


Описание работы над ботом.

▼ Эхо бот и пробный деплой на AWS

Свой проект я начал с того, что создал эхо бота на aiogram, почитал документацию Telegram.API и самого aiogram. Далее я попробовал задеплоить его на AWS с помощью lambda функции и GetAwayApi. Идея работы в том, что мы будем перенаправлять наши сообщения напрямую Амазону, а у лямбда функции будет стоять триггер на этот запрос. И тут я столкнулся с тем, что не до конца понимал, как правильно создать layer с библиотеками. В итоге благодаря этой статье разобрался и бот работал как надо, но на тот момент я не учел, что слои могут в сумме весить меньше 250мб...

Building Custom Layers on AWS Lambda

Many developers face issues when importing custom modules on AWS Lambda, you see errors like "No module named pandas" or "No module named numpy", and most times, the

 <https://towardsdatascience.com/building-custom-layers-on-aws-lambda-35d17bd9abbb>



▼ Инлайн кнопки, конечный автомат и асинхронность

Благодаря этому ресурсу у меня начало складываться представление, как будет работать мой бот.

<https://mastergroosha.github.io/telegram-tutorial-2/fsm/>

После того, как научился делать всякие кнопки и автоматы(состояния), ознакомился с понятием асинхронности(раньше думал, что это равно распараллеливанию), корутинами и тп.

Далее я написал бота с инлайн кнопками и состояниями. На этом этапе осталось написать классы моделей и придать двум кнопкам действия. Здесь столкнулся с такой проблемой, что не понимал, как отслеживать ситуации,

когда пользователь надергал кучу инлайн кнопок. Решил это, наверное, костыльно - благодаря словарю "num_inline_buttons_per_user", который отслеживает количество надерганных инлайн кнопок для каждого юзера и заданию каждой кнопке callback_data таким образом, что "callback_data.split('_')[2]" вернет число, которое будет означать какой по счету раз эти кнопки были созданы. И если это число не равно num_inline_buttons_per_user[message.chat.id], то значит юзер уже вызвал еще инлайн кнопки и зачем-то решил нажать на старые, но в таком случае мы просто удаляем эти кнопки, чтобы не было больше соблазна.

▼ Neural Style Transfer

Модельку брал отсюда, как рекомендовали в слаке.

<https://nextjournal.com/gkoehler/pytorch-neural-style-transfer>

Здесь в качестве весов выступает само изображение, а обучение занимает от 2-5 минут в зависимости от доступа к гри.

▼ Cycle Gan

Принцип обучения по сути описывается этими картинками, который кстати оправдывает слово "cycle" в названии :)

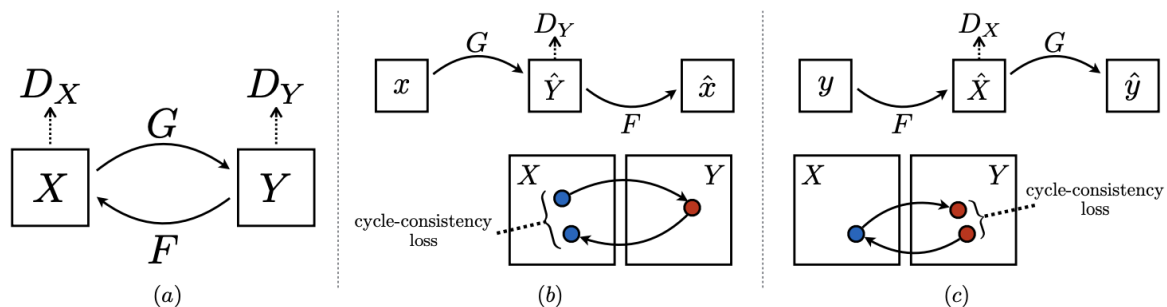


Figure 3: (a) Our model contains two mapping functions $G : X \rightarrow Y$ and $F : Y \rightarrow X$, and associated adversarial discriminators D_Y and D_X . D_Y encourages G to translate X into outputs indistinguishable from domain Y , and vice versa for D_X and F . To further regularize the mappings, we introduce two *cycle consistency losses* that capture the intuition that if we translate from one domain to the other and back again we should arrive at where we started: (b) forward cycle-consistency loss: $x \rightarrow G(x) \rightarrow F(G(x)) \approx x$, and (c) backward cycle-consistency loss: $y \rightarrow F(y) \rightarrow G(F(y)) \approx y$

Для обучения мы имеем два лосса, один из которых про дискриминатор

$$\mathcal{L}_{GAN}(G, D_Y, X, Y) = \mathbb{E}_{y \sim p_{data}(y)}[\log D_Y(y)] + \mathbb{E}_{x \sim p_{data}(x)}[\log(1 - D_Y(G(x)))]$$

А второй называется “Consistency Loss” и нужен для того, чтобы наше изображение все-таки оставалось похожим на изначальное

$$\mathcal{L}_{cyc}(G, F) = \mathbb{E}_{x \sim p_{data}(x)}[\|F(G(x)) - x\|_1] + \mathbb{E}_{y \sim p_{data}(y)}[\|G(F(y)) - y\|_1]$$

Предобученную модельку нашел на гите - переносит мультяшный стиль.

Работает меньше минуты, если не мгновенно, по крайней мере на моем скромном железе.

▼ Проверка бота на всякие исключения

На этом этапе вылезли некоторые косяки в логике обработки запросов, но, кажется, они были решены. Также пробовал писать боту одновременно с двух аккаунтов. С одного акка запустил обучение nst, а с другого писал ему и также запускал обучение. Никаких проблем не обнаружил - бот отвечал другому акку, параллельно обучая модельку для первого.

▼ Безуспешный Deploy 😞

Наверное больше половины всего времени я пытался понять, как можно все-таки задеплоить бота на AWS с помощью тех же лямда функций и сразу же столкнулся с тем, что подключаемые слои должны весить меньше 250мб, и тут я понял, что этот способ можно отметить и искать новый.

Тогда я понял, что можно сделать это с помощью ECR и EC2. В этот момент я разобрался с Docker, создал образ, запустил его локально и все работало как надо, но сколько я не пытался задеплоить все это дело на AWS, к сожалению так и не получилось.

Далее я попробовал деплой на heroku, бот даже работает на нем, но стоит начать обучать nst и все ломается из-за ошибки, с которой я так и не смог справиться, пробовал менять вебхук на пулинг, но не помогло.