Повторение эксперимента из статьи “Leveraging grammar and reinforcement learning for neural program synthesis”.

Задача синтеза программы состоит в том, чтобы автоматически сгенерировать программу, соответствующую
спецификации, такой как набор примеров ввода-вывода.

Предполагается, что во время обучения есть доступ к N обучающим выборкам, каждая из которых состоит из набора K состояний ввода/вывода и программы, корректно реализующей отображение:

$D = \left\{\left(\left\{IO_i^k\right\}_{k=1..K}, \lambda_i\right)\right\}_{i=1..N}$, такое, что: $\lambda_i(I_i^k) = O_i^k \; \forall i \in 1..N, \forall k \in 1..K$

где $\lambda_i(I_i^k)$– это конечное состояние после применения программы λ_i ко входному состоянию I_i^k. Задача – обучить генератор σ, который по набору примеров входов/выходов сможет сгенерировать программу, такую, что:

$$\sigma : \left\{IO_i^k\right\}_{k=1..K} \rightarrow \hat{\lambda}$$

Использованные данные

Сайт с данными для обучения доступен по ссылке: https://msr-redmond.github.io/karel-dataset/

Прямая ссылка на данные: https://rhgt8w.bn.files.1drv.com/y4mlPNtWRxNnyxHUQMq0LJu-wQ5T708OEsDl-eJ1NFdRMsgxGp8DeQNqdJklobolakwJbepphas2qzQpvfbnHo68tfeoptZ9X9-YUZSGp7SxA_1JPnUf-KnegM-ABeiOaRSkP4M5RmunS3l-q2DboPDFzwcovVY3gbFFwB7stv5nK8FRN37sHl50H37myAAm_1L-AXMetpigZUCOOlwQVbXXw

Далее, из данных для обучения были сформированы наборы по 1%, 3%, 10% исходного размера при помощи самописоного скрипта: https://github.com/michael2024-lw/labw_gandrl_nps/blob/master/DataLoader.py

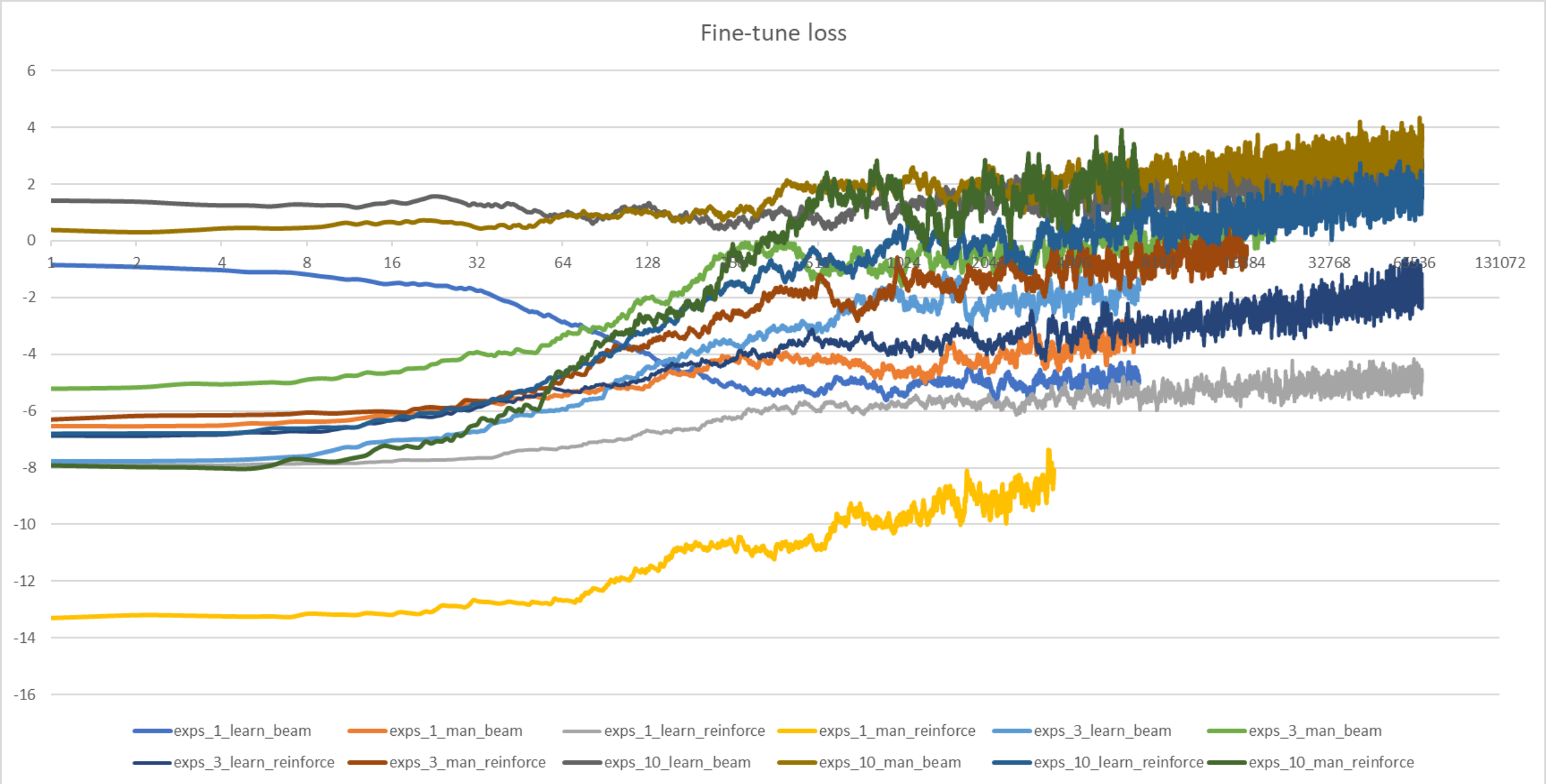
Постановка эксперимента

Репозиторий со всеми скриптами и данными находится по адресу: https://github.com/michael2024-lw/labw_gandrl_nps.git

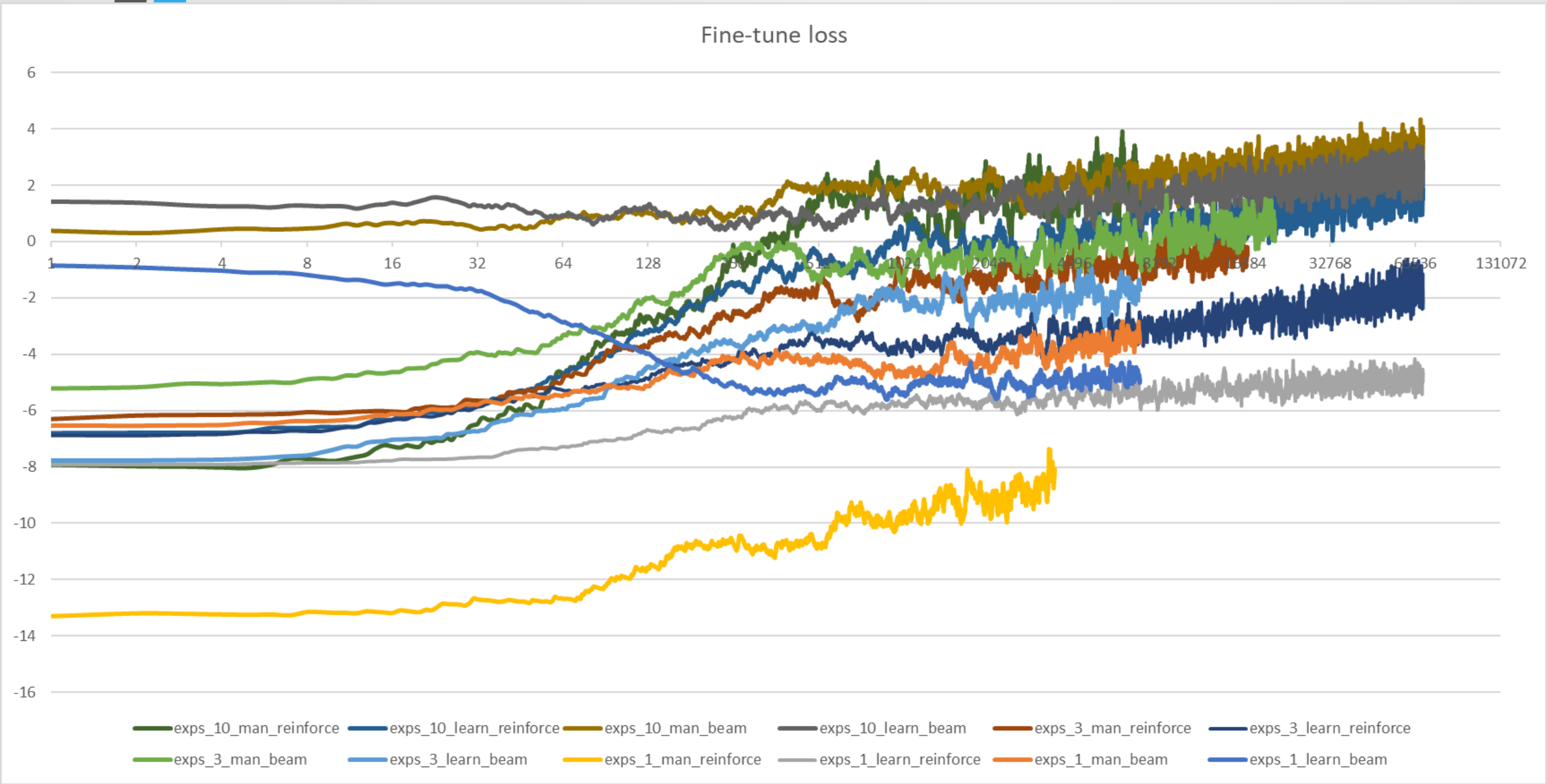
Команд для запуска эксперимента было довольно много, так что можно указать неизменную часть:

- train_cmd.py --kernel_size 3 --conv_stack "64,64,64" --fc_stack "512" --tgt_embedding_size 256 --lstm_hidden_size 256 --nb_lstm_layers 2 --signal supervised --nb_ios 5 --nb_epochs 100 --optim_alg Adam --batch_size 64 --learning_rate 1e-4 --train_file <path> --val_file <path> --vocab <path> --result_folder <path> --use_cuda [--use_grammar | --learn_syntax --beta 1e-5]
- train_cmd.py --signal rl --environment BlackBoxGeneralization --nb_rollouts 25 --init_weights <path> --nb_epochs 5 --optim_alg Adam --learning_rate 1e-5 --batch_size 8 --train_file <path> --val_file <path> --vocab <path> --result_folder <path> --use_cuda [--use_grammar | --learn_syntax --beta 1e-5]
- train_cmd.py --signal beam_rl --environment BlackBoxGeneralization --reward_comb RenormExpected --rl_inner_batch 4 --rl_use_ref --rl_beam 8 --init_weights exps/supervised_use_grammar/Weights/best.model --nb_epochs 5 --optim_alg Adam --learning_rate 1e-5 --batch_size 8 --train_file <path> --val_file <path> --vocab <path> --result_folder <path> --use_cuda [--use_grammar | --learn_syntax --beta 1e-5]

Результаты


Fine-tune loss
6
4
2
0
-2
-4
-6
-8
-10
-12
-14
-16
1
2
4
8
16
32
64
128
32768
131072
exps_1_learn_beam
exps_1_man_beam
exps_1_learn_reinforce
exps_1_man_reinforce
exps_3_learn_beam
exps_3_man_beam
exps_3_learn_reinforce
exps_3_man_reinforce
exps_10_learn_beam
exps_10_man_beam
exps_10_learn_reinforce
exps_10_man_reinforce

Результаты


Fine-tune loss
6
4
2
0
-2
-4
-6
-8
-10
-12
-14
-16
1
2
4
8
16
32
64
128
32768
131072
exps_10_man_reinforce
exps_10_learn_reinforce
exps_10_man_beam
exps_10_learn_beam
exps_3_man_reinforce
exps_3_learn_reinforce
exps_3_man_beam
exps_3_learn_beam
exps_1_man_reinforce
exps_1_learn_reinforce
exps_1_man_beam
exps_1_learn_beam

Сводная таблица

Тестируемая сеть	Критерии			
	exactmatch	fullgeneralize	semantic	syntax
1_learn_reinforce	14.16	19.68	20.12	100.0
1_learn_beam	21.12	29.92	30.8	100.0
1_man_reinforce	21.76	31.4	32.4	100.0
1_man_beam	22.36	32.68	33.64	100.0
3_learn_reinforce	27.0	40.88	41.96	100.0
3_learn_beam	29.68	45.08	46.4	100.0
3_man_reinforce	31.36	48.76	50.24	100.0
3_man_beam	31.64	49.84	51.2	100.0
10_learn_reinforce	35.24	55.28	56.92	100.0
10_learn_beam	37.32	59.64	61.56	100.0
10_man_reinforce	38.04	60.64	62.28	100.0
10_man_beam	41.44	65.56	67.68	100.0