizontal kernel at the line that connects the two vertical lines in the digit.

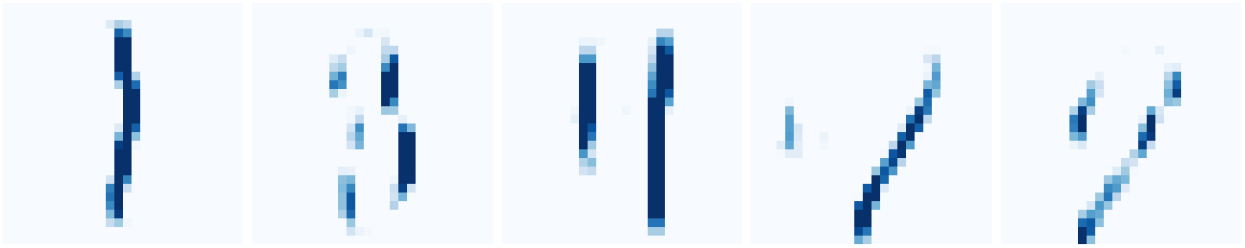
Intuitively, we may see how the amount of response to each kernel separates the digits. For example, both the one and seven have a tall line that is highlighted well by the vertical kernel, but only the seven has some correspondence to the horizontal kernel. There are many other kernels we may design as the foundation of our digit classification algorithm. However, this may be a very difficult task that requires a lot of hand-tuning and experimentation! The modern extension of this idea would be to learn or automatically derive good filters. This is the foundational idea behind *convolutional neural networks* (CNNs), which are the state-of-the-art in many machine learning and AI applications – in particular, computer vision. Discussing CNNs is beyond the scope of this course; however, it is important to note how the signal processing we learn in this class serves as the foundation of modern signal processing, deep learning, and AI techniques!

[1] https://en.wikipedia.org/wiki/MNIST_database



(a) Example 1. (b) Example 3. (c) Example 4. (d) Example 7. (e) Example 9.

Figure 7: Example images from MNIST handwritten digit dataset.



(a) Example 1. (b) Example 3. (c) Example 4. (d) Example 7. (e) Example 9.

Figure 8: Response of example images to vertical kernel. Darker blue pixels have a stronger (higher value) response.

(a) Example 1. (b) Example 3. (c) Example 4. (d) Example 7. (e) Example 9.

Figure 9: Response of example images to horizontal kernel.