Сейчас мы послушаем первый доклад. Это, наверное, один из самых известных людей здесь на Open Data Science Это Евгений Макаров. Давайте ему поаплодируем. Мы послушуаем прекрасный доклад. Спойлер: этот доклад может показаться немного жёстким, но это Open Data Science. Вы должны привыкнуть.

- Спасибо Евгению за то, что представил меня. Меня зовут Евгений Макаров. Я инженер в компании skidata, Мюнхен, Германия. Недавно я участвовал в старт-апе, который занимался последовательно аватарами, анимацией, виртуальными примерочыми. Мы создавали make-up по вашему желанию, который наносит макияж на ваше лицо. Это называется augmented reality.

Сегодня я бы хотел поделиться с вами теми вещами, которые мы узнали во время работы над этим проектом. Какие-то инсайты, которые мы получили, трюки, которые помогают улучишть качество, оценки ключевых точек на лице. Оценка ключевых точек на лице является базовым элементом многих систем, как оценка поведенческих паттернов человека, распознование лица для выравнивания лиц.

Задачей является нахождение ключевых точек на лице, привязанных обычно каким-то семантическим элементом лица: уголки глаз, уголки рта, крылья нос, овал лица и так далее.

Это применяется в редакторах изображений, применяется в технике чтения губ, чтения речи, используя картинку.

Также это применяется для immersive интерфейсов, которые позволяют вам взаимодействовать с компьютером.

Я очень рад, что рассказываю об этом в Минске, Минск является родиной компании Masquerade, которая с большим успехом создавало приложения, имеет большое количество пользователей. Эта компания была продана за большую сумму денег. Мне неловко выступать в городе, где работали специалисты высокого уровня. Я надеюсь, что моё выступление также будет полезным.

После этого все знают успех приложения Snapchat.

Есть такая компания Image-Metrics. Это очень успешная компания. Оно помогает всем, кто хочет создать приложение для обработки лиц.

Также существует компания Modiface, которая делала макияж для компании Лореаль. Эта компания была куплена компанией Лореаль в прошлом году. Казалось бы.. Кто-то может сказать, что это несерьёзное приложение, так как это макияж, маски, ушки. Но тем не менее такие приложения очень популярны, людям это нравится, спрос большой, поэтому эта отрасль развивается очень активно, и множество людей вкладываются в эту отрасль.

Задачей является получение координат X и Y на изображении ключевых точек. Вот здесь. Изначально вы имеете только изображение. Первым шагом при оценке этих ключевых точек на лице является запуск Face Detector, который выдаёт на выходе bounding box. Это может быть квадратным или прямоугольным, это зависит от дизайна вашей конкретно системы.

В него вы вставляете mean shape, это landmarks, которые у вас есть в обучающей выборке.

Вы рассчитываете среднее положение лица, обычно это является фронтальным, что логично. Затем вы fit mean shape into this box.

И вы хотите обучить систему машинного обучения, которая имеет на выходе вот эти координаты для mean shape , эта система может отдать вам на выходе координаты для какой-то другой shape.

Например, для повёрнутого лица или для лица, которое выражает какую-то эмоцию и так далее. Ваша задача - выучить разницу. Люди пробовали учить сходу, с нуля, не прибегая к этому mean shape. Они просто пытались искать эти координаты. Это оказалось сложно. Ёмкости модели не хватает. У этой модели не хватает таких способностей. Поэтому у людей ничего не получалось. Поэтому был предложен подход находить дельту.

Дельта X между каким-то начальным shape и тем, который вы хотите получить.

В этой области в какой-то момент произошла эволюция, люди перешли от генеративных моделей к дискриминативным моделям.

То есть стали учить регрессор на точки на основе признаков, извлекаемых из картинки.

До этого люди использовали создание каких-то параметрических моделей, играли параметрами модели, оптимизировали её по какому-то критерию качества и пытались встроить результат наилучшим образом во входное изображение.

Эта работа стала ключевой. Они предложили каскадную схему.

Они предложили не пытаться находить функцию регрессирующего mean shape сразу конечной форме,а делать это каскадным образом с последовательным уточнением X и Y координат.

На каждом этапе они брали 10 регрессоров и на каждом этапе уточняли, уточняли и уточняли.

Они находили дельта между нулевым и первым, дельта между первым и вторым, между вторым и третьм и так далее.

На каждом шаге они.. сейчас это кажется очевидной идеей, но в тот момент это было достаточно эволюционно.

Они на каждом этапе сэмплировали признаки лица.

В классической процедуре регрессии, например, с помощью градиентного бустинга или алгоритмов случайного леса, вы делаете экстракцию признаков один раз в самом начале.

Затем вы запускаете ансамбль или случайный лес.

И вы никогда больше не возвращаетесь к этапу экстракции признаков.

Здесь это работает не так. Они сказали: "Мы улучшили это, давайте сэмплировать эти точки здесь с более уточнённого shape".

Это действительно придало модели некую регуляризацию. Эта модель стала учиться.

За этим последовала работа Вахида Каземи, который предложил улучшение на основе деревьев решений, предложил процедуру оптимизации целевой функции.

Она была имплементировала во фреймворке, который, я уверен, многие из вас использовали.

Очень важная идея, которую предложили исследователи из Майкрософт Ресерч. Они предложили использовать очень простые признаки без извлечения градиентов и других сложных вещей.

Они просто предложили выбирать случайным образом пиксельное изображение, случайным образом выбирать пары пикселей и смотреть разницу интенсивности пикселей между этой парой. Это очень просто. Это всё можно быстро посчитать. Это удобно ложится на любую архитектуру процессора. Эта вещь, которая кажется очевидной, привела к большому успеху. Это стало работать в режиме реального времени на мобильных устройствах.

И даже сейчас, насколько я знаю, большинство коммерческих организаций в этой области применяют эту простую схему, несмотря на развитие нейронных сетей и свёрточных карт признаков. Эти системы по-прежнему доминируют в мире.

Ещё одной идеей, которая сейчас кажется очевидной, но привела к успеху таких регрессоров, является введение локальной системы координат.

Если вы хотите искать координаты ваших пикселей не в глобальной системе координат, привязанной к центру изображения, а в локальной системе, которая привязана к ближайшей landmark, лежащей на данном этапе регрессии.

Потому что они заметили, что..

Допустим, у вас есть эта точка, у неё есть какая-то дельта X дельта Y от начала координат и точка, отнесённая на такие же дельта X, дельта Y от начала координат на изображении с Леонардо Дикаприо.

Эта точка имеет другое семантическое значение, так как она находится на брови,а не на глазе.

Получается, может быть слишком большая вариабельность. И поэтому система просто не обучится, так как это выглядит, как случайные вещи, система не учит структуру. Они предложили использовать локальную систему координат:

вы случайным образом сэмплируете точку на изображении, находите ближайшую к ней landmark, например, на mean shape на начальной стадии. Затем вы высчитываете эти дельта X, дельта Y от этой landmark.

Для каждой точки у вас есть индекс ближайшей к ней landmark и Х,У координаты. Это всё, что вам надо помнить. Раньше вы помнили Х,У глобальной системы координат, теперь вы знаете Х,У локальной системы. Это может показать простым и банальным, но это привело к существенному увеличению точности и позволило обучать очень успешные регрессоры.  
  
Они предложили ещё одну идею. Мы на каждом этапе регрессора, когда мы сэмплируем признаки изображения, мы берём их относительно уже изменённого shape,уже уточнённого.

Нам больше не нужен этот mean shape, он нам нужен был только на первом этапе для инициализации.

Как только мы перешли к изменённому shape, мы когда-то сгенерили признаки для mean shape ещё в самом начале.

Мы 10 раз сэмплировали их на изображении случайным образом.

На нулевом этапе мы посчитали координаты относительно mean shape, а на первом этапе мы вынесли преобразование подобия между mean shape и изменённым shape и перессчитаем координаты точек, которые мы просэмплировали уже относительно изменённого shape. Вы видите, этот квадратик как был здесь, так и остался.  
  
  
Но оно немного повернулось. Ну окей. Это придало способность к последовательному refinement точек. Вам уже не нужна эта устаревшая информация, вы адаптируетесь к новой информации.

Ещё одна идея, которую они предложили, это использование фернов, так называемых фернов. Многие их знают под названием oblivious trees.

Например, они используются в современных фреймворках градиентного бустинга. Чем хороши такие oblivious trees?

Обычное дерево решений задаёт в узлах разные вопросы, и выясняет, больше или меньше порога ваш feature.

И вот вы задаёте один вопрос, если меньше, вы попадаете в этот узел, если больше, вы попадаете в этот узел и так далее.

Oblivious tree задаёт один и тот же вопрос на всех уровнях. Чем это удобно? Жизнь показала, что оно меньше переобучается.

Самое главное - вы уже здесь, на начальном этапе, знаете заранее, что вы будете задавать f0, f1, f2 каждому примеру из вашего обучающего множества.

Это очень удобно. Здесь нет условия if. Здесь вы не знаете, кому вы будете задавать f1, а кому будете задавать f2. Здесь нет параллели.

Вы заранее не знаете, какой признак у вас попадёт на условие if.  
В этом случае вы знаете, что все примеры получат один и тот же вопрос, и каждый пример из обучающей выборки получит свой бинарный код, скажем так.

Наподобие хеширования процедуры. Это очень удобно для тренировки, так как тренировка параллелится, очень удобно для инференса, потому что всё работает быстро даже на современных телефонах. Всё работает очень хорошо.

Они предложили просто пример, используя это дерево вопросов, oblivious tree.

В итоге вы попадаете в какой-то лист дерева и смотрите из всех примеров вашего train set, какая там дельта of landmarks и запоминаете это.

Внутри каждого каскада регрессора находятся вот такие ферны. В оригинальной работе было 500 таких фернов.

500 элементарных деревьев компенсируют недостающую дельту.

Потом этот каскад заканчивается, вы пересэмплируете features и повторяете процедуру.

После каждого каскада происходит пересэмплирование of features и пересчёт преобразования подобия относительно mean shape, который у вас был на начальном этапе.

"Feature selection" процедура ... я расскажу in a nutshell, потому что у нас не очень много времени, идея заключается в том, что у вас должны быть хорошие features, они должны быть коррелированы с target. Просэмплировав точки и найдя их pixel differences, вы должны понять, что вам удобнее всего для построения дерева, какая из них несёт больше информации.

Это отдельная процедура, но в целом можно сказать, что каждая feature должна хорошо коррелировать с target, и они должны быть комплиментарными, они должны дополнять друг друга.

Они предложили достаточно изящную процедуру, мы можем это обсудить потом, сейчас у нас нет на это времени. Это тоже является нововведением авторов из Microsoft Research.

В то же время вам никто не мешает использовать обычную процедуру поиска наилучших сплитов из градиентного бустинга.

В этом случае вы знаете, что все примеры получат один и тот же вопрос, и каждый пример из обучающей выборки получит свой бинарный код, скажем так.

Наподобие хеширования процедуры.

Это очень удобно для тренировки, так как тренировка параллелится, очень удобно для инференса, потому что всё работает быстро даже на современных телефонах.

Всё работает очень хорошо. Они предложили просто пример, используя это дерево вопросов, oblivious tree.

В итоге вы попадаете в какой-то лист дерева и смотрите из всех примеров вашего train set, какая там дельта of landmarks и запоминаете это.

Внутри каждого каскада регрессора находятся вот такие ферны. В оригинальной работе было 500 таких фернов. 500 элементарных деревьев компенсируют недостающую дельту. Потом этот каскад заканчивается, вы пересэмплируете features и повторяете процедуру.

После каждого каскада происходит пересэмплирование of features и пересчёт преобразования подобия относительно mean shape, который у вас был на начальном этапе.

"Feature selection" процедура ...

я расскажу in a nutshell, потому что у нас не очень много времени,

идея заключается в том, что у вас должны быть хорошие features, они должны быть коррелированы с target. Просэмплировав точки и найдя их pixel differences, вы должны понять, что вам удобнее всего для построения дерева, какая из них несёт больше информации.

Это отдельная процедура, но в целом можно сказать, что каждая feature должна хорошо коррелировать с target, и они должны быть комплиментарными, они должны дополнять друг друга.

Они предложили достаточно изящную процедуру, мы можем это обсудить потом, сейчас у нас нет на это времени.

Это тоже является нововведением авторов из Microsoft Research.

В то же время вам никто не мешает использовать обычную процедуру поиска.

Ещё одна идея, которая привела к хорошим результатам.. Они учили регрессировать модель не только от mean shape и ground truth, а.. они назвали это augmentation, я бы не сказал, что это аугментация, но ладно.

Они учили регрессировать от каких-то других shapes, которые были выбраны случайным образом из train set.

Они учились этому, и это добавляло модели обучающую способность. Очевидно, что это представляет некоторое пространство для фантазии. Вы можете выбирать их не случайным образом.

В самом деле, зачем вам учить регрессор регрессировать от девушки в мужчину?

Очевидно, что они далеко стоят друг от друга, и это вам не нужно.

В реальной жизни пользователь один, он смотрит на телефон, его shape не меняется, ему не нужны такие далёкие преобразования, далёкие поиски.

Это очень удобное место, где можно применить свою фантазию, как-то пытаться искать.

Ну, а на тесте вы просто запоминаете дельта landmarks, лежащие в листах фернов,

у вас есть какое-то количество каскадов, какое-то количество фернов внутри каскада и дельта landmarks, лежащие в каждом листе.

И вы просто их последовательно прогоняете, и у вас текущая тестовая картинка попадает в какие-то листы этих фернов, смотрит на свои дельта landmarks и добавляет их последовательно к shape.

Здесь всё достаточно просто. Очевидно, возникает сразу идея: как всё это улучшить?

Получается, что можно поиграть с гипер-параметрами регрессора, можно подумать над аугментациями, потому что, например, очевидно, все data sets содержат большое количество фронтальных оиц и испытывают нехватку лиц в профиле: левых, правых профилей, испытывают нехватку лиц, поднятых вверх, взглядов исподлобья.

Очевидно, здесь можно как-то улучшить процедуру генерации of features.

Например, вы можете сэмплировать эти пиксели не случайным образом, а с помощью априорного распределения, привязывать их к landmarks, надеяться что это даст вам более быструю сходимость регрессора.

Вы можете обучить несколько регрессоров: например, один регрессор учиться регрессировать от профилей в ground truth, а другой - от фронтальных лиц в ground truth.

И вы можете на начальном этапе выбирать: "Ага, похоже, что это лицо фронтальное, я выберу вот этот регрессор, специальный фронтальный регрессор и буду использовать его".

Чтобы вы не слишком много ждали от вашей модели. Конечно, можно использовать глубокие features выходов нейронных сетей, это является state of the art в настоящий момент.

Вы можете использовать синтетическую data для аугментации data set, но, если честно, это не приносило больших успехов.

Конечно, также есть GANs. Вы можете использовать их для аугментации data set.

Вы можете применять новые архитектуры нейронных сетей, какие-то новые, придумывать свои слои и прочее. Вы можете придумать новое ЛОС, и мы увидим это. Также вы можете применять старые архитектуры.

Оказывается, что нет нужды искать какую-то новую удивительную архитектуру, потому что можно добиться цели, используя проверенные методы.

Some practical hacks. Когда мы делали даже просто базовую версию регрессора, который знает каждый практикующий исследователь, вы можете просто поиграть с гипер-параметрами, и это принесёт успех.

Если вы можете генерировать features не случайным образом, а привязывая их к landmarks: не бросая их случайным образом на картинку, а ставить их поближе в ожидаемое место - это приносит улучшения.

Вы можете делать "test time avaraging": вы запускаете регрессор несколько раз и соединяете результаты.

Мы будем рассматривать результат текущих работа на датасете 300-W, это достаточно сложное собрание нескольких других датасетов, из которого люди просто создали протокол и сказали: "Ага, мы будем использовать вот эти несколько тысяч изображений: 3148 training images,

554 testing images, and 135 images from this dataset that can be challenging.

Люди сказали, что это протокол на текущий момен, это самый репрезентативый протокол, поэтому я постарался унифицировать.

Здесь идея тренировки нескольких регрессоров. Вы на каждом этапе находите какое-то приближение своего shape, вы его просчитываете, и вы ищите в датасете похожие на него, например, если даже лицо повёрнуто в ту же сторону или в другую сторону.

Как только вы нашли что-то похожее, вы видите, у вас есть такой, и он похож на то, что вы регрессировали.

Вы находите кластер лиц, похожих на это лицо, то есть мы выкинули лицо, повернутое вправо, крестик. Выкинули повернутое вправо вниз, крестик. А вот лицо, повернутое влево, мы продолжаем, сэмплируем из дата сета еще лица, повернутые влево, одно, второе, третье, и опять смотрим, которое же из них похоже на то, что мы ищем.

Опять находим, ага, похоже, что вот это лицо больше всего смахивает на него, тоже повернуто влево, рот открыт.

Ага, уточним, просэмплируем еще, то есть на каждом этапе вы находите в дата сете те лица, похожие на то, что вы в данный момент регрессируете, то есть опять же, вы пытаетесь помогать модели, вы не усложняете ей задачу, не пытаетесь заставлять ее регрессировать от какого-то там сильно отличающегося лица, а как бы постоянно пытаетесь уточнять, и люди предложили баесовский фреймворк для этого, для поиска в пространстве шейпов, ну и были очень успешны. Работа 2015 года. Этот чисто деревянный регрессор, на дереве.

Такая достаточно тоже.. Первая работа была в 2013 году, сверточная сеть, она выглядит достаточно наивно по текущим представлениям, ну я привожу ее здесь, отдавая дань уважения людям, которые были в 2013 году достаточно смелыми, так сказать самоуверенными, чтобы запускать нейронную сеть для такой сложной задачи.

Это вот они вообще регрессировали всего лишь 5 ключевых точек на лице, делали постепенные уточнения, то есть на первый каскад подавали всю картинку, потом уточняли позицию всего лишь одной ландмарки, уточняли ее еще, ну и они получили какие-то результаты, которые в данный момент конечно не выдерживают никакой конкуренции, они не выдерживали даже и тогда конкуренции с регрессами на деревьях.

Дальше люди догадались из опыта, то мультитаск нейронки учится лучше, то есть коль скоро у вас есть несколько задач, похожих друг на друга, вы хотите искать ландмарки, хотите искать углы поворота лица, хотите оценивать пол человека, задачи схожи, мы запускаем несколько голов, и она вроде как учится лучше, ну это в принципе многие практикующие инженеры знают об этом, и вот эта работа, она иллюстрирует впервые такой подход, и в принципе она была успешной.

Действительно, современные все state of the art подходы, они базируются на поиске хитмапов, то есть я вкратце опишу, это convoluyional pose machines, которые пришли из технологии оценки позы положения человека, то есть вы пропускаете, вы бежите по картинке, и классифицируете каждый раз то, что вы видите на картинке, у вас мультикласс классификатор, который классифицирует, принадлежит ли этот патчик вот какой... Да, вот здесь хорошо видно. Этот патчик...

Допустим, вы ищете колено. И вы классифицируете: ага, вот это мы видим колено или не колено? И получается какой-то конфиденс для этого, в какой-то степени уверенности. Потом вы берете все больше и больше патчики, потому что когда вы берете маленькие, у него не хватает, у этого паттерна, у него не хватает как бы точности. Он выдает вам часто ложные тревоги. А потом у вас когда патчик больше, вы ловите больше контекста и вы говорите: "А, ну вот действительно, похоже, что это колено". И вы вот это свечение ложной тревоги убираете с помощью вот этих более больших патчей.

И этот подход выдает вам хитмап ландмарок, который вы можете потом использовать. И даже работа 2018 года, они использовали этот хитмап для поиска позы лица. Дальше они находили такую хитмап ландмарок, фитили 3Д модель в него и говорили: "Ага, похоже, что это лицо, повернутое на вот такой угол азимутальный, угломестный плоскости картинки. Давайте возьмем его как начальное приближение регрессора и натреним все тот же хорошо знакомый нам ансамбль деревьев".

И это был.. тут наверное видно... получил очень хороший результат, на мысе это 4,55, это высоко. Предыдущая работа на последних каскадах, там было 5,60 что ли, то есть это было улучшение, это 2018 год. Это говорит о том, что можно использовать смесь.

Тут люди пытались использовать то, что все точки, через них можно провести линию, и как-то пытались это использовать, они предложили очень сложную архитектуру, это к вопросу о том, что вы можете играть с архитектурами очень диким образом, вот вы можете оценивать эту boundary лица и засовывать ее как image fuzion, фьюзить ее с фичемапами изображений, делать здесь преобразования, еще фьюжн-слои, достаточно сложный фьюжн, который я здесь не привожу.

И вот они пытались. Да, они что-то получили, но лично мне не кажется, что это перспективный подход. Здесь люди пошли еще дальше. Они заметили, что оно и то же лицо вот этой симпатичной девушки, приведенное в различных стилях, приводит к тому, что регрессор выдает разное положение ландмарок.

Вот посмотрите, здесь рот приоткрыт как будто, а здесь все три точки сошлись, то есть рот закрыт. В зависимости... нет у него устойчивости. Но опять же, вместо того, чтобы поиграть с цветовыми пространствами, они предложили очень сложную схему, они в фотошопе сгенерили из исходного дата сета картинки с новым стилем, зафантюнили на них easenet на классификатор стиля, который дает в ответ в каком стиле эта картинка. Кластеризовали их и натренировали cycle gan для этого, для того, чтобы генерировать картинки в новом стиле.

И после этого они этот модуль для тренировки применяли, по сути, такая аргументация на лету стилями. И даже если опять сложный поиск хитмапов, фьюжн, конкретинэйшн, они получили 424, то есть это state of the art был на тот момент, но вся эта модель весила 500 мегабайт по-моему, это какой-то ад. А теперь я хочу перейти уже ближе к завершению доклада, к двум практическим работам, которые мне очень нравятся. Это вот работа 2019 года, месяц назад вышла, proectical facial landmark detector. Здесь они сделали очень простую вещь: они...

Опять к вопросу о старых и новых архитектурах. Они взяли mobile net, 2 мегабайта, и заэкстрактили из него фичи и подал на вспомогательную сетку, которая высчитает углы поворота лица, java pitcheral. Дальше они эти углы поворота подают вот сюда, в лосс, и лосс выглядит таким образом, где Д - это просто евклидово расстояние между ландмарками, а дельта принимает значения 1, 2, 3. 3 угла у вас, 3 степени свободы вращения лица, и как только у вас... А Т - это ошибка оцененного угла и ground угла. То есть как только у вас ошибка появляется, косинус становится меньше 1, и у вас растет эта сумма, придавая больший вес этому сэмплу из тренировочного сета.

А вот эта омега - это еще один весовой коэффициент, они разделили тренировочную выборку на несколько классов, то есть, допустим, лица с улыбками, лица с открытым ртом, лица профильные, лица какие-то еще, и в зависимости от степени презентативности этого класса в общетренировочном сете, они придают вес, то есть обычный класс имбэланс, но они его засунули в лосс и получили очень удобную процедуру тренировки, вам ничего не надо делать, у вас здесь mobile net, какая-то достаточно простая сетка, оценивающая углы, и вот здесь у вас тоже хороший лосс, который делает hard negative maining на лету. Очень удобно, они получили 3,76. Это satte of the art. Еще одна очень хорошая идея - это wing loss. Опять практикующие инженеры знают, это нынче такой модный лосс. Люди заметили, что Л2 лосс сильно штрафует большие отклонения ландмарки от истинного значения.

Получается, вы ошиблись в одной ландмарке, ошибка большая, и вы начинаете тянуть остальные ландмарки, пытаетесь скомпенсировать эту ошибку, весь шейп лица у вас едет, но это не очень хорошо, это некрасиво. Они сказали: "Ага, мы будем использовать Л1 лосс". Многим известно что Л1 лосс с выбросами как-то лучше работает. Но они пошли дальше, они говорят: "А давайте посильнее штрафовать маленькие ошибки", то есть они предложили использовать функцию логарифма в окрестности маленьких ошибок. То есть когда ошибка большая, лосс растет линейно, а когда маленькая, производная у нее прям большая.

Оказалось, что действительно это приносит, они натренили очень простую нейронку пятислойную и получили фактически state of the art результаты, но они пошли дальше. Они оффлайн перед тренировкой аргументировали лица недостающие, профильные лица, и они сделали еще одну вещь, они сделали вспомогательную нейронную сеть, которая рефайнит баундбокс, то есть делает его чуть точнее, чем стоковый фейсдетектор, который у них был, они сделали такой рефайнмент. И от этого становится очень хорошо. Это здесь такая таблица, которую я привел, не знаю для чего, чтобы произвести на вас впечатление может быть. Как много всяких работ и как люди прогрессировали, и вот ошибку с годами уменьшали, вот она была 8, 7, 6, 5, 5, 4, 4, и вот эти ребята, они используют, кстати, easent 50, то есть достаточно хорошо известная архитектура, опять старая, то есть они не делали какой-то мудреный gan, они идентифицировали проблему. Профильных лиц мало, сэмплы с повернутыми лицами тяжелые, очень сильно влияет точность оценки баунд бокс детектора, они идентифицировали 3 проблемы, конкретизировали и и решили их. То есть не играя с какими-то там жуткими архитектурами. И немного про трекинг.

То есть вы можете, когда работаете с приложением, можете либо на каждом кадре делать детекцию желтеньким, из нее регрессировать ландмарки, это классический подход, можете запускать трекер. Вы запускаете на первый кадр трекер, трекер - это такая легкая версия, она помогает вам отслеживать лицо, обычно он легче детектора. То есть ваш телефон не так сильно нагружается. И опять же с помощью трекера, но эта версия более-менее устойчива, потому что трекер склонен ошибаться.

И еще очень важный момент хотелось бы затронуть, как вы можете понять, что у вас слетел ваш шейп. Вы ошиблись, вы рисуете юзеру не то, ваша глупая модель, она же не знает этого, а человеку это не нравится, у него слетела маска с телефона. И вот это подход, который говорит о том, что если у вас слетел баундинг бокс, ваш регрессор регрессирует что-то не то, лицо висит где-то на ухе, вам нужно собрать дата сет из таких позитивных примеров и негативных примеров, собрать на нем патчики, фичи, какие-то сифт фичи, можно поиграть, что угодно.

И можно обычный СВМ-классификатор использовать, это хорошо работает. И вы, применяя это, онлайн трекаете. И последнее, наверное, это вот текущее. Гугл зарелизила Arcore, мне опять же приятно гвоорить об этом в Минске, потому что это инженеры из Хаби бывшей, в основном это, я так понимаю, их достижение. Они сделали на основе нейронных сетей трекер вот такого достаточно сложного меша, здесь очень много ландмарок, они сделали егов реал тайме, использовав мощь цензор флоу лайт, которая, естественно, будучи инженером гугла, вам предоставляет.

И они добились хороших результатов, это очень быстро работает сетка, она не без недостатков. Мы можем это обсудить, конечно, они брали смесь синтетических данных и реальных 3Д данных, синтетические плюс реальные данные 2Д для того, чтобы сеть, обученная на синтетических данных, немного ее регуляризовать. Они не применяют никакого сложного трекинга, они на каждом кадре запускают детектор, делают сингл фрейм регрешн, используют сглаживание, но тем не менее, еще раз говорю, мне приятно говорить об этом в Минске, то есть это ваши земляки. От этой проблемы, вам бы хотелось персонализировать ваш трекер на людей на лету, вы уже поняли, что этот кастомер - это не какой-то там any person в мире, а это я, человек. Он здесь, никуда не девается.

Вы хотите как-то адаптировать свой регрессор, вы хотите делать хороший трекинг. Сейчас это очень старые подходы для этого применяются, и пока что никому не удалось еще как-то по-умному смерджить регрессор точек и трекер, пока что все это очень базовые вещи, поправка на оптический поток, калман, туда-сюда. Конечно, нехватка регрессии 3Д-шейпа. Здесь не хватает дата сетов. И очень важный вопрос - system design. У вас есть несколько блоков, несколько подсистем, вы хотите оптимизировать их работу вместе, вы хотите зафантюнить каждую из них под другую и получить системный результат, который очень хороший. Это лист открытых проблем, которым я хотел бы завершить. Спасибо вам большое, что вы в выходной пришли сюда, Арсений поблагодарил и я благодарю вас тоже, мне очень приятно было рассказать вам, надеюсь, кому-то это было полезно. Все. У нас очень мало времени на вопросы, я прошу прощения, немного затянул доклад.

- Женя, спасибо за... Слышно?

- Женя благодарит меня за доклад.

- Да, давайте, может еще раз можете похлопать, будет прикольно. Я умею манипулировать толпой, оказывается. Ну да, у нас есть 10 минут буквально на доклады, давайте, поднимаем руку, и вам поднесут микрофон.

- Здравствуйте, спасибо за доклад. Мне очень понравилось и было интересно. А вот скажите, смотрите, мы сначала накидываем точки, да, а потом пытаемся их отрегрессировать, а что делать, если у нас, например, лицо повернуто в профиль, и, допустим, глаза одного не видно. Ну или например волосы прикрывают.

- Это очень просто решается, эти точки просто исключаются из лосса, там есть такой нюанс, о котором я не сказал, у каждой точки есть лэйбл, визибилити, он равен 0 или 1, или вы можете сделать что-то advanced, какую-то вероятностную визибилити, ну это опять же место для открытого ресерча, вы можете написать об этом работу, стать знаменитым и все у вас будет хорошо. То есть там есть просто лэйбл визибилити, и если 0, то просто лосс от этой точки умножается на 0 и она не участвует в оптимизации.

- Спасибо.

- Да, давайте смелее поднимаем руки.

- В конце зала тоже можно поднимать руки.

- Вон молодой человек в дали.

- Спасибо за доклад, интересует такой вопрос: часто точки, которые предсказываются, они дрожат. Как это решать?

- Дрожат, да. Для того, чтобы не дрожали... Там есть несколько причин. Вы делаете wing loss, то есть сильно пенализируете маленькие отклонения, вы применяете какой-то сглаживающий фильтр, то есть вы можете трекать калманом, да, предсказывать состояния, то есть на один кадр вы будете задерживать картинку, которую вы будете показывать пользователю, но эта картинка, жизнь показывает, что это ничего. Вы применяете калмана, то есть предсказываете калманом, оцениваете, усредняете, то есть достаточно классическая схема. Это второй подход. Калмана линейного не надо применять, надо применять extended calman или... вот эти вещи, нелинейные, которые способны... Потому что это очень нелинейно, когда вы делаете какую-то мимику, я вот показываю, что-то лицом делаете, это нелинейные движения, вам линейный предиктор здесь не поможет. Поэтому оптический поток, ну, многие пытаются, но оптический поток - это достаточно простая вещь, и есть работа от фейсбука - Supervision by Registration, они делают поправку на... Они научились дифференцируемый оптический поток сформулировать, засунули его в нейронку, пропускают через него градиент, и они разделяют какие-то движения на ригидные и неригидные, ригидные - это когда у вас все лицо движется параллельным переносом, и вы это движение компенсируете через оптический поток, а какие-то неригидные, какую-то тонкую мимику человека, вы регрессируете как и раньше, потом, вы можете сглаживать одноевровым фильтром, например. То есть те точки, которые движутся быстро, вы окно усреднения берете поменьше, а те, которые движутся медленно, у вас окно усреднения побольше. Это кстати очень хорошо выглядит. Потом вы можете профили сильно сглаживать, делать большое окно усреднения.

Это не очень корректно с точки зрения науки, но с точки зрения wser experience это очень хорошо. То есть такое оно у вас, знаете, замораживается немного в конечных точках, можете так делать. И еще они дрожат из-за того, что фичи у вас нестабильные. То есть лицо... Человек держит телефон и лицо стабильное, а рука немного дрожит, и у вас немного меняются интенсивности пикселей. Тут вам помогут градиентные фичер экстракторы, то есть вы к этим pixel difference добавляете ещё какие-то более сложные фичи, какие-то сифты и туда-сюда.

Есть много подходов, но побеждает тот, кто грамотно их все применяет в правильной последовательности и не боится грязной работы: брать телефон в руку, смотреть, оптимизировать, крутить, брать, смотреть - это очень трудоёмкий процесс. Вы задаёте правильный вопрос, но, к сожалению, хорошее решение этой проблемы.. Ещё можно проецировать на PCA модель, вы можете посчитать по своему транссету PCA модель, уменьшить шумовые компоненты и каждый свой пересчитанный шейп вычислять проекцию на эту PCA модель. Тогда у вас в каждый момент времени shape будет находиться в пространстве вашей усечённой PCA модели. Это ещё один способ. Но он приводит к тому, что у вас шейпы становятся примитивными. Какие-то выбросы не будут регрессировать. Есть подходы, но ни один из них не является серебрянной пулей.

- Понятно, спасибо большое.

- Меня интересует ваше мнение: с точки зрения практической применимости, какую модель вы считаете наиболее перспективной по трём противоречивым параметрам - скорость работы, точность и занимаемая память.

- Я уже сказал, что мне очень нравятся две последние работы, которые я привёл в пример, потому что они на mobile net, это очень маленькая сетка, у неё очень много ФПС, она хорошо работает.

И вот, которая с wing loss, и вот ребята из Гугла сделали тоже нейронки, но я не очень люблю нейронки, потому что, если вы идёте в андроид разработку, устройств очень много, на них очень разные GPU, и вам очень трудно с точки зрения продукта удовлетворить требования всех пользователей. Вы будете получать очень противоречивые отклики, кто-то будет говорить, что всё классно, а у кого плохой телефон - будут говорить, что всё плохо. А вот CPU более менее один и тот же. Поэтому я люблю, и у нас в продаже регрессоры на деревьях. Просто мультитред, регрессор на деревьях. И как показывает опыт, на этих регрессорах можно выжать очень хорошее качество, не уступающее все этим нейроночным. Нужно просто грамотно с точки зрения системного подхода аккуратно подходить к оптимизации вашей системы. Это вопрос системного дизайна.

Персонально я не очень люблю вот эти все нейроночные вещи сейчас. Я верю, что через пару лет всё изменится. И вот google arcore работает на нейронке, он показывает хороший результат и на CPU, и на GPU. Это очень свежий фреймворк, я пока не могу ничего толком об этом сказать, к сожалению. Но похоже, что, как только censor flow light достигнет какой-то степени зрелости, и вы сможете деплоиться на CPU с приемлемым качеством, возможно, это будет прорыв. Пока что я больше верю системы на деревьях.

- Я думаю, это был последний вопрос, но не расстраивайтесь, если вы не успели задать вопрос, у нас есть отдельная networking zone, где вы всегда можете найти Женю, но возможно не всегда, потому что ещё частично он будет помогать со сбором датасета. Да. Женя, ты об этом не говорил?

- Да, у нас есть такая инициатива от ODS, это такая..

как назвать это..

- Научно-развлекательная.

- Non-commercial. Мы попытаемся собрать датасет различных эмоций, различных action units, bland shapes. У нас есть фотограф на stage 2, я очень вас прошу, если вам не лень, не жалко, и вы не боитесь, что ваша фотография окажется в лапах незнакомых людей, приходите, фотограф сделает 20 фотографий вас: с закрытыми глазами, с улыбкой, с подмигиванием и так далее. Мы хотим этот датасет зарелизить, выложить от ODS в научных целях, чтобы на нём можно было предобучать модели, например, emotion recognition, для анимации. Будет сделано объявление. Алексей Натёкин объявит это, никаких проблем. Каждый из тех, кто поучаствовал в создании, получает свою копию под подпись, что он не будет её применять в каких-то отрицательных способах. Non-commercial. Я вдохновляю вас участвовать в этой

активности.

- А еще если вы сфотографируетесь на датасет, то ваши градиенты гарантированно будут сходиться быстрее. Это проверено. Давайте поаплодируем докладчику.

Вы к этим pixel difference добавляете ещё какие-то более сложные features, какие-то SIFTs and so on.

Есть много подходов, но побеждает тот, кто грамотно их все применяет в правильной последовательности и не боится грязной работы: брать телефон в руку, смотреть, оптимизировать, крутить, брать, смотреть - это очень трудоёмкий процесс.

Вы задаёте правильный вопрос, но, к сожалению, хорошее решение этой проблемы..

Ещё можно проецировать на PCA модель, вы можете посчитать по своему транссету PCA модель, уменьшить шумовые компоненты и каждый свой пересчитанный shape вычислять проекцию на эту PCA модель.

Тогда у вас в каждый момент времени shape будет находиться в пространстве вашей усечённой PCA модели.

Это ещё один способ.

Но он приводит к тому, что у вас shapes становятся примитивными. Какие-то выбросы не будут регрессировать.

Есть подходы, но ни один из них не является серебрянной пулей.

-Понятно, спасибо большое.

-Меня интересует ваше мнение: с точки зрения практической применимости, какую модель вы считаете наиболее перспективной по трём противоречивым параметрам - скорость работы, точность и занимаемая память.

-Я уже сказал, что мне очень нравятся две последние работы, которые я привёл в пример, потому что они на (?41.21 mobile нете???), это очень маленькая сетка, у неё очень много ФПС, она хорошо работает.

И вот, которая (??41.28?), и вот ребята из Гугла сделали тоже нейронки, но я не очень люблю нейронки, потому что, если вы идёте в андроид разработку, устройств очень много, на них очень разные GPU, и вам очень трудно с точки зрения продукта удовлетворить требования всех пользователей.

Вы будете получать очень противоречивые отклики, кто-то будет говорить, что всё классно, а у кого плохой телефон - будут говорить, что всё плохо.

А вот CPU более менее один и тот же.

ПОэтому я люблю, и у нас в продаже регрессоры на деревьях. Просто мультитред, регрессор на деревьях.

И как показывает опыт, на этих регрессорах можно выжать очень хорошее качество, не уступающее все этим нейроночным. Нужно просто грамотно с точки зрения системного подхода аккуратно подходить к оптимизации вашей системы. Это вопрос системного дизайна.

Персонально я не очень люблю вот эти все нейроночные вещи сейчас.

Я верю, что через пару лет всё изменится.

И вот google arcore работает на нейронке, он показывает хороший результат и на CPU, и на GPU.

Это очень свежий фреймворк, я пока не могу ничего толком об этом сказать, к сожалению. Но похоже, что, как только (42.50?) достигнет какой-то степени зрелости, и вы сможете деплоиться на CPU с приемлемым качеством, возможно, это будет прорыв.

Пока что я больше верю системы на деревьях.

-Я думаю, это был последний вопрос, но не расстраивайтесь, если вы не успели задать вопрос, у нас есть отдельная networking zone, где вы всегда можете найти Женю, но возможно не всегда, потому что ещё частично он будет помогать со сбором датасета. Да. Женя, ты об этом не говорил?

-Да, у нас есть такая инициатива от ODS, это такая..

как назвать это..

-Научно-развлекательная.

-Non-commercial. Мы попытаемся собрать датасет различных эмоций, различных action units, bland shapes.

У нас есть фотограф на stage 2, я очень вас прошу, если вам не лень, не жалко, и вы не боитесь, что ваша фотография окажется в руках незнакомых людей, приходите, фотограф сделает 20 фотографий вас: с закрытыми глазами, с улыбкой, с подмигиванием и так далее.

Мы хотим этот датасет (??43.56), выложить от ODS в научных целях, чтобы на нём можно было предобучать модели, например, emotion recognition, для анимации.

Будет сделано объявление.

Алексей объявит это, никаких проблем.

Каждый из тех, кто поучаствовал в создании, получает свою копию под подпись, что он не будет её применять в каких-то отрицательных способах. Non-commercial.

Я вдохновляю вас участвовать в этой активности.

Если вы сфотографируетесь, то ваши градиенты гарантированно будут сходиться быстрее.

Это проверено.

Давайте поаплодируем докладчику.