Lecture 6

Очень большую роль в развитий computer vision сыграл датасет ImageNet. И челлендж, связанный с ним ILSVRC. Это соревнование по классификации на 1000 классов. Обучения на 1.2M, валидация 50k, 100k тестирование. И в основном архитектуры, которые мы тут рассматриваем архитектуры, занимали первые места в этом соревновании. И трюки, подходы, которые принесли выигрыш этим моделям интересно рассмотреть на самих же моделях. В начале

Classical solution

Основной лейтмотив в Deep Learning это создание мощных представлений за счет, на которых строится небольшое решение. Сейчас это нейронные сети, трансформеры, которые создает очень мощные представления данных. Раньше небыло способности создавать интересные представления и поэтому люди генерировали признаки достаточно вручную.

Одним из примером является SIFT (Scale-invariant feature transforms) vector. Мотивация следующая допустим у нас есть детектор углов, который может детектировать углы только на определенном масштабе. Тогда если мы приблизим или отдалим наш детектор уже не отработает корректно. Поэтому надо сделать так, чтобы мы могли детектировать изображения на разных масштабах. Зачем это нам нужно? Это нужно для того, чтобы мы могли классифицировать объекты на различных масштабах. Далее эти признаки тоже отправлялись дальше на классификацию после чего мы получаем результат классификации. Далее мы расскажем о том, как строились оригинальный alexnet

SIFT vectors.
https://www.robots.ox.ac.uk/~vedaldi/assets/pubs/simonyan13deep.pdf

AlexNet

Одна из первых сетей, которая показала эффективность использования была AlexNet. Это считается одна из первых успешных сверточных сетей, которые были применены. И показали, что эффективность предыдущих методов, завязанных на ручном извлечении признаков и большом числе трюков. Как решение это композиция сверток, активаций и пулингов. В начале можно увидеть, что у нас идёт свертки 11x11, 5×5, которые позволяют извлечь большое число информации из изображения повышая таким образом receptive field. Из аугментации применили dropout + augmentation (random crop, random flip, color augmentation).

Свертки max-pool. Relu активации. То есть она из первых сетей, которые превзошла классические подходы.

Они показали, что ReLU активации позволяют добиться такого же качества, что и tanh, но для этого требуется в шесть раза меньше времени.

Также они использовали Test-time augmentations. Вместо того, чтобы делать предсказание по исходной картинки. Они сделали 10 аугментированных версий (5 кропов и horizontal flip)

VGG

В следующем году на этом соревновании победила VGG Архитектура сети очень простая и отвечает подходу AlexNet - последовательность сверток с активациями, затем пулинг и так далее мы строим модель. Но так как модель победила в следующем году были сделаные полезные модификации. Первое - все свёртки 3x3. В чем суть? Как мы обсуждали ранее существует понятие receptive field. Это эквивалетно тому, какую часть от исходного изображения видел искомый пиксель. И получается, с точки зрения receptive field 2 свертки 3×3 эквивалетны свёртке 5x5. Но при этом мы имеем меньшее число параметров. За счет этого мы и выигрываем. Второе это обучение модели последовательно. Было 6 стадий. На каждой мы берем обученную архитектуру с предыдущего этапа и добавлялся новые обучаемые слои.

Inception

В этой архитектуре озаботились вопросом как выбрать оптимальные размеры свёртки. Возможно стоит на глубоких этапах применять свертки 5x5 и только на начальных этапах полезно делать свёртки 3x3. Поэтому авторы вместо того, чтобы думать какой размер оптимальный применять различные методы гипероптимизации добавили свёртки всех размеров и результат есть конкатенация. Теперь нейронная сеть может определять как ей выгоднее работать с изображением. Для того, чтобы уменьшить число фильтров после конкатенации они применяли 1x1 свертки. Суть их в следующем: используя 1x1 свёртки мы не меняем разрешение, но мы можем уменьшить число каналов и тем самым упростить модель. У нас было 1024 каналов, мы применяем 1x1 conv и получить 256 каналов. Меньше памяти + меньше параметров. Можно делать также 1x1 conv перед большими свертками, чтобы уменьшить сложность модели. Дополнительная мотивация в том, что признаки изображения очень похожи, поэтому в чем смысл использовать более точную свертку, если результат все равно останется скореллированным.

Из интересных решений использование промежуточных выходов моделей. До этого времени не использовали residual connection и их в принципе не было. То есть для такой глубокой сети актуальна проблема затухания градиента. Как еще решить? Давайте сделаем промежуточный выход. За счет этого градиент протекает на конечные слои

Inception как работа получила продолжения в виде второй и третьей версий. Идея упрощения сверток продолжилась и свертку 5x5 уже заменили на 2 свертки 3x3, а сама свертка 3x3 была замена на композицию 3x1 и 1x3. Pooling с точки зрения сохранения полезной информации неэффективен поэтому. Хотелось сохранить какую-то информацию, но использование обычных сверток слишком дорого. Поэтому авторы предложили сделать решение на пополам. Дополнительно они предложили использовать label smoothing. Мотивация у него следующая: зачем нам учить сеть давать более сильные предсказания если она уже уверена 0.9? Поэтому давайте сделаем так чтобы мы учили сетку не давать везде 1, а быть чуть неуверенным, таким образом, высвобождая ресурсы на то, чтобы выучить более сложные кейсы.

ResNet

Сейчас мы достигли моделей, у которых 22 слоя. И в этом случае возникают проблемы с таких моделей. Какие вы можете назвать?

- Затухание градиента

Вот на рисунке продемонстрировано, что модель в 18 и 34 слоя имеют одиннаковое качество, что является очень неэффективным. Мы хотим делать более глубокие сети и за счет этого улучшать качество, потому что есть тренд на увеличение глубины (более глубокая модель в теории должна давать более лучшее качество). Вдобавок, если мы работаем с изображениями размера не 32x32 (CIFAR10), а 512x512, то нам потребуется намного больше свёрток и слоёв, чтобы достигнуть большого receptive field.

И для решения этой проблемы была предложена архитектура ResNet и ее основной трюк residual layer. Он заключается в том, что при обучении таких глубоких нейронных сетей у нас возникают проблемы с затуханием градиентом. Вдобавок, мы хотим чтобы не терять эффективности 18 слоёв модели и хотим, чтобы добавление слоёв было аддитивным (например, как VGG). И для этого эффективен residual layer. Он заключается в том, что мы делаем Identity преобразование, таким образом мы сохраняем мощность модели как 18 слойной, но также позволяем градиенту протекать дальше.

Если что-то работает хорошо, а ResNet работал хорошо, то люди задумались: "а можно ли сделать его лучше?". И тут они начали эксперименты с тем, как модифицировать residual block для того, чтобы получить лучшее качество. На слайде видны различные модификации, которые были сделаны. То есть авторы попробовали поменять порядок нормализации, активации и свёрток, чтобы найти оптимальный рецепт. И авторы нашли, что простое identity работает эффективнее всего. Эффект достигается за счет того, что нет затухания градиента и поэтому модель работает лучше.

В дальнейшем были рассмотрены различные модификации residual layer, потому что его можно по разному реализовать и в результате была обучена сеть размером 1001 слой. Которая уже не сейчас не SoTA, но она показывает, что такая возможность существует.

Xception

В продолжении с темой упрощения сверток. Рассмотрим стандартизированный Inception module. В чем его суть? мы делим фильтры на несколько частей и каждый обрабытываем отдельно, а затем, чтобы учесть взаимодействия между фильтрами мы применяем 1x1 конволюции. А теперь остановимся и задумаемся, что мы учим используя свертки? Мы учим взаимодействия между spatial (width, height) and channel каналами. Inception module показывает нам, что в принципе нам не нужно полное взаимодействие между каналами и таким образом: "cross-channel correlations and spatial correlations are sufficiently decoupled that it is preferable not to map them jointly". И на этом базируется идея Xception.

А что если довести по предела идею разделения spatial information и channel information. Мы отдельно сворачиваем каждый канал свёрткой. Затем мы учитываем взаимодействие между каналами с помощью 1x1 свертки. Таким образом, мы получаем depthwise separable conv, эффективно работает с изображениями.

MobileNet

Исследовательский интерес хочет делать модели с лучшим качеством, но что если мы хотим использовать нейронные сети на компактных девайсах. Тут нам нужен маленьких размер моделей и их быстрота. Тут нам на помощь и приходит deptwise-separable conv с 1x1 conv. Они заменили стандартную свёртку на композицию 1x1 и затем depthwise. Можно посчитать сколько выигрыша это даёт нам по сравнению со стандартной свёрткой. Расчеты на слайде. Если взять число каналов как N и размер свертки 3, то мы выигрываем в ~9 раз.

MobileNet v2

В следующей версии авторы добавили residual connection и модифицировали основной блок. Сначала мы расширяем пространство каналов. Делаем там depthwise/pixelwise свертки и обратно проецируем в пространство меньшей размерности. Residul блок зарекомендовал себя очень хорошо поэтому добавление его это было лишь вопросом времени.

MobileConv хотел улучшить, что означает сокращение вычислений. А самое простое это сокращения числа каналов. Но если мы сократим число каналов, то сеть станет менее выразительной. Еще если мы будем использовать ReLU, то у нас в пространстве малой размерности очень быстро выродится значения. Чтобы этого не допустить предлагается использовать expanding lyer, который решает эту проблему.

Пример как ReLU вырождает данные на слайде.

EfficientNet

Эффективность нейронных сетей очень важна и известно, что улучшить качество модели можно увеличить число слоёв, число фильтров и разрешение модели. Они взаимосвязаны, то есть вместе с увеличением разрешения, нам хотелось иметь больше фильтров. И авторы заинтересовались этим варианто и предложили схему подбора оптимального соотношения между параметрами. Чтобы упростить поиск среди различных конфигураций, авторы предложили подобрать оптимальные параметры, чтобы увеличить размер модели в 2 раза. Затем просто увеличить модель в 4, 8, 16 раз. Схема работы видна на слайде. Мы хотим максимизировать качество используя следующее ограничение. Затем авторы равномерно увеличивали число параметров, следуя этому соотношению и получили улучшение качества. Авторы хотели показать как превзойти MobileNet поэтому в качестве main блока использова MBConv.

Во второй версии модели авторы стали вообще жуками И решили напрямую оптимизировать значения функции от скорости обучения, качества и размера параметров. Также они добавили новый блок в NAS (FusedMBConv), из-за того, что на ранних стадиях MBConv не эффективен и поэтому обычная конволюция может работать быстрее. Также они реализовали progressive training. Постепенно увеличивая размер изображения и усложняя аугментации.

ViT

И напоследок мы рассмотрим как можно применять трансформеры к задачам computer vision. Авторы статьи хотели, минимальным методами модифицировать трансформер, чтобы он работал эффективно. До visual transformer были различные попытки это сделать (например, ImageGPT от OpenAI). Но основная проблема в том, что если рассматривать пиксели в качестве токенов, то матрица attention получается слишком большая. (Рассматривая изображение размером 512x512 мы получим матрицу внимания размером (512×512)^2). Поэтому авторы предложили нарезать исходное изображение на патчи. Таким образом мы получаем матрицу attention, которая иметь уже допустимые размеры. Затем, чтобы уменьшить сложность модели аналогично 1x1 conv они применили линейный слой, чтобы уменьшить размерность токенов. И таким образом они +- адаптировали модель для задачи трансформера. Они еще поделали эксперименты с positional encoding, пытаясь внедрить 2d близость, но в итоге 1d learnable encoding оказался лучше и плюс минус давал близость соседних объектов.

Почему трансформер лучше чем CNN. Основная эффективность заключается в том, что трансформер не теряет эффективность при росте числа данных, а становится только эффективней. Одна из причин в том, что CNN имеет неявную регуляризацию. У нас есть инвариантность к трансляции, учитывается больше взаимодействие ближайших пикселей. Трансформер в свою очередь не имеет таких особенностей и поэтому он плох на малых датасетах, но с ростом числа данных его эффективность возрастает. Так как он не несёт никаких biases

Ссылки

https://vitalab.github.io/article/2017/03/21/xception.html

https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf

https://towardsdatascience.com/review-xception-with-depthwise-separable-convolution-better-than-inception-v3-image-dc967dd42568

https://machinethink.net/blog/mobilenet-v2/

https://napsterinblue.github.io/notes/machine_learning/computer_vision/inception/

https://patrick-llgc.github.io/Learning-Deep-Learning/

https://iaml-it.github.io/posts/2021-04-28-transformers-in-vision/

https://habr.com/ru/post/302242/

Статьи

https://arxiv.org/pdf/1603.05027.pdf

https://arxiv.org/pdf/1610.02357.pdf