Господа, здравствуйте, меня зовут Никита Сергеев. И я хочу вам рассказать, как работает AlphaGo. Поскольку у нас в распоряжении только 7 минут, буду краток, освещаемые вопросы вы видите на слайде, это постановка задачи, идея того, как альфаго принимает решения (алгоритм поиска по дереву), модели, используемые для принятия решений и их взаимосвязь и архитектуры используемых нейросетей и подаваемые фичи.

На руках имеется игра с полной информацией, аналогичная шашкам или крестикам-ноликам, только слегка побольше. Есть поле, за которое мы не выходим, легальные действия, функция перехода состояния поля, функция вознаграждения. Также мы вводим понятие полиси - распределение вероятностей совершенных ходов и вэлью фанкшн - ожидаемый исход игры, если действия будут совершаться как наиболее вероятные согласно полиси. Собственно задача в том, чтобы узнать оптимальную вАлью фанкшн при правильной игре обеих сторон.

О компонентах. Начнем с supervised policy network. Цель - получить вероятность того, куда поставит камень человек. Получаем на вход состояние доски, прогоняем его через сверточные слои - сначала паддингующий до 23x23 слой, потом слой из 192 фильтров 5x5, потом 11 слоев из 192 фильтров 3x3, и в конце фильтр 1x1 с разными биасами для каждой позиции, к чему впоследствии применяется softmax получаем на выходе доску с вероятностями хода соперника на каждой клетке. Обучается сеть на 160000 человеческих играх.

Теперь ролаут полиси. Цель - выдавать вероятность выпадения человеческого хода быстро. Это очень быстрая стратегия, которая является просто линейным классификатором. Ей на вход дают еще больше заготовленных фич. В общем, работа ролаут полиси сводится к тому, чтобы линейно сложить фичи по каждой клетке поля.

Перейдем к reinforcement policy. Сетка играет с одной из предыдущих версий себя (веса взяты с предшествующей итерации). Каждая итерация обучения состоит из 128 параллельно запущенных игр и каждые 500 из 10000 итераций веса у противника обновляются. Каждая игра доигрывается до конца и после получения результата полиси грэдиэнт обновляется таким образом. Здесь мы используем бэйзлайн как значение вэлью фанкшн чтобы уменьшить дисперсию.

Теперь собственно о вэлью фанкшн. Так как выигрышные позициии сильно коррелируют между собой, чтобы избежать переобучения мы делаем финт ушами. Выплевываем рандомное число U. Шаги до U производятся согласно полиси нетворк, утый шаг совершается рандомно, а все последующие, вплотьт до завершения игры, согласно рейнфорсмент полиси. Обучение и архитектура сети те же, что и у супервайзд полиси, за исключением того, что на выходе мы получаем не вероятности, а число - предсказанный исход игры.

Теперь об алгоритме поиска. В данной программе используется алгоритм, имя которому Monte-Carlo Tree Search. Начнем с того, что каждое ребро дерева поиска хранит в себе набор чисел (описываю что за число что есть). Запускается

параллельно несколько симуляций игр. На первом этапе - searching, происходит спуск по дереву до его листа такой, что от параметров ролаута отнимается/прибавляется число типапроигранных игр, чтобы другие параллельно запущенные ролауты не ходили сюда. Узлы для спуска выбираются вот так вот. По достижении листа он добавляется в очередь обработки вэлью фанкшн. Поскольку ролаут быстрее, он успевает проиграть игру из этого листа до конца и вернуться наверх, обновив соответствующие значения в ребрах. Затем, как только вэлью отработала, обновляются и ее параметры (расскахать про каждый параметр). Все повторяется до тех пор, пока в очереди у вэлью фанкшн лежат листья. Далее альфаго выбирает ход таким образом, чтоб в соответствующий ему узел за время поиска было совершено больше заходов.