字符识别

顾秀烨

Table of Contents

1. 概述

目前，用于车牌字符识别的主要方法有两大类：分类器方法与模式、模板匹配方法。许多车牌识别系统中，会有 Post processing 的步骤，目的是提高识别的准确率。

1.1 分类器

1.1.1 统计、混合分类器

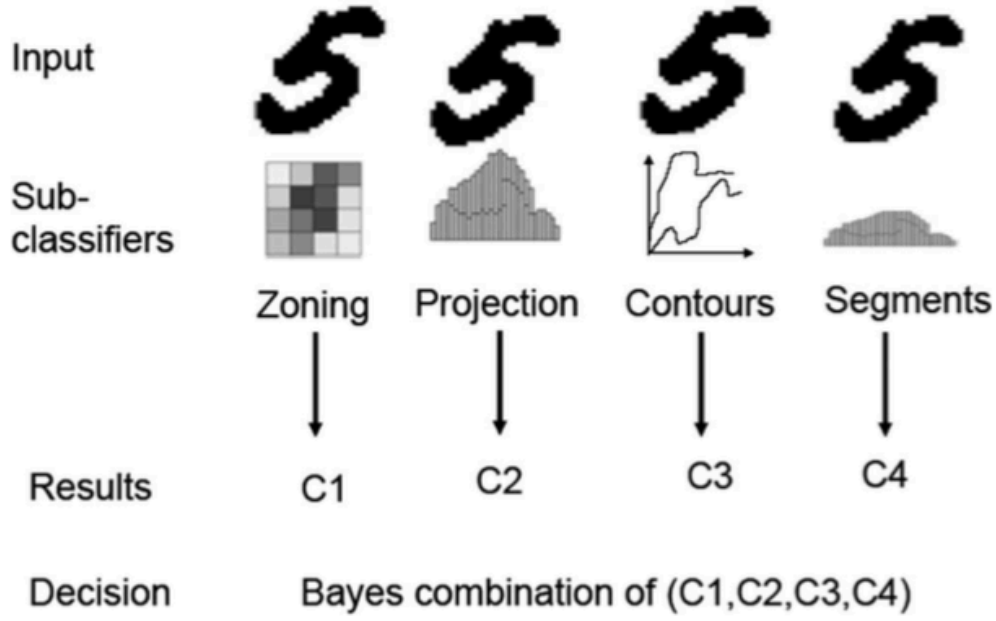
（1）隐式马尔科夫链：需要很复杂的预处理与参数化过程才能达到 95.7%的准确率。从前人的实验结果可以发现，HMM 方法要达到较好的结果需要做很细致的字符分析，所以将 HMM 用于车牌检测系统有一定局限性。

（2）SVM：目前，有许多车牌检测识别系统使用 SVM 方法。
其中，【2】使用了 4 个基于 SVM 的字符识别器，以识别大写字母、大写数字、小写字母、小写数字（韩国车牌）。

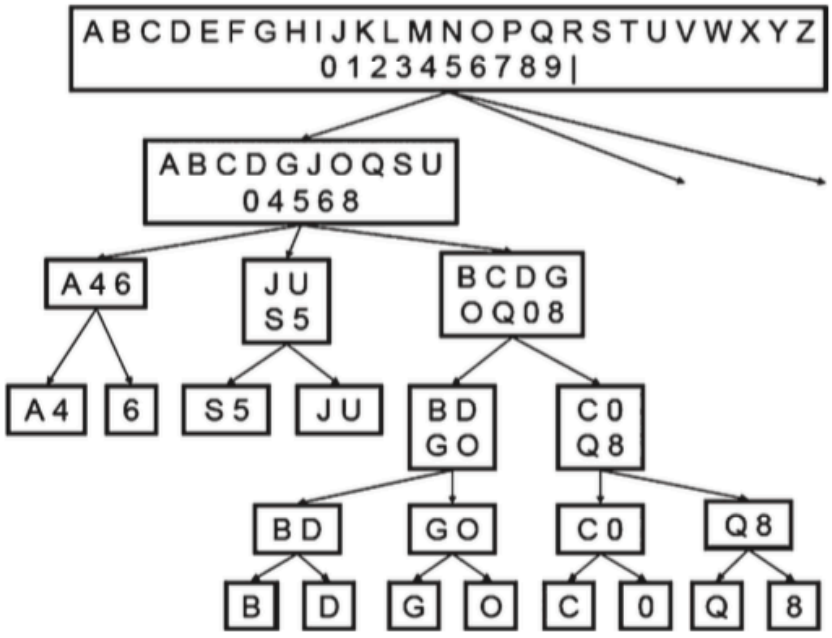
（3）许多研究者融合了两种分类方案，或者使用多阶段的分类方案，或者将多种分类器平行地组合起来。

例如，【3】中提出的将统计与结构识别方法组合起来的两阶段杂交识别系统，以提高鲁棒性并且获得更好的识别表现。在第一阶段，4 个统计到的子分类器（SC1，SC2，

SC3，SC4）独立地识别输入字符。并且，识别结果与贝叶斯方法组合起来。SC1 使用了 zoning density，SC2 使用了垂直投影，SC3 计算轮廓，SC4 计数每一行与每一列的线段数目（如下图）。然后，如果第一阶段的输出结果与预先指定的字符集相似，那么第二阶段（结构化方法）将作为第一阶段的辅助，获得更精确的分类结果。这一方法在超过 10000 张车牌的巨大测试集中获得了 95.41%的成功率。



（4）逐步细化的分类方法也是一种非常有效的方法，尤其是在要分成许多类，或者要系统地挖掘不同类别间的相似特性、挖掘层次化结构时。典型地逐步细化的分类方法如下图所示：



在【4】中，统一大小后的字符图片被划分为互不重叠的 5*5 的块，处理这些块获得边沿特性，根据这些特性使用逐步细化的分类方法进行分类。每对相似的字符被给予了更大的重视。在 520 张车牌中，单个字符识别率达到 99%。

1.1.2 神经网络

多层前向神经网络被广泛地用在车牌字符识别中。典型的训练方法是 BP 算法。其中，循环训练的次数、隐藏层的层数、隐藏层上的神经元个数，一般需要通过试错法来确定。

例如，在 easyPR 中，就使用了 ANN 做字符识别，但是开发者没有详述。

在【5】中，输入神经元为使用 DTCNN 生成的 24 个特征。并且，只有当某个输出神经元的置信水平超过 0.85、其他所有输出神经元的置信水平低于 0.25 时，输入字符才被识别为对应字符。

在这些神经网络方法中，相似字符对的识别、边框部分的识别被予以重视。在训练过程中，难以识别的样本会被训练更多次，在上图（逐步细化的分类方法）中，我们看到边框被作为特殊字符“|”处理。经过这些特殊处理，正确率提升到了 98.2%。

【7】实现了一个基于 ART 理论的非标准神经网络，ART 是一种非监督学习的技术，隐藏层中神经元的个数是动态的，而且保证了学习会收敛。

【8】使用了 self-organized neural networks 来处理噪音、畸形、破损、不完整对车牌字符识别所带来的影响。该方法专注于准确性而增加了计算复杂度和处理时间。使用 1061 张不同视角下的车牌测试集进行测试，准确率为 95.6%。为了克服相似字符对引起的误分类问题，作者预先定义了一个包含所有相似字符对的集合，一旦某未知字符被分类为该集合中的字符，需要进行额外的比较，该比较仅比较相似字符对中不相似的部分。（该方法比较系统完整，在第二节中会详述）

PNN（概率神经网络）也被用于车牌识别系统。PNN 训练起来更快速一些，因为隐藏层神经元只需训练一次，隐藏层的神经元个数由 training patterns 所决定。在【9】的 LPR 系统中，作者使用了两个 PNN，分别用于字母与数字识别。

1.2 模式、模板匹配方法

单一字体、固定大小、没有旋转的字符非常适合使用模式匹配技术。尽管这项技术被更多地用在二值图像中，合适的模板在灰度图像中也会取得很好的结果。

【10】成功使用了模板匹配，通过计算模板 g 在子图 f 上平移产生的 root-mean-square error，进行识别。

在许多系统中，将模板匹配技术与其他方法结合到了一起。

【11】【12】使用了 Hausdorff 距离：
Hausdorff 距离是描述两组点集之间相似程度的一种量度，它是两个点集之间距离的一种定义形式：假设有两组集合 A={a1, ⋯, ap}, B={b1, ⋯, bq}，则这两个点集合之间的 Hausdorff 距离定义为 H(A,B)=max(h(A,B),h(B,A))

其中,

$h(A,B)=\max_{a\in A}\min_{b\in B}\|a-b\|$

$h(B,A)=\max_{b\in B}\min_{a\in A}\|b-a\|$

$\|\cdot\|$ 是点集 A 和 B 点集间的距离范式（如：L2 或 Euclidean 距离）。

Hausdorff 距离是衡量两张二值化图片之间的差异的一种方法。这种方法拥有矩阵的所有数学属性。其识别率与使用神经网络分类器获得的识别率非常相似，但较神经网络稍慢一些。该方法可以作为识别方法的一种辅助手段，如果实时性的要求不是那么强烈的话。

1.3 后期处理

【8】中字符识别的最后一个步骤（预先定义的相似字符集），可以看做是一种后期处理。

另外，openALPR 系统中，也使用了 Post processing 以提高准确率。OCR 识别结果是一系列可能的字符串及其置信度。该系统会衡量低置信率的字符为空白的字符的可能性（低置信率可能不是车牌的一部分）。另外，该系统也会遵循当地车牌的规则，来决定最有可能的车牌字符。例如，被告知是密苏里州的车牌（[char][char][number]-[char][number][char]），置信率最高的三个字符串如下：

. CFOCIG
. CF0CIG
. CF0C1G

根据密苏里州的车牌规则，选择了第三个字符串。

2. 一些系统的具体实现方案

2.1 Real-Time License Plate Recognition on an Embedded DSP-Platform

该论文使用了 RBF 核的支持向量机，核函数：

$$K(x_i, x_j) = e^{-\gamma \|x_i - y_j\|^2} \qquad \gamma > 0.$$

其中，gamma 通过 n-折交叉验证在参数空间上搜索得到。

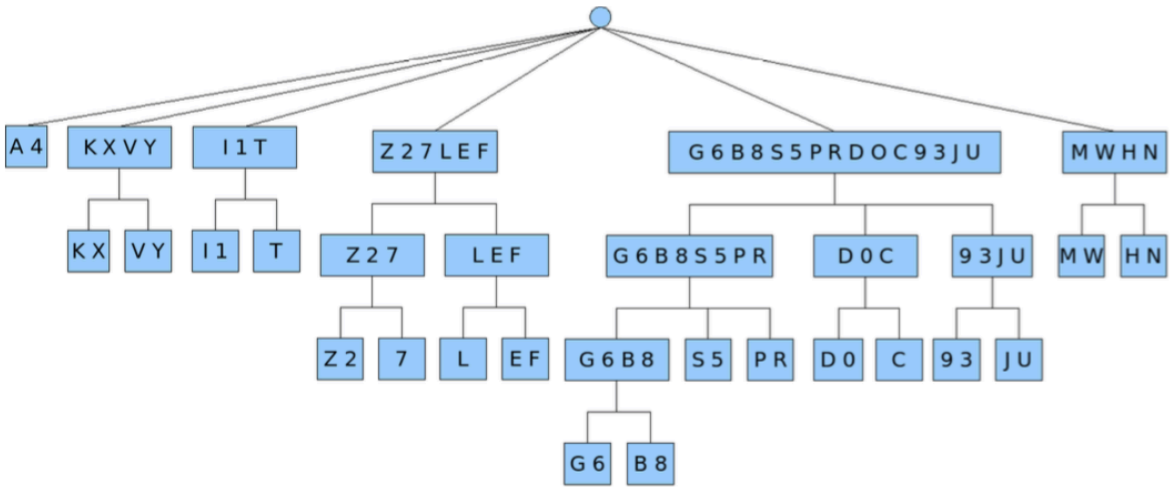
所有分割后的字符区域被缩放到一致的大小。直接使用像素值作为特征向量。作者使用了两种将 SVM 作用域区分多个类别的算法。

2.1.1 One Vs All

假设共有 k 个类别，则训练 k 个二分类的 SVM，只有当训练字符属于当前类别时，标记为正，其余均标记为负。这样，一共 k 个判别函数。选择使得判别函数值最大的判别函数所对应的那一类。

2.1.2 Tree-like structure

类似 1.1.1 中（4）coarse-to-fine 方法。

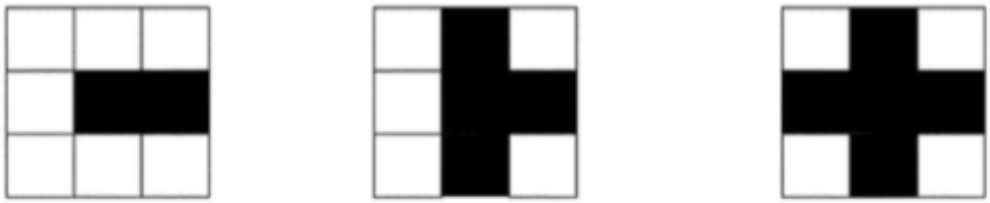


2.2 Automatic License Plate Recognition

这篇论文针对畸形的、破损的、不完整的、有噪音的输入字符，给出了一套可以容忍这些缺陷的字符识别方案。该方案主要由三个步骤组成：字符分类、拓扑分类、self-organizing 识别（SO）。

（1）字符分类
根据车牌字符组成的语义，将字符分为字母与数字。

（2）拓扑分类
在这篇文章中使用的拓扑特征有：字符上洞的个数、end-point、three-way node、four-way node。如下图所示：



Nodal types: (a) end-point, (b) three-way node, and (c) four-way node.

这些特征具有旋转、移动、缩放不变性，而且是定性的，比较容易检测到。以下规则被用于拓扑分类：

一个字符模板与输入字符是相容的，当他们满足以下两个条件：
1）他们的洞的个数之差在[-1, 1]范围内。
2）他们的 end-point、three-way node、four-way node 的个数之差在[-2, 2]范围内。

允许一定的偏差是因为，输入字符可能受到了破坏或是不完整的。
在实验中，拓扑分类的速度很快，可以缩短字符识别所需的时间。

（3）self-organizing（SO） 识别

核心思想是：给定一个未知字符与一个字符模板，使用神经网络中的权重来编码输入字符，字符模板作为一个刺激源，不断地刺激这个神经网络，导致神经网络的权重不断地变化，直到权重逐渐稳定（SO 层形态特征变得跟字符模板比较相似）。将权重的所有变化相加，用于衡量未知字符与字符模板之间的差异。最后将未知字符归入差异最小的那一类。SO 模型如下图所示：

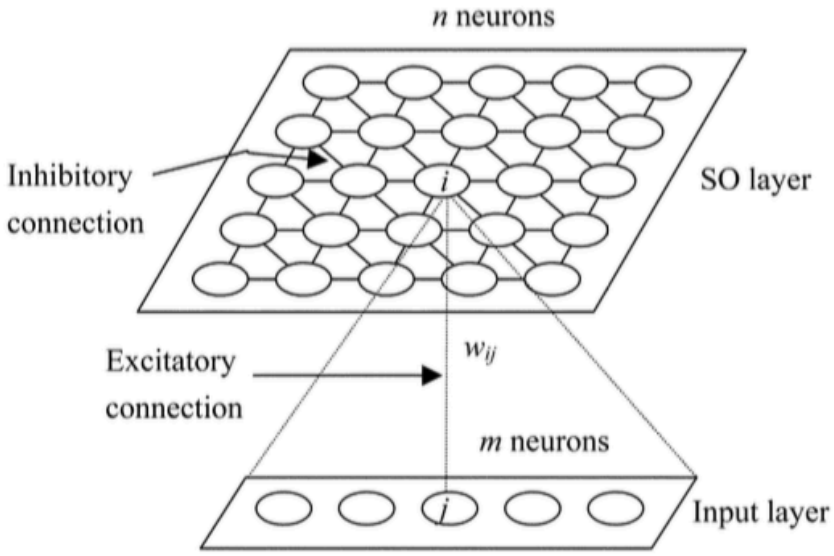


Fig. 5. Kohonen SO neural model.

具体实现：
1）首先，将字符分为 3 类—— 0-hole, 1-hole, 2-hole，根据字符中的洞的个数。每一类都有它自己的 SO 神经网络，他们的 SO 层的布局有些不同，如下图所示。每个 SO 神经网络由 40 个 SO 神经元与两个输入神经元组成。

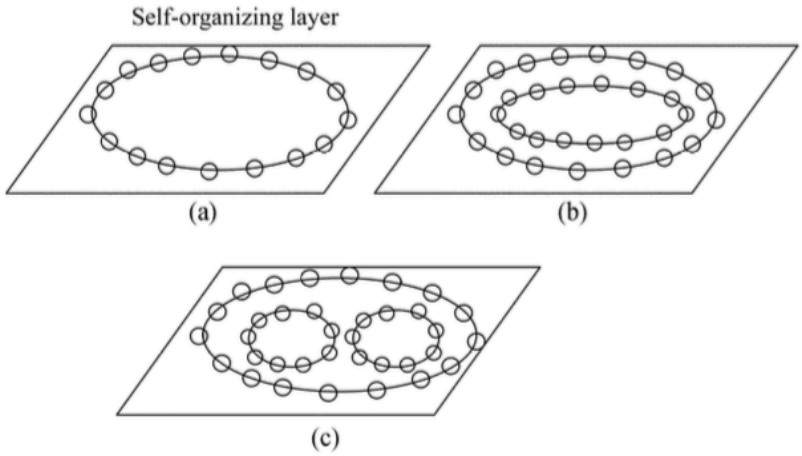


Fig. 6. SO layers: (a) 0-hole, (b) 1-hole, and (c) 2-hole SO layers.

2）输入字符，假设输入的未知字符为“C”，所有的字符都缩放到 16*16 pixels 的大小，与字符模板大小一致。

3）提取未知字符的轮廓，从任意一点开始，将轮廓分成 40 段等间距的小段。提取由 40 个小段的边界点的横纵坐标组成的二维坐标向量。

4）将这 40 个二维坐标向量赋值给 40 个 SO 神经元，作为他们的权重向量。这样以后，以权重向量的两个分量作为坐标轴构成的二维权重空间上，每个 SO 神经元都可以表示为这个空间中的一个点，如下图所示：

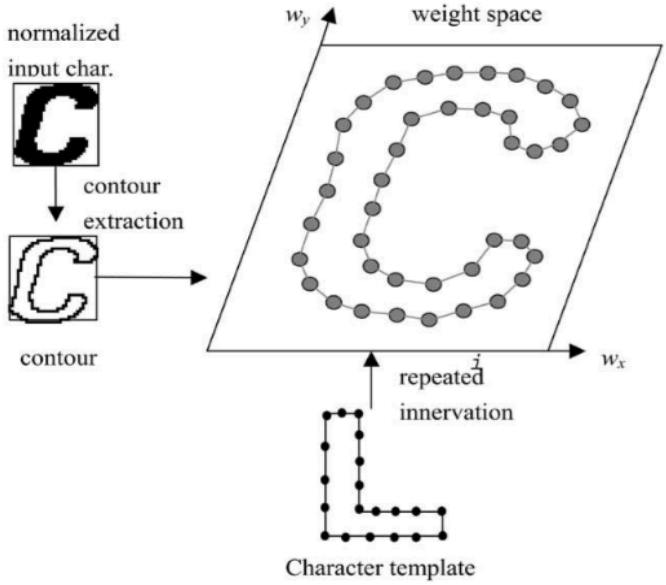


Fig. 7. Example of SO character recognition.

5）选择一个字符模板，假设为”L”，提取字符模板的轮廓，预先在字符模板的轮廓上等间距分段，取分段的端点。轮廓上的点作为刺激源，被一个一个地输入到神经网络中。

假设一个输入的轮廓刺激点为 v，SO 层的神经元为这个刺激物竞争，假设胜利的神经元为 n_c，由公式 $n_c = \arg(\max_{1 \le i \le 40}\{\mathbf{w}_i \cdot \mathbf{v}\})$ 所决定。其中，w_i 为 40 个 SO 神经元的权重向量。

6）n_c 与它最近的两个邻居组成集合 Nc，参与接下来的学习过程（更新集合 Nc 中 SO 神经元的权重）：

$$\Delta \mathbf{w}_k = \mathbf{d}_k g(\mathbf{r}_k - \mathbf{r}_c), \qquad k \epsilon N_c.$$

d_k 为$(v - w_k)$，我们使用弹簧模型计算 d_k，假设 SO 神经元由弹簧相连接。弹簧的弹性系数 $u_{k,k-1}$, $u_{k,k+1}$ 由神经元之间的权重模拟。弹簧模型如下图所示：

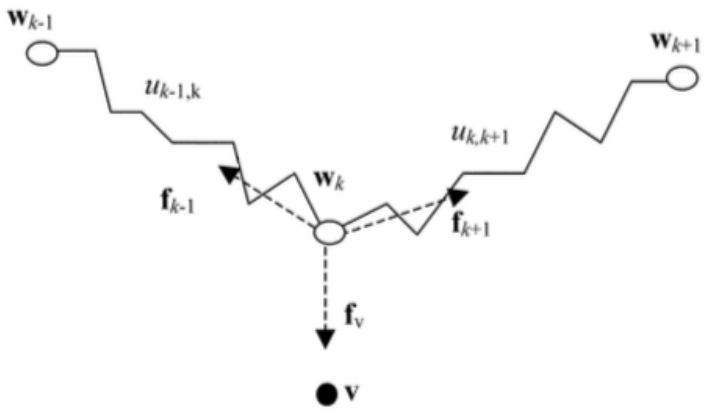


Fig. 8. Spring model.

v 为输入的点刺激源。在弹簧模型中，v 作为一个吸引源。学习过程相当于 v 用大小为 f_v 的力将 w_k 拉向自己 。此时，两段弹簧也对 w_k 作用拉力 f_{k-1}, f_{k+1}。另外，还有一个阻力 f_d（没有画在图上），用来分散弹簧模型中的能量，使得最后在外力 f_e 的作用下整个系统能够收敛到一个平衡态。

整个力学模型的公式为：

$$\mathbf{f}_v + \mathbf{f}_{k-1} + \mathbf{f}_{k+1} + \mathbf{f}_d = \mathbf{f}_e \tag{15}$$

where

$$\mathbf{f}_v = \frac{k_a}{(\|\mathbf{v} - \mathbf{w}_k\| + \varepsilon)^2} \frac{\mathbf{v} - \mathbf{w}_k}{\|\mathbf{v} - \mathbf{w}_k\|}$$

$$\mathbf{f}_{k-1} = u_{k-1,k} \left(\|\mathbf{w}_{k-1} - \mathbf{w}_k\| - l\right) \frac{\mathbf{w}_{k-1} - \mathbf{w}_k}{\|\mathbf{w}_{k-1} - \mathbf{w}_k\|}$$

$$\mathbf{f}_{k+1} = u_{k,k+1} \left(\|\mathbf{w}_k - \mathbf{w}_{k+1}\| - l\right) \frac{\mathbf{w}_k - \mathbf{w}_{k+1}}{\|\mathbf{w}_k - \mathbf{w}_{k+1}\|}$$

$$\mathbf{f}_d = -k_d \|\mathbf{f}_v + \mathbf{f}_{k-1} + \mathbf{f}_{k+1}\|^{1/2} \frac{\mathbf{f}_v + \mathbf{f}_{k-1} + \mathbf{f}_{k+1}}{\|\mathbf{f}_v + \mathbf{f}_{k-1} + \mathbf{f}_{k+1}\|} \tag{16}$$

其中，k_a 为引力系数，k_d 为阻力系数。Epsilon 为一个小的常量，以免在 w_k 接近 v 时 f_v 趋于无穷大。

d_k 可以使用距离公式计算：

$$\mathbf{d}_k = \mathbf{v}_{0k} + \frac{f_e}{m}\Delta t^2.$$

（私以为这里的 dk 应该等于 $d_k = v_{0k}\Delta t + \frac{f_e}{2m}\Delta t^2$ ）

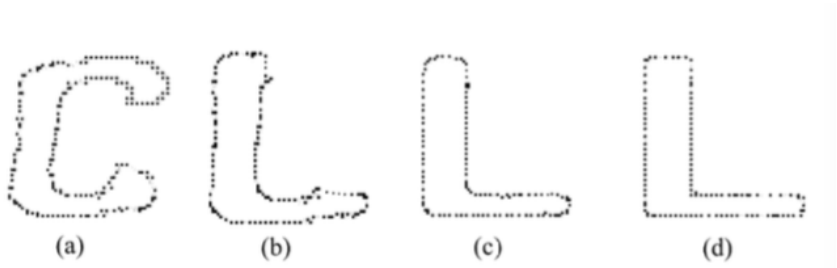
为了简洁性，假设神经元 n_k 的初速度 v_{0k} 等于 0，神经元的质量 m 等于 1，时间间隔也为 1.那么，结果为：

$$\mathbf{d}_k = \mathbf{f}_e = \mathbf{f}_v + \mathbf{f}_{k-1} + \mathbf{f}_{k+1} + \mathbf{f}_d.$$

另外，学习结果需要被学习程度 $g(r_k - r_c)$（学习率受高斯函数 g 的影响）与学习步长 ρ(t)所修改。所以，神经元 n_k 的权重的最终的改变量为：

$$\rho(t)\mathbf{d}_k g(\mathbf{r}_k - \mathbf{r}_c)$$

Nc 中的所有神经元都要经历上述学习过程，权重发生改变，这样以后，点刺激源 v 带来的刺激完成了。对于输入的刺激模板上的灭、每一个点刺激源，在这个例子中是字母“L”轮廓上的点，重复上述过程，则完成了这个模板的一轮循环。重复循环直到 SO 神经元的权重不再有显著的改变。SO 神经元的总的权重位移作为未知字符与字符模板之间的差异。下图展示了未知字符“C”在字符模板“L”的刺激性下，SO 层从”C“的轮廓变形为“L”的一些中间状态：



（4）后期处理

上述过程在识别字符对（8，B）和（O，D）时会有困难，尤其是字符变形时。作者预先定义了一个模糊集合，包含了字符 0，8，B，D。对于集合中的每一个字符，指定了相似字符对之间存在差异的部分，如下图所示：

在字符识别过程中，一旦某未知字符被归类到了模糊集合中的某个字符，需要进行一次额外的小比较，仅比较预先定义好的相似字符之间有差异的部分，以提高识别的准确率。

参考文献

【1】Christos-Nikolaos E. Anagnostopoulos, Ioannis E. Anagnostopoulos, Ioannis D. Psoroulas, Vassili Loumos, and Eleftherios Kayafas. License Plate Recognition From Still Images and Video Sequences: A Survey

【2】K. K. Kim, K. I. Kim, J. B. Kim, and H. J. Kim, “Learning-based approach, for license plate recognition,” in *Proc. IEEE Signal Process. Soc. Workshop, NNs Signal Process.*, 2000, vol. 2, pp. 614–623.

【3】X. Pan, X. Ye, and S. Zhang, “A hybrid method for robust car plate character recognition,” *Eng. Appl. Artif. Intell.*, vol. 18, no. 8, pp. 963– 972, Dec. 2005.

【4】Y. Amit, D. Geman, and X. Fan, “A coarse-to-fine strategy for multi- class shape detection,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 12, pp. 1606–1621, Dec. 2004.

【5】J. A. G. Nijhuis, M. H. ter Brugge, K. A. Helmholt, J. P. W. Pluim, L. Spaanenburg, R. S. Venema, and M. A. Westenberg, “Car license plate recognition with neural networks and fuzzy logic,” in *Proc. IEEE Int. Conf. Neural Netw.*, 1995, vol. 5, pp. 2232–2236.

【6】S. Grossberg, “Competitive learning: From interactive activation to adaptive resonance,” *Cogn. Sci.*, vol. 11, no. 1, pp. 23–63, Jan./ Mar. 1987.

【7】K. B. Kim, S. W. Jang, and C. K. Kim, “Recognition of car license plate by using dynamical thresholding method and enhanced neural networks,” in Computer Analysis of Images and Patterns, vol. 2756, N. Petkov and M. A. Westenberg, Eds. New York: Springer-Verlag, 2003, pp. 309–319.

【8】S.-L. Chang, L.-S. Chen, Y.-C. Chung, and S.-W. Chen, “Automatic license plate recognition,” *IEEE Trans. Intell. Transp. Syst.*, vol. 5, no. 1, pp. 42–53, Mar. 2004.
【9】C. Anagnostopoulos, E. Kayafas, and V. Loumos, “Digital image processing and neural networks for vehicle license plate identifica- tion,” *J. Electr. Eng.*, vol. 1, no. 2, pp. 2–7, 2000. [Online]. Available: http://www.medialab.ntua.gr/people/canag/journals.php
【10】Y.-P. Huang, S.-Y. Lai, and W.-P. Chuang, “A template-based model for license plate recognition,” in *Proc. IEEE Int. Conf. Netw., Sens. Control*, 2004, pp. 737–742.
【11】D. P. Huttenlokcer, G. A. Klanderman, and W. J. Rucklidge, “Comparing images using the Hausdorff distance,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 15, no. 9, pp. 850–863, Sep. 1993.
【12】J. Rucklidge, “Efficiently locating objects using the Hausdorff distance,” *Int. J. Comput. Vis.*, vol. 24, no. 3, pp. 251–270, Sep./Oct. 1997.
【13】Clemens Arth, Florian Limberger, Horst Bischof, “Real-Time License Plate Recognition on an Embedded DSP-Platform”.