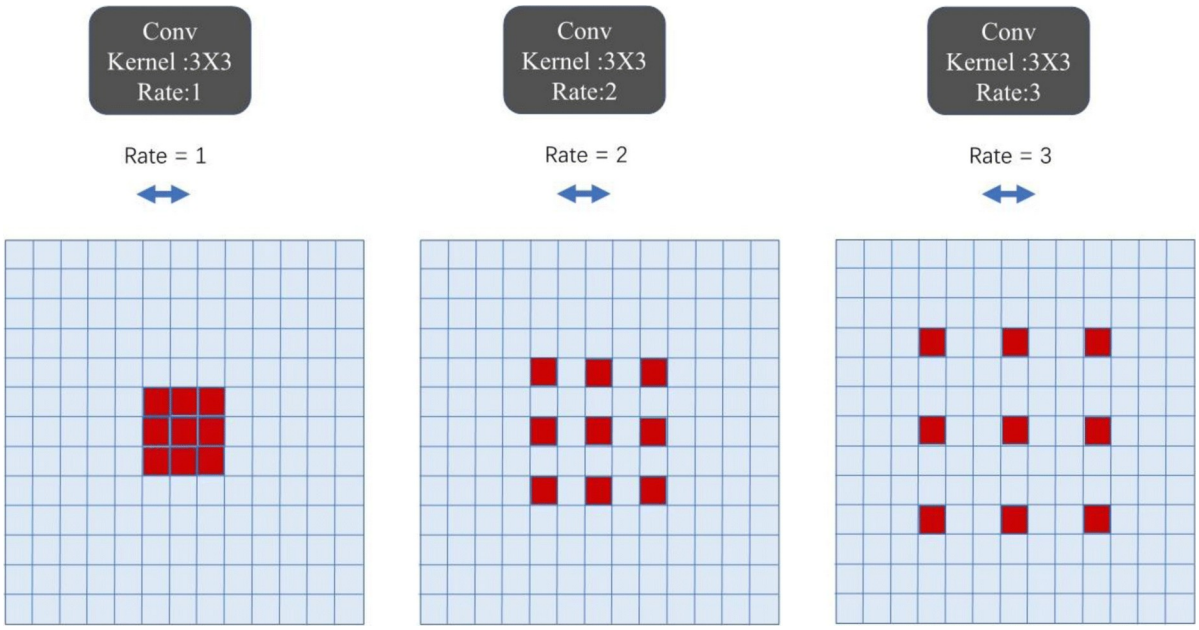
1) Честно говоря, исследование интернета на тему path-finding in ML создало впечатление, что данная тема не очень востребована. Все, что мне удалось найти -- reinforcement learning. Агент в некоторой среде осуществляет какие-либо действия, которые "поощряются" или "штрафуются" наблюдателем. Цель максимизировать поощрения. Т.е. в случае поиска путей мы будем поощрять модель за короткие пути и штрафовать за длинные.

2) Расширенные сверточные слои - сверточные слои с расширенными фильтрами, в которых в области между ненулевыми значениями фильтра заполняются нулями. На следующем рисунке формула для расширенной свертки в случае 1D.

$$y[i] = \sum_{k=1}^{K} x[i + r \cdot k] w[k]$$

Если r=1 (rate), имеем обычную свертку. При большей размерности получится следующая картинка:



Грубо говоря, расширенная свертка позволяет увеличивать поле зрения фильтра без увеличения числа параметров и количества вычислений. Можно сказать, что PC позволяет нам контролировать, как плотно вычислять отклики в FCN. Используется для semantic image segmentation (отнести каждый пиксель на картинке к какому либо классу.) Обычно объединяются в пирамиду, чтобы сеть лучше училась отличать объекты разных масштабов.

3) SGD

Единственное объяснение, почему при больших скоростях SGD лучше себя показывает для меня следующее: установив lr достаточно большим – мы “шагаем” широко (а с моментом еще шире). Адаптивные алгоритмы шагают “аккуратно”, но медленнее. Соответственно, если

функция достаточно “пологая”, широкие шаги нас быстрее приведут к минимуму. Однако, так можно минимум и “пропустить”.

4) Flatten

Была матрица на входе – получим длинный вектор. Строго говоря, flatten меняет форму тензора так, чтобы тот имел длину, равную числу элементов в нем (без учета batch dimension, мол сколько образцов – мы же не все изображения сразу схлопываем в вектор)