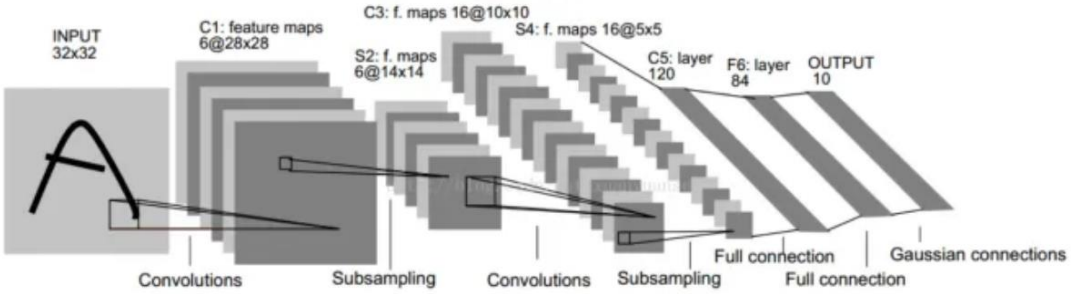
我们参考 Braincog 框架中提供的神经元节点模型以及转换器，将经典 LeNet5 卷积神经网络（CNN）转换为更为高效的脉冲神经元网络（SNN）模型。

LeNet5 网络介绍：

在本次项目中，我们使用的是 LeNet5 实现手写数字识别目标的。下面我将对 LeNet5 做出详细介绍。



LeNet5 的基本结构包括 7 层网络结构（不含输入层），其中包括 2 个卷积层、2 个降采样层（池化层）、2 个全连接层和输出层。

输入层：接收大小为的手写数字图像，其中包括灰度值（0-255）。在实际应用中，我们通常会对输入图像进行预处理，例如对像素值进行归一化，以加快训练速度和提高模型的准确性。

卷积层 C1：包括 6 个卷积核，每个卷积核的大小为，步长为 1，填充为 0。因此，每个卷积核会产生一个大小为的特征图（输出通道数为 6）。

采样层 S2：采用最大池化（max-pooling）操作，每个窗口的大小为，步长为 2。因此，每个池化操作会从 4 个相邻的特征图中选择最大值，产生一个大小为的特征图（输出通道数为 6）。这样可以减少特征图的大小，提高计算效率，并且对于轻微的位置变化可以保持一定的不变性。

卷积层 C3：包括 16 个卷积核，每个卷积核的大小为，步长为 1，填充为 0。因此，每个卷积核会产生一个大小为的特征图（输出通道数为 16）。

采样层 S4：采用最大池化操作，每个窗口的大小为，步长为 2。因此，每个池化操作会从 4 个相邻的特征图中选择最大值，产生一个大小为的特征图（输出通道数为 16）。

全连接层 C5：将每个大小为的特征图拉成一个长度为 400 的向量，并通过一个带有 120 个神经元的全连接层进行连接。120 是由 LeNet-5 的设计者根据实验得到的最佳值。

全连接层 F6：将 120 个神经元连接到 84 个神经元。

输出层：由 10 个神经元组成，每个神经元对应 0-9 中的一个数字，并输出最终的分类结果。在训练过程中，使用交叉熵损失函数计算输出层的误差，并通过反向传播算法更新卷积核和全连接层的权重参数。

该网络在 MINST 数据集中表现优秀，可以达到 98%以上的准确率。

MNIST 数据集介绍：

MNIST 数据集是手写数字识别数据集，由 60000 张训练图像和 10000 张测试图像组成。每张图像的大小为 28x28 像素，表示一个手写的数字。MNIST 数据集来自美国国家标准与技术研究所，National Institute of Standards and Technology (NIST). 训练集 (training set) 由来自 250 个不同人手写的数字构成，其中 50% 是高中学生，50% 来自人口普查局 (the Census Bureau) 的工作人员. 测试集(test set) 也是同样比例的手写数字数据。

在该项目中，我们直接使用 pytorch 的处理图像视频的 torchvision 工具集下载 MNIST 的训练和测试图片，数据集的图片是以字节的形式进行存储，在我们进行训练和测试时可以直接使用 torch.utils.data.DataLoader 进行加载。

转换细节：

1. 我们根据生物神经元的电生理特性，利用神经元动作电位的发放（Spike 脉冲）与细胞膜电位和阈值电压的关系，定义了 SpikeNode 类，用于模拟神经元真实的脉冲行为。相应的，我们还定义了动作电位和膜电位清除函数 clean_mem_spike，当更换训练样本批次时，通过调用该函数将 SpikeNode 的膜电位和 Spike 进行清零处理。SpikeNode 节点用于替换掉传统 CNN 网络中的 ReLu 和 LogSoftmax 激活函数。
2. 我们定义了 fuse 函数，该函数对卷积层的权重进行调整，按照批标准化的参数进行缩放和偏置，以融合批标准化层的信息，并创建新的融合后的卷积层。这样做的目的是将卷积层和批标准化层合并，减少模型中的参数数量，提高网络的推算效率。
3. 我们自定义了脉冲神经元的最大池化层：SmaxPool 类。该池化层可根据网络是否处于脉冲模式以及是否启用横向抑制机制进行不同的处理。在脉冲模式下，我们对累积的输入进行扩大操作，以强调输入中的最大值，并对其进行最大池化操作。随后将输入和经过扩大的累积输入相加，进行最大池化操作。最后将二者的差异作为输出。该设计可强调累积输入中的显著特征，同时考虑输入中的新信息。这更符合生物神经元中树突在时间上的刺激积累特性以及输入的重要性。此外，该池化层可以通过横向抑制特性操作减少累积的输入信息，从而更好地模拟了脉冲神经元之间的相互影响。
4. 根据以上转换规则，我们通过 fuse_norm_replace 函数将 LeNet5 中的卷积层、批标准化层、池化层以及激活层分别进行处理和替换。并在该函数中对全连接层进行权重和偏置的归一化操作。我们利用 hook 钩子函数记录的网络在前向传播过程中当前层和前一层的最大激活值，将当前线性层的权重按照前一层最大激活值的比例进行归一化。相似的，我们还对偏置和阈值电压进行处理，以确保脉冲神经网络中权重、偏置和激活阈值按比例

进行调整。

转换操作：

首先需要运行 lenet5.py，利用传统 LeNet5 网络对 MNIST 数据集进行训练，得到一个良好的 CNN 网络。然后运行 lenet_convert.py 加载权重并实例化转换器，便可以对转换后的脉冲神经网络进行推理。

代码解释：

lenet-convert.py

1. LeNet5 模型：LeNet5 类定义了标准的 LeNet-5 架构，用于图像分类。
2. SpikeNode 和 SMaxPool：这些是自定义模块，模拟脉冲神经元的行为和脉冲最大池化操作。
3. fuse_norm_replace 函数：该函数用于融合卷积层和批标准化层，将 ReLU 激活替换为 SpikeNode，并将最大池化层替换为 SMaxPool。
4. clean_mem_spike 函数：该函数在训练过程中更换批次时清除上一批次的内存和脉冲。
5. evaluate_snn 函数：该函数在 MNIST 测试集上评估 SNN 的准确性，测量随时间步骤的准确性，并可选择绘制结果。
6. 主要部分：代码的主要部分加载 MNIST 数据集，创建 LeNet5 模型的实例，并在测试集上评估 SNN 的准确性。还使用 fuse_norm_replace 函数对网络进行了一些修改，并将 SNN 的准确性与等效的人工神经网络（ANN）进行了比较。

lenet5.py

1. LeNet5 模型：定义了 LeNet-5 模型的架构，包括卷积层、批标准化层、ReLU 激活函数以及全连接层。
2. hook 函数：用于在每个前向传播过程中记录每一层的输出中占据 99%位置的值。这个函数是通过注册 forward hook 实现的。
3. train 函数：进行模型的训练，包括迭代训练集，计算损失，进行梯度反向传播，更新模型参数，并在每个 epoch 结束后评估在测试集上的准确性。
4. evaluate_accuracy 函数：评估模型在测试集上的准确性。
5. 主要部分：加载 MNIST 数据集，创建 LeNet-5 模型的实例，设置优化器和学习率调度器，然后调用 train 函数进行训练。最终，加载在测试集上表现最好的模型，并计算并输出其在测试集上的准确性。