ViT Project

В данной работе реализован подход, описанный в статье **An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale** https://arxiv.org/pdf/2010.11929.pdf.

Идея: применить модель на основе трансформера вместо сверточных сетей для задачи классификации изображений.

Архитектура сети

Архитектура модели подробно показана в статье:

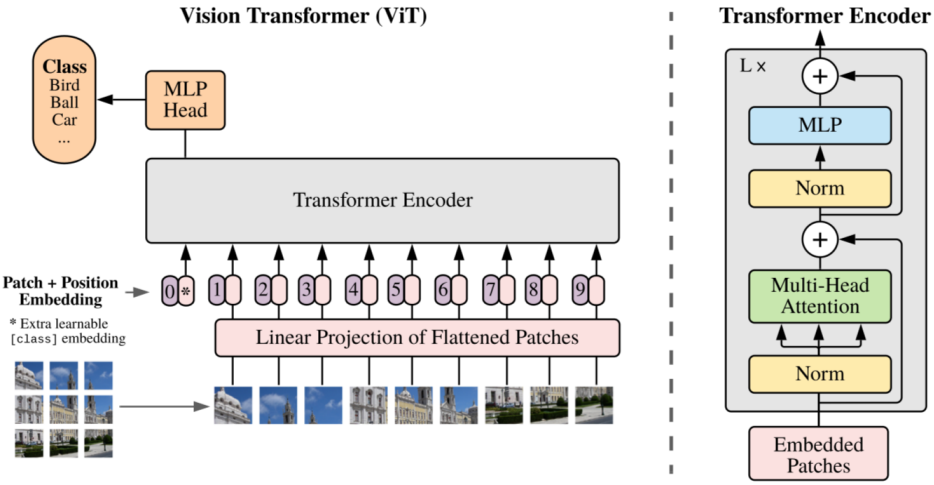

Рис. 1: Архитектура модели

Входное изображение нарезается на патчи некоторого размера (в статье предложено 16x16 пикселей), затем патчи вытягиваются в вектор и некоторым образом линейно преобразуются (Linear Projection), на выходе после Linear Projection к данным патчам приклеиваются обучаемые эмбединги «class» и прибавляются позиционные эмбединги, и затем такие последовательности подаются на вход Transformer Encoder. Внутри Transformer Encoder состоит из последовательных L блоков, на схеме показан один такой блок. Сначала применяется слой нормализации LayerNorm, потом идет Multi-Head Attention, LayerNorm и MLP. В блоке есть residual connections, они показаны черными стрелками. После выхода Transformer Encoder есть слой MLP Head, который отвечает за классификацию изображений.

Реализация

Интерфейс модели реализован с помощью классов на pytorch. Классы Attention (multi-head attention), TransformerEncoderBlock реализованы по формулам из статьи. Некоторые особенности реализации, которые не указаны в статье были подсмотрены в официальной реализации от авторов:

https://github.com/google-research/vision_transformer. Например, это инициализация некоторых обучаемых эмбедингов, в частности параметры позиционного эмбеддинга в официальной имплементации инициализированы нормальным распределением.

Обучение модели и эксперименты

Авторы обучали модель на огромном частном датасете JFT-300M dataset и тратили на этого тысячи TPUv3-core-days. Ввиду дефицита вычислительных ресурсов и памяти для хранения и обработки таких датасетов, параметры модели и объем данных были модифицированы под имеющиеся вычислительные возможности.

В статье авторы сранивают несколько моделей ViT с разным набором параметров: ViT-Base, ViT-Large, ViT-Huge.

Model	Layers	Hidden size D	MLP size	Heads	Params
ViT-Base	12	768	3072	12	86M
ViT-Large	24	1024	4096	16	307M
ViT-Huge	32	1280	5120	16	632M

Рис. 2: Детали модели

Для своих экспериментов я взяла модель Base и умножила значения параметров модели (Layers, Hidden size, MLP size, Heads) на 2/3. Авторы сравнивают модели на разных датасетах: ImageNet, CIFAR и др, я решила обучить с нуля свою модель ViT на TinyImagenet и посмотреть на качество. С одной стороны это не слишком большой датасет, с другой - достаточно разнообразный. Train часть датасета состоит из 200 классов ImageNet, в каждом классе 500 изображений 64x64.

Для обучения модели написана функция train, в которой сохраняется лучшая модель на валидации. Эксперименты и метрики train_loss, val_loss, val_accuracy логировались в neptune.ai, каждый эксперимент можно открыть по ID номеру и посмотреть по ссылке:
https://app.neptune.ai/calistro/vit/experiments?viewId=standard-view.

Было проведено несколько экспериментов: в первых экспериментах параметры задавались, как в статье: weight_decay=0.03 в оптимизаторе и dropout=0.1. Оказалось, что это слишком сильная регуляризация, и сеть не училась, accuracy было меньше 6 процентов за большое число эпох. В следующих экспериментах использовался lr_sheduler в оптимизаторе, и регуляризация была уменьшена до weight_decay=0.001, стало намного лучше, сеть начала учиться, за 100 эпох accuracy=0.25 на валидации (эксперимент VIT-10). lr_sheduler реализован в классе WarmupSchedule как в статье: первый процент от всех итераций обучения lr растет с нуля до 1e-3, после чего lr линейно уменьшается.

В последнем эксперименте (эксперимент VIT-12) я решила поэкспериментировать с размером патча, сделать его меньше: 8x8, поскольку TinyImagenet имеет невысокое разрешение 64x64 и патчи размера 16x16 могут быть слишком большие для него. Также я совсем убрала weight_decay, поставила dropout=0.05, а значения гиперпараметров ViT-Base умножила на 1/2 (см. таблицу ниже), чтобы компенсировать увеличение патчей, обрабатываемых трансформером.

Это получился самый успешный эксперимент, он обучался 150 эпох на batch_size=200 и достиг accuracy=0.38. Однако модель достаточно быстро достигла пика качества, потом стала переобучаться. Скорей всего модели хорошо помогло уменьшение размера патча, а чтобы не было переобучения можно было поставить небольшой weight_decay. Тем не менее для модели, обученной с нуля, это хороший показатель, сравнимый с обучением классических сверточных сетей.

Таким образом была протестирована рабочая реализация метода, при обучении модели на других датасетах надо настраивать параметры.

Проведенные эксперименты:

Experiment	Layers	Hidden size	MLP size	Heads	Patch size
VIT-1,2,4,7,10	8	512	2048	8	16
VIT-12	6	384	1536	6	8