Reprinted from the PROCEEDINGS OF THE WESTERN JOINT COMPUTER CONFERENCE
Los Angeles, California, March 1955

PRINTED IN THE U.S.A.

Pattern Recognition and Modern Computers*

O. G. SELFRIDGE†

INTRODUCTION

WE CONSIDER the process we call Pattern Recognition. By this we mean the extraction of the significant features of data from a background of irrelevant detail. What we are interested in is simulating this process on digital computers. We give examples on three levels of complexity corresponding to the subjects of the other three speakers here today. We examine in detail the problem on the second level, visual recognition of simple shapes.

Finally, we show how our attack on that problem can be extended so that the computer is essentially performing a learning process and constructing new concepts on the basis of its experience.

PATTERN RECOGNITION

By pattern recognition we mean the extraction of the significant features from a background of irrelevant detail. We are interested in simulating this on digital computers for several reasons. First, it is the kind of thing that brains seem to do very well. Secondly, it is the kind of thing that computing machines do not do very well yet. Thirdly, it is a productive problem—it leads naturally to studying other processes, such as learning. And, finally, it has many fascinating applications on its own.

We shall not review here the valuable work that has been done and is being done elsewhere.

EXAMPLES OF PATTERN RECOGNITION

Consider Fig. 1. The horizontal lines on the left differ from those on the right in having vertical spikes mostly at the left end. That is, here there are two patterns: those with a preponderance at the left end and those with a preponderance at the right end. The notion of simple preponderance or elemental discrimination is clearly one of the most primitive sources of patterns.

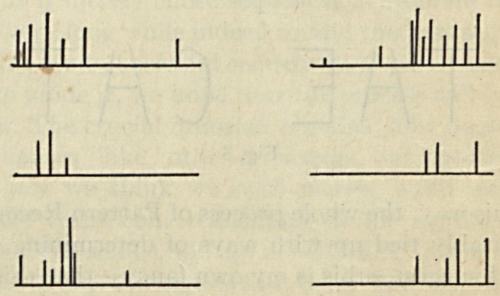

Fig. 1

Here we have filtered each line from perhaps 100 bits down to just one. It is this filtering that is pattern recognition.

Our next example is the visual recognition of simple shapes. This is a two-dimensional problem, of course, while the previous one was merely one-dimensional. Both the shapes in Fig. 2 are clearly squares though (1) they are in different places, (2) they have different sizes, (3) one is hollow, the other not, and (4) they have different orientations.

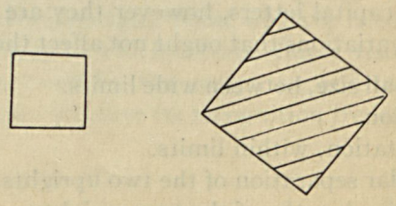

Fig. 2

Our final example, like our first, divides all the configurations of data into two classes. From every chess

* The work in this paper was sponsored jointly by the U. S. Army, U. S. Navy, and U. S. Air Force under contract with M.I.T.

† Lincoln Laboratory, Massachusetts Institute of Technology, Lexington, Mass.

position, that is, from every arrangement of chess pieces on a chess board, it either is or is not possible for White to force a win. Those from which White *can* force a win, we call "good" positions and the rest we call "bad." It is clear that recognition of this kind is a good deal more complicated than in either of the other examples. In itself, it is obviously equivalent to the whole game. For any dunce, playing White, could inevitably win by playing only moves which led to positions which he could recognize as "good."

Significance: The Criterion

As we have seen, pattern recognition involves classifying configurations of data into classes of equivalent significance so that very many different configurations all belong in the same equivalence class. I repeat our definition: Pattern recognition is the extraction of the *significant* features from a background of irrelevant detail.

Probably the most important word here is "significant." Now significance is a function of, first, context, and second, experience. As in Fig. 3, the shape is recognized with the help of its context. Now, of course, context is a function of experience. But more than that, experience alone affects the *kind* of thing we regard as significant.

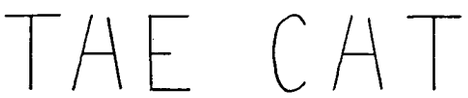

Fig. 3

In this way, the whole process of Pattern Recognition is inevitably tied up with ways of determining significance. I suggest—this is my own fancy—that this is the distinction usually made between machines and men. That men can learn by experience to extract and deal with the *significant* things and machines cannot . . . I do not, however, believe it is a valid distinction.

The other three speakers will talk in detail of pattern recognition on the three levels I have introduced to you. However, I should now like to examine the model used by the next speaker. His problem is the recognition of simple shapes in a visual field, such as block capital letters.

Letter Recognition

The problem is this: We want the computer to recognize block capital letters, however they are drawn. Let us list the variations that ought not affect the valuation:

1. Over-all size, between wide limits.
2. Position.
3. Orientation, within limits.
4. Angular separation of the two uprights.
5. Relative lengths of the two uprights.
6. Thickness, and changes in thickness, of the line segments, within limits.

and so on.

It is clear, I think, that if we tried to construct a template for every A, we should have an embarrassingly large number of them. *Some* filtering of the image has to be done in some way.

Now, people surely use many kinds of features to recognize things. The angularity and size of a letter are instances and they are not related in any obvious way. The features that our model extracts are those derivative from a few simple local image transformations.

Fig. 4 shows a block diagram of our model. The *original image* is transformed into a *secondary image* by operation A, B, or C. The secondary image itself may be transformed a number of times. Then, after the image has been transformed by a *sequence* of operations, the number of blobs remaining is counted by the black box called the Blob Counter. A typical original image is transformed sequentially in Fig. 5, showing the secondary images at each step.

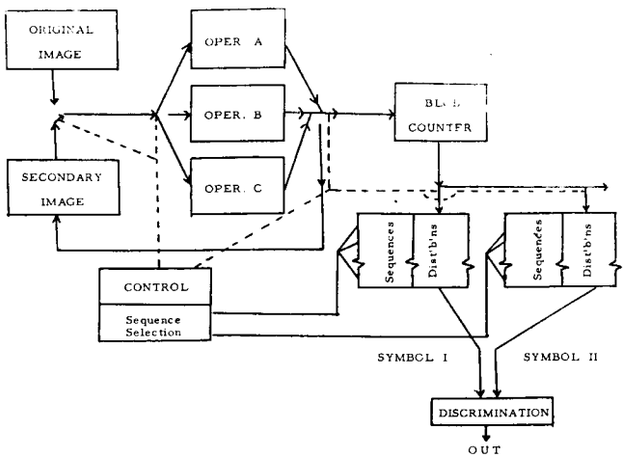

Fig. 4

The number coming out of the blob counter is then compared with the numbers stored for that particular sequence under the various symbols. If, after a number of sequences have been run on an image, the numbers check sufficiently well with the stored distribution of Symbol I, say, the computer may identify the image as Symbol I.

When we were constructing our transformations, we already had in mind some of the tasks they ought to do. First, we wanted to be able to eliminate small pieces of noise, isolated 1's in a field of 0's, say. The averaging operation, our first operation, does approximately this job. This operation emphasizes local homogeneity.

Applying this transformation to an image produces a new image where the intensity at each point is a kind of average of the intensities of the points surrounding it in the original image. Thus solitary 1's in a field of 0's (or vice versa) may be eliminated.

Our second operation emphasizes local discontinuities and tends to bring out edges and corners.

It does this by evaluating the radial asymmetry around each 1 in the image, and keeping only those 1's which have more than a certain amount. That is, our second operation emphasizes local inhomogeneity.

Other operations might well be variations of these, or perhaps something very different. Dr. Dinneen, the next speaker, will discuss the details of programming these operations and the results. He will also go into some extensions, and what kind of things we expect the operations to be able to do.

One of the things we know that they can do is to extract corners; for example, see Fig. 5. After clearing the image of noise with the averaging operation A, and edging twice with operation B, we have four blobs at the four corners of the original square. Now, clearly, having four corners is a significant criterion for squares: not that anything with four corners is a square, but no triangle has four, and no square has three. This sequence can, therefore, successfully distinguish squares from triangles.

Fig. 5

Note that many kinds of features cannot be handled by our particular model, which only counts certain kinds of features. For example, our model could never distinguish a C from a U; the latter might be merely the former rotated through 90 degrees, and all the operations that we have considered so far are unpolarized, that is, do not distinguish much between different directions.

Eventually, we hope to be able to recognize other kinds of features, such as curvature, juxtaposition of singular points (that is, their relative bearings and distances), and so forth.

I shall now discuss our plans for having the computer itself hunt for good sequences and assign the proper values. Every sequence is good or bad according as the numbers obtained from applying it to images tend to differ consistently for different symbols.

Now the search for "good" sequences—that is, those that can discriminate between symbols—can, of course, be programmed to be entirely random. For example, one could start with two sequences and always keep the better one, discarding the old, and picking a new sequence completely at random. But it seemed to us that this was not a realistic way for the computer to improve its performance.

Rather we shall try to have the computer do what I feel we do in such a case. It must build new sequences *like* the sequences that have worked well before. It must begin with sequences chosen at random, because it must have no *a priori* knowledge of the things it sees.

After the sequences have been tested for some time, the computer will have built up a distribution function for each sequence and each symbol.

We give our computer a few programmed operations, A, B, and C, and a random number generator; let it then pick out randomly a few short sequences of operations. It has no way of knowing yet whether these sequences yield any significance. We now feed the machine A's and O's, telling the machine each time which letter it is. Beside each sequence under the two letters, the machine builds up distribution functions from the results of applying the sequences to the image. Now, since the sequences were chosen completely randomly, it may well be that most of the sequences have very flat distribution functions; that is, they have no information, and the sequences are therefore not significant. Let it discard these and pick some others. Sooner or later, however, some sequences will prove significant; that is, their distribution functions will peak up somewhere. What the machine does now is to build up new sequences *like* the significant ones. This is the important point. If it merely chose sequences at random it might take a very long while indeed to find the best sequences. But with some successful sequences, or partly successful ones, to guide it, we hope that the process will be much quicker. The crucial question remains: how do we build up sequences "like" other sequences, but not identical? As of now we think we shall merely build sequences from the transition frequencies of the significant sequences. We shall build up a matrix of transition frequencies from the significant ones, and use these as transition probabilities with which to choose new sequences.

We do not claim that this method is necessarily a very good way of choosing sequences—only that it should do better than not using at all the knowledge of what kind of sequences has worked. It has seemed to us that this is the crucial point of learning.

In general, if one wants a computer to do some job, one must merely specify sufficiently precisely what that job is. In some way the enormous amount of information in experience must be filtered down to the significant or valuable part. This process, which we call pattern recognition, may be more or less complex, as the other speakers illustrate. What we have tried to do is to specify one way in which certain simple visual patterns can be recognized by the computer, and in which the computer may improve its recognition by learning.