Глубокое обучение и pytorch

Свёрточные нейронные сети
- nn.Conv[1,2,3]d
- nn.BatchNorm[1,2,3]d

Цели
- Обучить бОльшее количество параметров, чем в классическом машинном обучении

Обучение
- Оптимальная остановка обучения
  - Смотреть на прогресс на последних k итерациях

Библиотеки
- torch
  - nn
    - functional
      - Библиотека, содержащая в себе все полезные функции (потерь, оптимизаторы, расписание lr и пр.)
    - Module
      - Данный класс содержит в себе структуру описания обновлений градиентов при форвард и бэквард-пассах
        - updateGradInput
        - updateGradOutput
    - Sequential
      - Класс содержит два метода для форвард и бэквард-пассов (банальное применение модулей/слоёв один за одним и в обратном порядке)
      - Наследуется от nn.Module (архитектура)
  - utils
  - torchvision
    - Набор датасетов для работы с компьютерным зрением

Torch
- Устройство обычной нс в pytorch
  - Каждая нс — отдельный класс, который наследуется от nn.Module
- Принцип работы модулей в nn.Sequential
  - Проходясь по каждому модулю в нашем блоке, мы производим требуемые операции из блока, затем после последнего слоя считаем функцию потерь, градиент, пропускаем градиент обратно.
  - ЕСЛИ какая-то из переменных весов обозначена как requires_gradient, то будут считаться градиенты для ВСЕХ наборов весов
  - Всё сохраняется в оперативку карточки, для того, чтобы *потом* вычислить backprop
    - Это и ведёт к проблеме нехватки памяти
- Не влезает *что-то* в сеть. Подходы:
  - Уменьшение батча («шардинг»)
    - Режем батч на равные части, прогоняем сетью, конкатенируем (?уточнить)
  - Чекпойнты градиентов
    - Расставляем некоторые «важные» точки — там будут сохраняться градиенты (и нигде больше).
      На этих промежуточных точках мы сохранили данные, всё, что между ними/после — не сохраняется.
      От этого мы выигрываем по памяти, но вычисления нужно производить на треть дольше (forward + backward + forward после бэкварда)
    - torch.utils.checkpoint
- Степени двойки для слоёв (размеры слоёв часто выбирают как степени двойки)
  - Сделано для того, чтобы эффективно вести параллельные вычисления
  - Структура процессоров изначально устроена так, чтобы обрабатывать за раз 128 или 256 бит (разрядность, в компьютерах это 32 и 64)
  - Если размер выбирать не как степень двойки, то работа модели замедлится — происходит паддинг нулями
- Сохранение модели
  - Сохранение модели «полностью»
    - Пригодится, если мы работаем онли в одной тетрадке как раз при остановке обучения (сохраняем при очередном улучшении качества, итерируемся дальше без сохранения до следующего улучшения)
    - Нужно описать сетку точь-в-точь как она сохранена (все модули и блоки)
    - torch.save(<modelname.pth>, '<pathname>')
    - Зачастую это плохой метод, ибо (?уточнить)
  - Сохранение параметров модели
    - torch.save_parameters_only()

NLP
- Лоссы
  - Cross-entropy
    - softmax + log transform on the end
  - Negative log-likelihood