Nervous System #29: Teaching Machines to See Faces

BY DAVID KALAT

With the aggressive pace of technological change and the onslaught of news regarding data breaches, cyber-attacks, and technological threats to privacy and security, it is easy to assume these are fundamentally new threats. The pace of technological change is slower than it feels, and many seemingly new categories of threats have been with us longer than we remember.

Nervous System *is a bimonthly blog that approaches issues of data privacy and cyber security from the context of history—to look to the past for clues about how to interpret the present and prepare for the future.*

People use facial recognition systems increasingly to access smartphones and bank accounts, to assist with policing and border crossings, to organize photo libraries, and for other applications. As facial recognition systems become more common, that familiarity can hide that training a computer to recognize a face is a complex computational challenge. Humans take this natural ability for granted (it is a facility so powerful that we can even "see" faces in shadows on Mars or in burn marks on toast). For a computer, however, the task must be reduced to a mathematical process.

Researchers first started trying to teach computers to recognize human faces in the 1960s, but modern facial recognition technology began with a landmark paper published twenty-nine years ago this month. Two researchers at the Massachusetts Institute of Technology turned what had previously depended on manual labor by computer programmers into a mostly automated process.

Matthew Turk and Alex Pentland's "eigenface" method is not about teaching a computer to recognize a *person* as a living three-dimensional being that occupies space. The premise is to simplify the task by approaching it as a two-dimensional project, recognizing a face inside a photographic image. That image is a grid of pixels, each of which has a certain value of brightness or darkness. The entire photograph can be represented by a matrix of data points—each pixel's grid coordinates and luminosity.

For discussion's sake, imagine a collection sample of one hundred distinct images that will be used to train the system. They might represent not a hundred distinct faces, because there may be multiple shots of the same faces, but a hundred different pictures. Each is formatted the same way, with the same dimensions and resolution, so that every pixel in every picture has a correspondent pixel in every other picture.

The next step is to take the average value of each pixel. In other words, for pixel 1, sum up the luminosity of all one hundred variations and divide that by a hundred. Do the same for pixel 2, and so on to the end. The resulting picture is a blurry ghostlike representation of the average of every face in the sample set.

Every actual picture in the sample set can be recreated by taking this ghostly average and applying a series of transformations. This is where something seemingly magical happens. From a machine's point of view, the transformations are just dumb math—but a *human* watching this process unfold would describe the results in an entirely different way. The transformations have the effect of mapping certain facial

features—make the eyes more almond shaped, lengthen the hair, widen the nostrils, make the smile more lopsided—but nothing in the algorithm maps any such thing. It simply happens that the kinds of differences that separate individual faces in the sample from the average tend to correlate to the kinds of things a witness might tell a police sketch artist when trying to reconstruct a given face.

The system compares images against this base set, subtracts the common elements they share, homes in on the distinctive features that make a given image different, and assigns mathematical weights to how a given image compares to the base set. It turns out to be a method that neural networks can learn to do. The approach reduces greatly the processing time to compare a given image against a large database of source images.

These transformations are called "eigenfaces," after the concept of "eigenvectors" in linear algebra. The concept is that certain essential transformations from some idealized norm characterize each individual. Apply the right eigenfaces to the average, and it is possible to restore any of the original samples.

Crucially, though, it is *also* possible to apply a combination of eigenfaces to the average to construct an image that was not part of the starting sample set. If a certain characteristic combination of eigenfaces is needed to restore a *known* facial image, and a substantially similar combination of eigenfaces defines a new image, then there is a mathematical basis to conclude the two images are visually similar.

Early researchers in facial recognition technology had to spend grueling hours hand coding the critical facial features on a batch of photographs to establish the mathematical basis for the machine algorithms used in pattern recognition. Those early experiments were promising, but they depended on humans to identify the key facial features like eyes, nose, and mouth on the base sample set.

By contrast, the eigenface method involves calculations that can be performed quickly and reliably, by a machine that has no concept of "eyes," "nose," or "mouth."

In "Face Recognition Using Eigenfaces," Turk and Pentland condensed the highly complex multidimensional features of a human face into a simple two-dimensional matrix. In addition to allowing for computerized face detection and face recognition, the technology reduces drastically the data storage and transmission needs associated with the facial images. In the three decades since its publication, other technologies have been developed, but many modern face recognition systems still use a version of this technique.

Legaltech® news