Efficient models: Продвинутый прунинг

Влад Голощапов @GradientWitnesses @kraidiky

>>> Пределы возможного

Дано:

Модель: GPT-2 10M от Карпаты, типоразмера 6d (6 голов, 6 слоёв, 6*64 эмбединг) 256 длинна контекста
Датасет: 40 тысяч строк из произведений Шекспира (tinyshakespeare: 10M токенов train, 100K val)
Может быть обучена: до perplexity 4.20, 2000 итераций, дальше начинается переобучение
Прореживаем: неструктурированным прунингом с дообучением, не одинаково по слоям, все уловки.

Вопрос:

На сколько процентов сеть можно проредить пожертвовав лишь незначительной частью perplexity?

Делайте ваши ставки!

Дано:

Модель: GPT-2 10M от Карпаты, типоразмера 6d (6 голов, 6 слоёв, 6*64 эмбединг) 256 длинна контекста
Датасет: 40 тысяч строк из произведений Шекспира (tinyshakespeare: 10M токенов train, 100K val)
Может быть обучена: до perplexity 4.20, 2000 итераций, дальше начинается переобучение
Прореживаем: неструктурированным прунингом с дообучением, не одинаково по слоям, все уловки.

Вопрос:

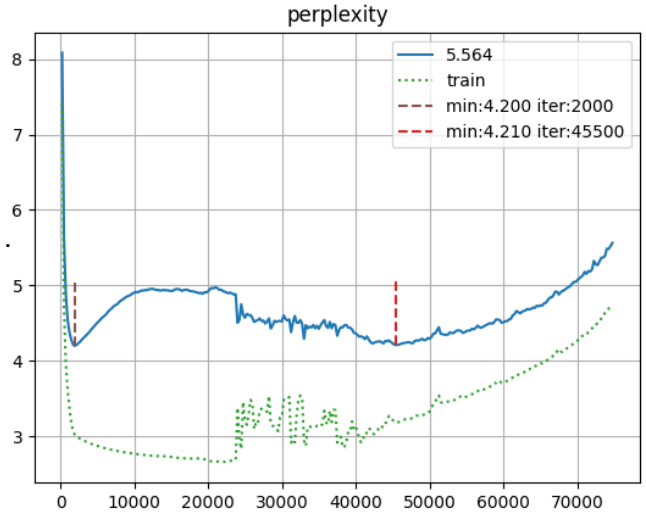
На сколько процентов сеть можно проредить пожертвовав лишь незначительной частью perplexity?

Ответ:

step: 45500
train loss: 1.1587752103805542
val loss: 1.4421539306640625
ppl: 4.210293769836426
zeros: 99.14%/x116.3/zeros:10,652,684 active:92,404

Что на этом можно выиграть?

Память: 3/160, инференс: всё слжоно, трейн: все ещё сложнее…



>>> Разновидности разреженности

Разреженность весов бывает разная:

- Спарс-аттеншен, и другие аттеншены, без которых современные LLM невозможны.
- Специальная 2:4 и 4:16, и т.п.
- Блочная разреженность постоянная (Диагональньная, Батерфляй, и тому подобное)
- Блочная разреженость переменная.

Прунинг тоже бывает разный:

- Структурный, приводящий к уменьшению размеров матриц, что, кстати, не всегда даёт экономию.
- Неструктурный, который вообще плохо поддерживается на железе.
- Блочный, поканальный, понейронный или посвёрточный, как частный случай.

Разреженность делится на:

- Полностью поддерживаемую на железе и софте
- Не поддерживаемую в данный момент
- Плохо поддерживаемую или находящуюся в работе (тут неструктурная или 4:16 на будущих Хуавеях)

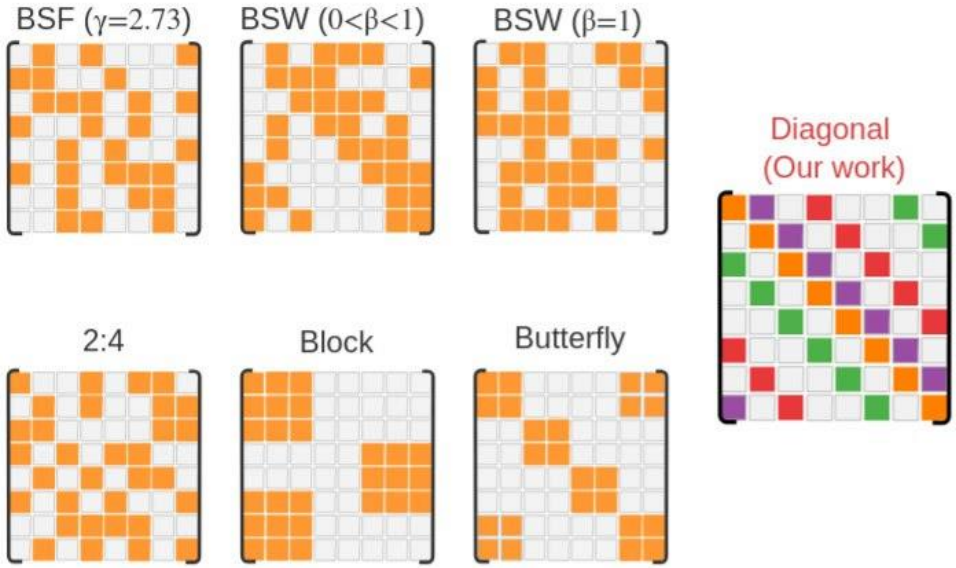
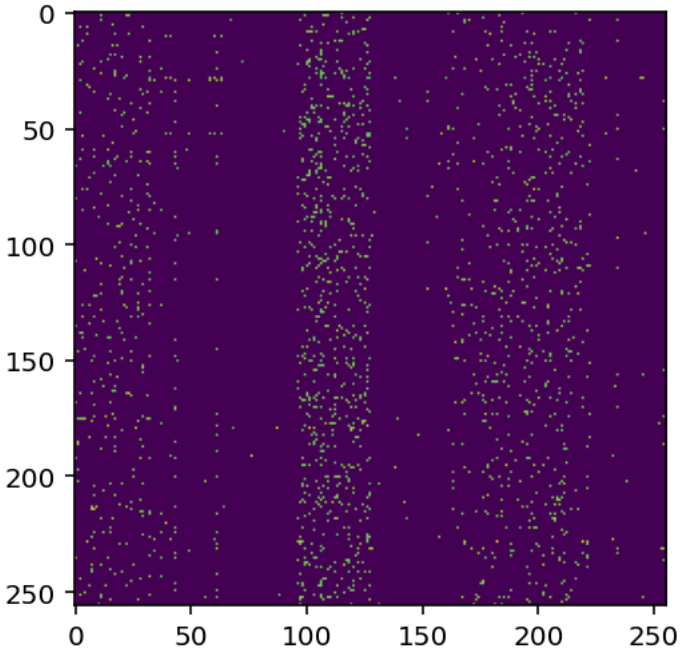


Figure 2: Overview of different sparsity patterns from the literature used in sparse training methods. Bipartite Scale-Free (BSF) and behavior of Bipartite Small-World (BSW) networks with varying β is explained in Apdx. I



Аттеншен матрица 3М сети на Шекспире

https://arxiv.org/pdf/2506.11449

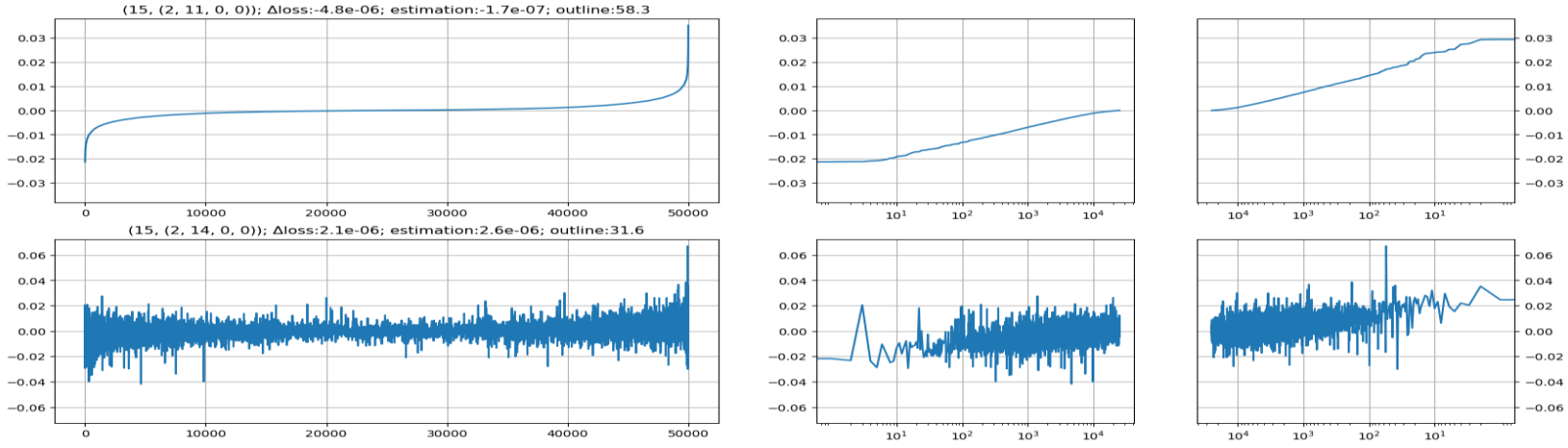
>>> Бесполезны ли бесполезные веса?

Проверим влияние весов на работу сети:

- MobileNetV3 на ImageNet
- Обнуляем один единственный вес средний во всех отношениях и смотрим как изменится loss на 50K валидацинной выборки. Сортируем по изменению.
- Видим, что для ~10 картинок влияние ~2% loss, для ~100 влияние 1%, для ~1000 ,5%, и т.д.

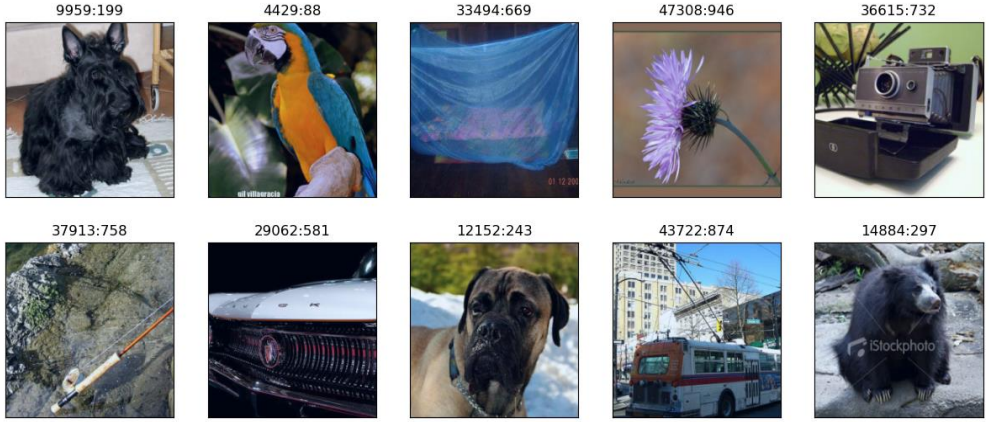
Выводы:

- Вес очень не бесполезен для небольшой части тестовой выборки.
- Те, для кого он полезен и те для кого он вреден находится примерно в балансе в результате обучения.
- Но очень важен он только для части тестовых примеров.
- Два веса одного и того же нейрона могут быть значимы для разных примеров.



Для этих вес полезен

Для этих вес вреден

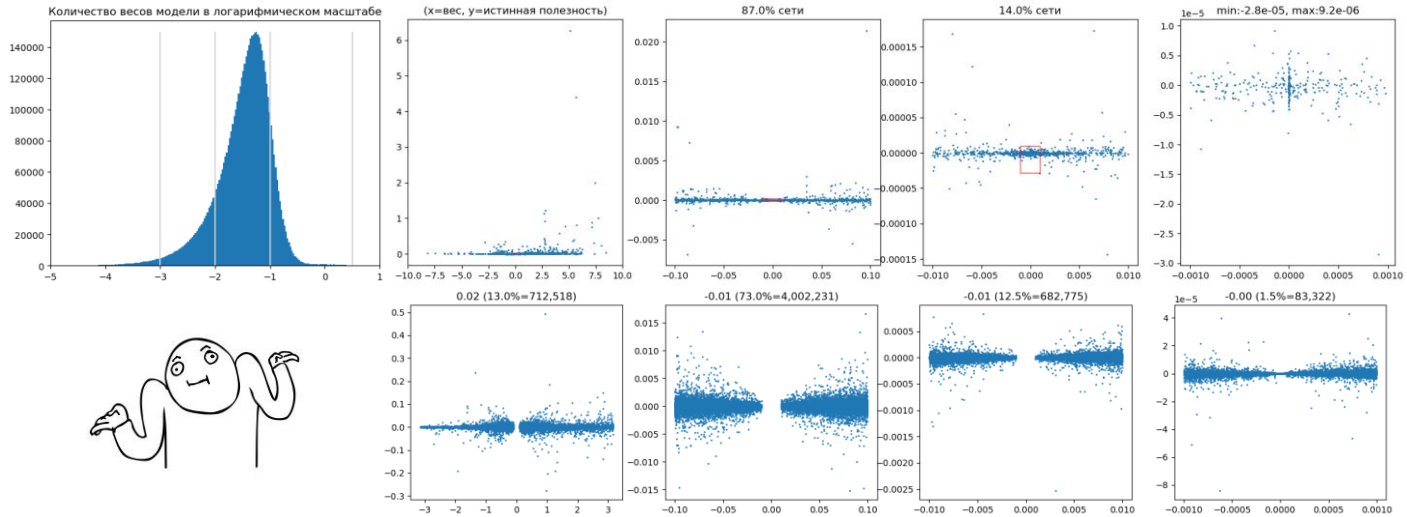


>>> w*dLoss/dw (wdw)
правильное решение?

Истинная полезность – проверка предположений:

- Истинная полезность – то на сколько изменится loss если обнулить отдельно взятый вес, нет другого способа проверить себя.
- Расчёт на трейне безумно дорог, будем проверять валдейшене.
- На пробу посчитаю истинные полезности для 5219 весов разных характерных размеров. На validation выборке чтобы сравнить с wdw метрикой на том же validation. ~1 минута на вес.

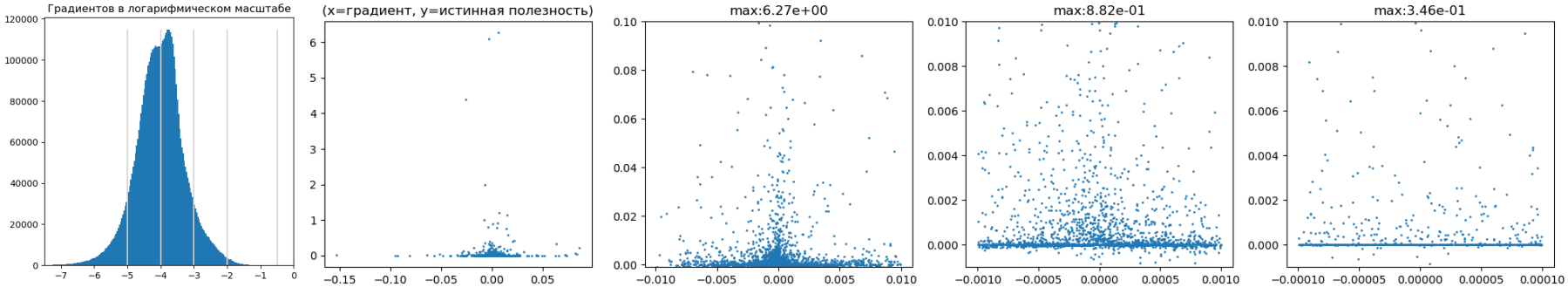
Магнитудный метод:



- Маленький вес – означает малую полезность, но большой вес не значит ничего.
- Есть много работ, где предполагают, что большие веса – важные, и не проверяют этого.
- Маленькие веса составляют небольшую часть сети, что требует диковинного дообучения и регуляризации.

Градиентный метод

- В некоторых статьях рассматривают даже предположение, что у кого меньше градиент, тем и лучше он подходит для удаления.
- Метод реализован в уважаемой библиотеке SparseML и ещё много где.

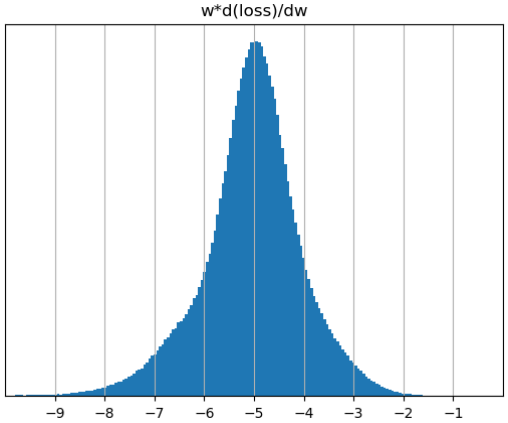


- Градиент с полезностью хоть и связан, но как-то совсем не очевидно. Важных весов даже кажется больше среди тех, у кого градиент маленький.
- Градиент потребуется при разморозке, но для заморозки или удаления весов сам по себе он кажется мало полезен.

Другие методы

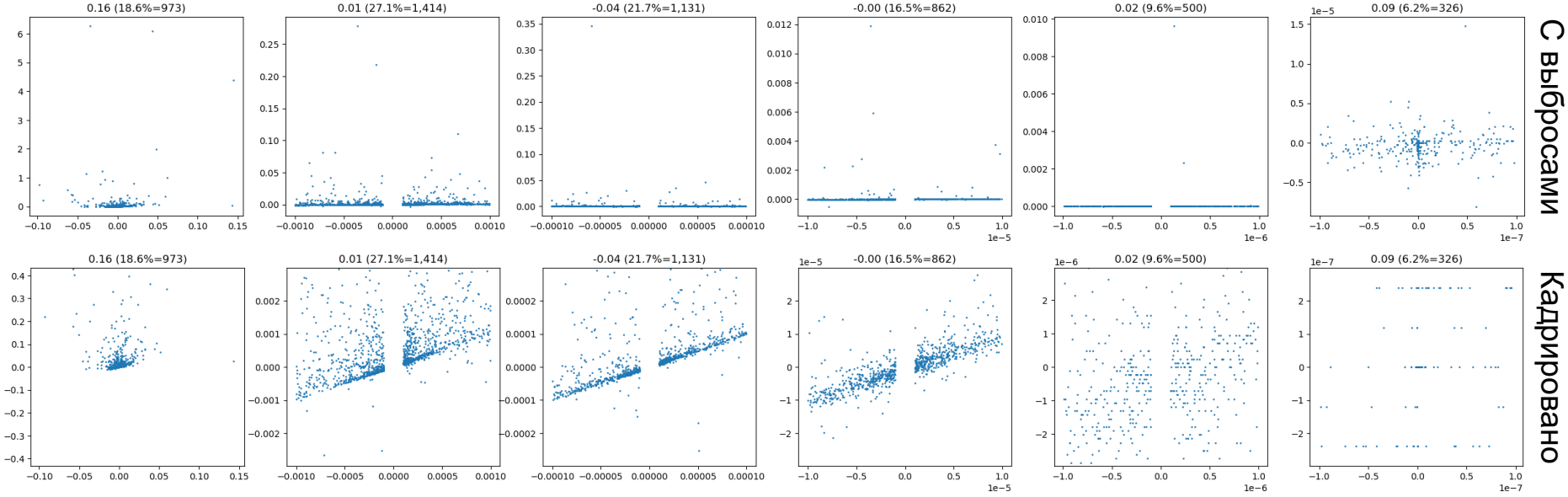
- Существует ещё много странноватых методов, использующих, например, крайне трудно и, обычно, приближённо считающийся гессиан и другие критерии. Любой из них можно проверить аналогичным способом.
- Проверим один из них, который называют Тейлоровским, SM (почему-то) критерием, я его зову wdw: w*dLoss/dw

WDW критерий



- Если разложить нейросеть в ряд Тейлора, то сразу после константы будет идти параметр Δw*dLoss/dw, это очень удобно, потому что при дообучении градиент вы всё равно вынуждены насчитывать.
- Однако на сколько такое разложение соответствует действительности, видимо, никто особо не проверял, иначе бы эта картинка лезла на вас из каждого утюга.
- Здесь явно видна корреляция между метрикой и истинной полезностью, ну и выбросы, конечно.

WDW критерий



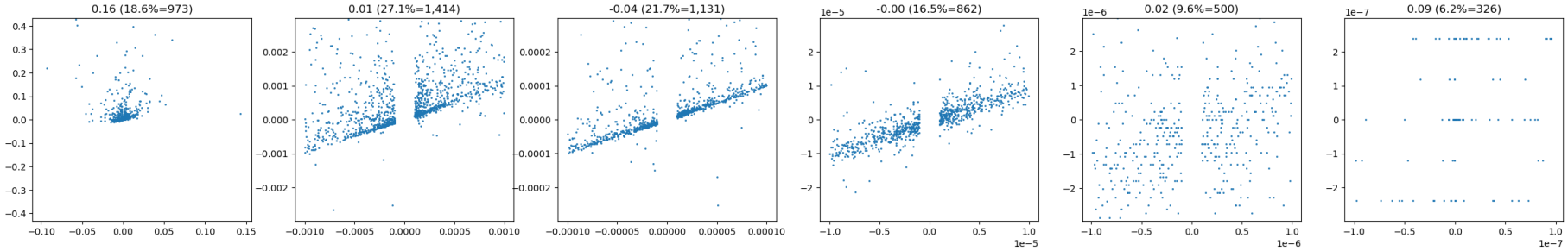
- Видны веса, которые лучше не трогать, потому что они большие, и выбросы там тоже большие.
- Видны веса на столько малые, что даже в fp32 мы видим шум дискретизации, а не закономерности.
- Видны веса, которые лучше всего подходят для того, чтобы их резать.
- Видно, что мы очень хорошо умеем определять полезность весов, но это свежеобученая сеть, которую мы ещё не прунили. Как только она подвергнется прореживанию картина станет более смазанной.

- Шок-контент: **У WDW ЕСТЬ ЗНАК!**

>>> На сколько линейны нейросети?

Степень линейности сети

- На большом масштабе нейросеть сильно нелинейна, невыпукла, и во всех остальных отношениях тоже сложная.
- На маленьком масштабе можно посмотреть в какой степени её поведение линейно и зависит больше от первой производной.



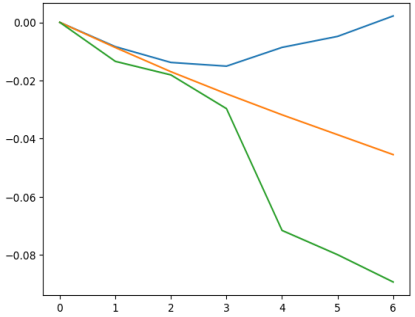
Мы видим, что вторая производная есть, потому что есть выбросы вверх, но есть и выбросы вниз, это значит, что есть и третья производная, но выбросы вниз все маленькие, значит есть и четвёртая производная, а мы и вторую то производную вычислить можем только в очень грубом приближении.

Не будем пытаться красиво решать задачу, я пытался и не смог. Зайдём с другой стороны – будем понемногу удалять веса и смотреть на суммарный эффект.

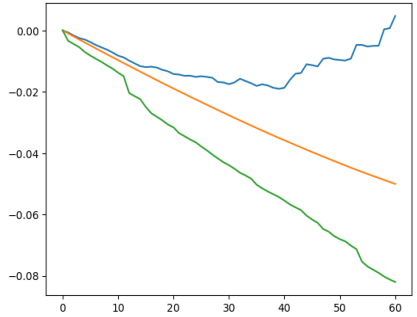
Постепенное удаление

- Нельзя удалить одинаковое количество весов с малым wdw и надеяться что-то увидеть, потому что если сеть строго линейно себя повдёт суммарный эффект будет нулевой. Понятно почему?
- Удаляем только веса с wdw больше 0, тогда если сеть линейна loss должен уменьшаться, а он точно не сможет заниматься этим до бесконечности или даже просто долго. Понятно почему?

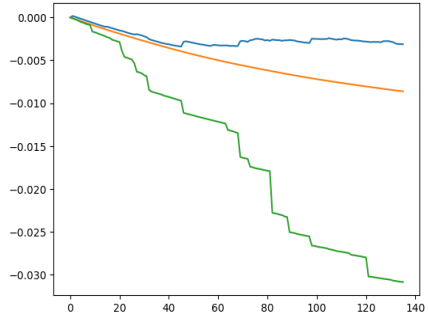
wdw > 1e-2



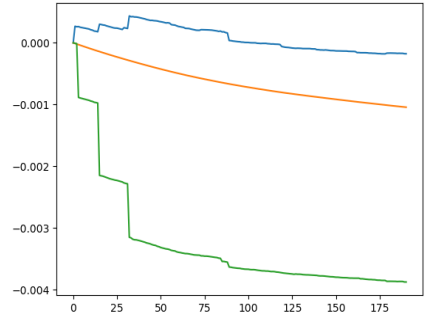
wdw > 1e-3



wdw > 1e-4



wdw > 1e-5

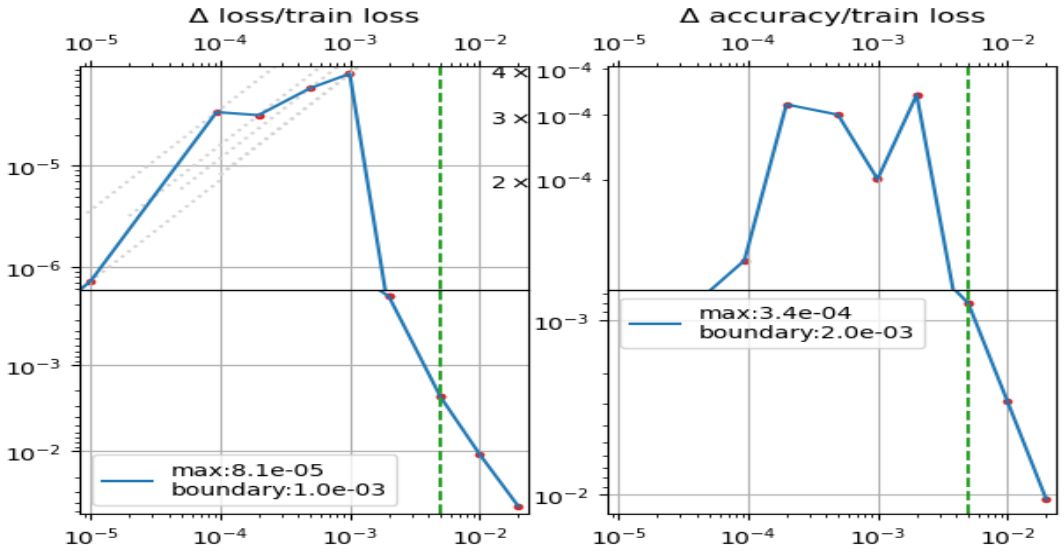


Синий – суммарная полезность, Оранжевый – сумма истинных полезностей, зелёный – сумма ожиданий по wdw.

- Видно, что во-первых, сеть до некоторых небольших пределов сеть ведет себя как линейная регрессия.
- Удаление 6 весов wdw~1e-2, по эффекту схоже с удалением 60 весов wdw~1e-3, по всей видимости, нескольких сотен весов wdw~1e-4 и так далее. То есть для сети значение имеет не количество удалённых весов, а то, на какую сумму wdw мы их удаляем.

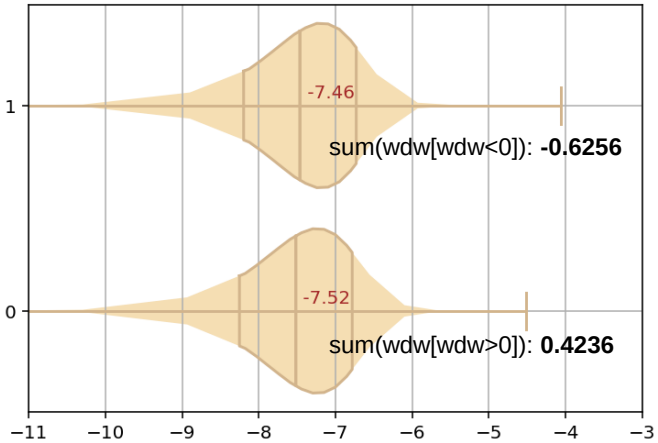
Проверка границ линейности

- Для прунинга мы заинтересованы в удалении наибольшего количества весов, поэтому резать надо того у кого wdw маленький, но при этом не имет смысл резать только положительные если зацепляем шум, то есть в любом случае режем выше некоторого порога. Таким образом выбираем наименьших из тех, что выше wdw > 1e-7
- Отрезаем на определённую сумму wdw на трейне, и смотрим что с loss, потом на кратно большую, и так далее.
- Видно что до суммы 1e-3 в данном состоянии сети loss улучшается, а дальше сеть перестаёт себя вести даже отдалённо линейно.
- По мере мере изменения состояния сети график может выглядеть не на столько прекрасно, но вы можете иногда его перерисовывть чтобы понимать общее состояние сети.

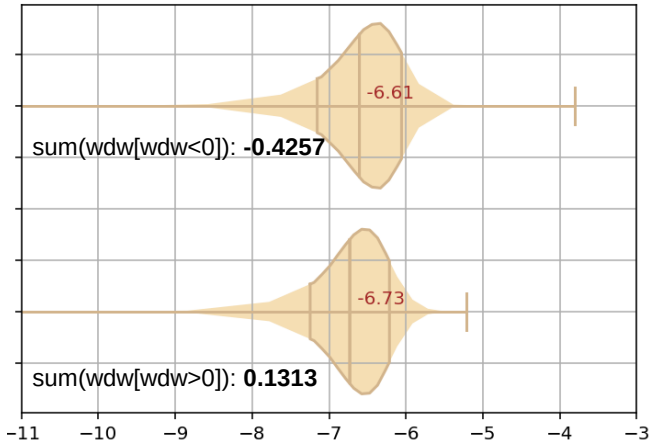


Распределение весов по wdw и суммарный wdw сети

Среднее значение wdw в сети ожидаемо около 0, но что если посмотреть отдельно на положительные и отрицательные wdw и их распределение? На левой иллюстрации языковая задача с первого слайда на сети 3M compression ratio: x2.9 (65.64%), у свежей непрореженной сети в зависимости от архитектуры эти цифры могут достигать десятков и сотен. Что нам это говорит?



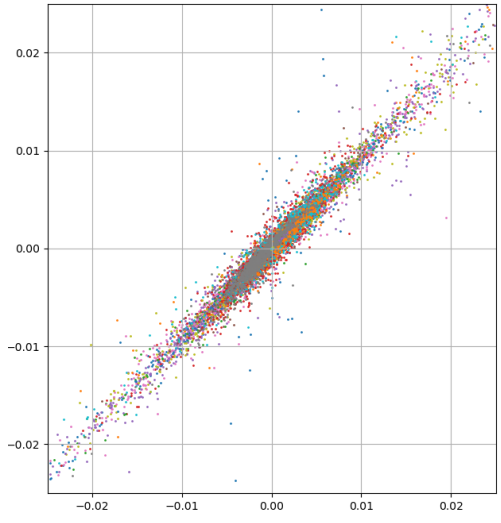
До начала прореживания

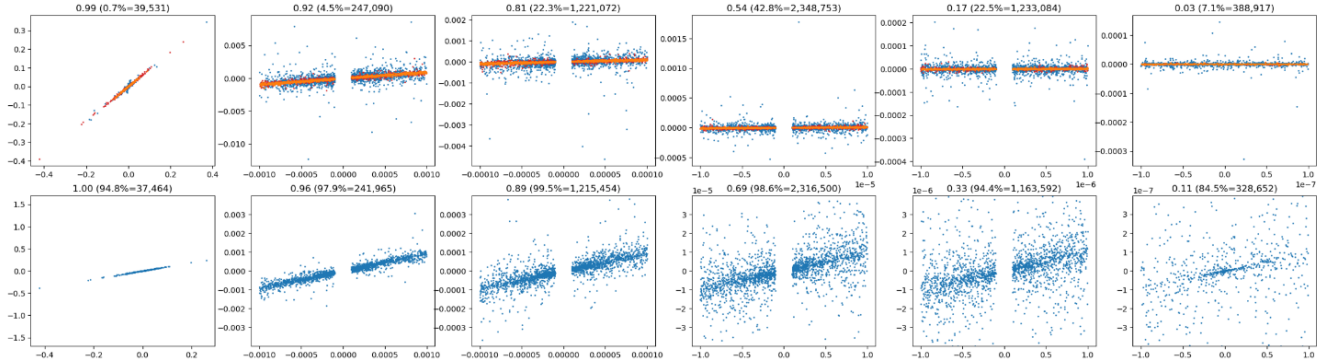


После прореживания ~x3

Корреляция wdw между train/val

- Большинство иллюстраций выше было получено на val выборке, потому что она меньше и нас интересовала в первую очередь она. Но напрямую действовать при прунинге мы можем только на train. Значит надо посмотреть на сколько они скоррелированы.



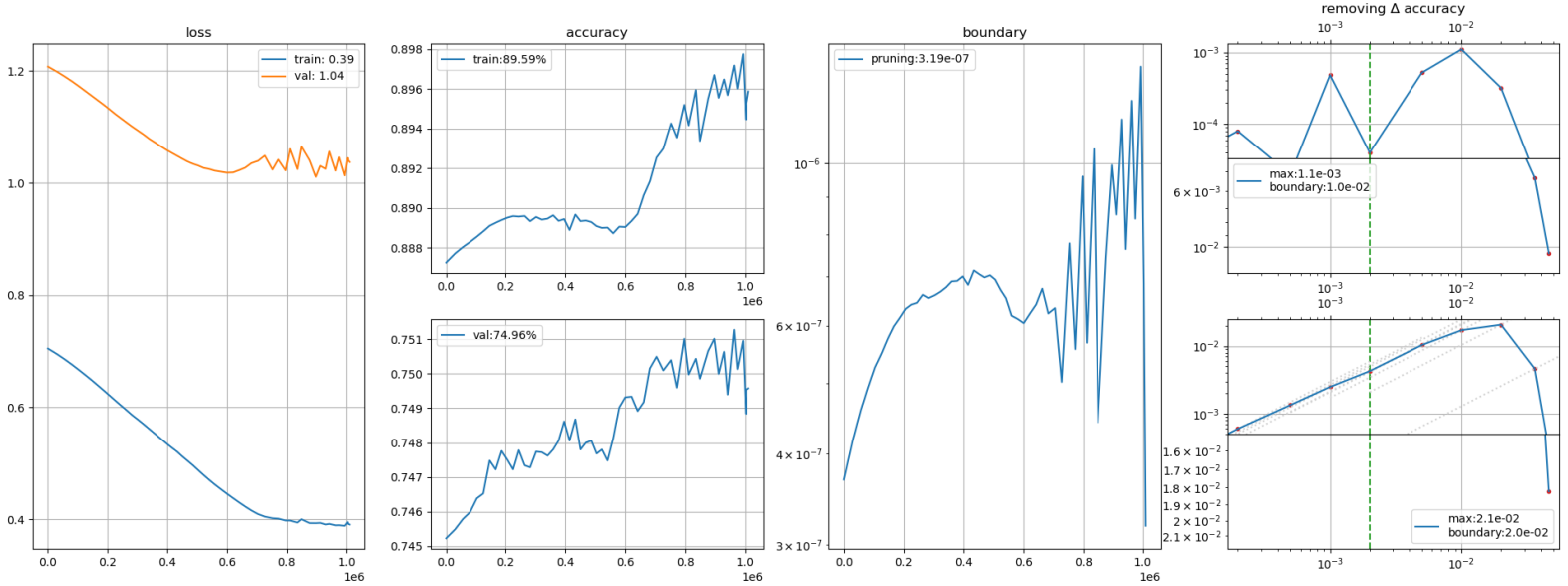


Вторая строка это уже почищенные от выбросов и перемасштабированые вертикаль к горизонтали 1/3, но даже и без чистки хорошо видная корреляция, то есть действуя на train мы с высокой вероятность будем резать интересующие нас веса на val

А ещё видно, что глубже 1e-7 рассчитывать на корреляцию не стоит, хотя с другой стороны и вклад у них копеечный

Пример сжатия без дообучения, умный прунинг это тоже такое обучение

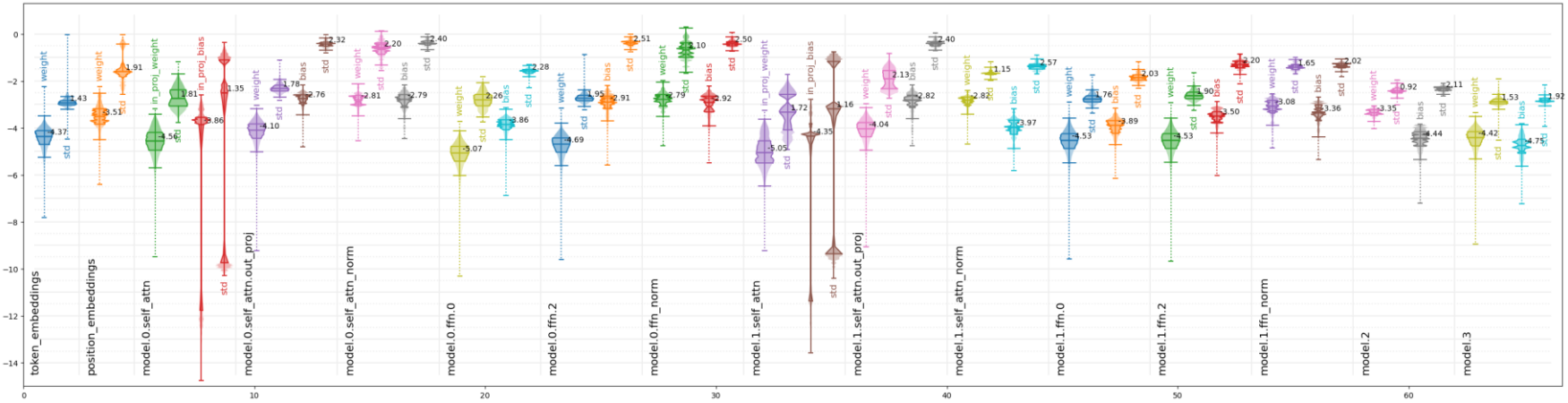
- Исходя из того, что вы уже знаете, логично предположить, что возможно такое прореживание нейросети, при котором accuracy на val будет не портиться, а улучшаться, хотя бы в некоторых пределах. Практического применения пример не имеет, если только у вас не бесконечный компьют, только изменить пределы возможного:
- Ну и ещё, возможно пригодится на каггле в борьбе за последние тысячные, потому что это ещё один инструмент регуляризации, которого, вероятно, нет у ваших конкурентов, и который, как правило, лишним не бывает.
- Обнуление первого миллиона весов в MobileNet_v3. Сильное падение в начале частично связано с пережатым датасетом, на полностью корректном превосходство над бейзлайном не 0.5%, а всего несколько сотых, но оно есть.
- Без алгоритма чистки выбросов вы буквально так, вероятно, не сделаете. Но с дообучением можно попробовать.



>>> Равно полезны ли слои? Кого резать?

Одинаково ли слои подходят для резни?

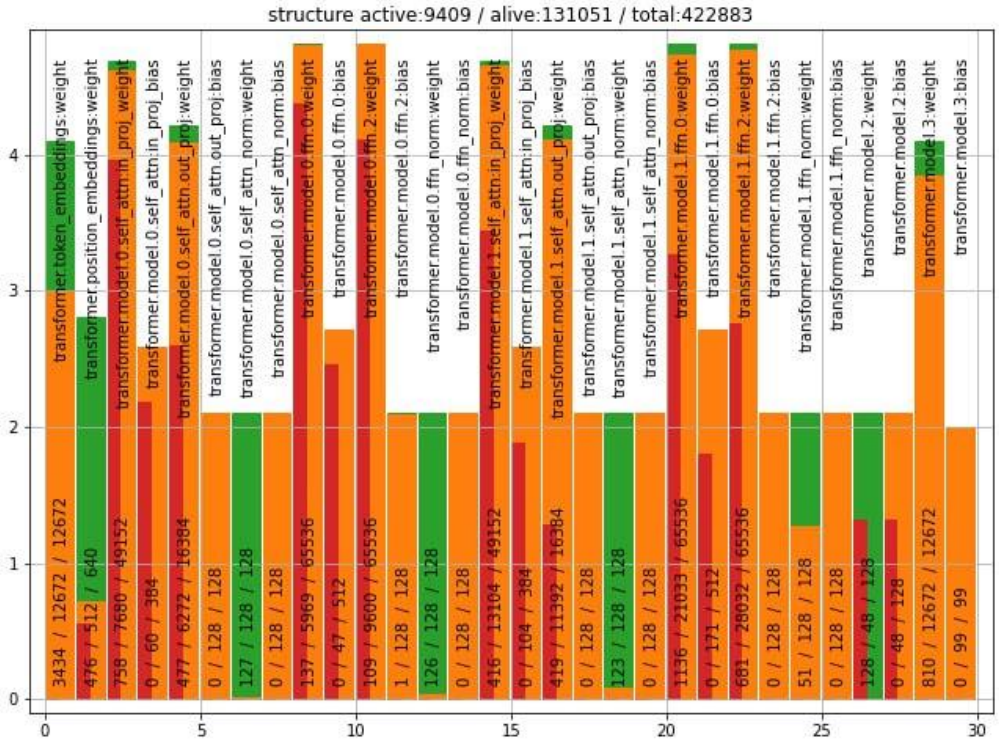
- Обычно в работах говорят что-то вроде: Давайте для простоты иметь одинаковую степень прореживания на всех слоях. В лучшем случае одинаковый критерий типа магнитуды для всех слоёв, да и ладно, и не проверяют дальше.
- Посмотрим как выглядит распределение градиентов по слоям в небольшом пожатом трансформере:



На этом месте вы уже сами должны догадаться в чём проблемка. По вертикали десятичный логарифм градиента, то есть разница в градиенте между слоями больше порядка.

Рассмотрим экстримально сжатую сеть

- Это трансформер, 2 слоя, 4 головы, задача известная гугловская задача на которой был продемонстрирован гроккинг. Степень сжатия ~x45 acc val 100%



- Не считаем леер нормов, которые лучше вообще не трогать, потому что они вырубают сразу каналы у нас:
- Самый незатронутый слой с эмбедингами выжило 2/3 весов, или последний fully connectd на 6.4%
- Самый прореженный ffn.2.weight – 0.17%
- И это ещё не самый пожатый, на это задаче можно x75 получить и больше.

Вывод:

- Какой бы алгоритм спарсификации вы не предпочитали он не должен предполагать равного прореживания разных слоёв.

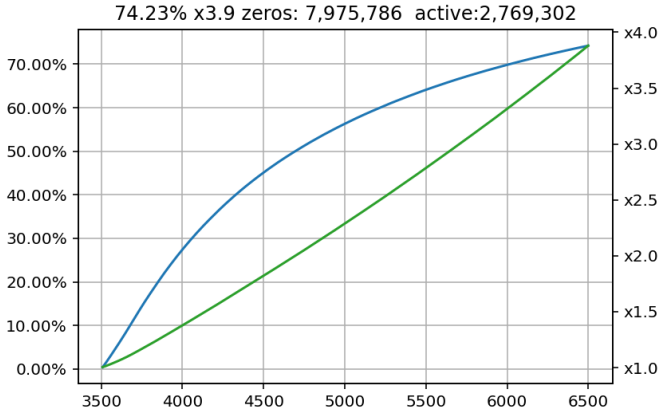
>>> Прунинг это такая регуляризация

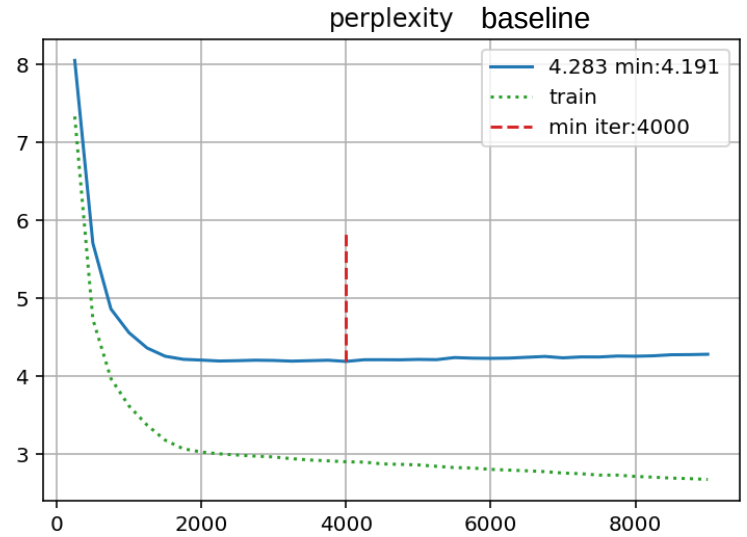
Теоретические обоснования

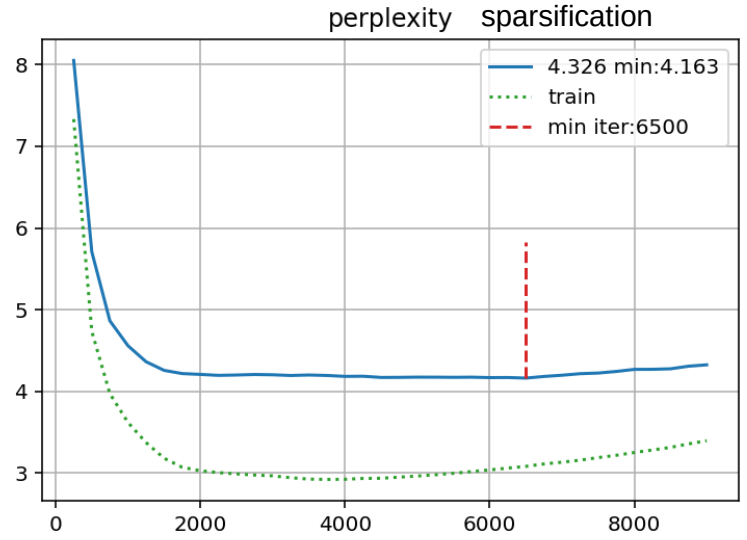
- Представьте, что вы аккуратно отрубаете сети веса, заставляя выражать ту же мысль все меньшим количеством сигналов, но при этом размер доступного сети эмбединга не ументьшается. Она всё ещё может формулировать свои эмбединги столь же разнообразно, но, меньшим количеством сигналов.

Берём сеть и просто врубаем прунинг

- Врубили начиная с 4000-ого шага, и довольно интенсивный, за 6500 итераций снесли примерно ¾ весов Хорошо видны типичные последствия дополнительной регуляризации:
  - Train перестаёт улучшаться.
  - Validation продавливается ниже.
- Но тут же мы видим и негативные последствия – Слишком передавленная регуляризацией сеть начинает разрушаться.
- Это потому что у нас кроме прунига функции регуляризации выполняет ещё dropout=0.2 и weight_decay=0.8, слишком сильно мы стучим сети по шапке.

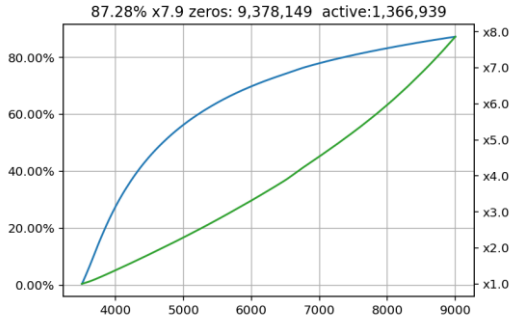


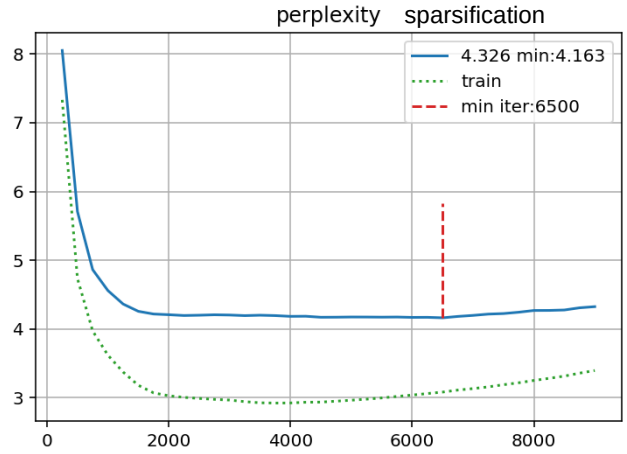


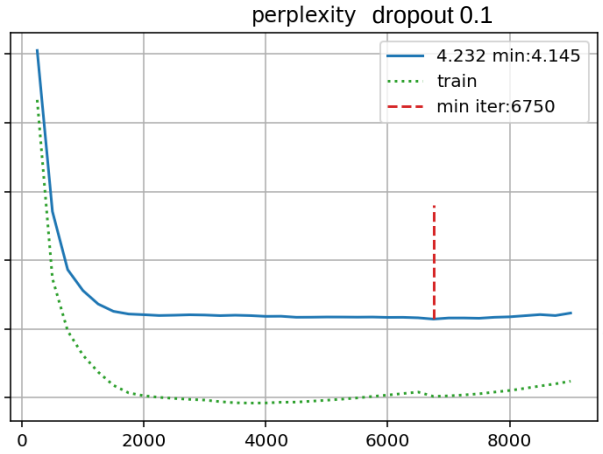


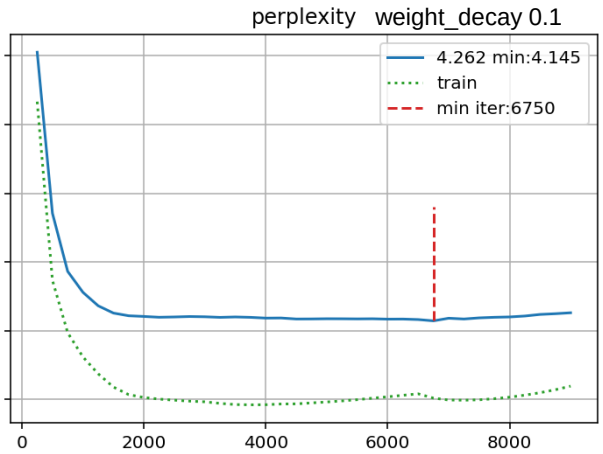
Берём сеть и отключаем часть регуляризации

- Давайте снизим dropout в два раза, до 0.1?
- Трейн чуть-чуть улучшился и даже validation немного утащил, это нам повезло просто.
- А давайте ещё weight_decay попробуем с 0.8 до 0.1 уменьшить.
- Ну трейн ещё более зрелищно ушел вниз, но val стал хуже, видимо это слишком радикально на этом этапе.
- В общем, если мы хотим чего-то добиться с гиперпараметрами придётся играть буквально на кончиках пальцев.
- И видно, что спарсификация дает регуляризацию в другом режиме нежели другие приёмы, её сила постепенно накапливается по мере прореживания сети.





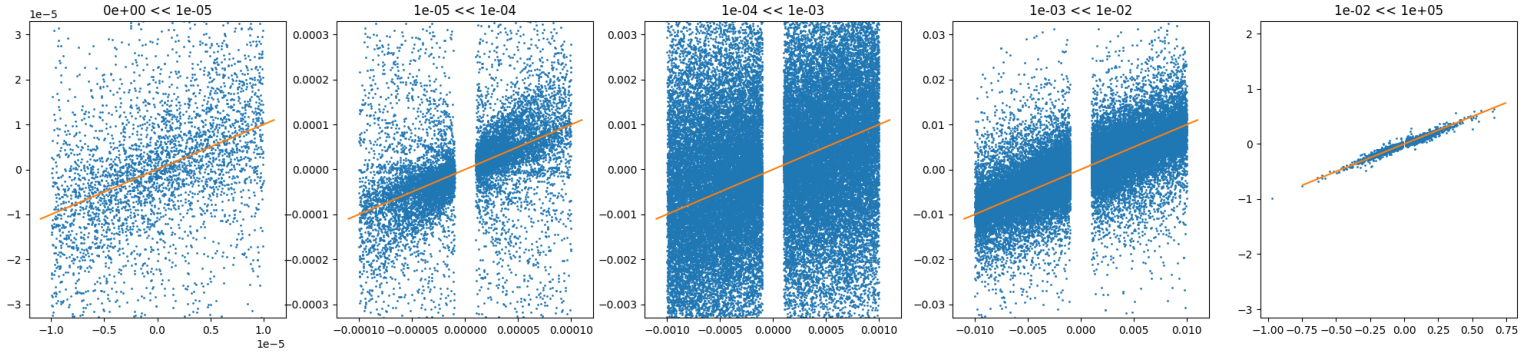




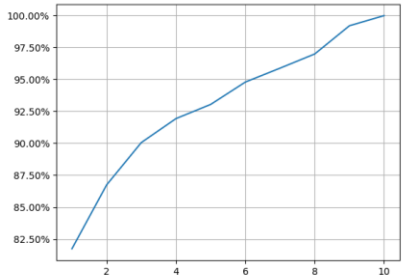
>>> Размер сабсета для прунинга и сглаживание

Сравниваем градиент пакетный с частичным

- Для того чтобы всё это делать нам нужен пакетный градиент по всему датасету, потому что, как мы помним, большую часть вклада в loss создаёт малое количество примеров выборки, но это адски дорого, а для безбрежных текстовых датасетов просто невозможно.
- Возьмём текстовую задачу, и уже обученную сетку, первый раз соберём его на большом участке датасета, а во второй раз накапливаем его понемногу и рисуем на диаграмме на сколько он похож, на градиент накопленный на большом участке, и, что очень важно для нашего прунинга - какая доля wdw совпадает с окончательной по знаку.



- Смотрим какая часть датасета достаточна, чтобы нам было не страшно руководствоваться, учитывая, что дообучение закроет часть огрехов.
- Такой большой градиент нам собрать не удастся, почти наверняка, но зато мы можем собирать градиент с большим окном округления экспоненциальным скользящим средним.



Где добыть хорошие градиенты?

- Для всех этих манипуляций вам потребуется коллектор градиентов.
  Мой можно качнуть с репозитория: https://gitverse.ru/kraidiky/in_sight
  там ещё много всякого полезного что вы тут видели, но вообще там 25 строк,
  можно и самому побыстрому наваять.

1. Делаем экспоненциальное скользящее среднее
2. На первых шагах beta = max(1/self.steps, self.beta) чтобы не получать просадки на первых шагах, но всё равно полезно прогревать воизбежание проблем на первых шагах, или, если прунинг включается не сразу, переложить туда уже слегка скруглённые значения из сорстояния оптимизатора.

- И сохранять накопленное в чекпоинтах, конечно.

- Большинство оптимизаторов имеют у себя в стейте градиенты округлённые скользящим средним, можно их использовать, если степень сглаживания для вас приемлема.
- Adam – по умолчанию хранит градиент округлением на 10 итераций и квадраты градиентов, если вас нужно с очень приличным округлением в 1000 (betas=(0.9, 0.999)), Но в репозитории Карпаты второй бетас часто берётся скромные 20 (betas=(0.9, 0.95)).
- В знаменитом репозитории modded-nanogpt оптимизатор Muon, условное 20 окно сглаживания градиента: momentum=0.95
- Такие сглаживания вероятно недостаточны для прунинга, но можно попробовать чтобы сэкономить x1-2 памяти, или попробовать не будет ли обучение в вашем случае терпимо относиться в большим значениям betas/momuntum

>>> Прунинг без разморозки
- компьют на ветер

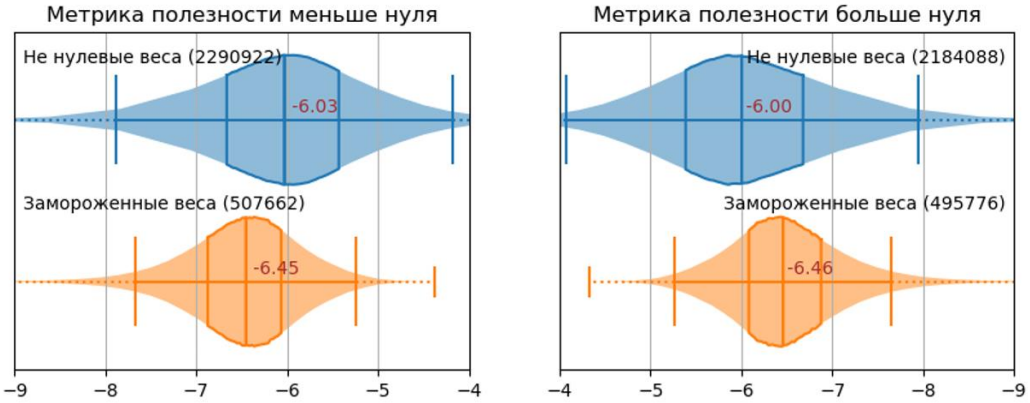
В чём разница между методами маскировки и пороговыми?

Минимум три основных подхода к спарсификации

- Честное обучение спарсифицированных матриц, мы об этом мечтаем потому что:
  + Обучать мало весов, градиент тоже считать для малого количеств весов. До двух порядков экономии.
  + Малое количество весов как инструмент дополнительной генерализации. В порывах до гроккинга.
  + Зависимость флопсов от размеров эмбединга не квадратичная, привет аттеншену.
  - Современный GPU не приспособленны.
  - Неочевидно как собирать градиенты отсутствующих весов, хотя способы есть, но они непростые.
- Маскировка весов. Запоминаем маску какие веса не нулевые, возможно не храним её даже, а вычисляем перед шагом оптимизации, и после шага обучения их зануляем. Можно даже прямо внутри оптимизатора, для экономии. Классическая работа на эту тему RigL, хотя я этот метод лет на 5 раньше применял. В чём плюсы и минусы:
  + Имитируем честное спарсифицированное обучение с его плюсами, кроме экономии флопсов.
  + Имеем все нужные градиенты.
  + Чётко контролируем сколько у нас нулевых весов.
  - Вынуждены считать всё, не можем учить 300B сеть начиная с 3B активных весов.
- Ничего нигде не храним, берём критерий и обнуляем веса по нему каждый раз.
  + Ещё более просто. Для магнитудного критерия это просто L0 регуляризация.
  + Если какой-то вес окажется полезен он станет активен без каких либо дополнительных телодвижений.
  - Можно контролировать линию отсечения, но крайне трудно – сумму удаляемого, а мы знаем как это полезно для аккуратности процесса.
  - В случае заметного изменения нормы градиента при обучении, а такое бывает при дестабилизации сети, очень много весов активируются.
  - Такой алгоритм, даже если хорошо сработает, не переносим на этап обучения никак. Только инференс.
  ? Я не знаю достижимы ли экстримальные уровни прореживания при таком подходе. Но раз никто 300B сети не инференсит на кастомерских картах, видимо какие-то свои подводные камни есть.

Критерии разморозки?

- На картинке wdw для прунинга без дообучения.
- Видно что многие веса, которые были вредны для сети на 1e-7 в момент обрезания теперь стали очень полезны, и их можно возвращать.
- В способе с масками чтобы решить кого размораживать можно как в RigL использовать градиенты и это очень логично, если вы учитесь с SGD, но если Adam, уже не очень.

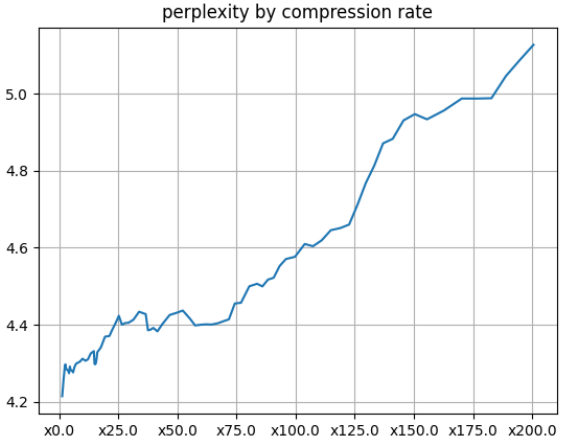


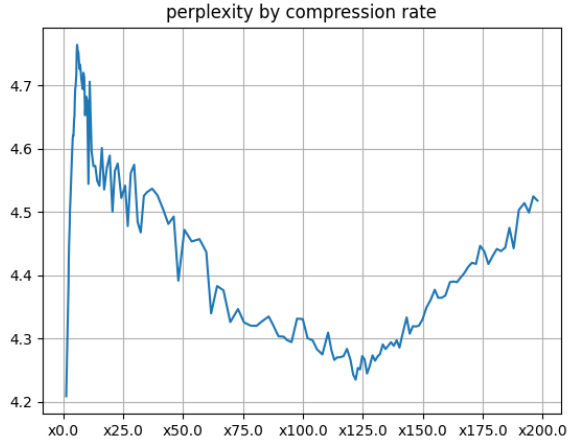
- Как критерий разморозки можно использовать ожидаемый прирост wdw после одного шага оптимизации. Для этого легко воспроизвести алгоритм спятаный в недрах Adam, и вам пригодятся округлённые квадраты градиентов, или просто обнулять по маске не всех выбирая кого оставить.
- А есть ещё Muon, который SOTA но при этом принципиально не умеет работать с прореженными матрицами, тут метод обнулять не всех становится, на сколько я знаю безальтернативным для масочного метода.
- Последствия размораживания весов после прунинга очень плохо корреллируют с их wdw от них ожиданием, поэтому либо делаем это не в тот такт, в который пруним, либо размораживаем с маленьким ненулевым весом, или вообще изменением маски без изменения веса если она у вас хранимая

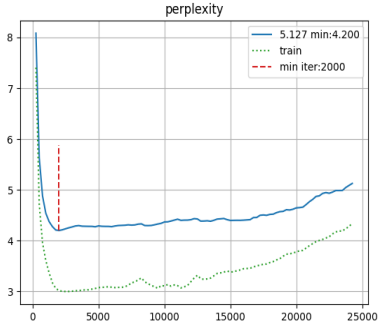
Прунинг без разморозки – компьют на ветер?

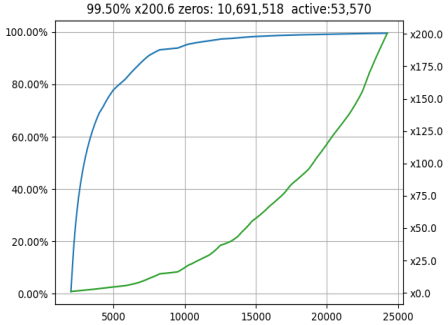
А теперь просто сравним результаты экстримальной спарсификации с маскированием и разморозкой и без.

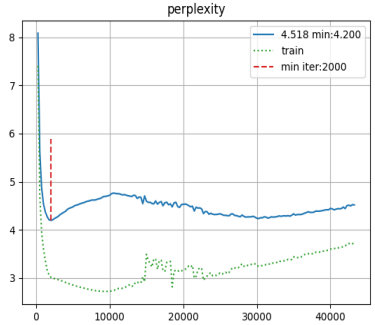
Определите какая из картинок относится к какому случаю. ☺

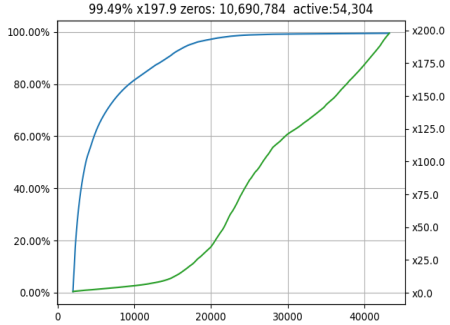








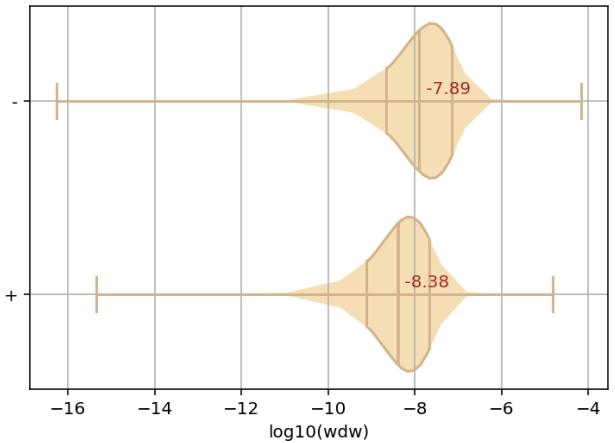


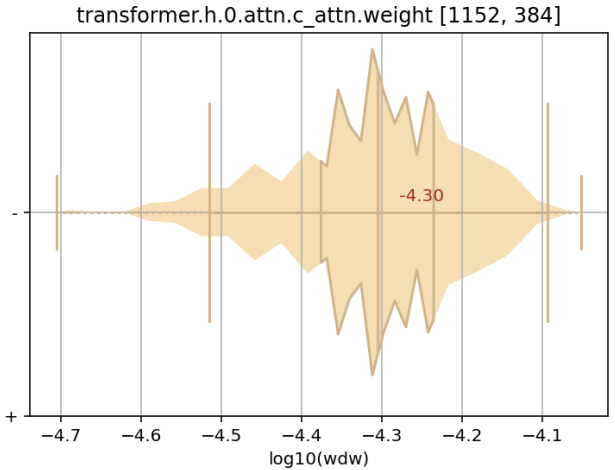


>>> Поканальный и понейронный прунинг

В чём проблемы и в каких диапазонах характерные значения метрик

- Как мы много раз видели сегодня wdw отдельных весов колеблется от ±1e-2 до ±1e-15, есть из чего выбирать и кого прунить. Посчитаем то же самое для каналов каких-нибудь квадратных больших матриц в свеженькой непрореженной сети.
- В данном случае, очевидно, закон больших чисел работает против нас.
- Ансамбль синапсов – структурная единица нейронной сети, моя статья 2015-ого года, до сих пор не устарела.
- Со свёрточными сетями всё сильно проще, потому что свёрток мало в штуках и мы можете за разумное время буквально измерить истинную полезность отдельных свёрток, теперь вы знаете достаточно чтобы проделать это самостоятельно.





>>> Нейросети не точная,
а естественная наука

Поиск эксперименты и перепроверка посылок

- Проблема научных статей – математики мало что могут без смелых, но часть проверяемо неверных предположений:
  - Гессиан положителен полуопределён
  - Мы в локальном минимуме, градиенты почти нулевые
  - Большая часть весов бесполезна
  - Отдельные слои независимы
  - Активации разных слоёв нормально распределены и нескоррелированы

Эти и многие другие предположения проверяемо неверные, но авторы статей, чаще всего не могут от них отказаться, потому что не смогут применить математику, а без неё не считово!

- Очень часто в ходе экспериментов придумывают какую-то гипотезу, просто встраивают её в три строки в код и глядя только на loss/perplexity/accuracy и не меняя гиперпараметров даже между сериями экспериментов, не то что внтури серии строят на этом выводы. Ну такое.
- Только собирая и визуализируя много информации о внутреннем состоянии и работе сети можно чего-то добиться в сложных случаях.
- Нейросеть крайне сложная многокомпонентная система на которую многие факторы действиют по разному, ещё и по разному в разном контексте. Если этого не учитывать почти никакие выводы будут не верны, а интересные результаты недостижимы.

- Теперь, зная много нового о прунинге попробуйте самостоятельно ответить на вопрос почему не смотря на то, что существует правильный ответ, но практикуется и множество других.

СПАСИБО ЗА ВНИМАНИЕ!!!