Отчет по курсовой работе на тему

Оптимизация нейронных сетей для решения задачи классификации изображений

Выполнил студент группы P42182
Плюхин Д.А.

Описание проекта

Название проекта: Оптимизация нейронных сетей для решения задачи классификации изображений.

Постановка задачи: Выбор архитектуры нейронной сети и демонстрация использования методов оптимизации для ускорения ее работы на выбранном датасете с сохранением точности.

Состав команды: Плюхин Дмитрий Алексеевич, студент группы P42182.

Ссылка на репозиторий (см. ветки master и resnet):
https://github.com/zeionara/swift-models

Использованные технологии

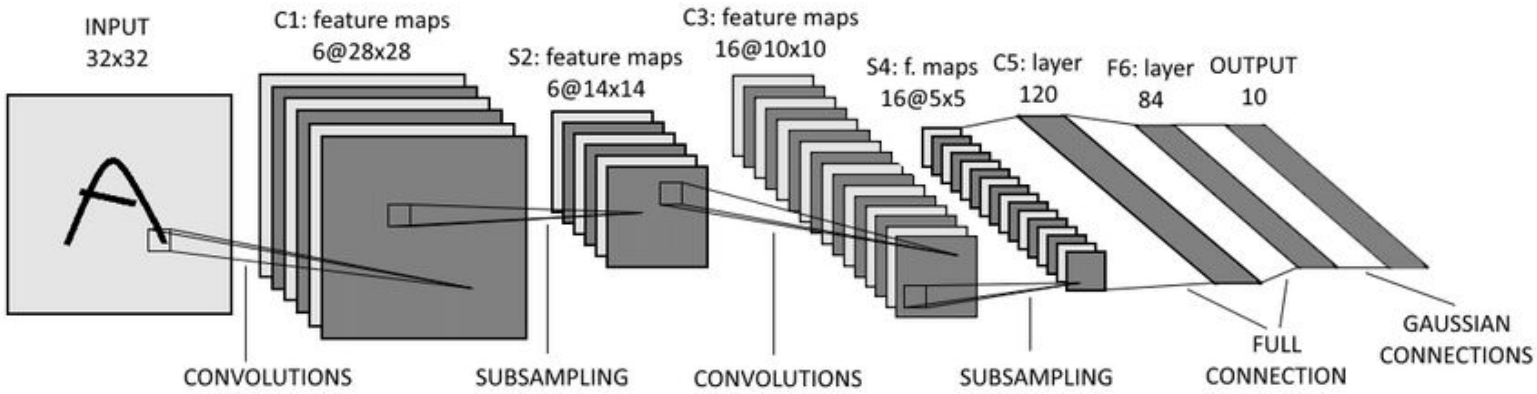
В ходе работы использовался фреймворк **Tensorflow** для языка программирования **Swift**, реализация методов оптимизации производилась путем расширения модулей обучения и тестирования моделей нейронных сетей из библиотеки **swift-models**.

Использованные модели

LeNet-5

Классическая архитектура сверточной нейронной сети.

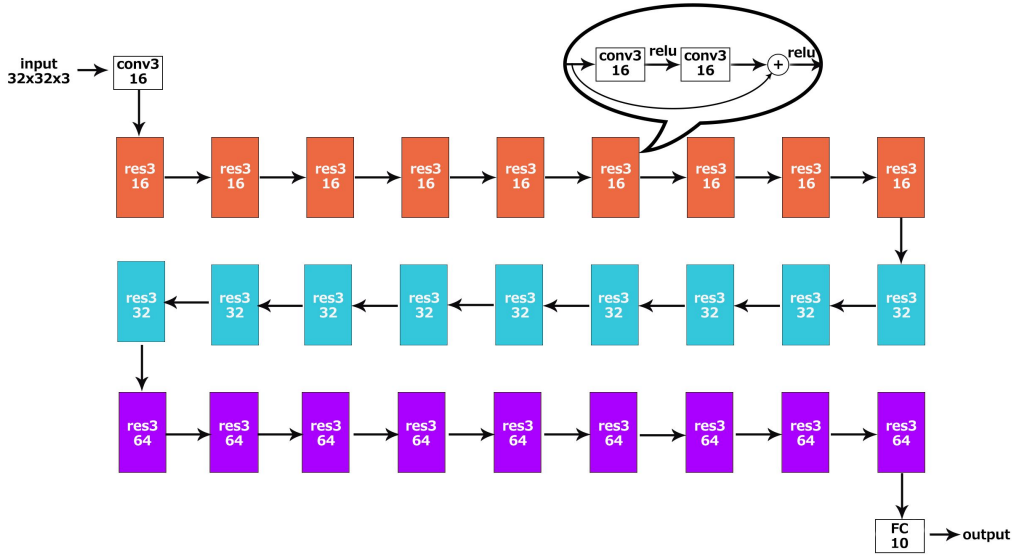
Представляет собой простую последовательность чередующихся слоев свертки и пулинга (в работе использовалась функция подсчета среднего значения).



ResNet-56

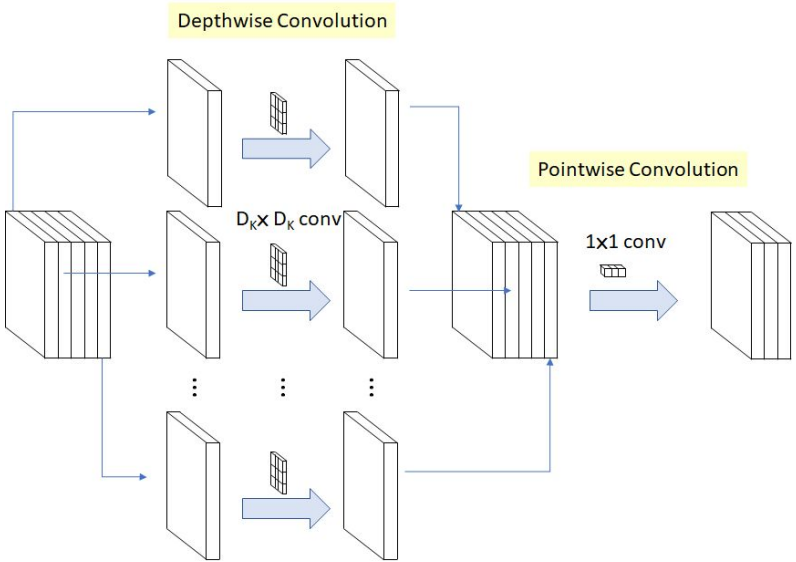
Модель с архитектурой ResNet, демонстрирующая эффект использования residual связей и предназначенная для решения более сложных задач.

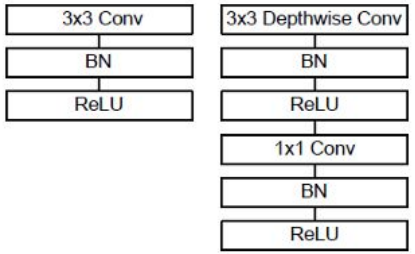
В отличие от LeNet-5, состоит из большого количества слоев свертки.

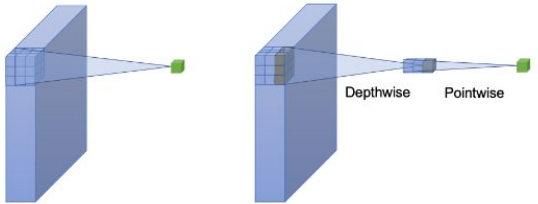


MobileNetV1

Архитектура нейронных сетей, оптимизированная для использования на устройствах с ограниченными ресурсами за счет реализации идеи “Separable convolutions”, позволяющей снизить количество параметров.





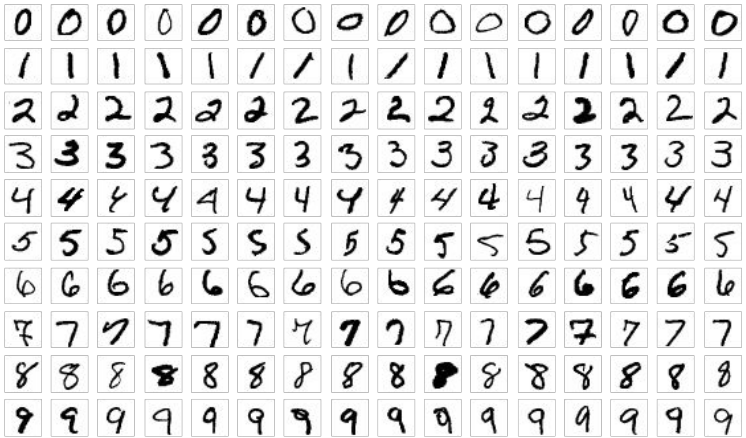


Использованные датасеты

MNIST

Простой датасет, предназначенный для демонстрации обучения и тестирования алгоритмов обработки изображений.

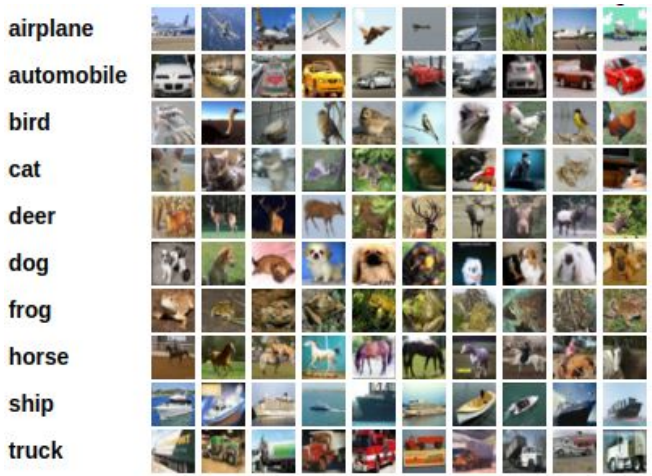
Состоит из 70 000 изображений рукописных цифр (60 000 составляют обучающую выборку, оставшиеся 10 000 - тестовую).



CIFAR-10

Датасет, представляющий собой набор цветных изображений малого размера, каждое из которых относится к одному из 10 классов.

Состоит из 60 000 файлов (50 000 составляют обучающую выборку, оставшиеся 10 000 - тестовую).



Результаты

Базовые модели и методы, использованные в процессе оптимизации:

LeNet-5 (для оценки точности использовался датасет MNIST):

1. Дистилляция;
2. Снижение разрядности весов;
3. Использование precompiled kernels;
4. Применение более простых функций активации;
5. Использование GPU вместо CPU.

ResNet-56 (для оценки точности использовался датасет CIFAR-10):

1. Дистилляция;
2. Использование precompiled kernels;
3. Использование более экономичной модели (MobileNetV1);
4. Использование GPU вместо CPU.

Дистилляция (LeNet)

При дистилляции модели LeNet с одновременным применением снижения разрядности весов (используется тип данных Float вместо Double) и применении более простой функции активации (relu вместо gelu), удается добиться существенного ускорения обучения и тестирования модели:

	time (seconds)			time multiplier (teacher vs student)	accuracy			accuracy multiplier (teacher vs student)
	teacher	student	student without a teacher		teacher	student	student without a teacher	
training (10 epochs)	449.977	191.255	84.828	2.352759405	0.9831	0.9787	0.9801	1.00449576
inference (10 epochs)	1.9012	0.4047	0.4379	4.69780084				
training (50 epochs)	2310.531	933.906	410.348	2.474050922	0.9886	0.9871	0.9864	1.001519603
inference (50 epochs)	1.887664	0.412416	0.415505	4.577087213				

Использование GPU вместо CPU (LeNet)

При запуске исходной и дистиллированной моделей на GPU удается ускорить обучение до 3 раз, а тестирование - до 6.

	time (seconds)		time multiplier	accuracy		accuracy multiplier
	cpu	gpu		cpu	gpu	
teacher (training)	750.952	235.443	3.189527826	0.9847	0.9835	1.001220132
teacher (inference)	5.9152	0.9836	6.013826759			
student (training)	316.126	111.172	2.843575721	0.9772	0.9776	0.9995908347
student (inference)	1.0662	0.3544	3.008465011			

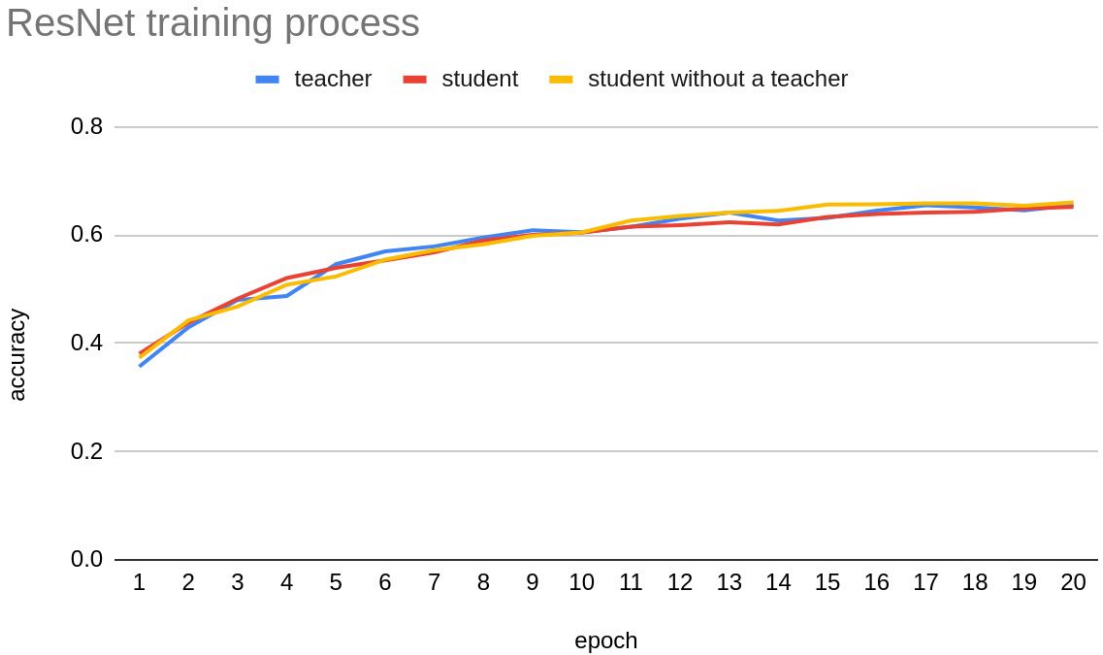
Дистилляция (ResNet)

При дистилляции модели ResNet удается добиться существенного ускорения обучения и тестирования модели, а применение GPU средней мощности дает больший эффект, чем использование гораздо более мощного процессора.

	time (seconds)			time multiplier (teacher vs student)	accuracy			accuracy multiplier (teacher vs student)
	teacher	student	student without a teacher		teacher	student	student without a teacher	
training (10 epochs on i7)	1210.7036	934.37595	879.6833	1.295734977	0.6184	0.583	0.6266	1.060720412
inference (10 epochs on i7)	**23.9271**	**17.20142**	18.69336	1.390995627				
training (20 epochs on xeon gold)	490.96794	436.61277	328.62594	1.124492854	0.6301	0.648	0.6665	0.9723765432
inference (20 epochs on xeon gold)	14.11345	**9.74661**	9.76775	1.448036805				
training (20 epochs on gtx 1650)	371.83965	300.6013	259.01433	1.236986167	0.6552	0.6517	0.6603	1.005370569
inference (20 epochs on gtx 1650)	5.98935	**4.15199**	4.32435	1.442525151				

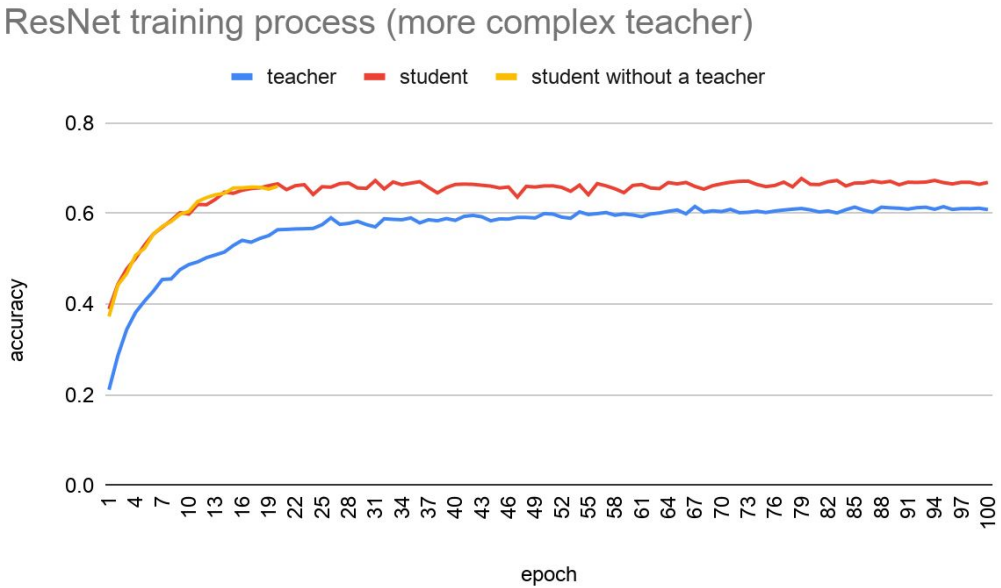
Дистилляция (ResNet)

Графики зависимости точности от номера эпохи имеют одинаковый вид, после 15 эпохи происходит насыщение модели и прекращается заметный рост точности.



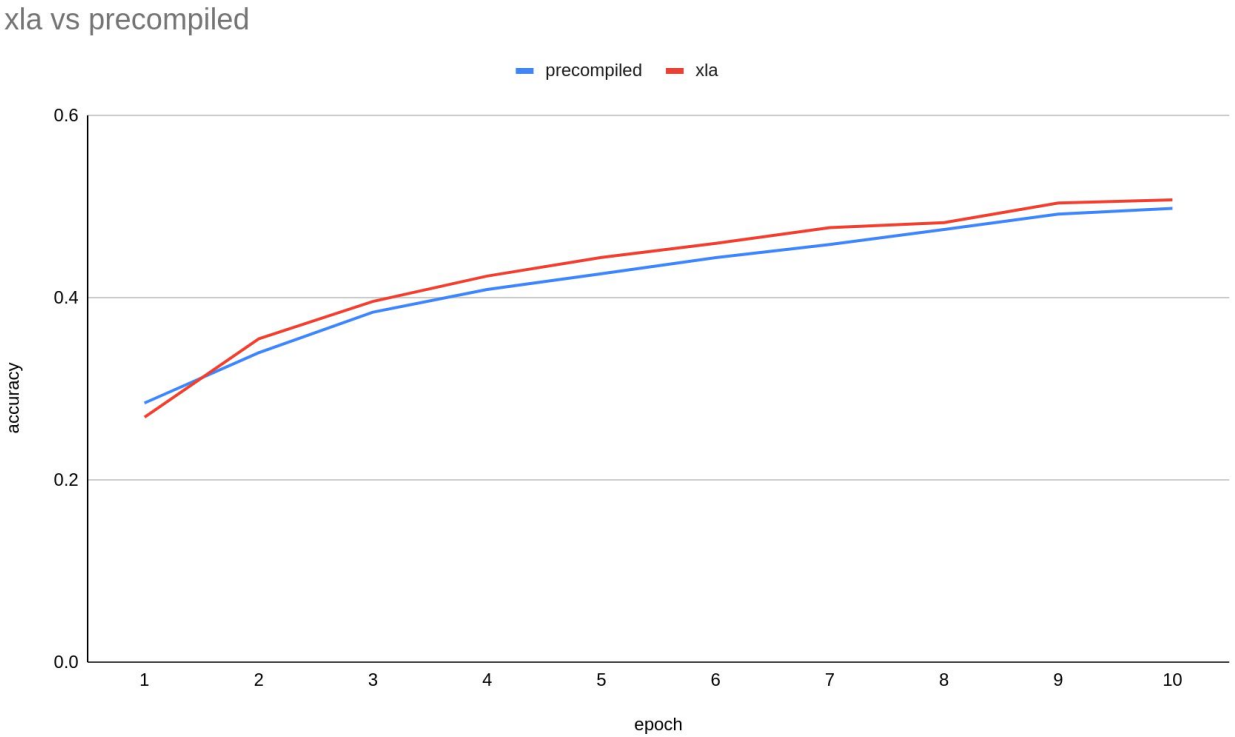
Дистилляция (ResNet)

Использовании более мощной версии ResNet в качестве “учителя” не позволяет улучшить результаты. При этом training time сокращается более чем в 2 раза (447.64154 / 196.06485), а inference time - почти в 3 раза (8.21510 / 2.77490).



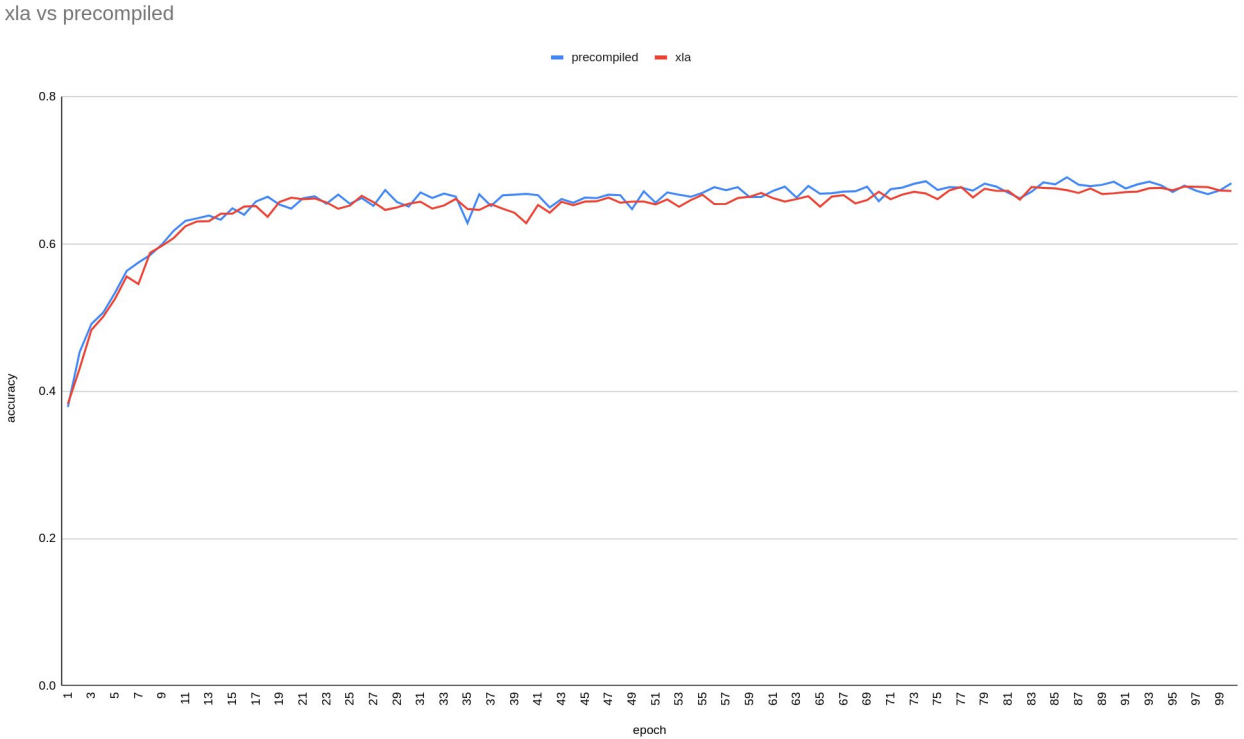
Использование precompiled kernels (ResNet)

При использовании precompiled kernels в случае запуска модели на **CPU** inference time снижается почти в 4 раза (95.76320 / 25.10940), training time - в 2.6 раза (1299.80991 / 490.84959), а точность остается такой же.



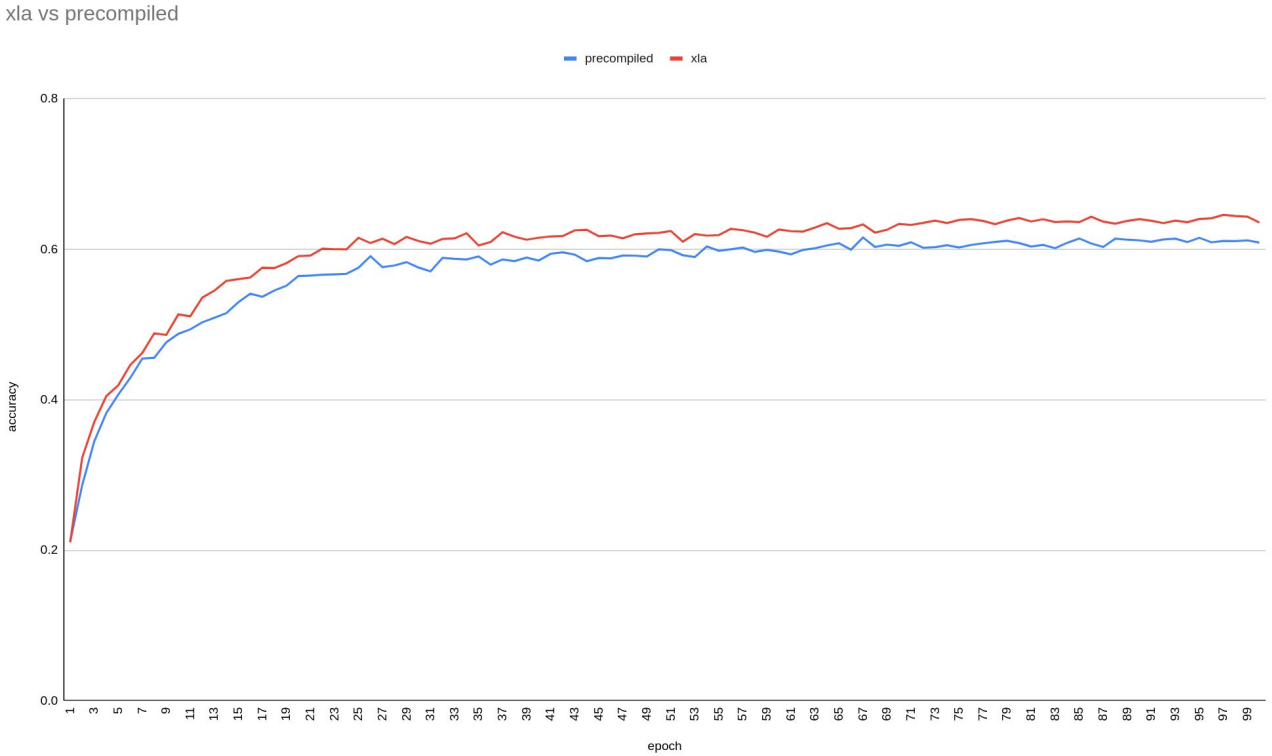
Использование precompiled kernels (ResNet)

При использовании precompiled kernels в случае запуска модели на **GPU** inference time снижается почти в 1.7 раз (7.47482 / 4.33609), но training time **повышается** более чем в 2 раза (230.56995 / 111.77132), а точность остается такой же.



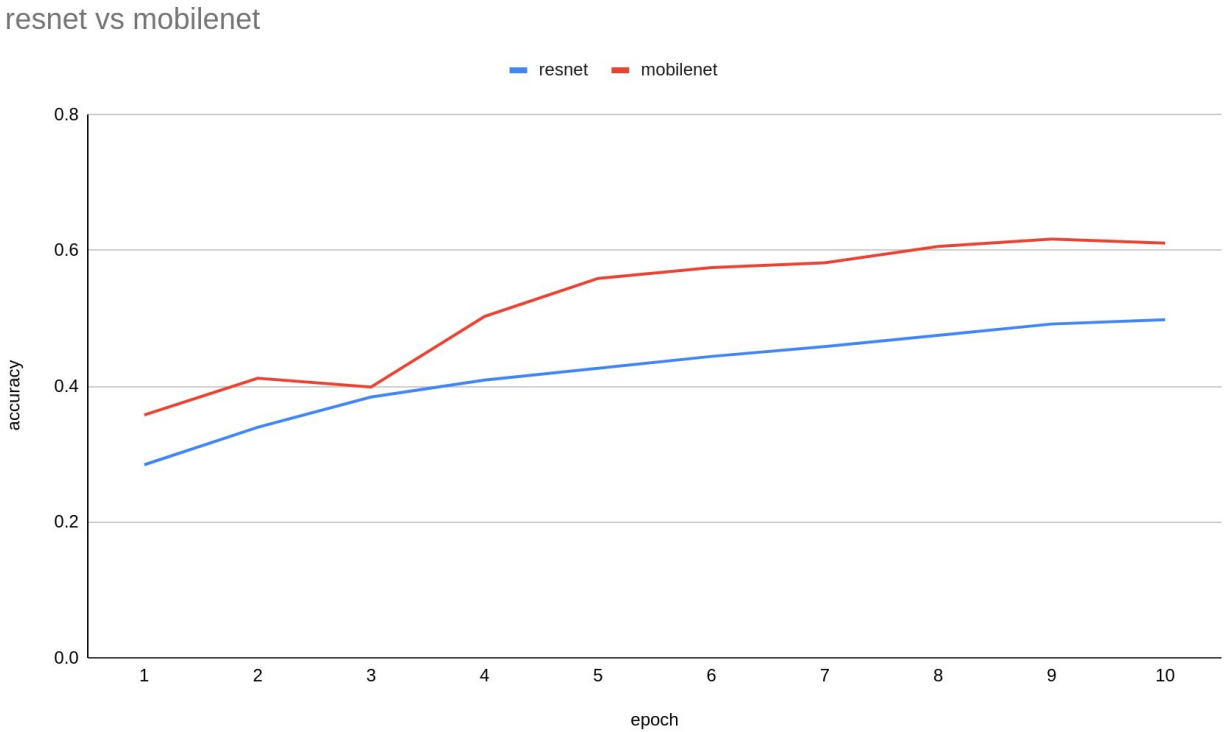
Использование precompiled kernels (более “тяжелый” вариант ResNet)

При использовании precompiled kernels в случае запуска модели на **GPU** inference time снижается более чем в 2 раза (18.47336 / 8.21510), но training time **повышается** в 1.8 раз (447.64154 / 247.84829), точность остается приблизительно такой же.



Использование более экономичной модели (ResNet)

При использовании более экономичной модели inference time снижается почти в 4 раза (24.089 / 6.085), а сопоставимой точности удается достичь уже после 4 эпохи.



Выводы

1. Наибольший эффект получается путем использования модели с более компактной архитектурой;
2. Применение GPU заметно ускоряет обучение модели независимо от мощности используемого до этого CPU;
3. Применение компиляторов для сборки пользовательских kernels (например, XLA) не всегда приводит к хорошим результатам;
4. Дистилляция модели может привести к менее высоким результатам, чем обучение более простой модели без учителя.

Спасибо за внимание

Ссылка на репозиторий проекта: https://github.com/zeionara/swift-models
Ссылка на гугл-таблицу с результатами измерений: http://bit.ly/nno-project-data
Ссылка на директорию с логами запусков моделей: https://github.com/zeionara/swift-models/tree/resnet/logs