中南大學
CENTRAL SOUTH UNIVERSITY

黄品溢个人学习笔记

Title 2022.05.24 学习笔记

Tips

学生姓名 黄品溢

指导教师 王磊

学　　院 计算机学院

目　录

第一章　关于 Transformer

原理图：

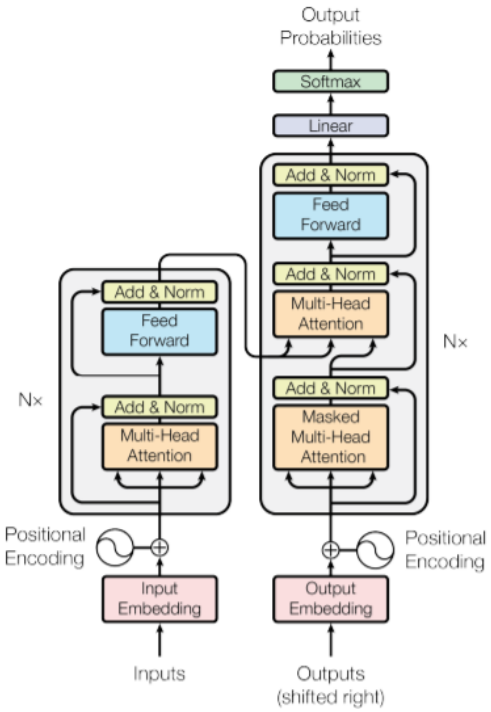


图 1.1　Tranformer 原理图（论文原图）

1.1　Masked Self-Attention 和 Self-Attention 的区别

(1)self-attention 的每个输出都考虑了所有输入

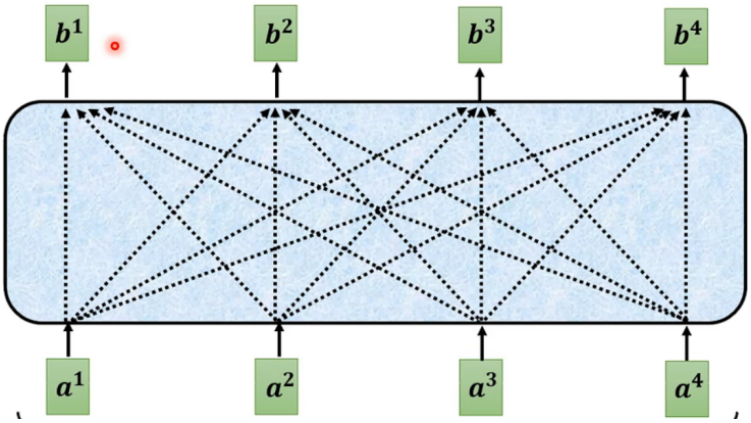


图 1.2　self-attention 并行工作原理图

其具体的实现方法为：

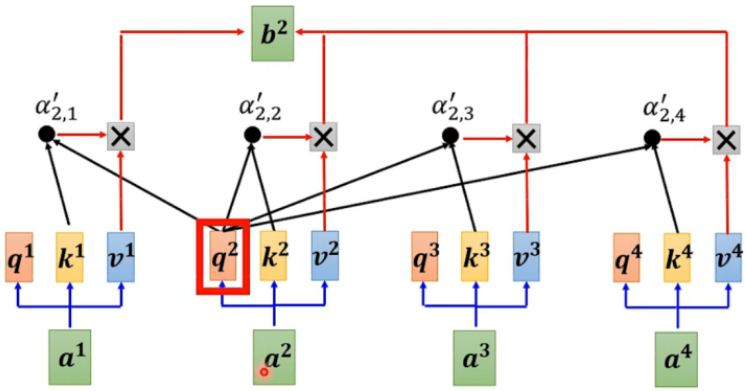


图 1.3　self-attention 具体得到各个输出的示意图

(2)masked 之后，就只能考虑之前的输入（有点类似 RNN，类似产生了时序关系）

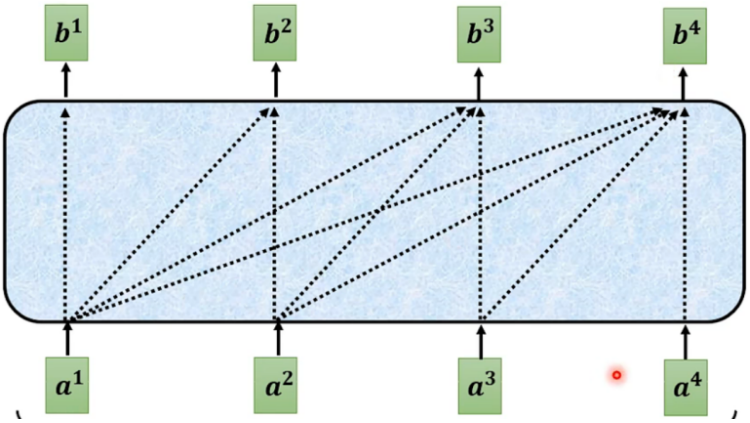


图 1.4　masked self-attention 原理图

mask 之后的具体实现方法为：

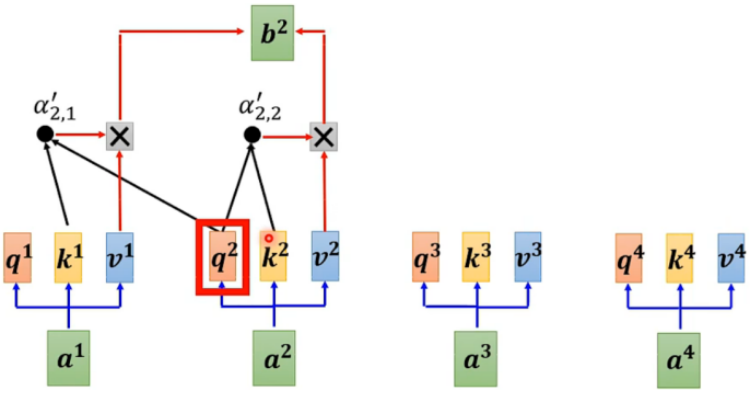


图 1.5　masked self-attention 具体得到各个输出的示意图

其中，b^2 的计算就只与 a^1 和 a^2 的 k^1 和 k^2 有关了，而非全部的 key。

1.2 为什么要使用 Masked？

考虑到 decoder 的工作原理，其输入是一个一个输入、一个一个输出的，当计算 b^2 的时候，a^3 及之后的所有输入都还未到来，因此只能考虑之前的所有输入。

这与 Encoder 当中的 Self-Attention 并行计算并不相同。

第二章　Auto-Regressive 自回归模型 VS Non Auto-Regressive 模型

2.1　两个模型的对比

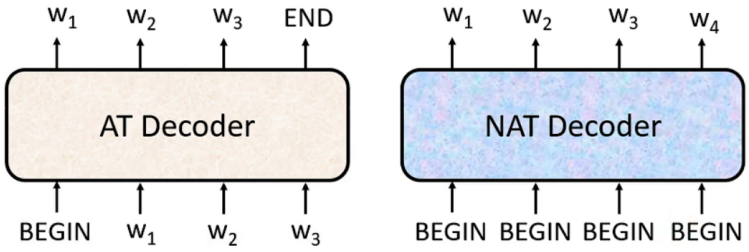


图 2.1　AT 模型和 NAT 模型的输入、输出对比

1. 关于输入方式的不同：
   - AT 模型一次给一个输入，并将该时刻输出作为下一时刻的输入，并以 END 结尾。
   - NAT 模型同时给一定数量的 BEGIN 向量，同时 生成相同数量的输出，结束。
2. 怎样知道应该输出多少个字符？
   - a. 第一种方法是，再添加一个分类器 classifier，输入一个要处理的对象，输出一个数字，该数字就是 NAT Decoder 同时输入的 BEGIN 的数量。
   - b. 第二种方法是，假设保证输出长度不超过 300，就输入 300 个 BEGIN。Decoder 输出后，在其中第一个 END 处截断，只取前面的部分。
3. NAT 模型的优势：
   - 并行计算：同时输入一定数量的 BEGIN，就能同时并行计算得到相应数量的输出；而 AT 模型中只能一个一个计算，速度更快。
   - 它可以控制输出的长度
4. 但是，NAT 模型的性能往往比 AT 模型的性能差。

2.2　Encoder - Decoder 实现细节

Encoder 和 Decoder 交互的部分叫做 Cross Attention，这也是 Tranformer 中最重要的部分。

其中，左边的两个红箭头分别是来自 Encoder 的 *Key*、*Value*（*K*、*V*）矩阵，绿箭头是来自 Decoder 的 *Query* 矩阵。

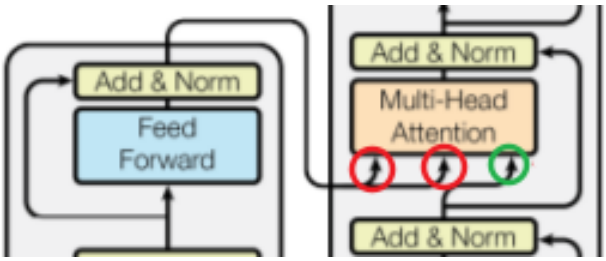


图 2.2　Cross Attention 部分的理解

Attention 的计算公式为：

$$Attention(Q,K,V) = Softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{2.1}$$

使用 CrossEntropyLoss 交叉熵训练，使得总体的交叉熵值越小越好。这里的训练时，可以使用一个 Teacher Forcing（using the ground truth as input）的情况进行训练，即使用正确的输出作为 Decoder 的每个输入。

第三章　关于光流 Optical Flow

3.1　关于光流：光流特点是运动信息

1. 光流在几个假设下起作用：
   - 对象的像素强度在连续帧之间不会改变。
   - 相邻像素具有相似的运动。
2. 稀疏光流：对某些像素点、角或者是边来说进行跟踪

   Lucas - Kanade 光流方法：

   (a) 三个假设：
   - 亮度恒定：像素点随时间变化时，其亮度值（像素灰度值）是恒定不变的。这是光流法的基本设定。（所有光流法都必须满足）
   - 小运动：时间的变化不会引起位置的剧烈变化。
     这样才能利用相邻帧之间的位置变化引起的灰度值变化，去求取灰度对位置的偏导数。（所有光流法都必须满足）
   - 空间一致：即前一帧中相邻像素点在后一帧中也是相邻的。
     这是 LK 光流法独有的假定。因为为了求取 x，y 方向的速度，需要建立多个方程联立求解。而空间一致假设就可以利用邻域 n 个像素点来建立 n 个方程。

   (b) 大致步骤：
   - 首先进行角点检测 cv2.goodFeaturesToTrack()，LK 稀疏光流只能跟踪角点，将所有角点添加至一个列表中，以供后续进行跟踪
   - 前一帧的角点和当前帧的图像作为输入来得到角点在当前帧的位置
   - 当前帧跟踪到的角点及图像和前一帧的图像作为输入来找到前一帧的角点位置
   - 得到角点回溯与前一帧实际角点的位置变化关系
   - 以上一振角点为初始点，当前帧跟踪到的点为终点划线

   (c) LK 金字塔：如当物体运动速度较快时，假设 3 不成立，那么后续的假设就会有较大的偏差，使得最终求出的光流值有较大的误差。因此需要使用 LK 金字塔进行求解。

3. 稠密光流：

   基于前后两帧所有像素点的移动估算算法，其效果要比稀疏光流算法更好。由于要计

算图像上所有点的光流，故计算耗时，速度慢。需要使用某种插值方法在比较容易跟踪的像素之间进行插值，以解决那些运动不明确的像素。

与稀疏光流类似，稠密光流有两个目标：

(a) 使得配准后的两张图像上每个对应点尽可能相同；

(b) 对于纯色区域的点，其光流计算式稍有不同，需要加入一个平滑项。平滑项的目的就是，对于特征比较弱的区域的点，其偏移量尽量向强特征点的偏移量靠近，或者说相邻两个特征点的偏移量相差不能太大，也就是偏移量的变化率不能太大。

由此可得总体的目标函数如下：

$$\min_{u,v} E(u,v) = \int [\varphi(T(x,y) - I(x+u, y+u)) + \lambda \cdot \psi(|\bigtriangledown u|, |\bigtriangledown v|)] dxdy \tag{3.1}$$

前一部分是第一个目标的目标函数，后一部分是第二个目标的目标函数，其中 λ 是权重因子，φ 和 ψ 被称为两个误差函数，其大致可以分为两大类：绝对值函数和二次函数，分别称为 L1-光流（L1-范式）和 L2-光流（L2-范式）。

希望能减少图像中物体的运动速度。一个直观的方法就是，缩小图像的尺寸。假设当图像为 400×400 时，物体速度为 [16, 16], 那么图像缩小为 200×200 时，速度变为 [8, 8]。缩小为 100×100 时，速度减少到 [4, 4]。

所以在源图像缩放了很多以后，原算法又变得适用了。所以光流可以通过生成原图像的金字塔图像，逐层求解，不断精确来求得。简单来说上层金字塔（低分辨率）中的一个像素可以代表下层的两个。每一层的求解结果乘以 2 后加到下一层。

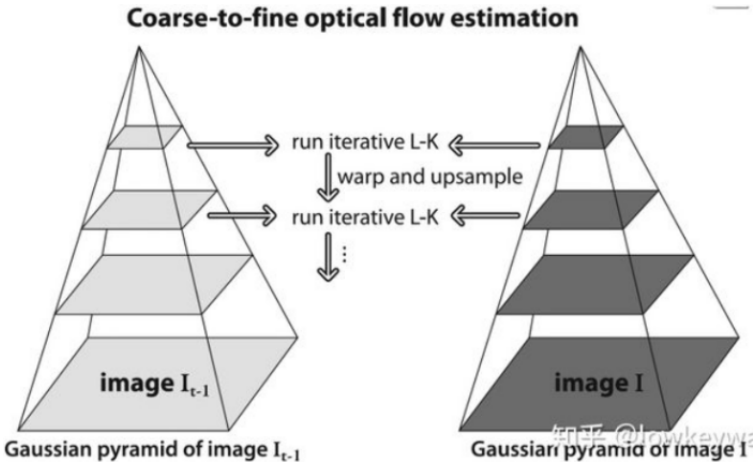


图 3.1　LK 金字塔示例

这就是 calcOpticalFlowPyrLK 中的 maxLevel 参数，图中有四层，则 $maxLevel = 3$，其中下层的宽度和高度均为上层的 2 倍。

3.2 LBP（*Local Binary Pattern*）局部二值模式

3.2.1 Basic LBP

不仅仅是“纹理特征”，因为可以把视频看作 3D 维度，即 x,y,t 三个维度，因此可以在 3D 维度里提取 LBP 特征，即 LBP 也可以包含运动信息。

LBP 特征中的像素值均为**灰度值**，因此需要先转为灰度图像。

原始的 LBP：

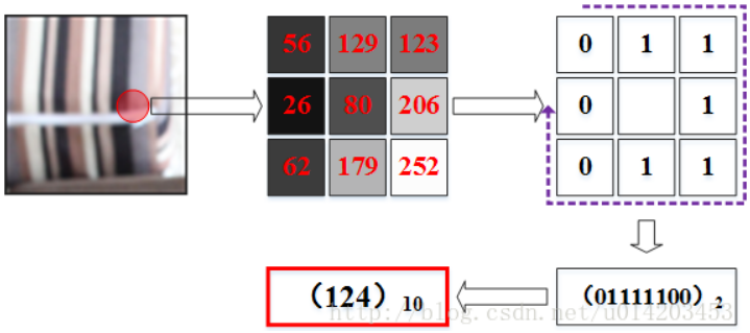


图 3.2 LBP 提取纹理特征过程示意图

每一个图片可以划分为若干个 Block，每一个 Block 内划分为若干个 Cell。在 Cell 内部只用求一个 LBP 值，而非在图片上到处求。

3.2.2 改进后的圆形 LBP

使用符号为 $LBP_{r,p}$ 其含义为半径为 r，总共等距取 p 个点。由于在圆弧上取点，因此坐标不可能全为整数，非整数的点的灰度值一般采用**双线性差值**方法来确定。

3.2.3 旋转不变性 LBP

从 LBP 的定义可以看出，LBP 算子是灰度不变的，但却不是旋转不变的，图像的旋转就会得到不同的 LBP 值。

不断旋转圆形邻域得到一系列初始定义的 LBP 值，取其最小值作为该邻域的 LBP 值。

当一种编码在旋转过程中，能取到的所有值，在旋转不变 LBP 模式下，都取同样的值，如 $(0001)_2 = (0010)_2 = (0100)_2 = (1000)_2 = 1$

如此一来，8 位的 LBP 就可以被编码为 **36** 种不同的取值。

旋转 LBP 模式同样存在缺陷，大量的实验证明 LBP 模式的 36 种情况在一幅图像中分布出现的频率差异较大，得到的效果不是很好。因此人们提出了 uniform LBP。

3.2.4 Uniform LBP，LBP 的等价模式

原始的 LBP 算子，随着邻域内采样点数的增加，二进制模式的种类是急剧增加的。过多的二值模式对于特征的提取以及信息的存取都是不利的。

将 LBP 算子用于人脸识别时，常采用的 LBP 模式的统计直方图来表达人脸信息，而较多的模式种类将使得数据量过大，且直方图过于稀疏。因此，需要对原始 LBP 模式进行降维，使得数据量减少的情况下能最好的代表图像的信息。

3.2.5 LBP 直方图

对于 LBP 直方图来说，横坐标是 LBP 编码值，纵坐标是占比：

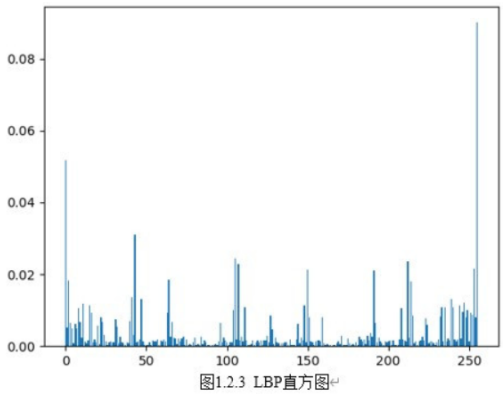


图 3.3 LBP 直方图

LBP 可以得到图像的纹理部分：

首先将图像转为灰度图像，之后求每个非边缘像素点的 LBP 值，并将该像素点的值设为该值，遍历整个图像上的所有点，最后就可以得到一个图像的纹理部分。

3.3 VLBP（*Volume LBP*，体积局部二值模式）

对于 $VLBP_{L,P,R}$ 来说，L 表示时间间隔（如间隔 L 帧取样）、P 表示圆周上等距取 P 个点、R 表示半径为 R。

对于时域来说，在 $t_c - L$ 时刻、t_c 时刻和 $t_c + L$ 三个时刻进行取值。其中 t_c 时刻的圆心表示了图像的整体亮度，对纹理分析没有帮助，则去掉。因此一共有 3P+2 个点。

大致实现方式如下，由外到内表示时间轴顺序：

按照顺序进行 LBP 编码，其权重依次从 2^0 到 2^{3P+1}，乘以 0 或 1 组成一个 3P+2 位的数。因此 VLBP 的直方图横坐标长度为 2^{3P+2}。

其中 0 和 1 是上图中的点依次和 tc 时刻圆心点的灰度值进行对比求得的。

其融合了时间轴上的信息和 LBP 特征，因此能较好地提取动态纹理特征。

其最重要的一点是，由 3P+3 个点的联合分布中提取出了 3P+2 个点的联合分布。

这一步产生了一定的信息损失，但由此实现了灰度不变性。

这一步变换必须是各点相互独立才能实现，而这里进行了相互独立的假设，实际情况可能不是如此，因此会产生一定的信息差（*However, we are willing to accept a possible small loss of information as it allows us to achieve invariance with respect to shifts in gray scale*）。

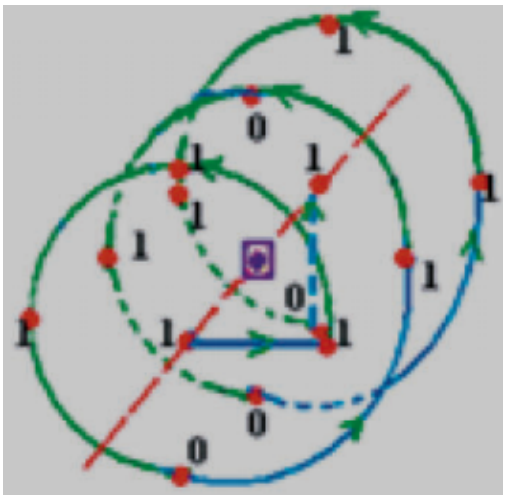


图 3.4　VLBP 取点顺序图

$$V = v(g_{t_c-L,c} - g_{t_c,c},\ g_{t_c-L,0} - g_{t_c,c}, \cdots,$$
$$g_{t_c-L,P-1} - g_{t_c,c},\ g_{t_c,c},\ g_{t_c,0} - g_{t_c,c}, \cdots,$$
$$g_{t_c,P-1} - g_{t_c,c},\ g_{t_c+L,0} - g_{t_c,c}, \cdots,$$
$$g_{t_c+L,P-1} - g_{t_c,c},\ g_{t_c+L,c} - g_{t_c,c}).$$

(a) 各点的灰度值的联合分布原式

$$V \approx v(g_{t_c,c})v(g_{t_c-L,c} - g_{t_c,c},\ g_{t_c-L,0} - g_{t_c,c}, \cdots,$$
$$g_{t_c-L,P-1} - g_{t_c,c},\ g_{t_c,0} - g_{t_c,c}, \cdots, g_{t_c,P-1} - g_{t_c,c},$$
$$g_{t_c+L,0} - g_{t_c,c}, \cdots, g_{t_c+L,P-1} - g_{t_c,c},\ g_{t_c+L,c} - g_{t_c,c}).$$

(b) 进行各点无关假设，并提出后的联合分布式

图 3.5　损失一定信息量实现灰度不变性的原理

由此一来就得到了联合差分分布，就可以采用 LBP 方法提取纹理特征了。

对于特征点数量 P 的选取：若 P 太大，则直方图将过长；若 P 太小，则意味着会损失很多信息。由于数字的增长十分迅速，这一点限制了 VLBP 的扩展和适用性。

同时，当 L 大于 1 时，相邻帧的信息就会被忽略了。

3.4　双线性差值的 OpenCV 计算公式

在图中，已知各点的像素值，求 $x,y \in (0,1)$ 的双线性差值。

其计算公式为：

$$f(P) = f(x,y) \approx \begin{bmatrix} 1-x & x \end{bmatrix} \begin{bmatrix} f(0,0) & f(0,1) \\ f(1,0) & f(1,1) \end{bmatrix} \begin{bmatrix} 1-y \\ y \end{bmatrix} \tag{3.2}$$

或者可以写成：

$$f(P) = f(x,y) \approx (1-x)(1-y)f(0,0) + x(1-y)f(1,0) + (1-x)yf(0,1) + xyf(1,1) \tag{3.3}$$

3.5　LBP-TOP（*LBP from Three orthogonal planes*）

普通的 LBP 只考虑了 XY 轴，因此只能用于单张图片进行纹理特征提取。

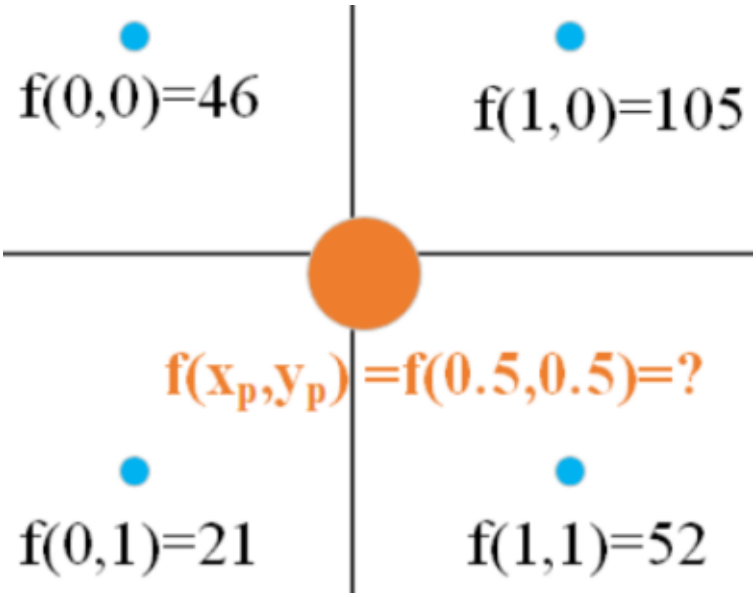


图 3.6　双线性差值计算公式

在 LBP-TOP 中额外考虑了一个随时间变化的 t 轴，由此可以适用于视频片段的纹理特征提取。LBP-TOP 分别考虑了 X-Y、X-T 和 Y-T 轴，在三个面上分别进行 LBP 特征提取，将三个面的直方图直接连接起来，就得到了 LBP-TOP 的直方图。

LBP-TOP 的直方图横坐标 bin 长度为 $2^P \times 3$，其中 2P 是指其编码一共有这么多种可能，3 是指三个正交平面直接相连。

对于 LBP-TOP 来说，T 轴上的半径和 XY 上的半径不宜相同。尤其对于高分辨率、低帧率的图像来说，在 T 轴上相同的时间间隔内，纹理可能发生剧烈变化。因此整体看来，LBP-TOP 的取样是一个椭球。

LBP-TOP 接收六个参数，如：$LBP-TOP_{P_{XY},P_{XT},P_{YT},R_X,R_Y,R_T}(x_{c,t},y_{c,t})$

LBP-TOP 对于面部表情识别来说，最佳的半径选择（x, y, t）是 $\{1,1,2\}$，而在微表情识别来说是 $\{1,1,4\}$。

3.6　LBP-SIP（*LBP with Six Intersection Points*，具有六个相交点的 LBP）

注意，在 LBP-TOP 的计算中，每个邻居像素被使用多次，并且在相交平面集合中只有 6 个不同的邻居点。

LBP-SIP 认为相交的六个点就已经带来了足够描述时空动态纹理的足够信息，因此只需要对这六个点进行特征提取并进行分类就已经足够了。

LBP-SIP 接收四个参数，如：

LBP-SIP 性能优于 LBP-TOP，原因是高纬度的特征通常会受到维度诅咒，即所表示的数据变得越来越稀疏，影响了分类能力。这一点 LBP-SIP 做得更好，其缩减了特征的维数，使得数据更紧凑、计算复杂度更低。

第四章　使用神经网络进行微表情识别

4.1　双时间尺度卷积神经网络（*Dual Time Scaled CNN, DTSCNN*）

对于传统意义上的 CNN 来说，训练一个神经网络需要一个比较大的数据集。而微表情识别这一方面并没有这么大的数据集，因此无法使用传统的 CNN 进行识别。

在现存的微表情数据集上进行训练 CNN，所产生的后果是会带来及其严重的过拟合，这对于微表情识别来说是不能接受的。

而 DTSCNN 则尝试了从三个不同的层面来防止类似问题的发生：

1. 使用两个浅层的神经网络进行特征提取
2. 使用数据增强和更高的 Dropout 比率
3. 使用了 CASME I 和 CASME II 来训练神经网络

为了使神经网络能够获取更高级别的特征，使用神经网络训练的特征是光流，而不是原始数据。

同时，为了解决已存在的微表情数据集过小的问题，DTSCNN 设计了独特的数据增强方法，以获得相对来说足够大的训练集：

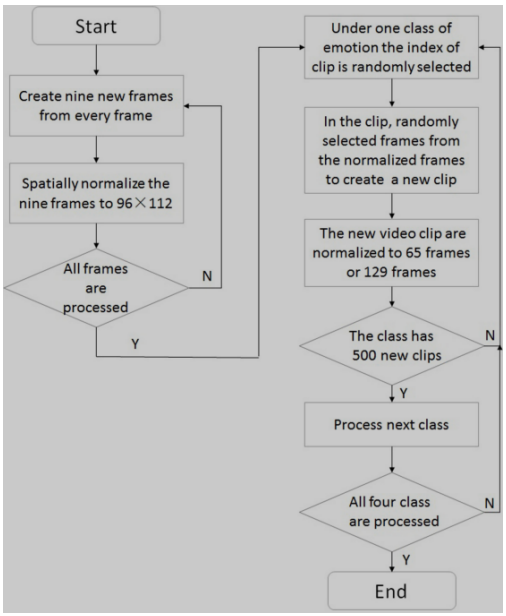


图 4.1　DTSCNN 流程图

同时还将最后 FC 层的 Dropout 比例设置为了 0.5，这也在一定程度上缓解了过拟合问题。

4.2 基于面部图表示学习和面部动作单元融合的微表情识别 *FGRMER*

FGRMER 其全称为：*Micro-Expression Recognition Based on Facial Graph Representation Learning and Facial Actcion Unit Fusion*

4.2.1 FGRMER 的步骤大致为

1. 使用 DConv 将各个 patches 视为许多个通道，学习其中的特征，即点学习。
2. 使用 Encoder of Transformer 来学习点之间的关系，即边学习。由此就学习到了一个面部图表示。
3. 使用 AU-GCN，通过嵌入和 GCN 学习动作单元矩阵。
4. 设计了一个将动作单元矩阵与面部图表示进行结合的融合模型。

4.2.2 FGRMER 相对于其他方法的优势

肌肉移动映射在微表情领域上时，其变化主要是几何变化。相对来说，纹理的变化是非常微弱的，并且容易收到各种其他因素的干扰。

由此来说，基于面部纹理特征的**LBP** 方法不会是微表情方面最好的方法。

MagNet 通过转移学习 (Transfer Learning) 将深度学习引入了微表情增强方面，产生了当时最好的效果。这证明了增强后的几何特征对微表情识别有着十分重要的贡献。

但是 MagNet 随后的操作中，其将二维特征粗鲁的直接降成了一维，这造成了空间信息的损失；同时这个方法也没有考虑到 Action Unit(AU) 的机制。因此这个方法同样具有提升的空间。

关于 AU 特征矩阵的获取，使用到了**MER-GCN** 模型 [20]。

第五章　杂七杂八的知识

5.1　图像金字塔

图像金字塔实际上是一张图片在不同尺度下的集合，即原图的上采样和下采样集合。金字塔的底部是高分辨率图像，而顶部是低分辨率图像。我们将一层一层的图像比喻成金字塔，层级越高，则图像越小，分辨率越低。

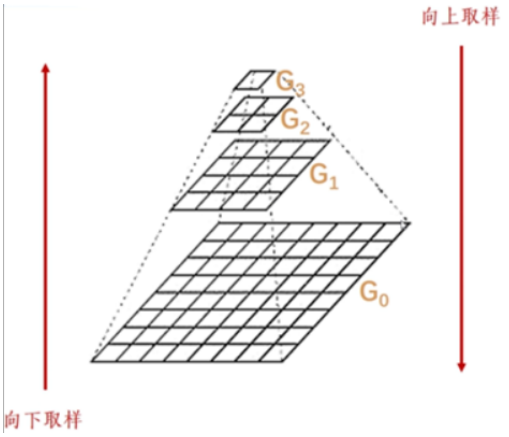


图 5.1　金字塔示例图

1. 高斯金字塔 G_i：

   采用高斯滤波的方法形成的金字塔。

   上采样：将行列扩充为原来的 2 倍，偶数行列置 0，然后采用高斯滤波。

   下采样：先用高斯滤波（高斯核要和上面相同），再去掉偶数行列。

2. 拉普拉斯金字塔：

$$L_i = G_i - PyrUp(PyrDown(G_i)) \tag{5.1}$$

   即拉普拉斯金字塔的每一层是根据高斯金字塔计算得来的，高斯金字塔的每一层减去其先下采样再上采样后的值。

   拉普拉斯金字塔保留的是残差，为图像还原做准备。

5.2　关于矩阵求导

5.2.1　对于标量函数和向量函数

标量函数：输出为标量（一个数）的函数，输入可以为一个向量。

向量函数：输出为向量（或矩阵甚至是张量）的函数，如

$$f(x)=\begin{bmatrix} f_1(x)=x \\ f_2(x)=x^2 \end{bmatrix}, R\to R^2 \tag{5.2}$$

输入：可以是标量、向量、矩阵

输出：可以是标量、向量、矩阵

5.2.2 矩阵求导的**本质**

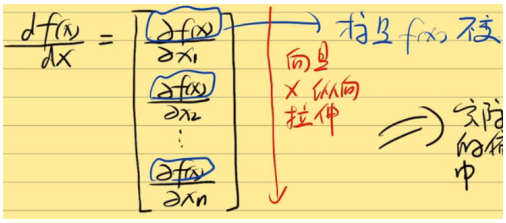
$\frac{dA}{dB}$：矩阵 A 中的每个元素对矩阵 B 中的每个元素求导

若 $A_{p\times q}$ 和 $B_{m\times n}$，则 $(\frac{dA}{dB})_{p\times q\times m\times n}$

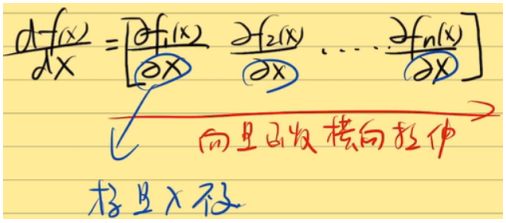
5.2.3 求导方法

标量不变，向量拉伸；前面 $f(x)$ 横向拉，后面 x 纵向拉例子：

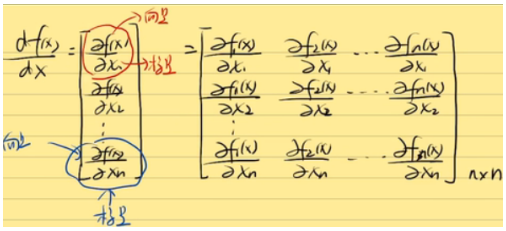
1. $R^n\to R$，即 $x=(x_1,...,x_n)$，$f(x)=f(x_1,...,x_n)$



2. $R\to R^n$，即 $f(x)=[f_1(x),...,f_n(x)]^T$



3. $R^m\to R^n$，即以上两种组合：



5.2.4 二次型求导公式

$$\frac{dx^TAx}{dx}=(A+A^T)x \tag{5.3}$$

第六章　关于相关代码

6.1　cv2.goodFeaturesToTrack()

该函数查找图像或指定图像区域中最突出的角点。

corners = cv.goodFeaturesToTrack(image, maxCorners, qualityLevel, minDistance[, corners[, mask[, blockSize[, useHarrisDetector[, k]]]]])

输入的参数分别是：

- image: 输入图像，是八位的或者 32 位浮点型，单通道图像，所以有时候用灰度图
- maxCorners: 返回最大的角点数，是最有可能的角点数，如果这个参数不大于 0，那么表示没有角点数的限制。
- qualityLevel: 图像角点的最小可接受参数，质量测量值乘以这个参数就是最小特征值，小于这个数的会被抛弃。
- minDistance: 返回的角点之间最小的欧式距离。
- mask: 检测区域。如果图像不是空的 (它需要具有 CV_8UC1 类型和与图像相同的大小)，它指定检测角的区域。
- blockSize: 用于计算每个像素邻域上的导数协变矩阵的平均块的大小。
- useHarrisDetector：选择是否采用 Harris 角点检测，默认是 false。
- k：Harris 检测的自由参数

6.2　cv2.calcOpticalFlowPyrLK()

迭代计算 LK 稀疏光流函数

lk_params = dict(winSize=(15, 15), maxLevel=2, criteria=(cv2.TERM_CRITERIA_EPS | cv2.TERM_CRITERIA_COUNT, 10, 0.03))

p1, st, err = cv2.calcOpticalFlowPyrLK(img0, img1, p0, None, **lk_params)

- p1 表示光流检测后的角点位置；
- st 表示是否是运动的角点；
- err 表示是否出错；
- img0 表示输入前一帧图片；
- img1 表示后一帧图片；
- p0 表示需要检测的角点；
- lk_params：

- winSize 表示选择多少个点进行 u 和 v 的求解；
- maxLevel 表示空间金字塔的层数；