Проект на тему: Свёрточные нейронные сети

КЛАССИФИКАЦИЯ ЦВЕТОВ С ПОМОЩЬЮ СВЁРТОЧНЫХ НЕЙРОННЫХ СЕТЕЙ

Величко Максим Иванович
maks99123vell@mail.ru | 334786

Задача

В работе я познакомился с различными архитектурами сверхточных нейронных сетей и их обучением на GPU (англ. graphics processing, графический процессор) на языке программирования Python 3 и фреймворка Torch (PyTorch).

В практическом задании необходимо я обучил еще одну сверточную архитектуру для задач классификации цветов.

В выбранной архитектуре также разобрался с основными её параметрами и принципами работы.

Выполнение задания

Теория

Для начала разберем готовую модель.

В начале требуется подключить необходимы библиотеки. Будут использоваться библиотеки для обработки данных (numpy, pandas), использования математических операций, вывода данных и фреймворк torch.

Далее подключим GPU google Colab для вычислений. После подключения необходимо соединить файлы с Google Drive с Google Colab.

Далее будет составлена сверточная нейросеть с 21 слоем. Будут использованы слои 2D свертки для обработки изображений Conv2D, слои для объединения сигналов и признаков MaxPool2D, слои для сглаживания Flatten и слои с функциями активации (Linear, Relu). Слои разделяются по свойствам изображений. Последний слой имеет 5 выходов для разделения цветов по 5 видам. После чего модель отправляется на GPU.

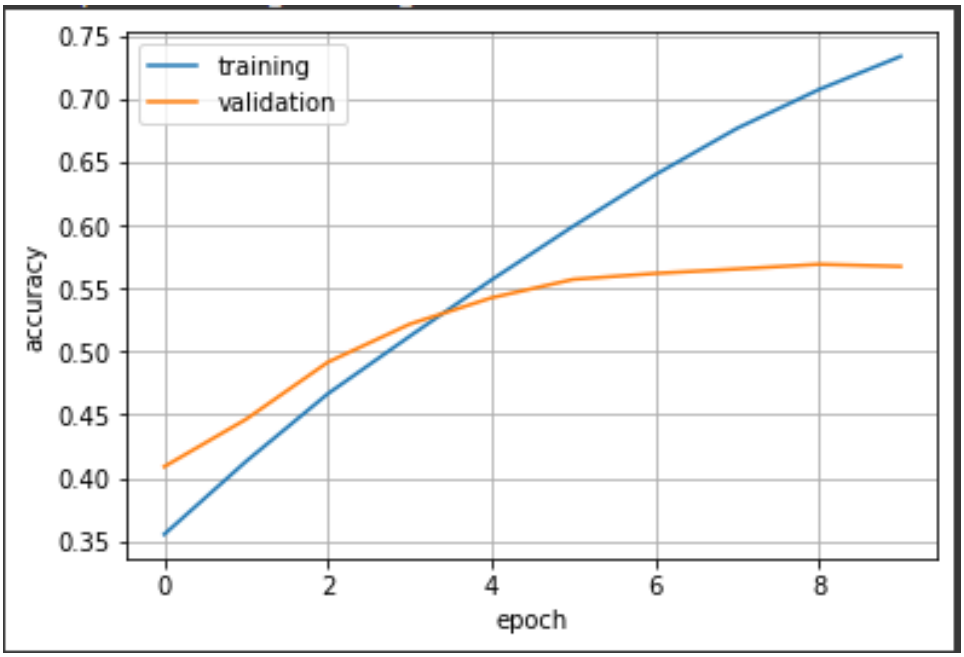
Следующим этапом является обучение. Для этого будут использованы размер мини-выборки 32, оптимизатор adam с коэффициентом скорости обучения 0.001.

Для работы с данными датасет будет разбит на train и validation. Будут заданы dataloader’ы (объекты для итеративной загрузки данных и лейблов для обучения и валидации). Также будут созданы функции для подсчета точности Accuracy и для обучения и валидации.

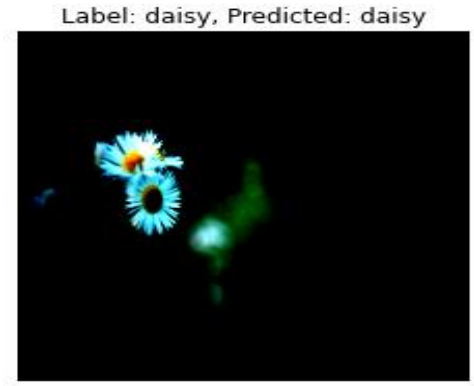
В следующем шаге мы можем заняться непосредственно обучением. Для обучения будут использованы вышеописанные функции и параметры с 10 эпохами обучения.

Несмотря на использования сторонних вычислительных мощностей, обучения длится около 30 минут из-за большого количества входных данных и сложности нейросети. По результатам обучения можно заметить тенденцию увеличения точности и снижения функции потерь.

Для оценки обучения использован ниже представленный график, созданный с помощью библиотеки matplotlib. График отображает зависимость точности от эпохи для тренировочного и валидационного сетов. Оба являются возрастающими.



Для сравнения предсказаний и реальных классов можем отобразить небольшой сет изображений.



Практика

В практической части была обучена ещё одна сверточная архитектура для задачи классификации цветов. В данной архитектуре были разобраны основные параметры.

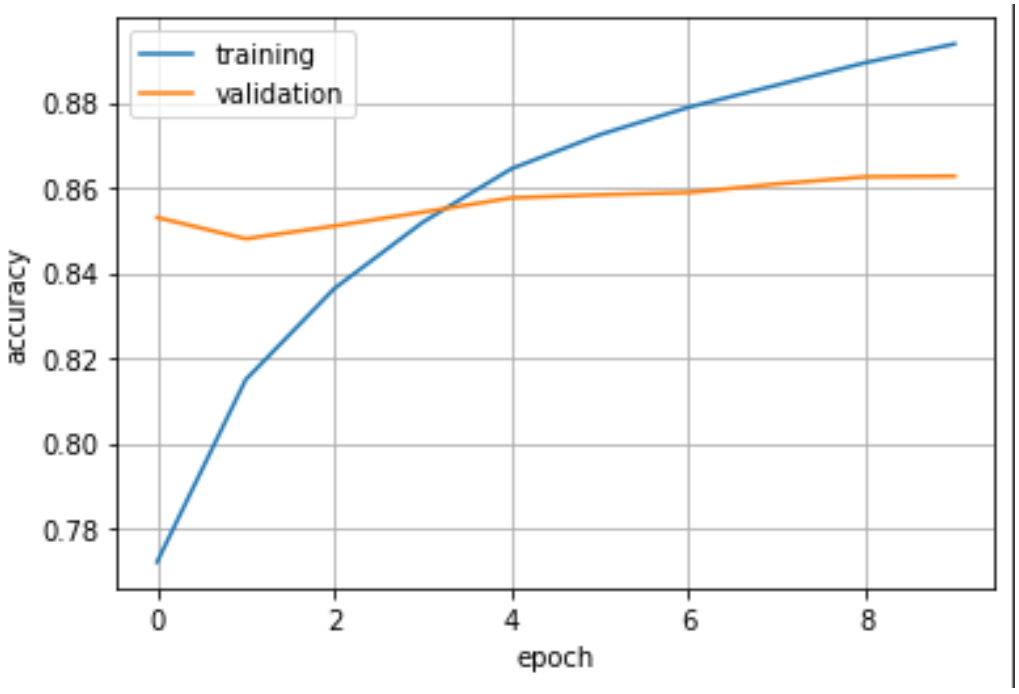
Была использована предобученная модель AlexNet из фреймворка pyTorch. Была использована функция заморозки весов модели. Далее был изменён последний слой модели на 5 выходов для классификации цветов по 5 видам.

После отправки модели на GPU и выбора параметров (алгоритм оптимизации SGD с коэффициентом 0.01, 10 эпох) мы можем обучить модель.

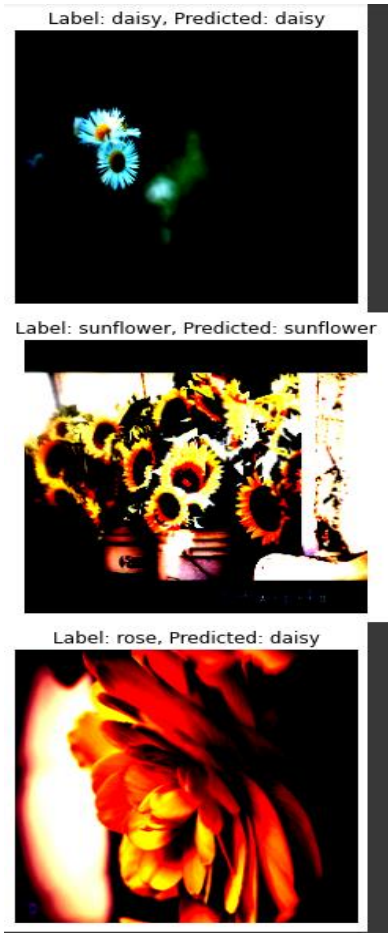
Время обучения модели сильно сократилось в связи с увеличением коэффициента скорости обучения оптимизатора и предобученности модели. Опытным путем выяснено, что время обучения сократилось на 23 минуты.

Оценка модели будет проходить образом, описанным в теоретической части.

Мы можем наблюдать возрастание точности для тренировочных данных и усреднение значений валидации в промежутке 0.85 – 0.87, что также выше, чем показатели модели в теоретической части. Кроме того, тенденция к повышению точности и снижению значений функции потерь сохранились, однако видно, что для валидации характер изменений непостоянный.



Ниже представлены некоторые изображения с предсказанием нейросети и фактическим видом. На данной малой выборке можно заметить лучшее определение вида (3 совпадения против 6).



AlexNet

Описание AlexNet. Архитектура является 12-слойной (3 полносвязных, 3 pooling слоя, слой нормализации, 5 сверточных слоев).

```
AlexNet(
  (features): Sequential(
    (0): Conv2d(3, 64, kernel_size=(11, 11), stride=(4, 4), padding=(2, 2))
    (1): ReLU(inplace=True)
    (2): MaxPool2d(kernel_size=3, stride=2, padding=0, dilation=1, ceil_mode=False)
    (3): Conv2d(64, 192, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
    (4): ReLU(inplace=True)
    (5): MaxPool2d(kernel_size=3, stride=2, padding=0, dilation=1, ceil_mode=False)
    (6): Conv2d(192, 384, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (7): ReLU(inplace=True)
    (8): Conv2d(384, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (9): ReLU(inplace=True)
    (10): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (11): ReLU(inplace=True)
    (12): MaxPool2d(kernel_size=3, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
  (avgpool): AdaptiveAvgPool2d(output_size=(6, 6))
  (classifier): Sequential(
    (0): Dropout(p=0.5, inplace=False)
    (1): Linear(in_features=9216, out_features=4096, bias=True)
    (2): ReLU(inplace=True)
    (3): Dropout(p=0.5, inplace=False)
    (4): Linear(in_features=4096, out_features=4096, bias=True)
    (5): ReLU(inplace=True)
    (6): Linear(in_features=4096, out_features=1000, bias=True)
  )
)
```

Используются слои из ранее описанной нейросети (Conv2D, MaxPool2D, Relu, linear) и слои Dropout (исключение). Из особенностей можно выделить нелинейность Relu, отсеивание половины нейронов.

Данная модель работает по принципу параллельной нейросети. Это значит, что происходит параллельная обработка данных на нескольких узлах. Таким образом большинство задач разделены на множество меньших, которые решаются параллельно. К параметрам относятся размер пакета 128, предобученность, отображение прогресса и вес.

Свёрточная нейронная сеть является специальным видом нейронной сети, использующейся в основном для распознавания образов. Идея таких сетей заключается в чередовании сверточных слоев и слоев предвыборки. Таким образом, главное отличие от полносвязной нейросети заключается в использовании свертки и пулинга.

Transfer learning (трансферное обучение) - подраздел машинного обучения, целью которого является применение знаний, полученных из одной задачи, к другой целевой задаче. Можно сказать, что это передача процесса обучения с одного набора данных на другой.
Предобученая нейросеть – это нейросеть, которая до процесса обучения уже способна определять какие-либо признаки.

Функция заморозки весов служит для запрета изменения весов предобученой модели. Это нужно для того, чтобы во время обучения модель не меняла веса, которые она получила как предобученая. Без заморозки предобученая модель будет совершать большие ошибки в классификации.

Блок “Сверточная сеть с нуля” показывает часть кода, где прописываются слои нейросети. Сначала выбирается последовательная модель фреймворка PyTorch. Далее внутри модели задаются слои нейросети. Сверточные слои Conv2d выполняют операцию свертки (каждый фрагмент изображения умножается на матрицу свёртки поэлементно, а результат суммируется и записывается в аналогичную позицию выходного изображения) с помощью заданных параметров. in_channels определяет количество карт во входном изображении, out_channels – количество карт в выходном слое, kernel_size – размер матрицы, stride – шаг свёртки, padding – внутренние отступы со всех сторон элемента. На основе этих параметров определяется размер фильтра и карта признаков.
Пулинговый слой MaxPool2D используется для снижения количества параметров для обучения путем объединения. Его параметры kernel_size и stride используются для настройки новой карты признаков. nn.MaxPool2d(2, 2) использует карту 2x2.
Последняя часть блока использует Flatten для сглаживания непрерывного интервала в тензор; ReLu для отсеивания отрицательных значений, их заменой на 0; Linear для вычисления оценки по каждому классу.