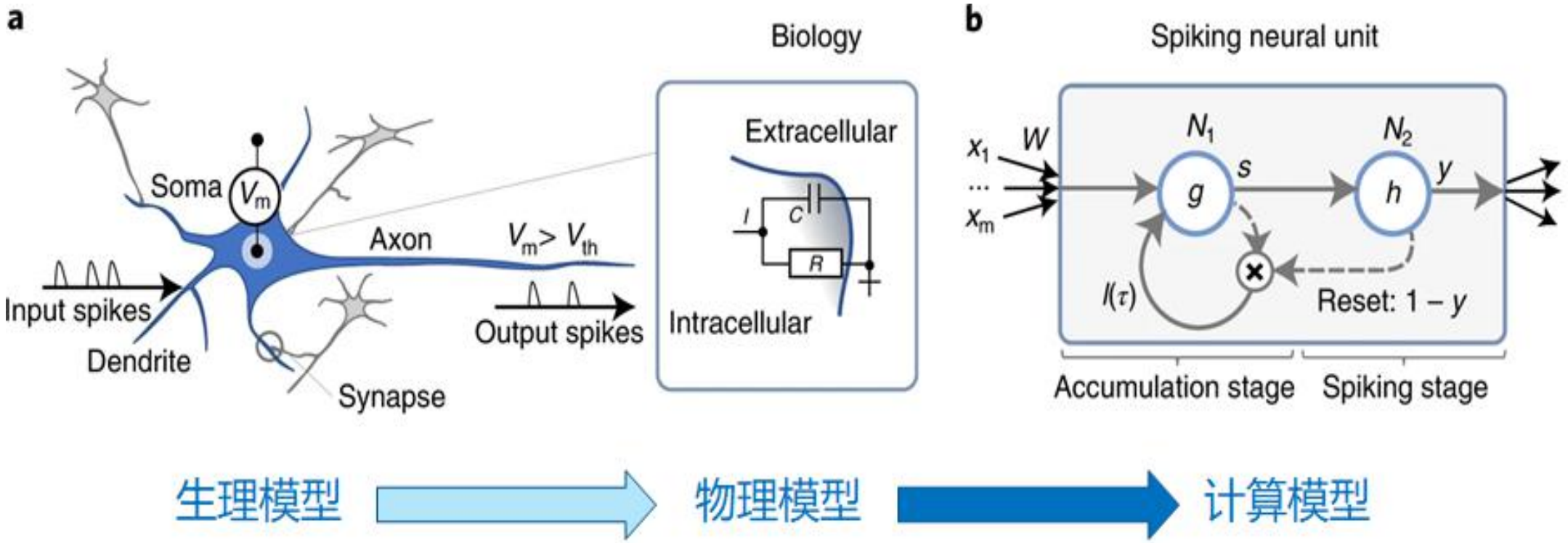
# 脉冲神经网路(Spike Neural Network)

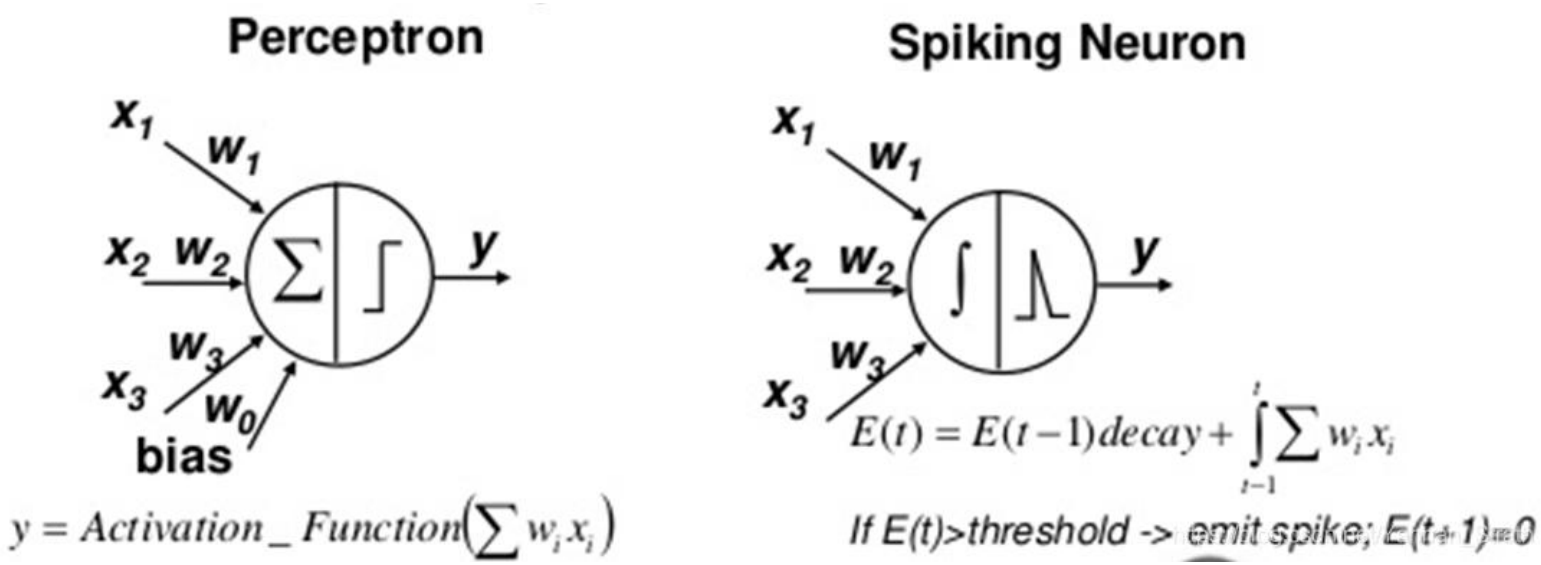
# 脉冲神经网路(Spike Neural Network, SNN)




生理模型 → 物理模型 → 计算模型

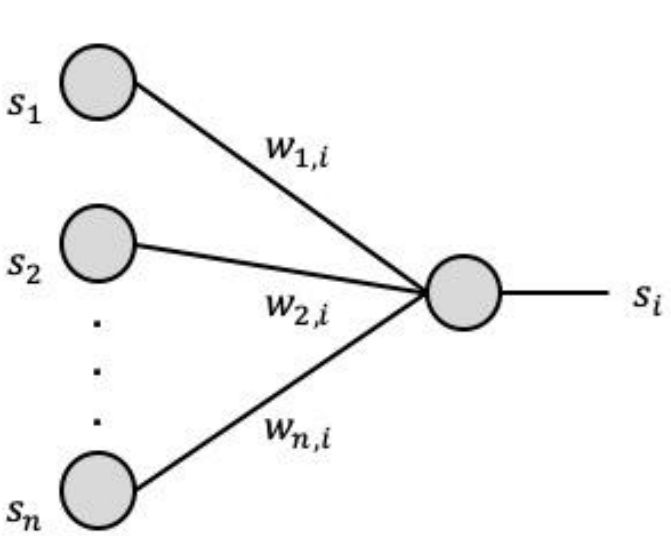
# SNN神经元

- 在SNN中， 很重要的一点是引入了时序（temporal）相关的处理形式。
- 一般使用**合放（integrate-and-fire）脉冲神经元**，通过脉冲交换信息
- 信息是被编码在**脉冲序列的时间序列（spike train）**中的。
- 例如：
  - 高频率的一组脉冲序列可以代表一个较高的值而低频率的脉冲则代表低值。 在一个固定的时间窗中，单个脉冲出现的位置也可以代表相应的值/信息。

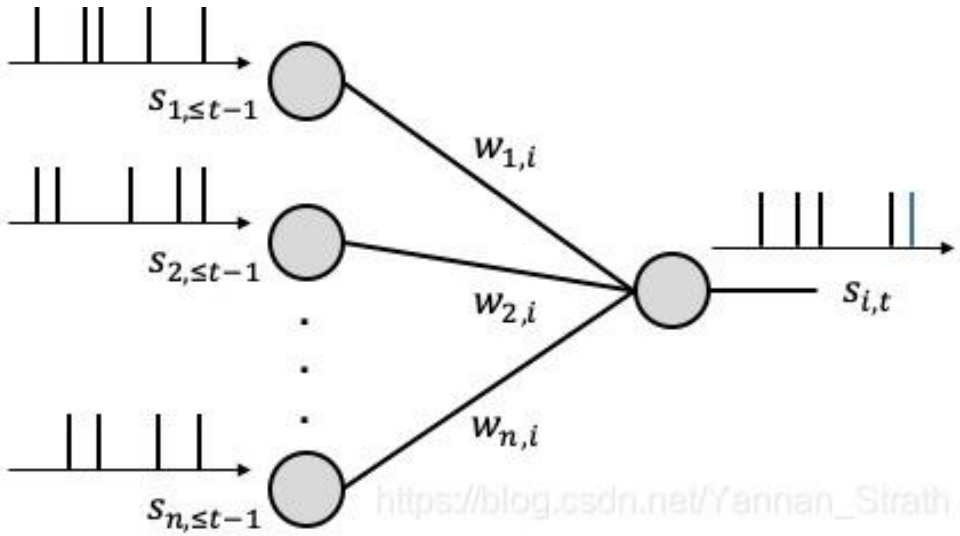


# SNN的信息载体（information carrier）

- ANN 使用的是高精度浮点数而SNN使用的是spikes 或可以理解为1和0，SNN增加了信息在网络中的稀疏性。
- 这些spike在网络中有相同的幅度和duration.



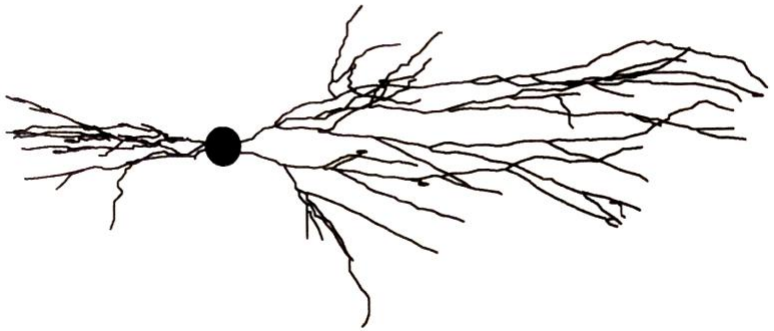

ANN




SNN

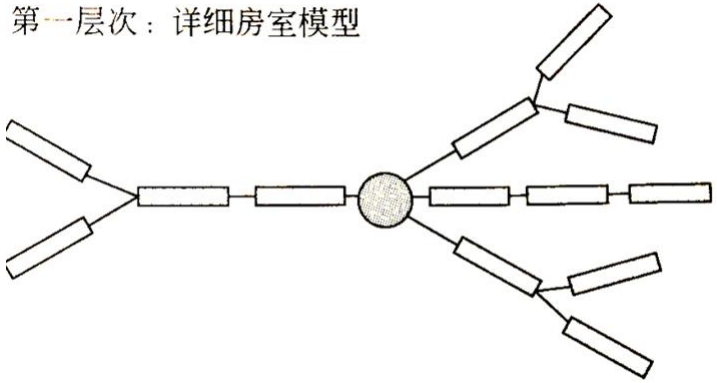
# 脉冲神经元的建模层次

生物神经元的几何形态

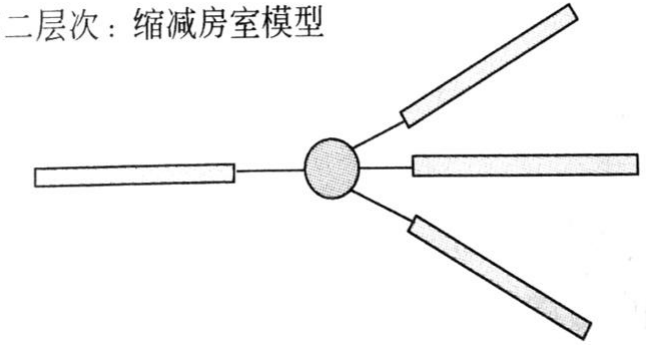


建模层次: 详细房室模型（detailed compartmental model）、缩减房室模型（reduced compartmental model）和单房室模型（single compartmental model）。

第一层次：详细房室模型



第二层次：缩减房室模型

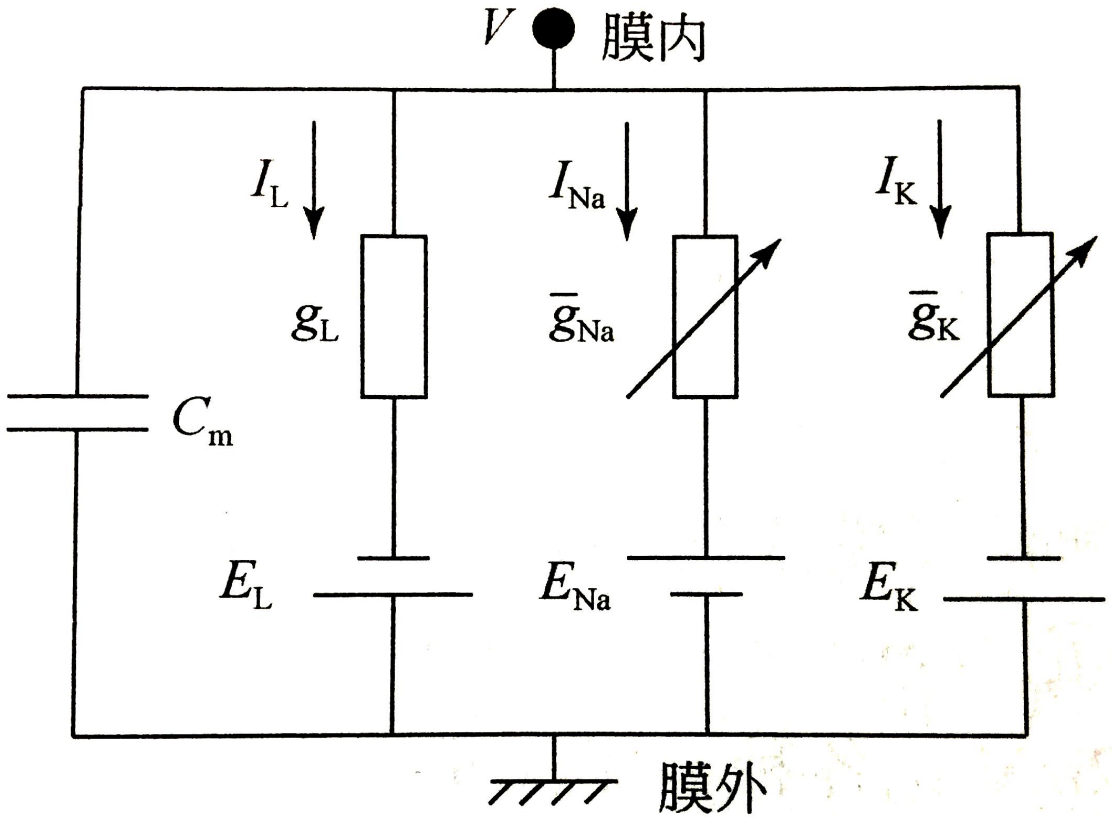


第三层次：单房室模型

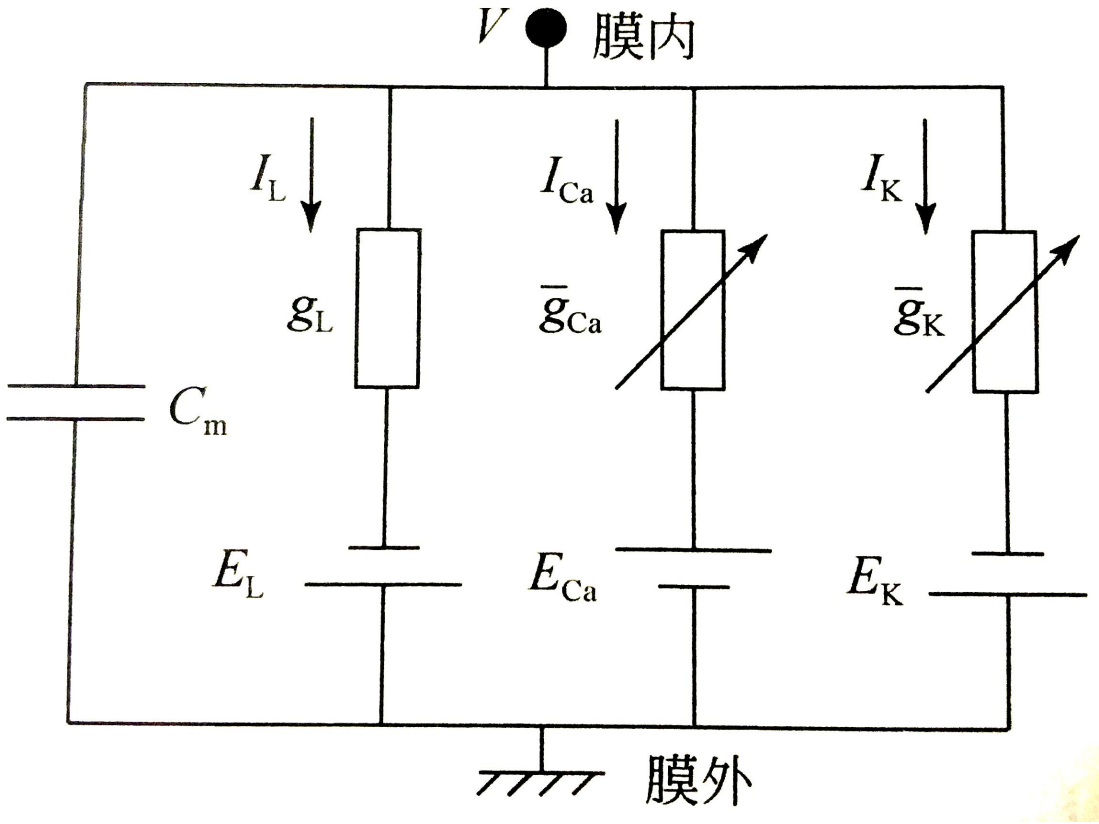
# 单房室脉冲神经元模型

- 单房室模型忽略了神经元的形态特征，将神经元抽象成一个点，仅考虑各种离子电流对神经元阈下行为和脉冲生成的影响。
- 最为典型的单房室脉冲神经元模型就是
  - 描述神经元膜电位和离子电导变化关系的 Hodgkin-Huxley模型。
  - 固定阈值的脉冲发放模型-Integrate-and-Fire神经元模型
- 在过去的几十年涌现了很多单房室脉冲神经元的计算模型，根据神经元的构造方法的差异将这些模型分为四类：
  - ①生物可解释性的生理模型；
  - ②脉冲生成机制的非线性模型；
  - ③固定阈值的脉冲发放模型；
  - ④分段线性化的解析模型。

# 生物可解释性的生理模型




Hodgkin-Huxley神经元模型

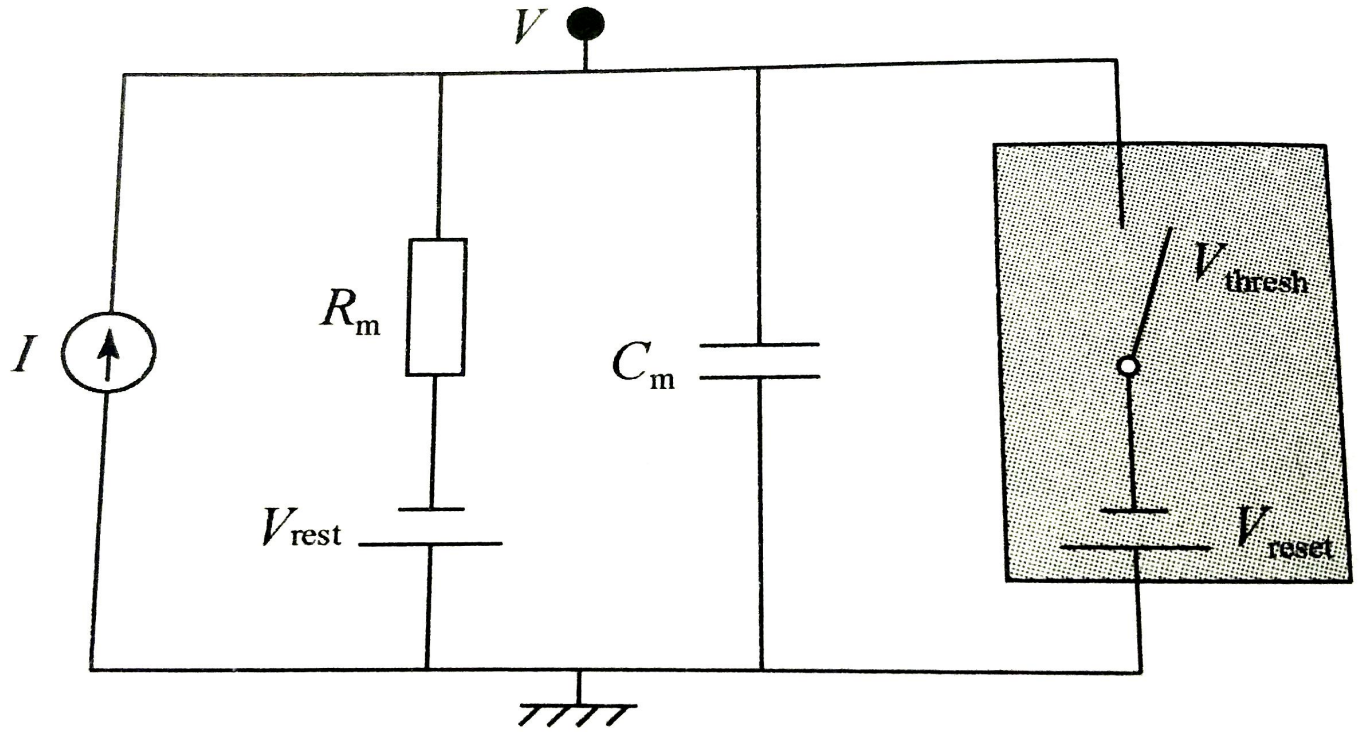



Morris-Lecar神经元模型

# 固定阈值的脉冲发放模型-合放（integrate-and-fire）脉冲神经元模型

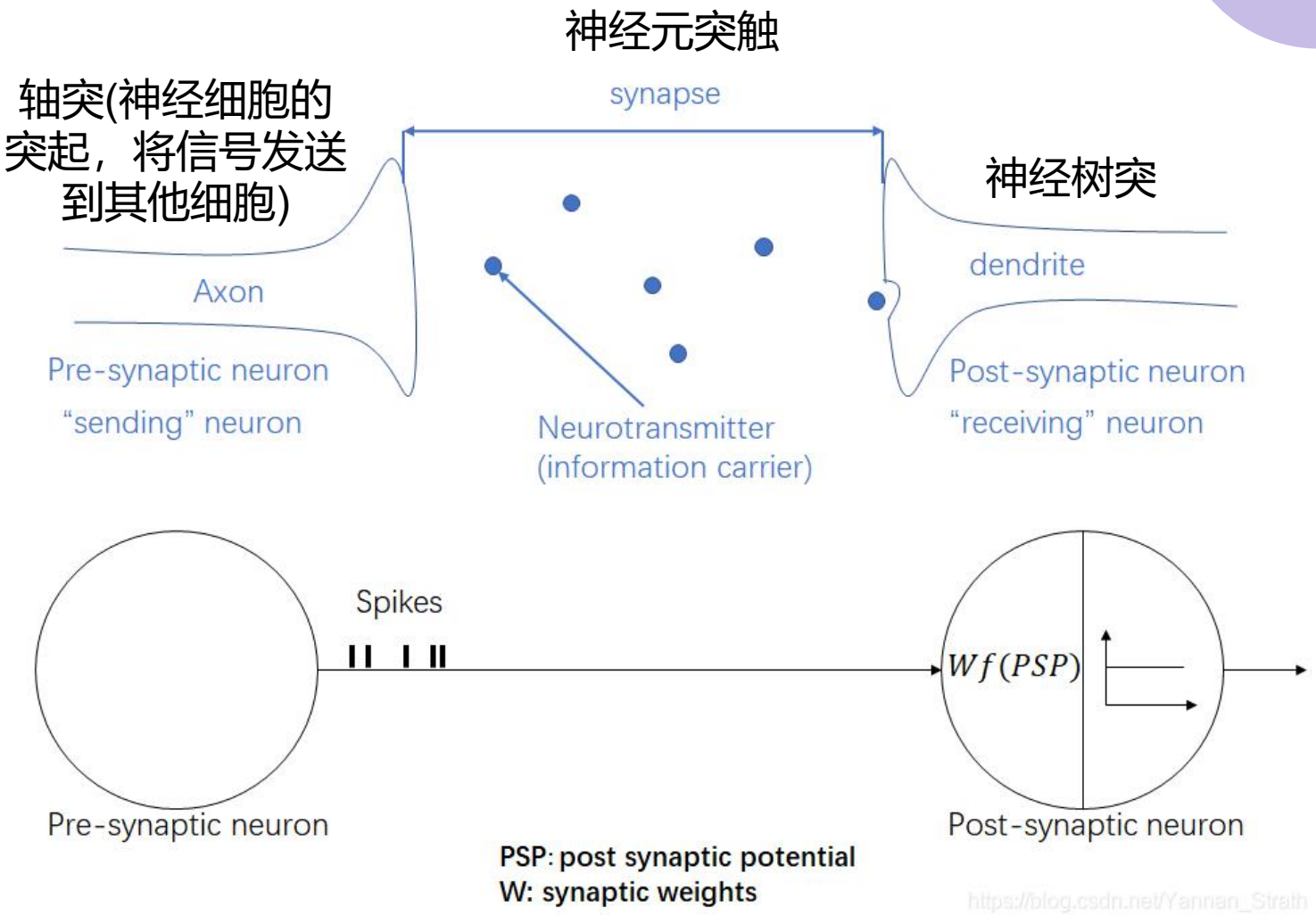
- 漏电Integrate-and-Fire模型非常直观地描述了神经元的**膜电位V**和其**输入电流I**之间的关系。
- 可以看出来式子为电容C定律的时间导数，电荷量Q = CV。
- 当施加输入电流时，膜电压会随时间增加，直到达到恒定阈值$V_{th}$，此时将出现增量函数尖峰，并且电压会重置$V_{reset}$为其静止电位，此后模型将继续运行。

$$I(t) = C_m \frac{dV_m(t)}{dt}$$



# Integrate-and-fire模型的Spiking Neuron

- 以生物突触结构为基础构建的脉冲神经元（spiking neuron)。
- 想象有两个spiking neuron
  - **一个为突触前神经元（pre-synaptic neuron）作为spiking的发出者。**
  - **一个为突触后神经元（post-synaptic neuron) 作为spike的接受者。**
- spiking neuron所进行的处理是接受由突触传递而来的脉冲， 依据**突触权重**通过**spiking function**产生**突触后膜电压（post synaptic potential (PSP))**

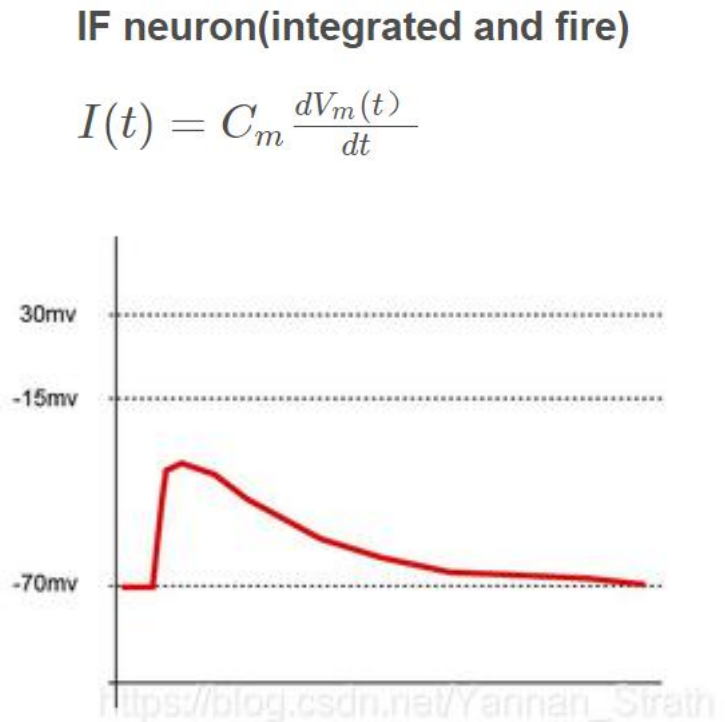


# Integrate-and-fire神经元的post synaptic potential (PSP)

- PSP简单的解释就是**神经元上的膜电压变化。**
  - 如图就是一个神经元接受到spike后膜电压 u（t）随着时间 t 的变化。
  - 膜电压在接受到脉冲输入前会一直保持在 - 70 -70−70mV 的地方， 这个值通常叫做**静止值（resting value）**。
  - **当接受到刺激后， 会产生电压变化的幅值。**
  - 在变化结束后， 膜电压会归位回起始的静止值。
- Integrate-and-fire的差分方程（differential equation）模型:

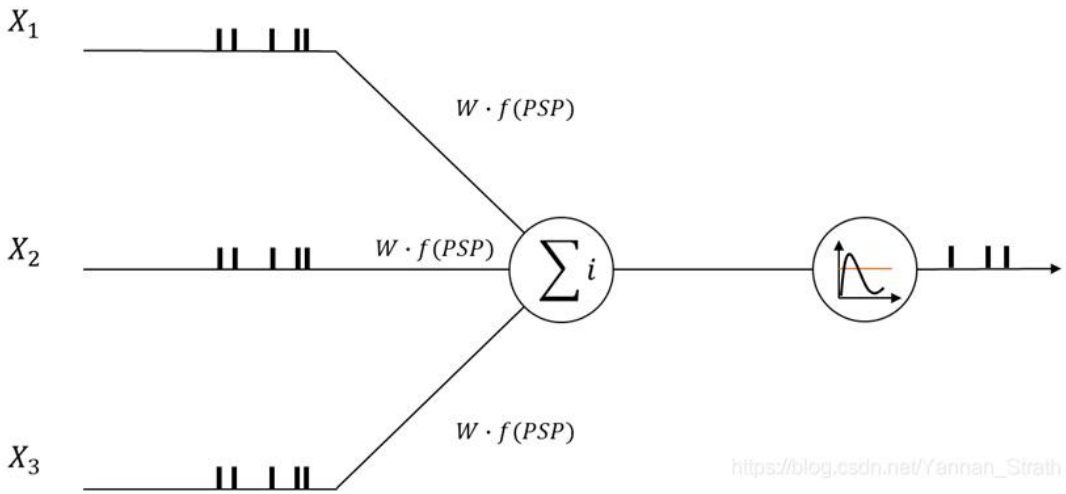
$$V(t) = V(t-1) + L + X(t),$$
$$\text{If } V(t) \geq \theta, \text{ spike and reset } V(t) = 0,$$
$$\text{If } V(t) < Vmin, \text{ reset } V(t) = Vmin.$$

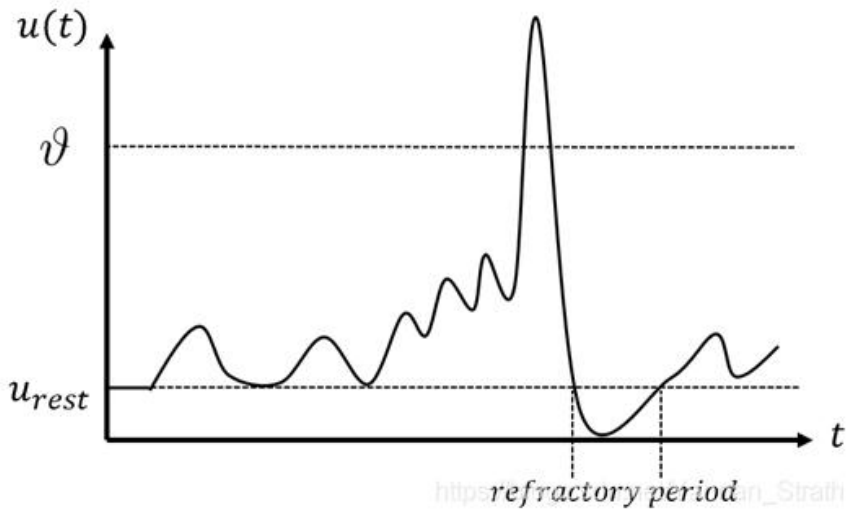
- L是泄露参数（常量），X(t)是与该神经元连接的所有突触在t时刻的输入和。当V(t)超过阈值，神经元发射一个脉冲，它的膜电位V(t)复位为0。膜电位V不允许低于它的静息电位 。



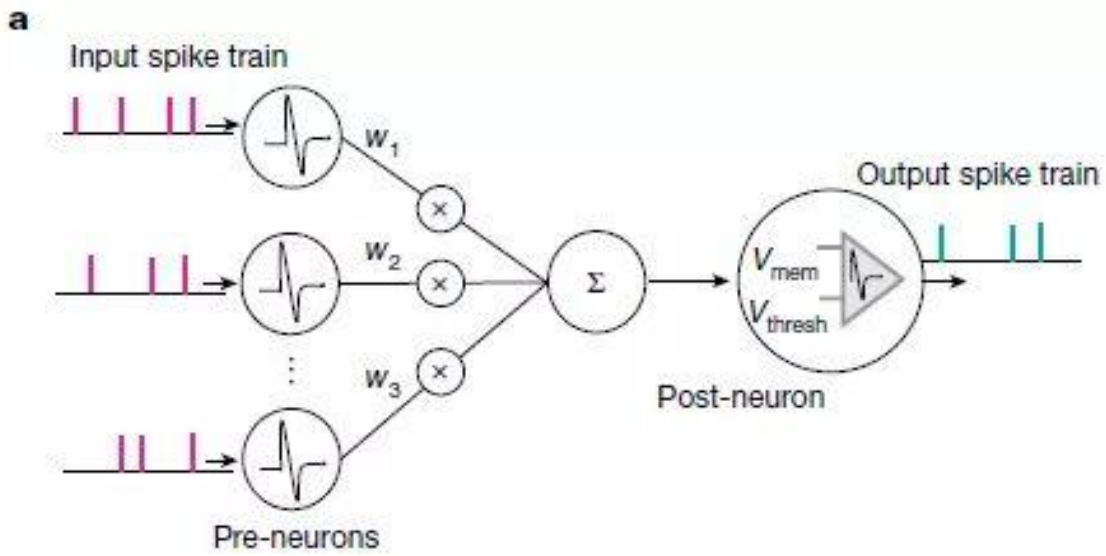
# 单个spiking neuron接受多个spike输入

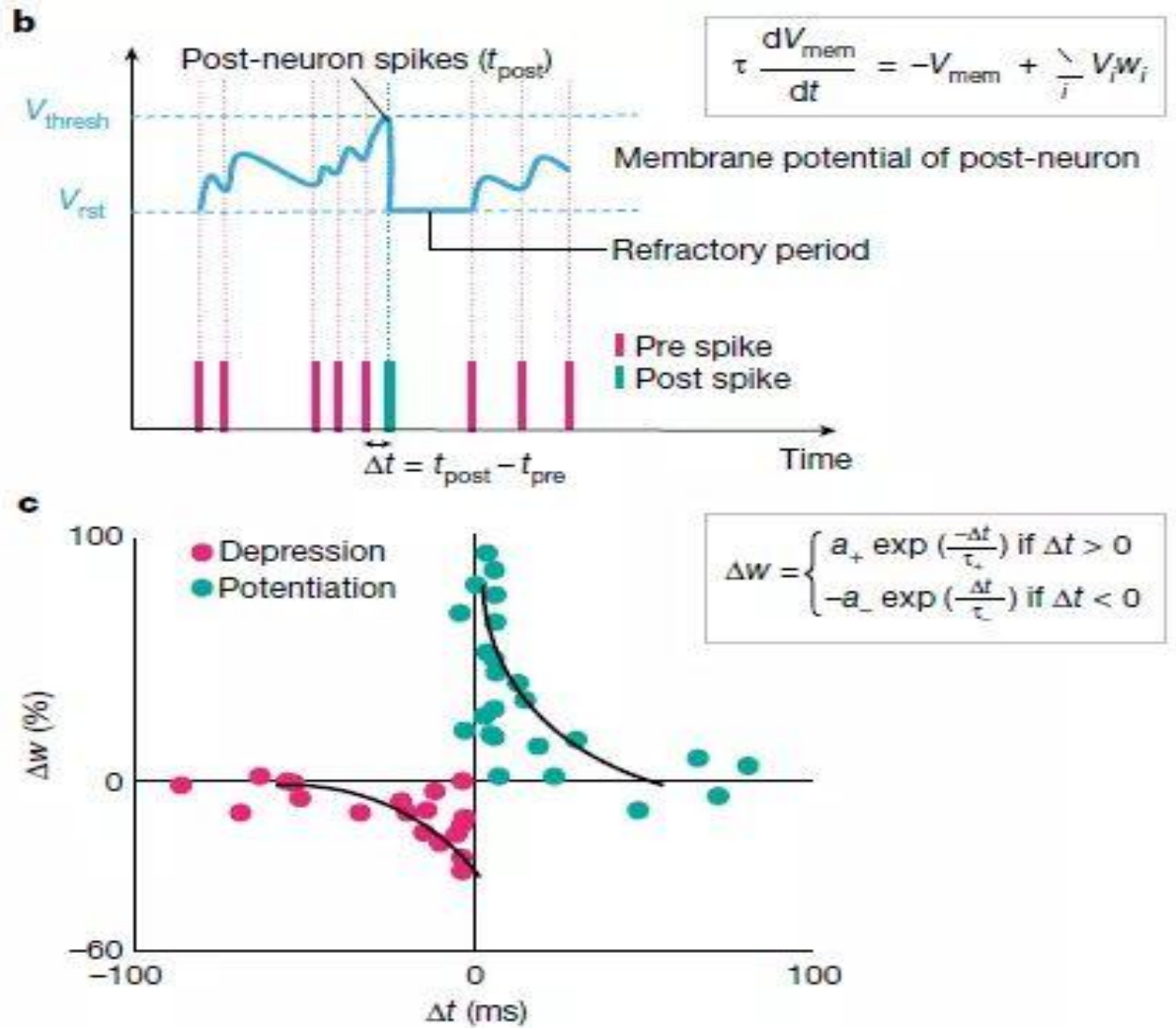
- 一次完整的spike generation过程：
  1. 突触后神经元会首先按照接受到的时间整合脉冲， 将他们变为膜电压的变化的叠加。
  2. 当膜电压超过预先设置好的阈值（$\vartheta$ ）时，突触后neuron被认定为收到了足够的刺激从而发出一个脉冲（spike）。
  3. **在发出脉冲后， 膜电压会被重置并且突触后神经元会在一段时间内无法处理接收到的脉冲(refractory period)。**
  4. 在refractory period之后， 膜电压会恢复到静止值从而可以准备下一次脉冲的产生。



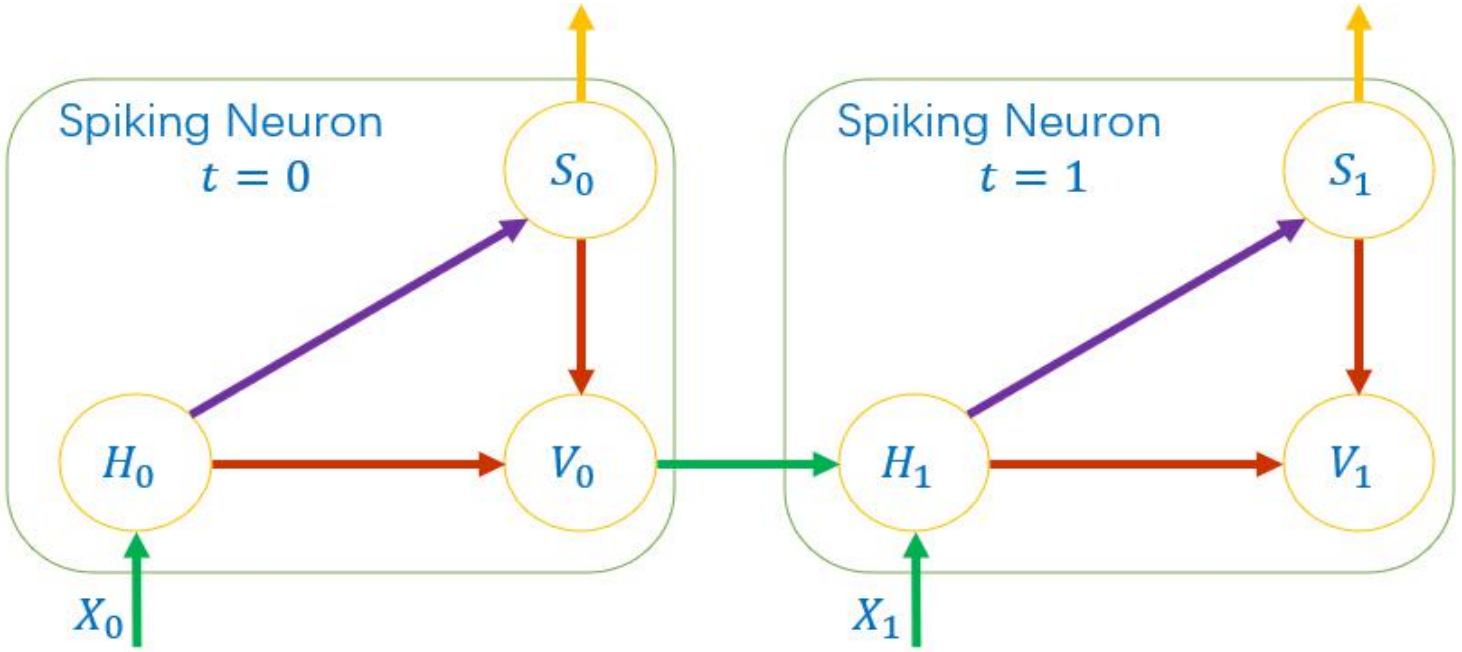
# 神经元的动态过程


a
Input spike train
$w_1$
$w_2$
$w_3$
Σ
$V_{mem}$
$V_{thresh}$
Output spike train
Post-neuron
Pre-neurons


b
Post-neuron spikes ($t_{post}$)
$V_{thresh}$
$V_{rst}$
Membrane potential of post-neuron
Refractory period
Pre spike
Post spike
$\Delta t = t_{post} - t_{pre}$
Time
$\tau \frac{dV_{mem}}{dt} = -V_{mem} + \sum_i V_i w_i$
c
100
0
−60
−100
0
100
Depression
Potentiation
Δw (%)
Δt (ms)
$\Delta w = \begin{cases} a_+ \exp(\frac{-\Delta t}{\tau_+}) \text{ if } \Delta t > 0 \\ -a_- \exp(\frac{\Delta t}{\tau_-}) \text{ if } \Delta t < 0 \end{cases}$

# 神经元的动态过程

- 为充电，放电，重置三个过程




充电：$H_t = f(V_{t-1}, X_t)$

放电：$S_t = \Theta(H_t - V_{threshold}) = \begin{cases} 1, H_t \geq V_{threshold} \\ 0, H_t < V_{threshold} \end{cases}$

重置：$V_t = r(H_t, S_t)$

$X_t$: $t$时刻的输入

$H_t$: $t$时刻充电后的膜电位

$S_t$: $t$时刻释放的脉冲

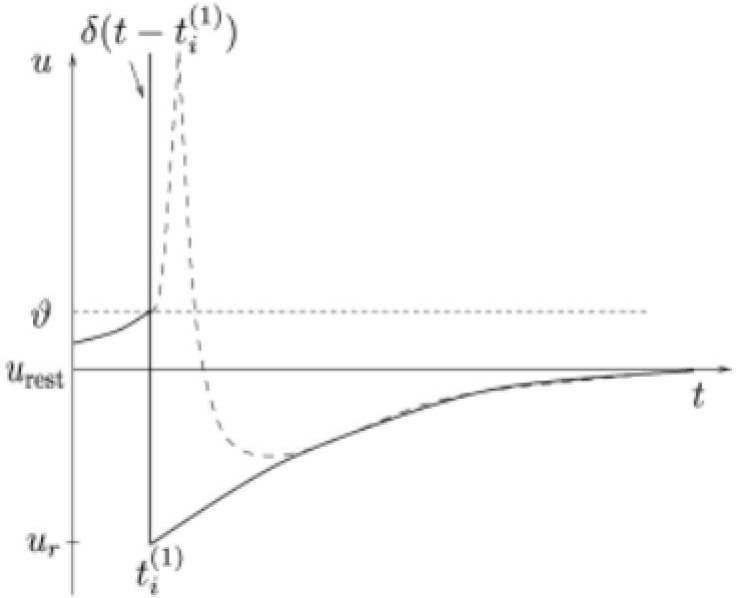
$V_t$: $t$时刻释放脉冲后的膜电位

# Leaky Intergrate and Fired (LIF) Model

- 目前使用最多的是Leakyintegrity-Fire（LIF）**模型。
- IF模型只有一个电容，没有并联的电阻，因为电阻实际等效于泄露电流，对应LIF模型。
- LIF流程:
  - Leaky指泄露，表示如果神经元输入只有一个时，不足以让膜电势超过阈值，由于细胞膜不断进行膜内外离子交换，膜电势会自动发生泄露逐渐回落到静息状态；
  - Integrate指积分，表示神经元会接收所有与该神经元相连的轴突末端（上一个神经元）到来的脉冲；
  - Fire指激发，表示当膜电势超过阈值时，神经元会发送脉冲。神经元发送脉冲后会进入超极化状态，然后是不应期（Refractory Period），在不应期内即使给予刺激也不会反应，即神经元不再接收刺激，保持静息电位。

LIF模型由两部组成:

$$\text{Leaky intergration: } \tau_m \frac{\mathrm{d}u}{\mathrm{d}t} = -\left[u(t) - u_{\text{rest}}\right] + RI(t)$$
$$\text{Reset } : \lim_{\delta \to 0; \delta > 0} u\left(t^{(f)} + \delta\right) = u_r$$



# SNN学习方法

- 转换法
  - 将传统神经网路转换为SNN
  - 将训练好的网络在输入，运算和输出上全面转换为以二进制spike为处理载体的网络。
  - 所有神经元要用相应的spiking neuron来替换，训练所得得权重要进行量化。
- 基于脉冲的方法
  - 在基于脉冲的方法中，SNN使用时间信息进行训练，因此在整体脉冲动力学中具备明显的稀疏性和高效率优势。研究人员采用了两种主要方向：
    - 无监督学习（没有标记数据的训练）
    - 监督学习（有标记数据的训练）
      - 早期监督学习成果是ReSuMe和tempotron，它们证明了在单层的SNN中，可以使用脉冲时间依赖的可塑性（STDP）的变体去进行分类。
      - SpikeProp及其相关变体已派生出通过在输出层固定一个目标脉冲序列来实现SNNs的反向传播规则。最近的成果对实值膜电位使用随机梯度下降法，是为了让正确输出神经元随机激发更多的脉冲（而不是具有精确目标的脉冲序列）。
      - 这些方法在深度卷积SNNs的小规模图像识别任务上取得了最新进展，例如美国国家标准与技

# 转换法-传统神经网路NN转换为SNN

- 由于ANN中的ReLU神经元非线性激活和SNN中IF神经元的发放率有着极强的相关性，可以基于类似的相关性将训练好的ANN转换为对应的SNN，省去了直接训练SNN的难题。
- 使用权重调整（weight rescaling）和归一化方法将训练有素的NN转换为SNN，
- 将非线性连续输出神经元的特征和尖峰神经元的泄漏时间常数（leak time constants），不应期（refractory period）、膜阈值（membrane threshold）等功能相匹配。
- 在图像分类的大型脉冲网络中（包括ImageNet数据集），这种方法能够产生了有竞争力的精确度。
- 但是，这种方法有其内在的局限性。例如在使用双曲线正切（tanh）或归一化指数函数（softmax）后，非线性神经元的输出值可以得正也可以得负，而脉冲神经元的速率只能是正值。因此，负值总被丢弃，导致转换后的SNNs的精度下降。
- 转换的另一个问题是在不造成严重的性能损失的前提下获得每一层最佳。最近的研究提出了确定最佳放电率的实用解决方案，以及在NN的训练过程中引入其他约束（例如噪音或泄漏修正线性单元（leaky ReLUs））以更好地匹配尖脉冲神经元的放电率。
- 转换的方法可为图像识别任务提供最先进的精度，并与NN的分类性能相当。
    - 缺点：从DLNs转换的SNNs的推理时间变得很长（约几千个时间步长），导致延迟增加、能耗增加。

# 无监督学习

- 大多是借鉴传统人工神经网络的无监督学习算法，是在Hebb学习规则不同变体的基础上提出的。
  - Hebb学习规则是一个无监督学习规则。
- SNN常见的无监督学习-STDP方法
  - Henry Markram提出STDP，Spike Timing Dependent Plasticity学习方法。
  - 神经科学的研究成果表明，生物神经系统中的脉冲序列不仅可引起神经突触的持续变化，并且满足**脉冲时间依赖可塑性（spike timing-dependent plasticity，STDP）**机制。
  - 在决定性时间窗口内，根据突触前神经元和突触后神经元发放的脉冲序列的相对时序关系，应用STDP学习规则可以对突触权值进行无监督方式的调整。

# 无监督学习-STDP方法

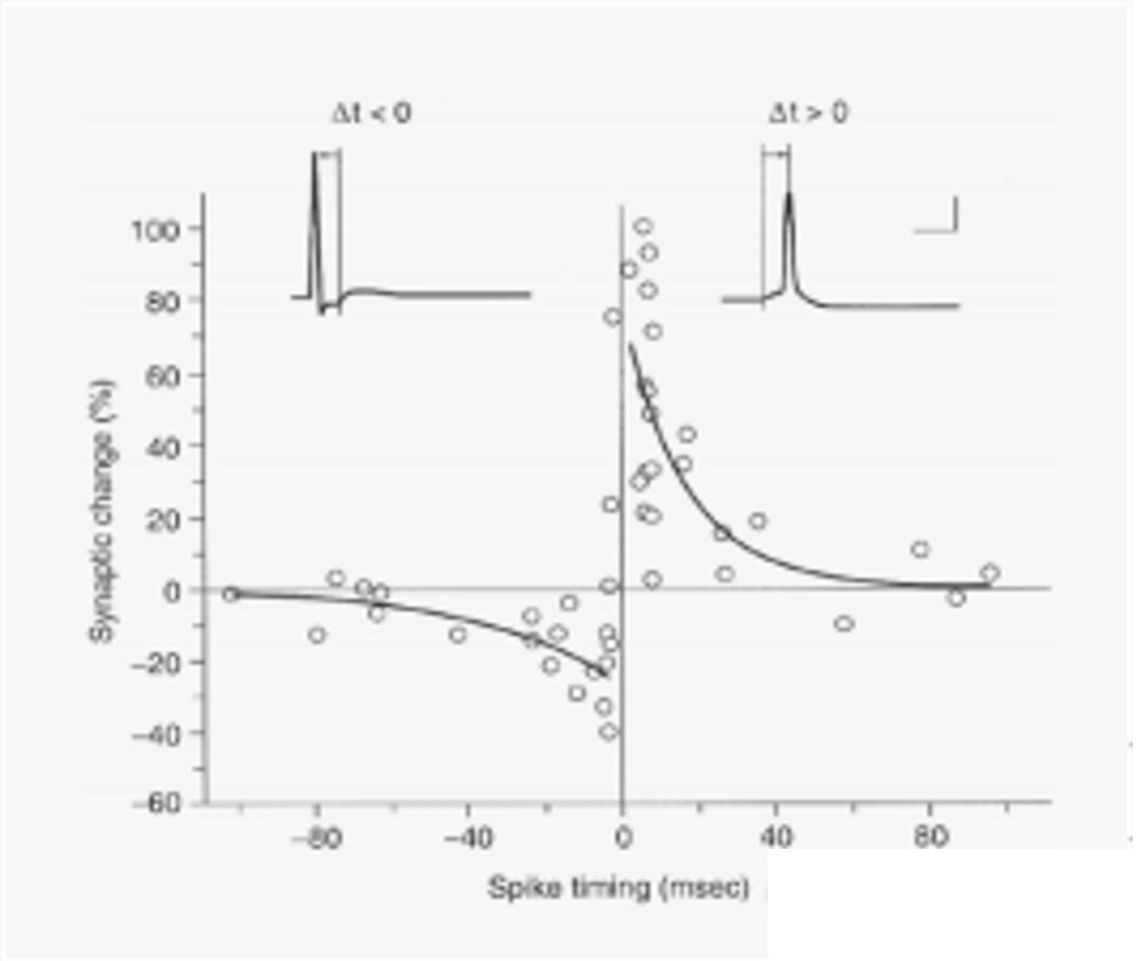
- 生物STDP的最常见形式具有非常直观的解释。
- 它根据神经元学习的先后顺序，调整神经元之间连接的强弱。
- 如果突触前神经元在突触后神经元之前不久触发（大约10毫秒），则连接它们的权重会增加。
- 如果突触后神经元在突触后神经元后不久触发，则时间事件之间的因果关系是虚假的，权重会减弱。
- 常见的STDP规则:

$$\Delta w = \begin{cases} Ae^{\frac{-(|t_{pre}-t_{post}|)}{\tau}} & t_{pre} - t_{post} \leq 0, A > 0 \\ Be^{\frac{-(|t_{pre}-t_{post}|)}{\tau}} & t_{pre} - t_{post} > 0, B < 0 \end{cases}$$

- w是**突触权重**， A > 0 和B < 0 通常是指示**学习率**的常数参数， t是时间学习窗口的**时间常数**（例如15ms）。
- 效果的强度由**衰减指数调制**，衰减指数的大小由突触前和突触后脉冲之间的时间常数比例时间差控制。

# 无监督学习-STDP方法

- SDTP是在大脑中发现的神经元之间权重连接的更新规则。
- SDTP是如果一个神经元A的激活在另一个神经元B的激活之后很快就发生，时间差小于5ms时，B到A的连接权重就会增加约70%，而相反A到B的连接权重就会衰减20%。
- 通俗的说，就是当A和B先后激活时，具备紧密先后关系的双方会加强联系，而具备相反关系的，就会渐行渐远。
- 从这个角度，也可以想象到神经元之间往往建立的是单向的加强联系，如果A到B不断增强，那么B到A就不断减弱，而如果是同时发生，一般两者会不好不坏。



# 无监督学习-STDP方法

- 如果在其他神经元j传递信息之后，它才产生反应，那么类似于因果关系，它和传递信息的神经元之间连接G(j→i)会加强；
- 如果它产生反应之后，其他神经元j才传递信息来，那么这个信息就有可能被忽略，即该神经元与传递信息的神经元间的连接G(j→i)会减弱。
- 例子:
  - 如果每次你跳完一段新疆舞后，就给狗食物吃，次数多了，狗在看你跳舞时，变会认为离吃食物时间不远了，而流口水；
  - 但是如果每次你在给狗吃东西之后才跳一段新疆舞，那么狗会认为你有病，大倒胃口，“控制”流口水的神经元与传递你跳新疆舞的神经元之间的连接就被减弱了。

# SNN监督学习算法

- 基于梯度下降的监督学习算法
  - 利用神经元目标输出与实际输出之间的误差以及误差反向传播过程，得到梯度下降计算结果作为突触权值调整的参考量，最终减小这种误差。
  - 是一种数学分析方法，在学习规则的推导过程中，要求神经元模型的状态变量必须是有解析的表达式，主要采用固定阈值的线性神经元模型，如脉冲响应模型（spike response model）和Integrate-and-Fire神经元模型等。
- 基于突触可塑性的监督学习算法
  - 利用神经元发放脉冲序列的时间相关性所引起的突触可塑性机制，设计神经元突触权值调整的学习规则，这是一种具有生物可解释性的监督学习。
- 基于脉冲序列卷积的监督学习算法
  - 通过脉冲序列内积的差异构造脉冲神经网络的监督学习算法，突触权值的调整依赖于特定核函数的卷积计算，可实现脉冲序列时空模式的学习。

# 监督学习-ReSuMe（远程监督学习）方法

- 所有以上模型均由单个脉冲神经元组成，该脉冲神经元接收来自许多突触前脉冲神经元的输入，目的是训练突触，使突触后神经元产生具有期望的脉冲时间的脉冲序列。
- ReSuMe将最初用于非脉冲线性单位的Widrow-Hoff（Delta）规则调整为SNN。
  - Widrow-Hoff学习算法是一个近似最速下降法，其中性能指标为均方误差。
- Widrow-Hoff规则权重变化与期望输出减去观察输出成正比，如下所示。

$$\Delta w = \left(y^{\mathrm{d}} - y^{\circ}\right) x = y^{\mathrm{d}} x - y^{\circ} x$$

- 其中x 是突触前的输入，$y^{d}$ 和$y^{o}$ 分别是期望的输出和观察到的输出。当按照RHS所示进行扩展并针对SNN重新定义时，该规则可以表示为STDP和anti-STDP的总和。即，训练兴奋性突触的规则采用以下形式：

$$\Delta w = \Delta w^{\mathrm{STDP}}\left(S^{\mathrm{in}}, S^{\mathrm{d}}\right) + \Delta w^{\mathrm{aSTDP}}\left(S^{\mathrm{in}}, S^{\mathrm{o}}\right)$$

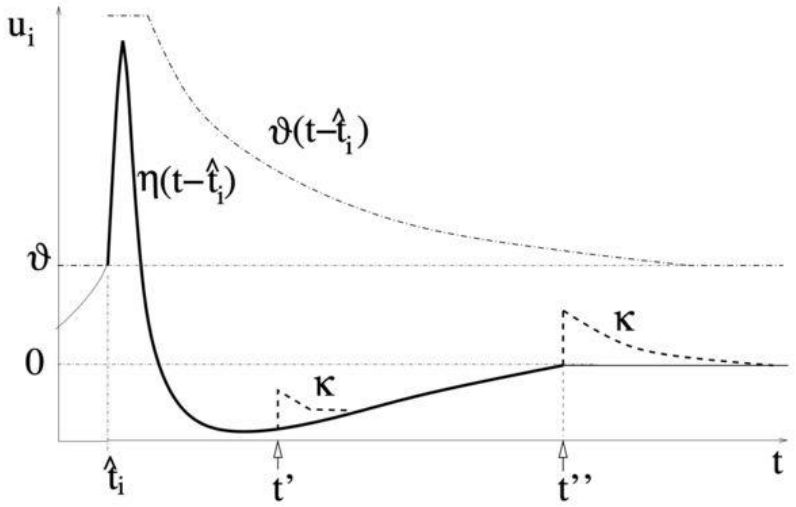
- Δ $w^{STDP}$是突触前和期望脉冲序列的相关性的函数，而Δ w $^{aSTDP}$取决于突触前和观察到的脉冲序列。因为学习规则使用的是教师神经元（期望的输出）和输入神经元之间的相关性，所以没有直接的物理联系。这就是为什么在“远程监督学习”中使用“远程”一词的原因。

# 监督学习-SpikeProp学习

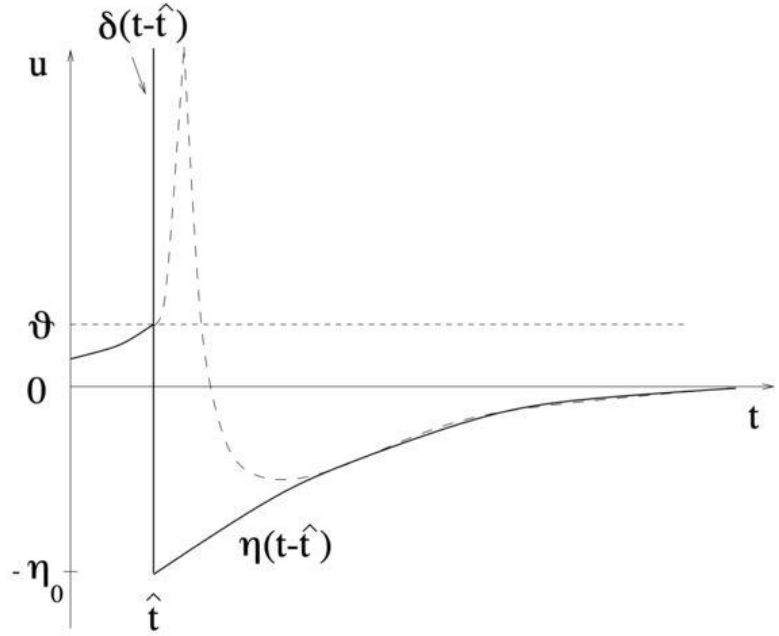
- 第一个通过反向传播误差训练SNN的算法，他们的损失 函数考虑了脉冲时序，SpikeProp能够使用三层体系结构针对时间编码的XOR问题对非线性可分离数据进行分类。
- 脉冲神经元使用脉冲响应模型（SRM）。
- 使用SRM模型，避免了对隐层单元的输出脉冲取导数的问题，因为可以将这些单元的响应直接建模为连续值PSP，并将其应用于预测的输出突触。
- 这项工作的局限性在于每个输出单元都只能精确地发放一个脉冲，连续变量值（例如在时间扩展的XOR问题中）必须被编码为可能相当长的脉冲时间延迟。
- SpikeProp的后序高级版本Multi-SpikeProp适用于多脉冲编码。
  - 使用与SpikeProp相同的神经体系结构，新的脉冲时序编码和脉冲时间误差公式改善了脉冲反向传播算法。
  - Wu等人提出了SNN中反向传播的最新实现，在多层SNN中进行时空梯度下降。

# Spike Response Model (SRM)

- 标准的SRM主要考虑两个部分：
  - 一是脉冲发放后的膜电势变化，二是对外部刺激电流(电极电流或突触前神经元发放的脉冲的突触后电流)。



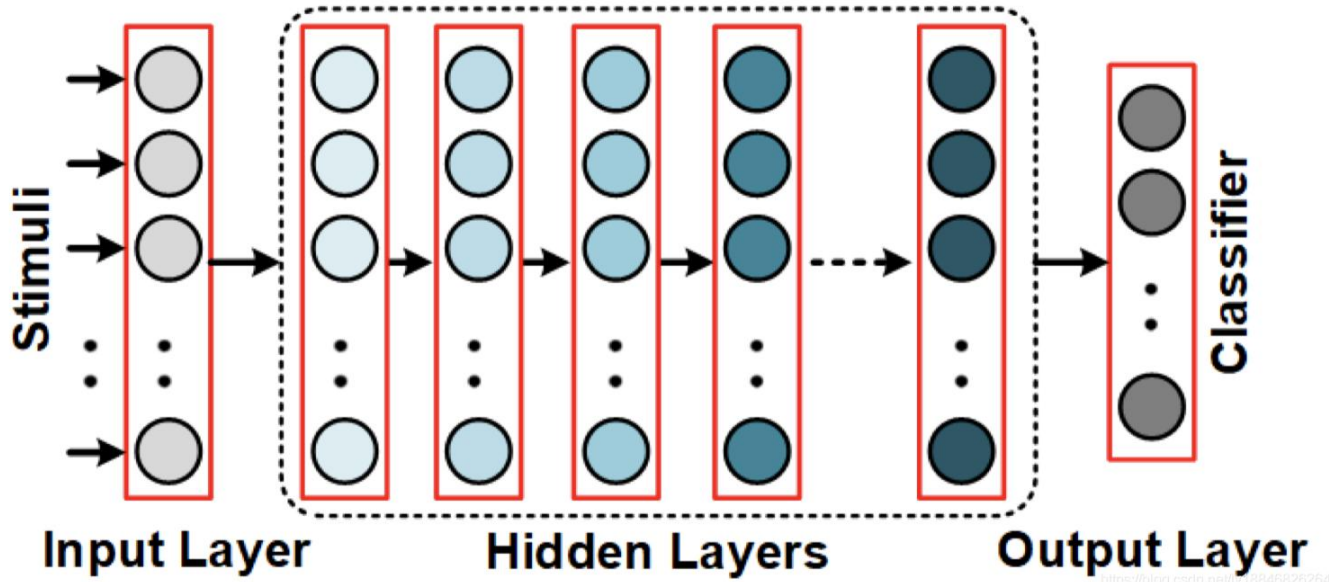

标准SRM




简化SRM

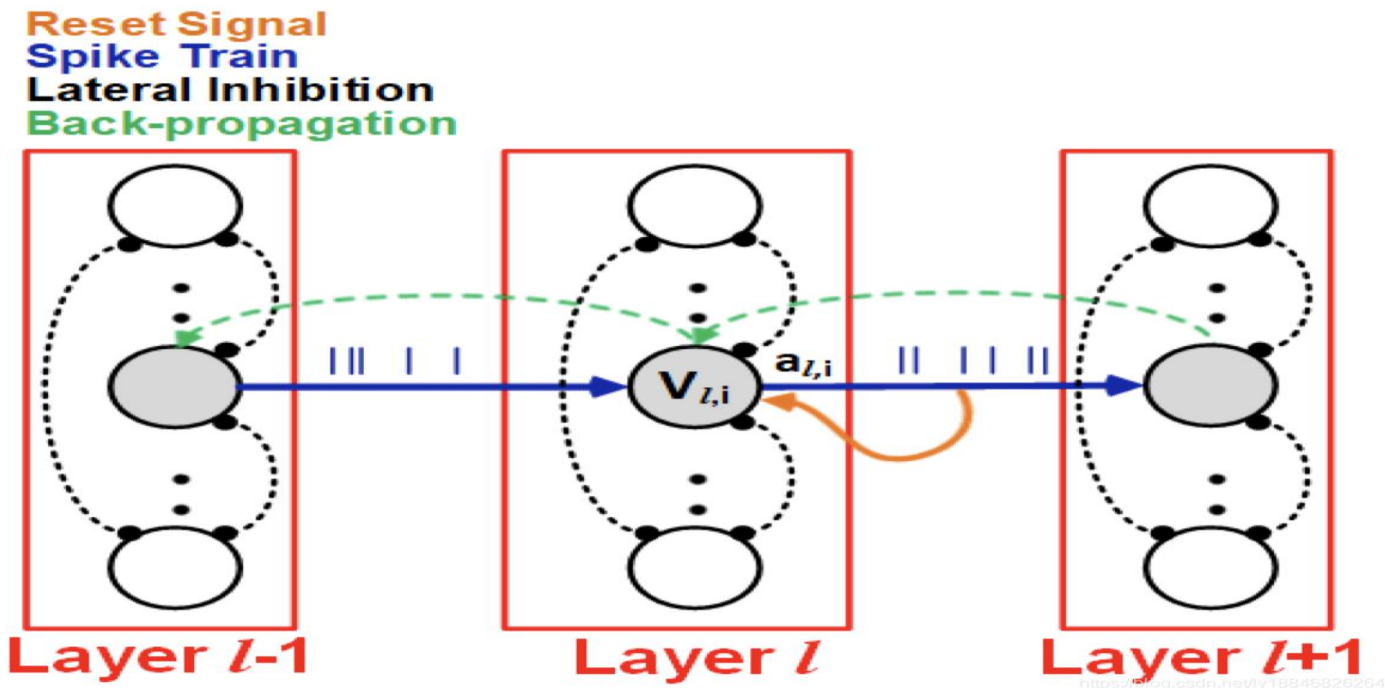
# SNN中的深度学习

- SNN在许多模式识别任务中也显示出令人鼓舞的性能。
- 但是，直接训练的脉冲深层网络的性能不如文献中的传统DNN好。
- 因此，由于在DNN硬件实现中的重要性，具有出色性能的可与传统深度学习方法相比的脉冲深层网络（脉冲DNN、脉冲CNN、脉冲RNN或脉冲DBN）是一个具有挑战性的话题。



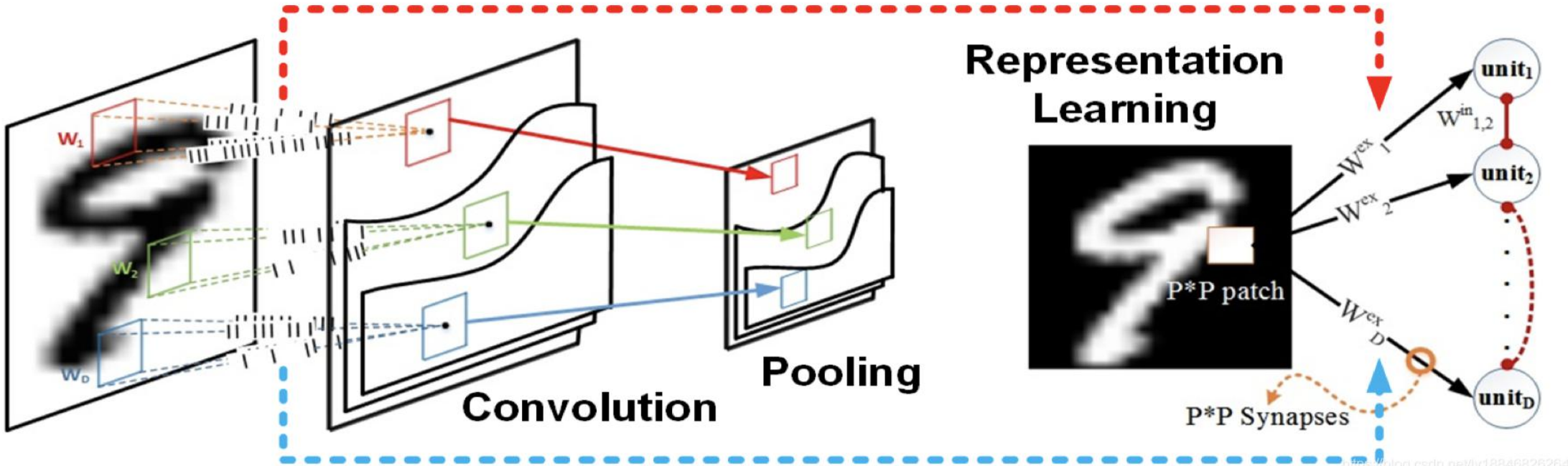
# Deep SNN

- 由Lee等人提出的配备反向传播功能的Deep SNN。
- 神经元的激活值a，i由神经元的膜电位给出。由神经元的兴奋性输入，侧向抑制和阈值计算出的可微分激活函数用于使用链法则进行反向传播。
- 当前层（第1层）的输出激活值在反向传播算法中用作下一层的输入。



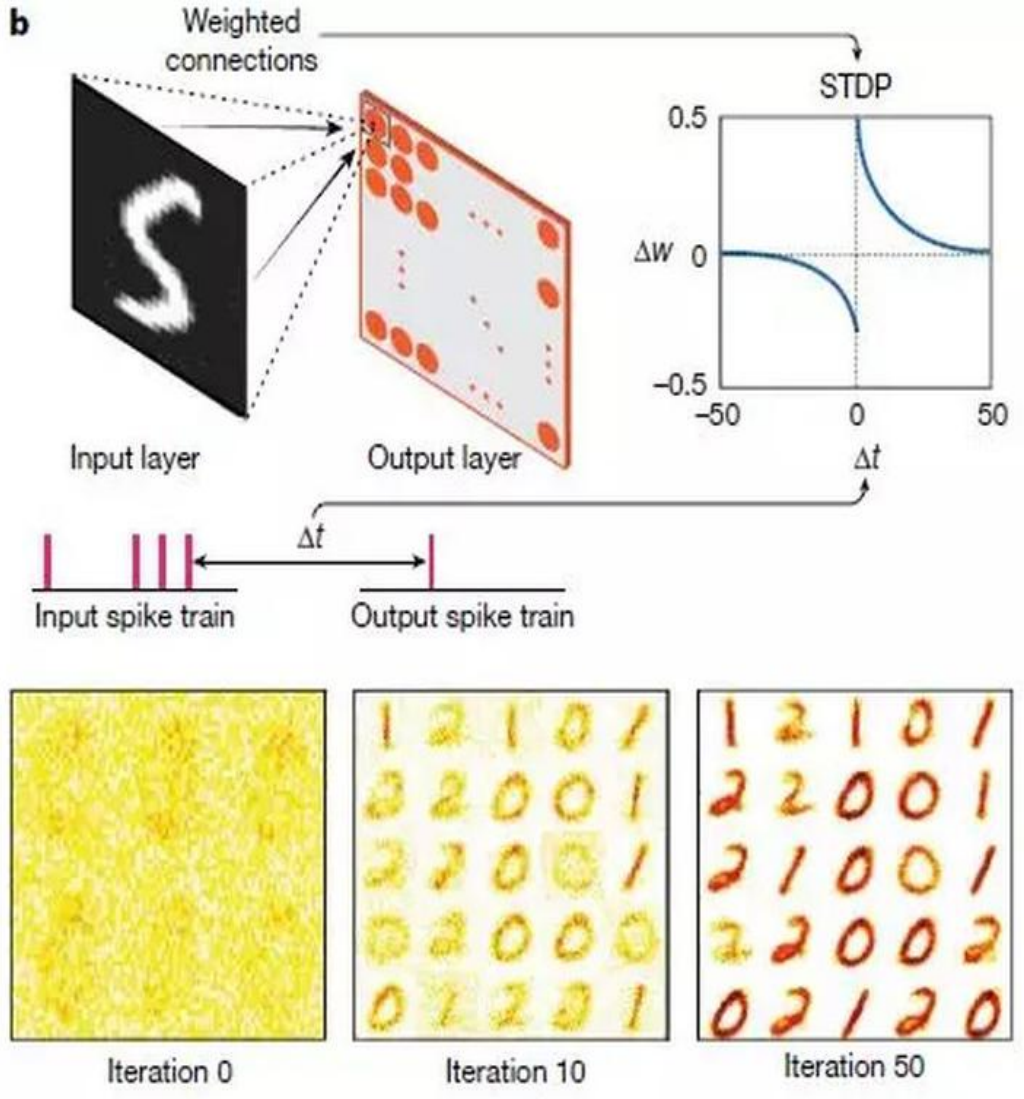
# 无监督脉冲卷积神经网络

- 表示学习（SAILnet）用于对脉冲CNN进行分层无监督学习。
- 连接到表示层中神经元的兴奋性突触权重指定卷积滤波器。
- 这个架构决定了可以利用单层SNN中的表示学习来训练逐层增强的CNN。



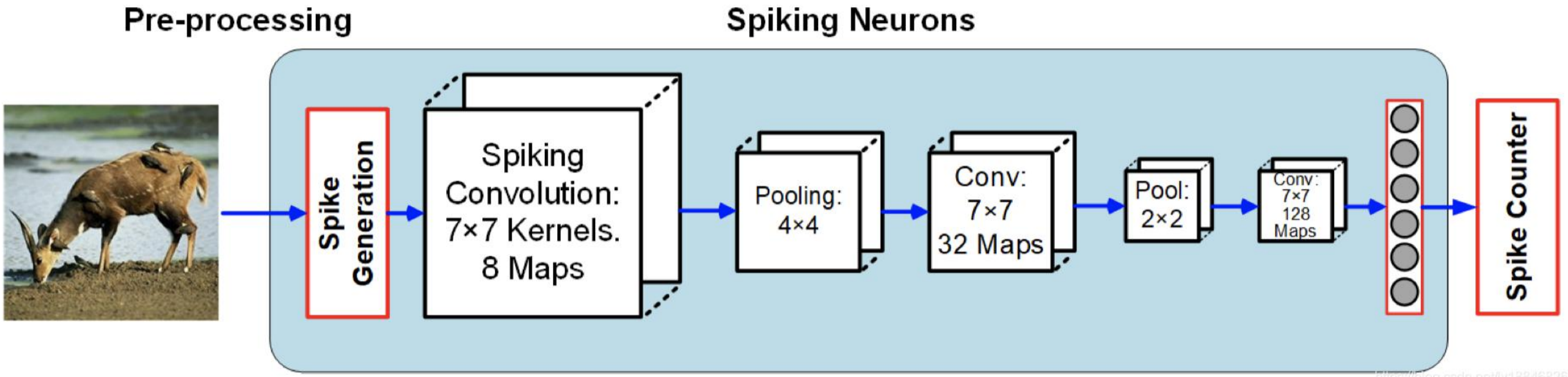
# 无监督脉冲卷积神经网路

- 基于STDP学习规则的局部无监督SNN训练。
- 通过局部学习（我们将在后面的硬件讨论中看到），有机会使记忆（突触存储）和计算（神经元输出）更紧密地相结合。
- 这种架构更像人脑，也适合节能芯片上实现。
- Diehl等人率先证明了完全无监督的SNN学习，其精度可与MNIST数据库深度学习相媲美



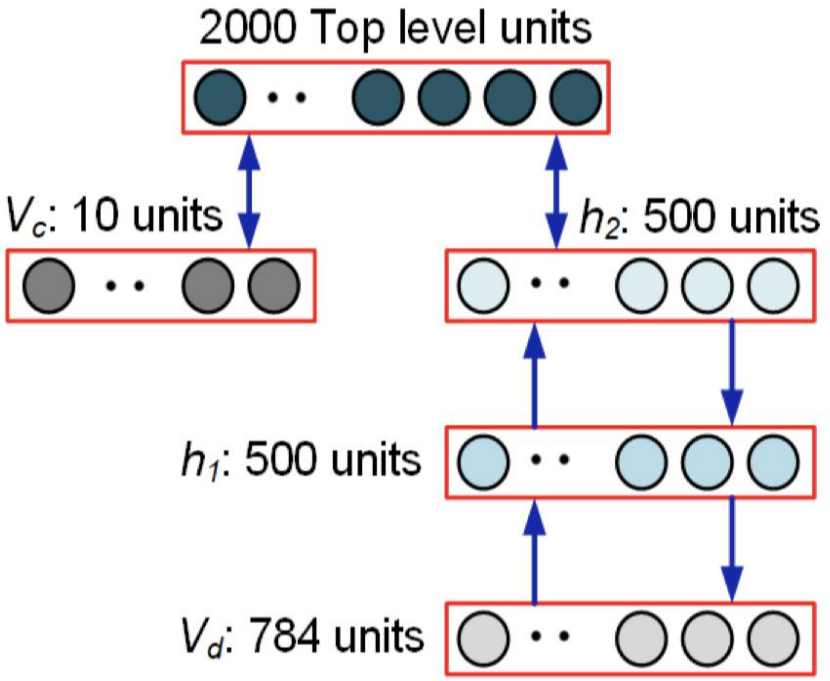
# 脉冲CNN架构

- 由Cao等人开发的脉冲CNN架构。
- 经过预处理的输入图像会根据像素强度转换为脉冲序列。
- **脉冲层使用由非脉冲CNN训练的权重。**
- 最后一个组件选择具有最大活动性（脉冲发放率）的神经元作为图像类别。

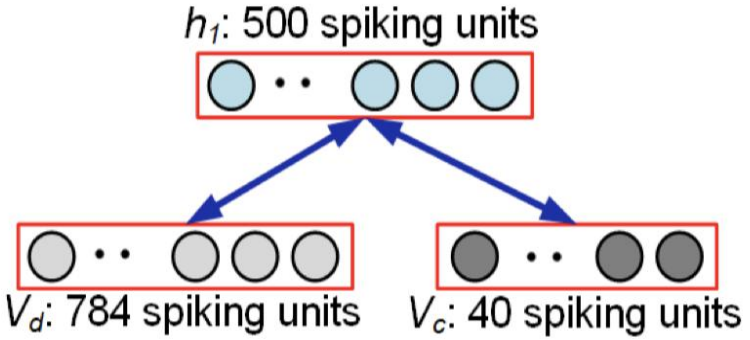


# 脉冲深度信念网络

- 将先前训练的DBN转换为脉冲神经网路。
- 将DBN转换为LIF脉冲神经元网络以进行MNIST图像分类。
- 网路如图:
  - a）Hinton等人提出的DBN，用于MNIST图像分类。该网络由具有500、500和2000个表示神经元的三个堆叠的RBM组成。输入和输出分别包括784（作为像素数，28 x 28）和10（作为类数，0，...，9）神经元。
  - b）Neftci等人引入的脉冲RBM体系结构。由500个隐藏神经元、784个输入神经元和40个类神经元（824个可见神经元）组成。



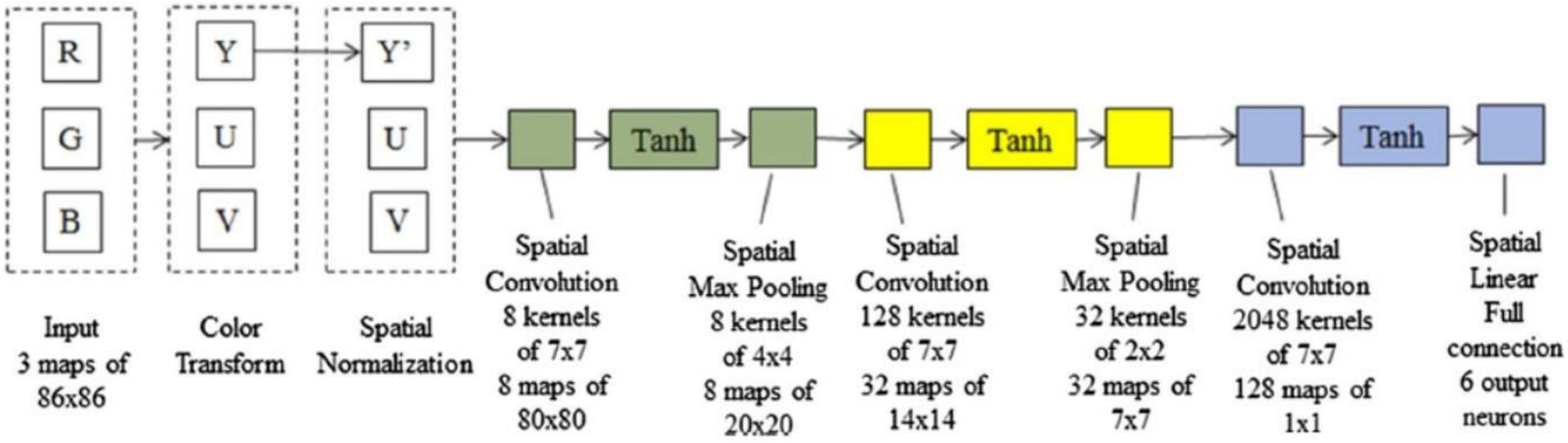

(a)



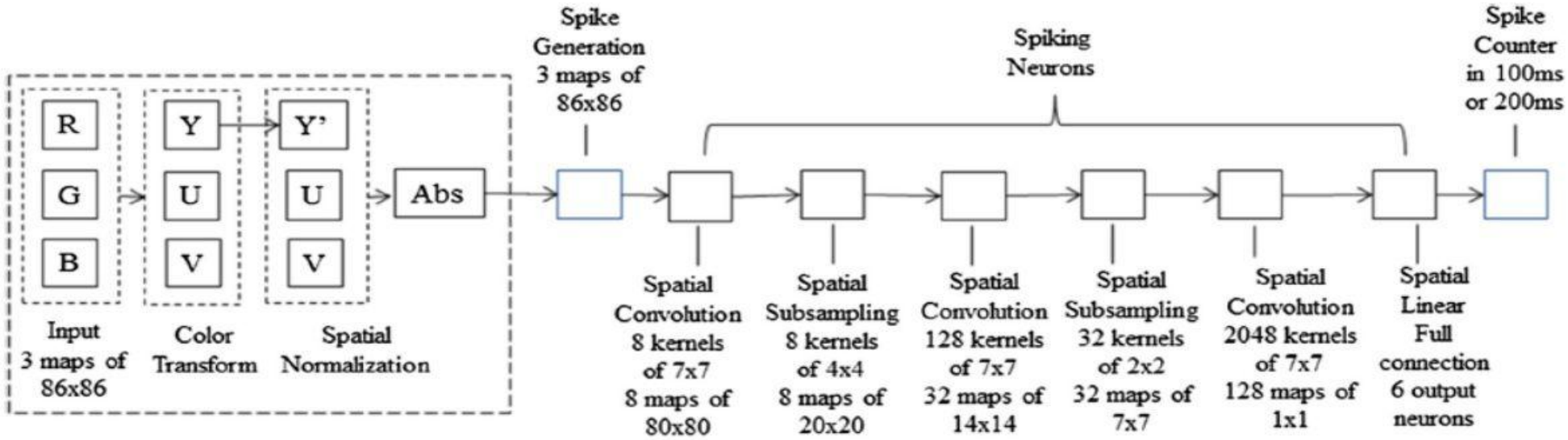

(b)

# CNN转换为SNN

CNN




SNN



# CNN转换为SNN的精度损失

- 源于一下几个方面：

1.CNN层的负输出值在SNN中很难精确的表示，负值主要来源于下面的CNN计算：

a. tanh()的输出范围为-1到1；

b. 在每个卷积层，每个输出feature map的值都是输入和权重的乘加和再加上偏置的结果，这有可能是个负值；

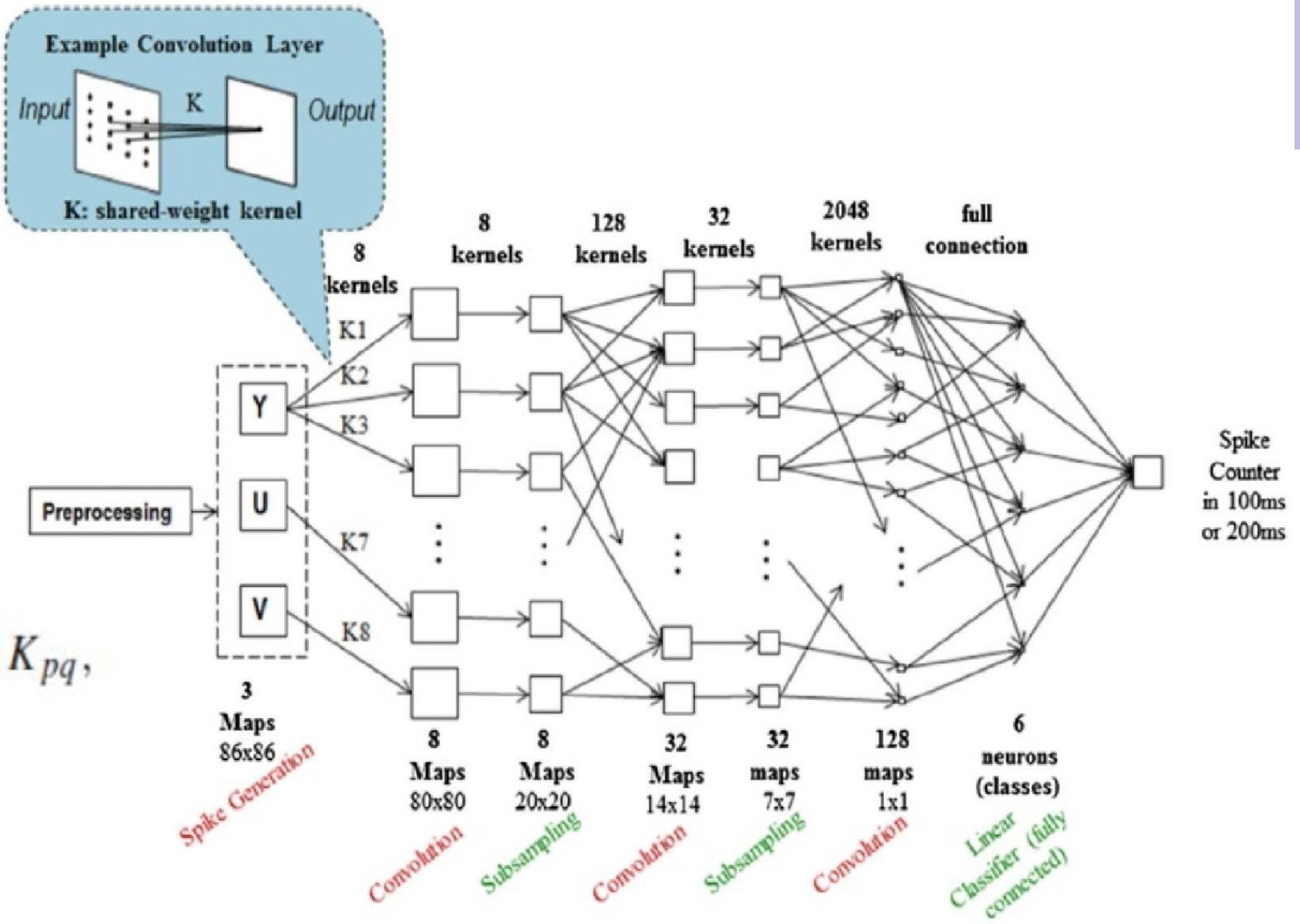
c. 预处理后的网络输入也有可能是负的；

2. 不像在CNN中，在脉冲网络中没有什么好办法表示偏置，每个卷积层的偏置可以是正也可以是负，这在SNN中不容易表示；

3. Max-pooling需要两层脉冲网络，在CNN中，空间上的最大池化就是在一个区域内求最大值，在SNN中，我们需要两层网络来实现它，横向抑制后紧跟着小区域的池化操作，这个方法需要更多的神经元，而且由于额外的复杂度造成了精度损失；

# 脉冲卷积神经网路例子

- SNN结构由预处理、脉冲产生、第一个空间卷积和线性子采样层、第二个空间卷积核线性子采样层、第三个空间卷积核线性分类层、脉冲计数组成。
- 可以观察到SNN结构中没有了ReLU模块，因为ReLU的属性在SNN中由脉冲神经元反映。



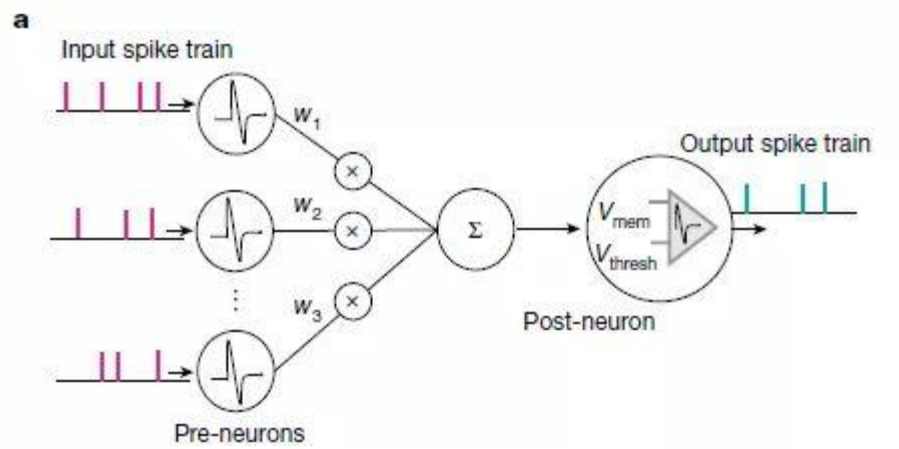

$$X_{ij}(t) = \sum_{p,q=-3}^{3} A_{p+i,q+j}(t)\, K_{pq},$$

这里的 $A_{p+i,q+j}(t)$ 是来自前层的输入脉冲（0或者1），

$K_{pq}$ 是被同一个map中的神经元共享的 7x7 的卷积核权重。

# Spike-YOLO

- 使用**IF(integrate-and-fire)神经元**，并使用DNN-to-SNN转化方法将SNN应用到更复杂的目标检测领域中。
- 面临以下两个问题：
  - **常用的SNN归一化方法过于低效，导致脉冲发射频率过低。**由于SNN需要设定阈值进行脉冲发射，所以要对权值进行归一化，这样有利于阈值的设定，而常用的SNN归一化方法在目标检测中显得过于低效。
  - **在SNN领域，没有高效leaky-ReLU的实现**，因为要将YOLO转换为SNN，YOLO中包含大量leaky-ReLU，这是很重要的结构，但目前还没有高效的转换方法。
- 使用channel-wise归一化(Channel-wise normalization)和阈值不平衡的有符号神经元(signed neuron with imbalanced threshold)来分别解决以上问题。




$$V^l_{\mathrm{mem},j}(t) = V^l_{\mathrm{mem},j}(t-1) + z^l_j(t) - V_{\mathrm{th}}\Theta^l_j(t),$$

$$z^l_j(t) = \sum_i w^l_{i,j}\Theta^{l\text{-}1}_i(t) + b^l_j,$$

$$\Theta^l_i(t) = U(V^l_{\mathrm{mem},i}(t) - V_{\mathrm{th}}),$$

# Spike-YOLO-Channel-wise normalization

- 之前Layer-wise normalization

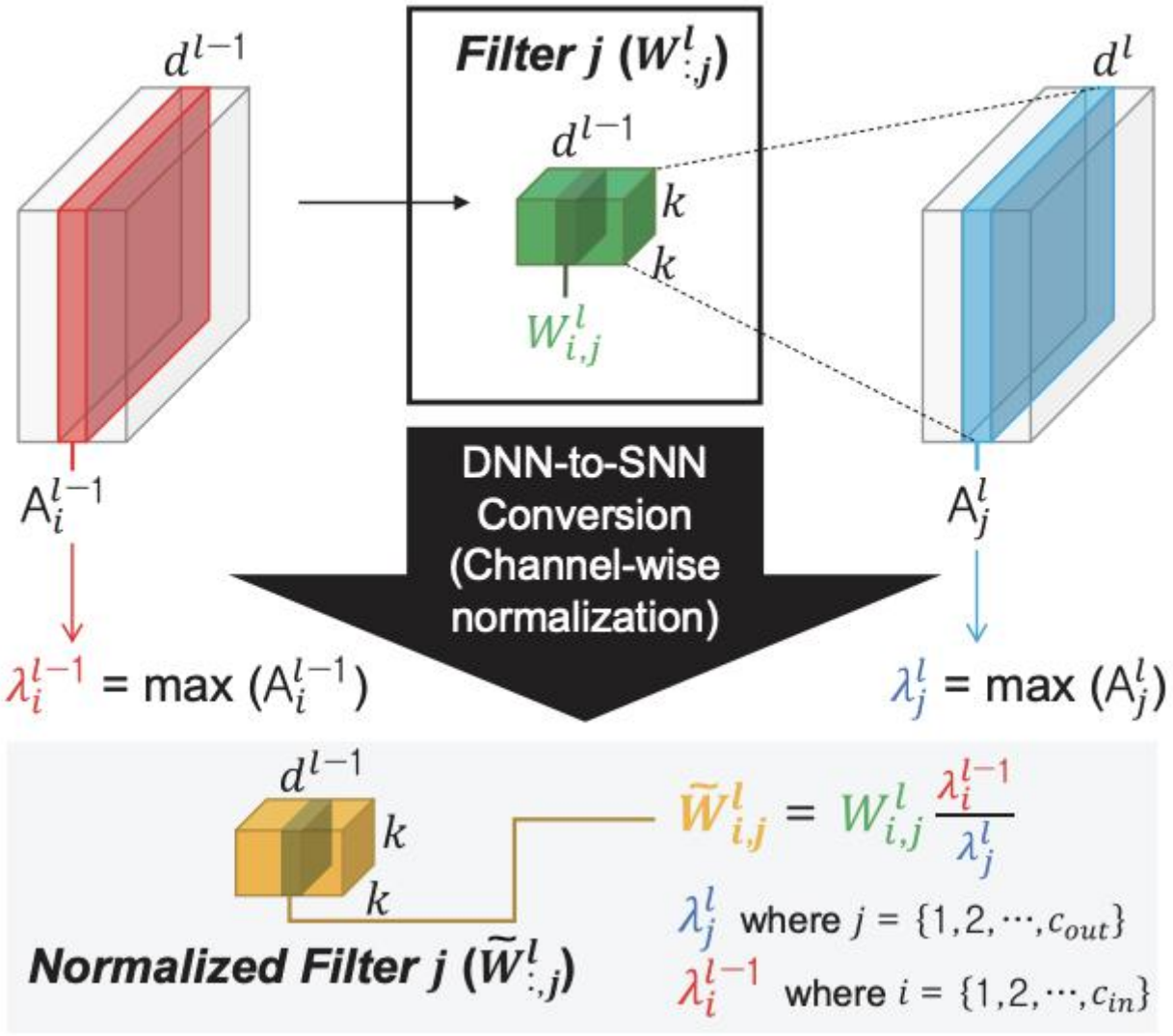
$$\tilde{w}^l = w^l \frac{\lambda^{l-1}}{\lambda^l} \quad \text{and} \quad \tilde{b}^l = \frac{b^l}{\lambda^l},$$

- w，b和为权重，λ 为输出特征图最大值。

- Channel-wise normalization

$$\tilde{w}^l_{i,j} = w^l_{i,j} \frac{\lambda^{l-1}_i}{\lambda^l_j} \quad \text{and} \quad \tilde{b}^l_j = \frac{b^l_j}{\lambda^l_j},$$

- i和j为维度下标，l层权值w通过在每个channel使用最大激活值λ 进行归一化，该值依然是从训练集计算的。避免传递的信息会越来越小。



# SNN对比

- 下表显示了用于开发深度SNN及其结构的模型以及它们在不同数据集上的准确率。
- 表显示了脉冲模型的两个轨迹：
  - 1）使用在线学习和
  - 2）使用离线学习（部署）。
- 后一种方法展示了更高的性能，但通过将离线训练的神经网络转换为相关的脉冲平台，避免了训练多层SNN。
- 另一方面，在线学习为SNN中提供了多层学习，但准确率较低。
- 此外，如预期的那样，在图像分类方面，脉冲的CNN的精度高于脉冲的DBN和全连接的SNN。

| Model | Architecture | Learning method | Dataset | Acc |
|---|---|---|---|---|
| **Feedforward, fully connected, multi-layer SNNs** | | | | |
| O'Connor (2016) [137] | Deep SNN | Stochastic gradient descent | MNIST | 96.40 |
| O'Connor (2016) [137] | Deep SNN | Fractional stochastic gradient descent | MNIST | 97.93 |
| Lee (2016) [57] | Deep SNN | Backpropagation | MNIST | 98.88 |
| Lee (2016) [57] | Deep SNN | Backpropagation | N-MNIST | 98.74 |
| Neftci (2017) [138] | Deep SNN | Event-driven random backpropagation | MNIST | 97.98 |
| Liu (2017) [108] | SNN | Temporal backpropagation (3-layer) | MNIST | 99.10 |
| Eliasmith (2012) [129] | SNN | Spaun brain model | MNIST | 94.00 |
| Diehl (2015) [130] | SNN | STDP (2-layer) | MNIST | 95.00 |
| Tavanaei (2017) [118] | SNN | STDP-based backpropagation (3-layer) | MNIST | 97.20 |
| Mostafa (2017) [109] | SNN | Temporal backpropagation (3-layer) | MNIST | 97.14 |
| Querlioz (2013) [139] | SNN | STDP, Hardware implementation | MNIST | 93.50 |
| Brader (2007)[128] | SNN | Spike-driven synaptic plasticity | MNIST | 96.50 |
| Diehl (2015) [140] | Deep SNN | Offline learning, Conversion | MNIST | 98.60 |
| Neil (2016) [144] | Deep SNN | Offline learning, Conversion | MNIST | 98.00 |
| Hunsberger (2015) [177], [178] | Deep SNN | Offline learning, Conversion | MNIST | 98.37 |
| Esser (2015) [141] | Deep SNN | Offline learning, Conversion | MNIST | 99.42 |
| **Spiking CNNs** | | | | |
| Lee (2016) [57] | Spiking CNN | Backpropagation | MNIST | 99.31 |
| Lee (2016) [57] | Spiking CNN | Backpropagation | N-MNIST | 98.30 |
| Panda (2016) [173] | Spiking CNN | Convolutional autoencoder | MNIST | 99.05 |
| Panda (2016) [173] | Spiking CNN | Convolutional autoencoder | CIFAR-10 | 75.42 |
| Tavanaei (2017) [171], [172] | Spiking CNN | Layer wise sparse coding and STDP | MNIST | 98.36 |
| Tavanaei (2018) [174] | Spiking CNN | Layer-wise and end-to-end STDP rules | MNIST | 98.60 |
| Kheradpisheh (2016) [170] | Spiking CNN | Layer wise STDP | MNIST | 98.40 |
| Zhao (2015) [169] | Spiking CNN | Tempotron | MNIST | 91.29 |
| Cao (2015) [183] | Spiking CNN | Offline learning, Conversion | CIFAR-10 | 77.43 |
| Neil (2016) [179] | Spiking CNN | Offline learning, Conversion | N-MNIST | 95.72 |
| Diehl (2015) [140] | Spiking CNN | Offline learning, Conversion | MNIST | 99.10 |
| Rueckauer (2017) [142] | Spiking CNN | Offline learning, Conversion | MNIST | 99.44 |
| Rueckauer (2017) [142] | Spiking CNN | Offline learning, Conversion | CIFAR-10 | 90.85 |
| Hunsberger (2015) [177] | Spiking CNN | Offline learning, Conversion | CIFAR-10 | 82.95 |
| Garbin (2014) [181] | Spiking CNN | Offline learning, Hardware | MNIST | 94.00 |
| Esser (2016) [182] | Spiking CNN | Offline learning, Hardware | CIFAR-10 | 87.50 |
| Esser (2016) [182] | Spiking CNN | Offline learning, Hardware | CIFAR-100 | 63.05 |
| **Spiking RBMs and DBNs** | | | | |
| Neftci (2014) [203] | Spiking RBM | Contrastive divergence in LIF neurons | MNIST | 91.90 |
| O'Connor (2013) [204] | Spiking DBN | Offline learning, Conversion | MNIST | 94.09 |
| Stromatias (2015) [205] | Spiking DBN | Offline learning, Conversion | MNIST | 94.94 |
| Stromatias (2015) [206] | Spiking DBN | Offline learning, Hardware | MNIST | 95.00 |
| Merolla (2011) [207] | Spiking RBM | Offline learning, Hardware | MNIST | 94.00 |
| Neil (2014) [208] | Spiking DBN | Offline learning, Hardware | MNIST | 92.00 |

# CNN vs. SNN

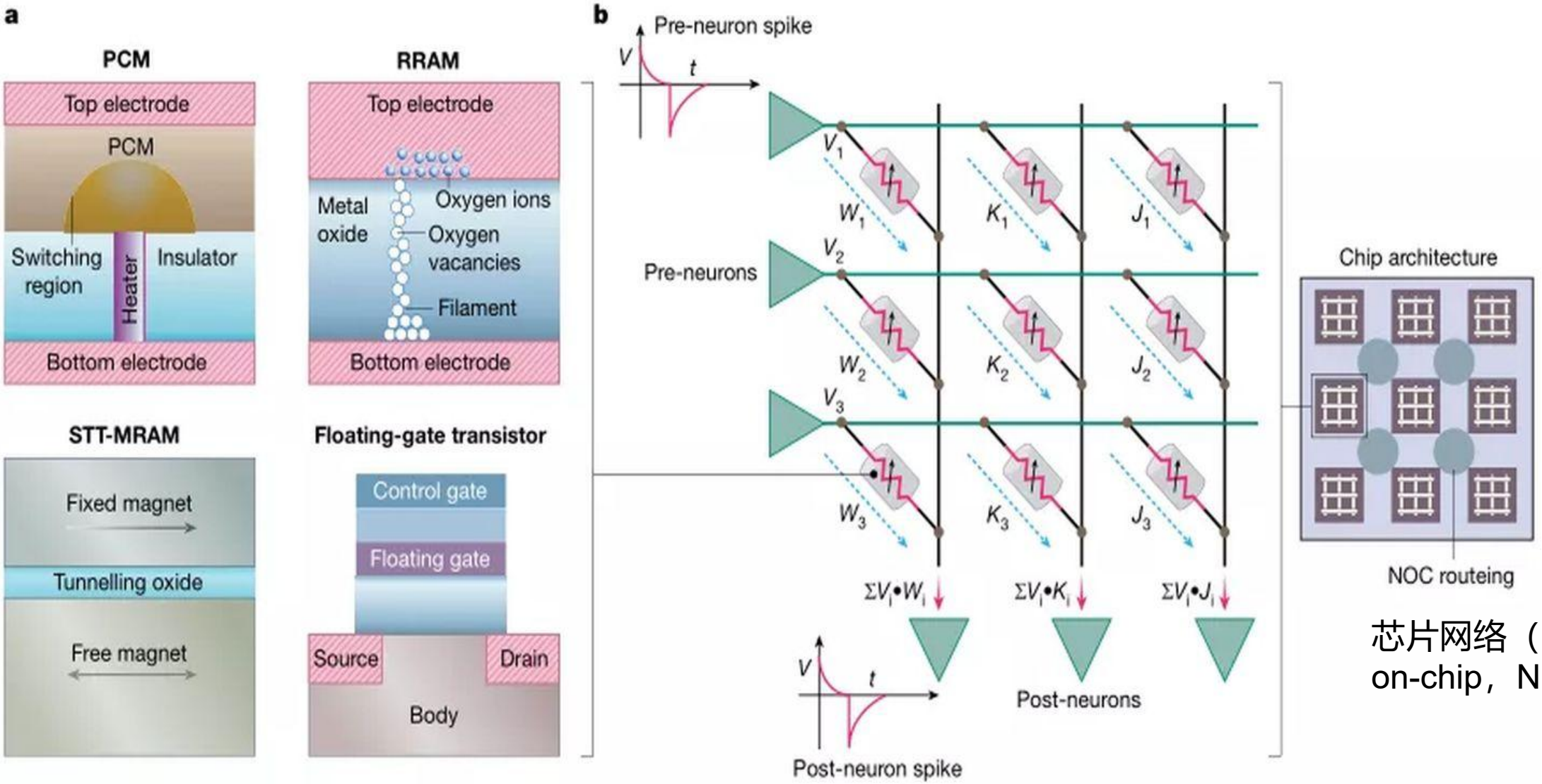
| | Convolutional Neural Networks | | Spiking Neural Networks | |
|---|---|---|---|---|
| | Characteristic | Result | Characteristic | Result |
| Computational functions | Matrix Multiplication, ReLU, Pooling, FC layers | Math intensive, high power, custom acceleration blocks | Threshold logic, connection reinforcement | Math-light, low power, standard logic |
| Training | Backpropagation off-chip | Requires large pre-labeled datasets, long and expensive training periods | Feed-Forward, on or off-chip | Short training cycles, continuous learning |

* Math intensive cloud compute

* Low power edge deployments

# SNN芯片

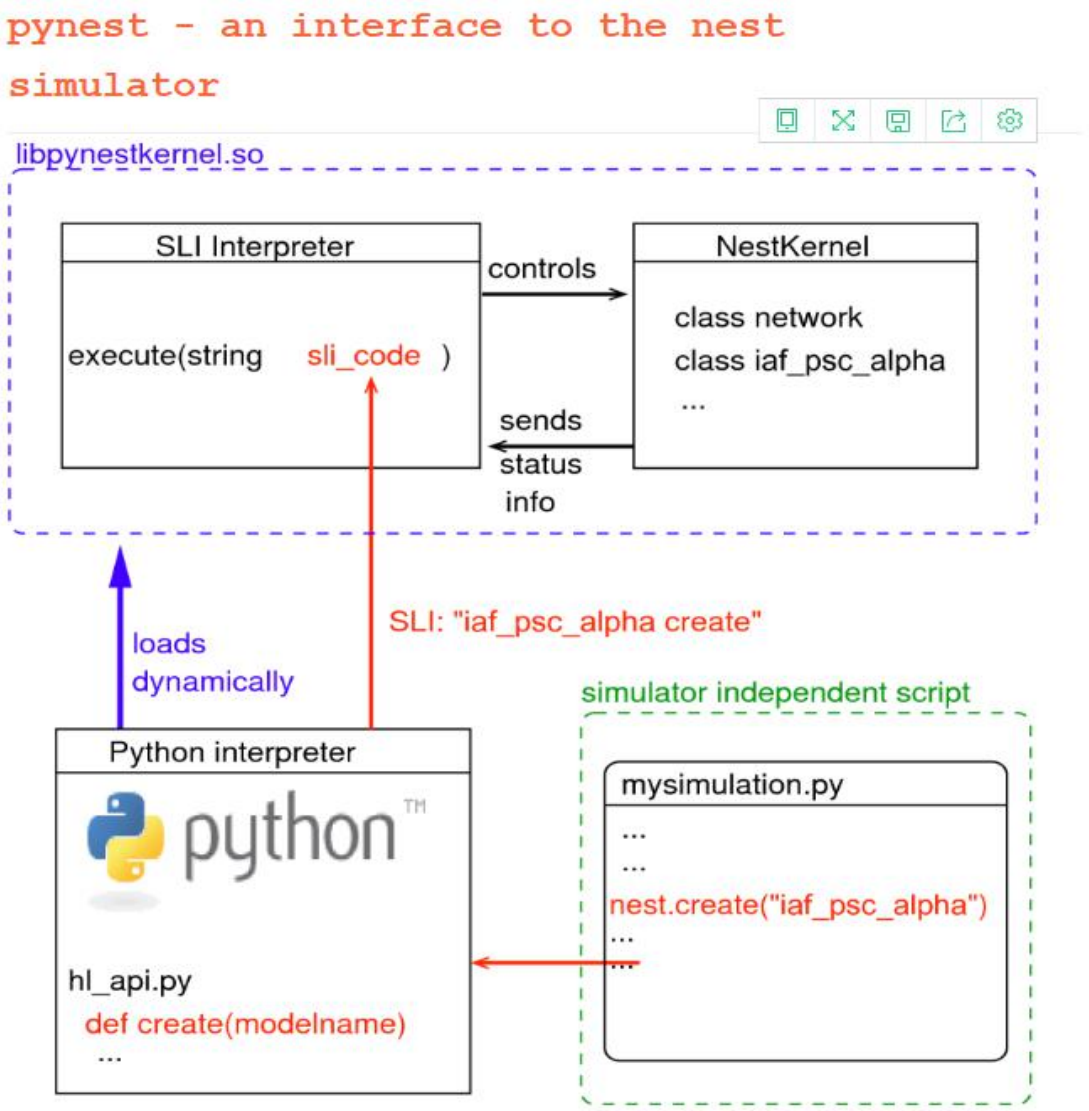
相变内存（PCM）,可变电阻式内存（RRAM），自旋传递扭矩磁性随机读写存储器（STT-MRAM）




芯片网络（networks-on-chip，NOCs）

# SNN工具 - Nest仿真器

- The Neural Simulation Technology Initiative
- 神经仿真工具（NEST：www.nest-initiative.org）
- 用于仿真大型的点神经元异构网络。
- 它是根据GPL许可发布的开源软件。
- 该模拟器带有Python 2的接口。
- 支持众多神经元模型、突触模型和复杂网络结构。
- Python提示或在ipython中交互使用PyNEST。



# 其他SNN工具

- The Brian Simulator
  - The Brian spiking neural network simulator
  - github项目地址：brian-team/brian2
  - Brian 是一个免费的开源模拟器
- BindsNET
  - A Machine Learning-Oriented Spiking Neural Networks Library in Python。
  - bindsnet是一个python包，它使用pytorch函数在CPU或GPU上模拟脉冲神经网络（SNN）。
  - bindsnet是一个脉冲神经网络模拟库，旨在开发用于机器学习的生物启发算法。
- 惊蜇(SpikingJelly)
  - 是一个基于 PyTorch ，使用脉冲神经网络(Spiking Neural Network, SNN)进行深度学习的框架。
  - https://git.openi.org.cn/OpenI/spikingjelly/
  - 使用PyTorch作为自动微分后端，利用C++和CUDA扩展进行性能增强，同时支持CPU和GPU。框架中包含数据集，可视化，深度学习三大模块。目前社区主要由北大媒体学习组和鹏城实验室人工智能中心运营管理。

# 问题

- SNN的相关工作原理?
- SNN类型分类?
- SNN转换法学习的相关工作原理?
- SNN无监督学习的相关工作原理?
- CNN与SNN的差异?