НИУ ВШЭ
Курсовая работа

Тестирование методов оптимизации машинно-обучаемых межатомных потенциалов

MTP и ETN · BFGS · SciPy BFGS · SGD · Momentum · Adam · MUON · Adam + L-BFGS

Заренков Пётр Иванович
Факультет компьютерных наук НИУ ВШЭ
Руководитель: Новиков Иван Сергеевич
2026

Контекст и постановка задачи

Зачем нужны ML-потенциалы и почему важна оптимизация

Зачем нужны ML-потенциалы

Проблема DFT

- DFT даёт точные энергии и силы, но каждый расчёт дорог.
- Молекулярная динамика требует тысячи и миллионы вызовов E и F.
- Для больших атомных систем прямой DFT становится узким местом.

Идея ML-потенциала

- Обучить функцию энергии Eθ(R) на DFT-разметке.
- Силы получать как производные энергии по координатам.
- Сохранить физические симметрии и ускорить симуляции.

Что даёт работа

- сравнение MTP и ETN;
- проверка оптимизаторов;
- анализ качества и времени;
- оценка устойчивости запусков.

Вывод: ML-потенциал полезен только если его параметры удалось хорошо и устойчиво оптимизировать.

Почему оптимизатор — ключевой элемент

Что усложняет оптимизацию

- loss объединяет energy и forces с разными масштабами;
- силовых компонент намного больше, чем энергетических значений;
- поверхность loss может быть плохо обусловлена;

- MTP и ETN имеют разную параметризацию и разные траектории обучения;

- mini-batch методы добавляют шум оценки градиента.

Зачем сравнивать методы

- одинаковая модель может давать разное качество при разных оптимизаторах;

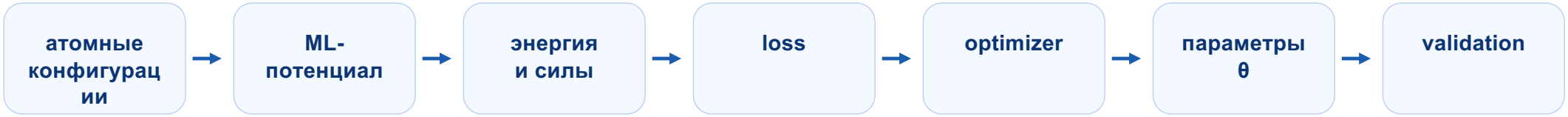
- final loss не всегда совпадает с лучшей validation EPA RMSE;
- важны не только ошибки, но и время, outliers и стабильность;
- success-флаг SciPy надо читать вместе с validation metrics.

Критерии сравнения оптимизаторов

Критерий	Что измеряет	Зачем нужен
Качество	validation EPA RMSE, forces RMSE	показывает точность потенциала
Цена	train time, число шагов	показывает вычислительную стоимость
Устойчивость	5 запусков, std, outliers	показывает зависимость от инициализации

Вывод: Задача курсовой — проверить не только качество модели, но и поведение оптимизаторов.

Общая схема обучения потенциала



Замкнутый цикл обучения

- модель предсказывает энергию и силы для конфигураций;
- loss сравнивает предсказания с DFT-разметкой;
- optimizer изменяет параметры θ и запускает следующий шаг;
- после обучения считаются train/validation metrics.

Что сравнивается

- семейства методов: SGD, Momentum, Adam, BFGS, MUON;
- метрики: EPA RMSE, forces RMSE, train time, final loss;
- ансамбли по 5 инициализациям там, где это применимо;
- динамика loss для объяснения траекторий обучения.

Вывод: Оптимизация связывает физическую задачу, ML-модель и итоговое качество симуляций.

Постановка задачи

$$L(\theta) = w_E \sum_{k=1}^{K} \left(E_\theta(R_k) - E_k^{DFT}\right)^2 + w_F \sum_{k=1}^{K} \sum_{i=1}^{N_k} \left\| F_{\theta,i}(R_k) - F_{i,k}^{DFT} \right\|^2$$

Что минимизируется

- θ — параметры потенциала;
- Eθ и Fθ — предсказанные энергия и силы;
- Eref и Fref — DFT-разметка;
- wE, wF задают баланс energy/force частей.

Как оценивается метод

- validation EPA RMSE — ключевая energy-метрика;
- validation forces RMSE — качество по силам;
- train time — вычислительная цена;
- loss histories — диагностика mini-batch методов.

Вывод: Оптимизатор сравнивается по качеству, времени и устойчивости, а не только по final loss.

Модель MTP: Moment Tensor Potential

В MTP энергия конфигурации раскладывается на локальные атомные вклады: окружение каждого атома описывается moment tensor descriptors, из которых строятся инвариантные базисные функции для предсказания энергии.

Как устроен MTP

- энергия системы раскладывается в сумму локальных атомных вкладов;
- окружение атома описывается moment tensors: радиальная часть + угловые моменты;
- базис учитывает переносы, повороты и перестановки атомов;
- усложнение модели повышает ёмкость, но увеличивает цену обучения.

Почему важен для оптимизации

- MTP — сильный baseline в MLIP-4;
- full-batch loss достаточно гладкая для BFGS;
- модель чувствительна к масштабу начальной точки в SciPy BFGS;
- на MTP хорошо видно различие first-order и quasi-Newton методов.

Вывод: MTP — удобная базовая модель, где особенно заметен выигрыш квазиньютоновских методов.

Модель ETN: Equivariant Tensor Network

ETN строит энергию через тензорную сеть: атомное окружение кодируется эквивариантными признаками, которые сохраняют информацию о геометрии системы, а итоговая свёртка сети даёт инвариантную энергию конфигурации.

Как устроен ETN

- параметры задаются тензорной сетью с контролируемыми ranks;
- эквивариантные признаки согласованно меняются при поворотах;
- энергия остаётся инвариантной, а силы согласуются с геометрией;
- shape/rank управляют ёмкостью и сложностью оптимизации.

Что проверялось

- tuning rank и shape на AgPd;
- native BFGS против SciPy BFGS;
- SGD, Momentum, Adam и MUON на AgPd;
- перенос сравнения на MoNbTaVW;
- проверка, переносимы ли выводы с MTP.

Вывод: ETN компактнее и геометрически структурированнее, но оптимизационно ведёт себя иначе, чем MTP.

Данные, задачи и метрики

Блок	Датасет	Модель	Что проверялось
Task 1	AgPd	MTP / ETN	native BFGS, tuning ETN rank/shape
Task 3	AgPd	MTP / ETN	SciPy BFGS и начальная точка
Task 4	AgPd	MTP / ETN	SGD, Momentum, Adam, MUON, Adam+L-BFGS
Task 5	MoNbTaW	ETN	возмущение параметров
Task 6	MoNbTaVW	ETN	перенос сравнения на другой датасет

Метрики

- train/validation energy RMSE и EPA RMSE;
- validation forces RMSE;
- train time, final loss, steps/epochs;
- nit/success для SciPy BFGS;
- loss histories для mini-batch методов.

Устойчивость

- ансамбли из 5 запусков;
- mean/median/std и outliers;
- отдельное сравнение качества и времени.

Вывод: Единый набор метрик нужен, чтобы сравнение не зависело от конкретного скрипта обучения.

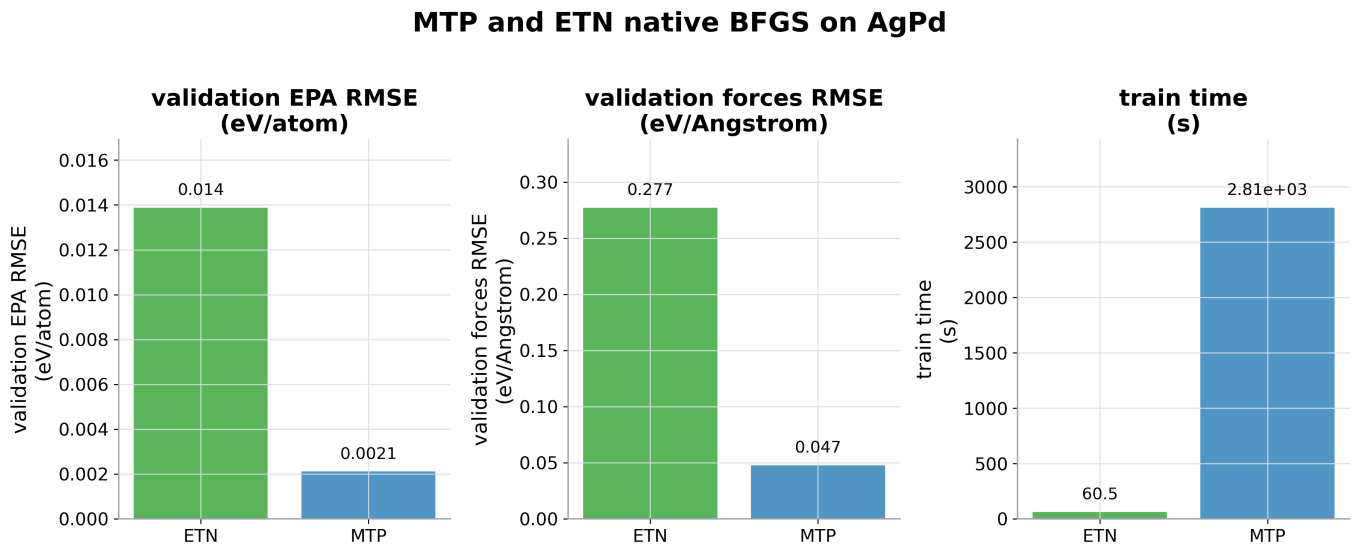
Task 1

MTP и ETN на AgPd при native BFGS

Task 1: MTP vs ETN по validation RMSE

Численные результаты

модель	val EPA	forces	time
MTP native BFGS	0.00210	0.04746	2810.3 s
ETN native BFGS	0.01387	0.27712	60.5 s

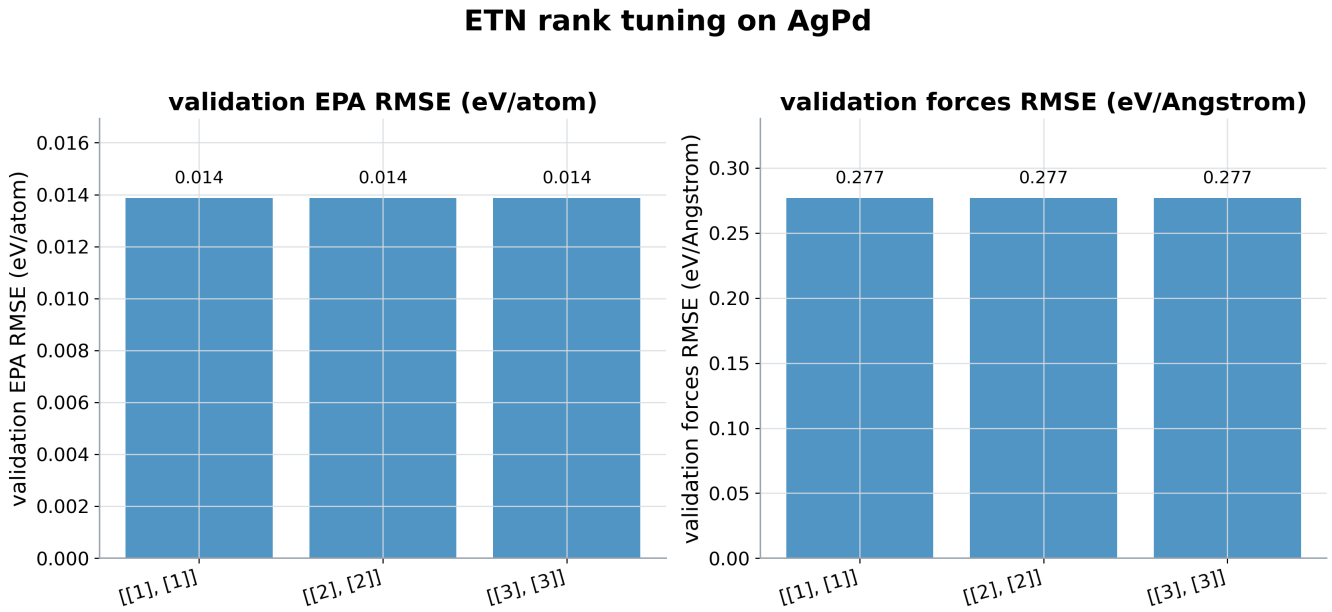


Вывод: MTP при native BFGS даёт меньшую validation error, но требует заметно большего времени.

Task 1: tuning ETN rank

Численные результаты

rank	val EPA	forces	time
[[1],[1]]	0.01387	0.27712	58.6 s
[[2],[2]]	0.01387	0.27712	58.6 s
[[3],[3]]	0.01387	0.27712	53.7 s

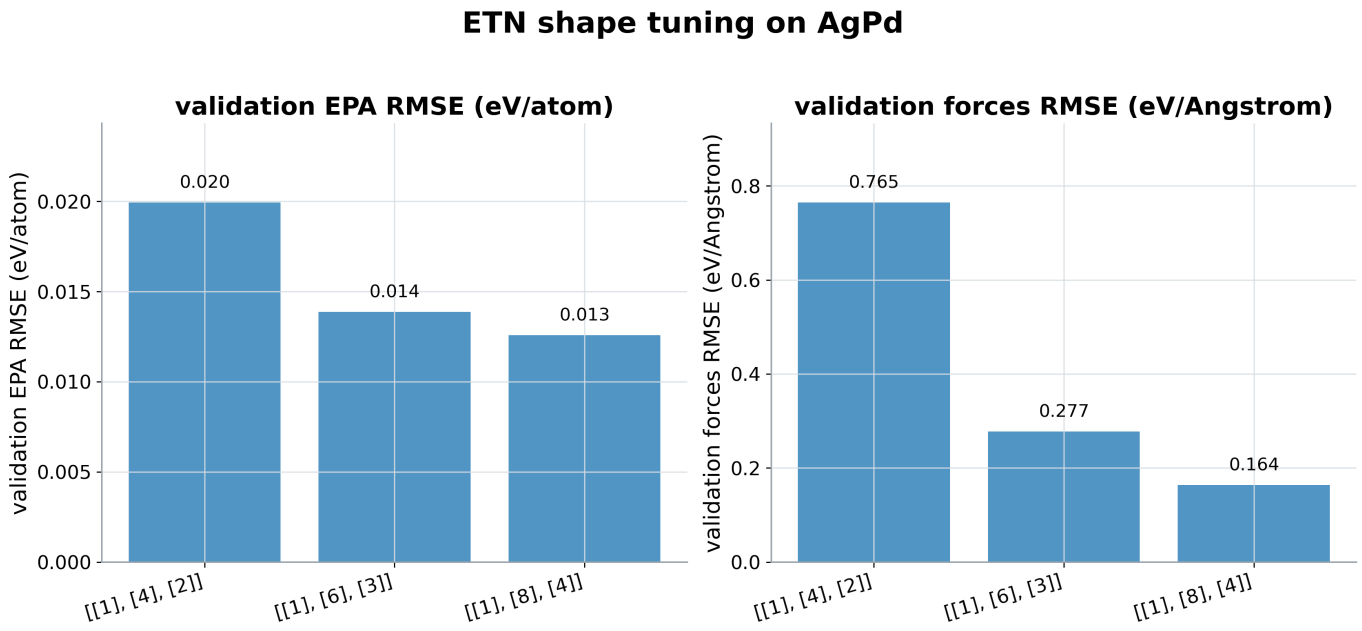


Вывод: Варианты rank почти не изменили качество ETN в этом эксперименте.

Task 1: tuning ETN shape

Численные результаты

shape	val EPA	forces	time
[[1],[4],[2]]	0.01994	0.76469	42.8 s
[[1],[6],[3]]	0.01387	0.27712	58.0 s
[[1],[8],[4]]	0.01258	0.16412	73.0 s



Вывод: Shape [[1], [8], [4]] дал лучший результат среди протестированных, но увеличил время.

Оптимизаторы

Методы первого порядка, квазиньютоновские методы и MUON

Карта оптимизаторов

Эволюция методов

- GD: базовая идея движения против градиента;
- SGD: ответ на дорогой full-batch шаг;
- Momentum: ответ на шум и зигзаги SGD;
- Adam: ответ на разные масштабы координат;
- BFGS: ответ на отсутствие информации о кривизне;
- MUON: попытка нормировать матричное направление шага.

Mini-batch

- SGD
- Momentum
- Adam

Quasi-Newton

- native BFGS
- SciPy BFGS
- Adam+L-BFGS

Matrix / precond.

- MUON pad sqrt
- MUON factorization

Главная ось сравнения

Класс	Информация	Цена шага
SGD/Adam	градиент mini-batch	дешевле, но шумнее
BFGS	градиент + кривизна	дороже, но точнее
MUON	матрич. нормировка	дороже Adam, датасет-зависимый выигрыш

Вывод: Методы отличаются тем, какую информацию о функции потерь используют на одном шаге.

Gradient Descent

$$\theta_{t+1} = \theta_t - \eta \nabla L(\theta_t)$$

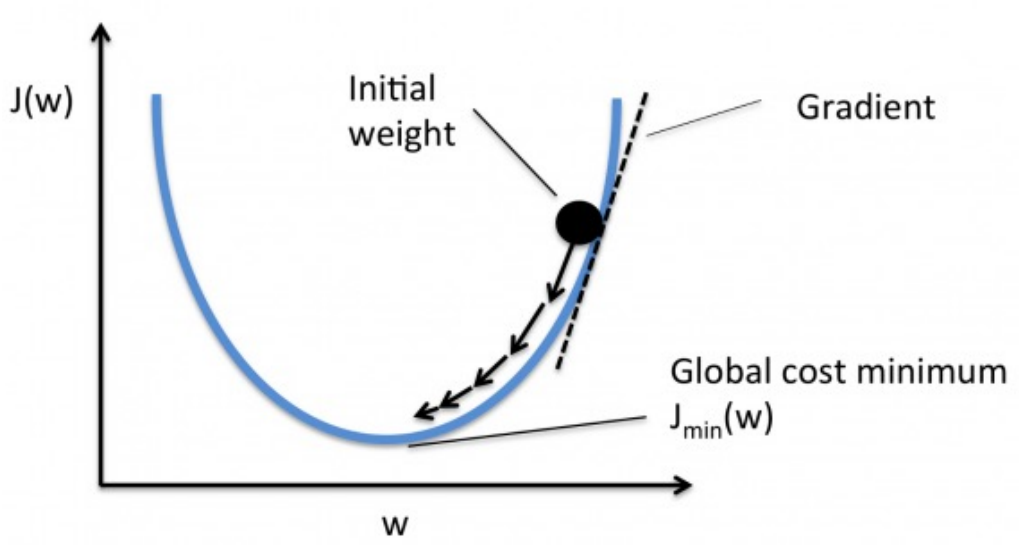
Зачем нужен GD

При обучении ML-потенциала мы подбираем параметры θ, чтобы предсказанные энергии и силы были как можно ближе к DFT-разметке. Для этого минимизируется функция потерь $L(\theta)$.

Градиентный спуск — базовый метод, который каждый раз сдвигает параметры в сторону наибольшего уменьшения ошибки.

Интуиция

- градиент показывает, как нужно изменить параметры, чтобы loss уменьшался
- learning rate η задаёт размер шага
- слишком большой шаг может сделать обучение нестабильным;
- слишком маленький шаг даёт медленное обучение и выход на плато.



Вывод: SGD дешевле на шаг, но требует аккуратной настройки и часто выходит на плато.

Gradient Descent / SGD

$$\theta_{t+1} = \theta_t - \eta g_t$$

$$g_t = \nabla L_B(\theta_t)$$

Мотивация

- полный градиент стабилен, но требует прохода по всему датасету;
- SGD появился как способ делать много дешёвых шагов по mini-batch;
- для ML-потенциалов это важно из-за большого числа конфигураций и силовых компонент;
- цена ускорения — шумный градиент, сильная зависимость от lr и clipping.

Что проверялось в работе

- валидация custom SGD против Torch SGD;
- подбор learning rate и gradient clipping;
- подбор batch size;
- сравнение с Momentum, Adam и BFGS.

метод	val EPA	forces
custom SGD	0.2690	2.806
native BFGS	0.00210	0.0475

Вывод: SGD дешевле на шаг, но требует аккуратной настройки и часто выходит на плато.

Momentum

$$v_{t+1} = \beta v_t + (1-\beta) g_t, \quad \theta_{t+1} = \theta_t - \eta v_{t+1}$$

Почему появился Momentum

- обычный SGD может зигзагить в узких долинах loss;
- одно обновление использует только текущий шумный mini-batch;
- Momentum добавляет «инерцию»: направление шага усредняется по истории;
- это ускоряет движение в устойчивом направлении и сглаживает шум.

Ограничение

- метод всё ещё зависит от learning rate и clipping;
- если масштаб шага выбран плохо, плато остаётся;
- для MTP Momentum улучшил SGD, но не приблизился к BFGS.

MTP	val EPA	forces
Momentum	0.0509	1.149
SGD	0.0937	1.425

Вывод: Momentum сглаживает SGD, но не решает проблему масштаба шага полностью.

Adam

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t, \quad v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2$$

$$\theta_{t+1} = \theta_t - \eta\, \hat{m}_t / (\sqrt{\hat{v}_t} + \varepsilon)$$

Зачем нужен Adam

- Momentum усредняет направление, но использует общий масштаб шага;
- в ML-задачах разные координаты параметров имеют разные масштабы градиента;
- Adam хранит первый момент как направление и второй момент как нормировку;
- bias correction стабилизирует первые итерации.

Почему это важно здесь

- energy и force части дают градиенты разного масштаба;
- Adam оказался лучшим mini-batch baseline для MTP;
- но без информации о кривизне он уступил BFGS.

MTP	val EPA	forces
Adam	0.0293	0.3386
native BFGS	0.00210	0.0475

Вывод: Adam сильнее SGD/Momentum за счёт адаптивного масштаба шага, но остаётся слабее BFGS.

BFGS / quasi-Newton

$$p_k = -H_k \nabla L(\theta_k), \quad H_k \approx (\nabla^2 L(\theta_k))^{-1}$$

Почему нужен quasi-Newton

- first-order методы знают только локальный наклон loss;
- если поверхность плохо обусловлена, направление градиента неэффективно;
- BFGS восстанавливает приближение обратного Гессиана по изменениям градиента;
- получается более информативное направление шага.

Почему BFGS силён в MLIP

- работает на full-batch loss без mini-batch шума;
- хорошо подходит для гладких регрессионных задач;
- в SciPy даёт line search и диагностику nit/success;
- но чувствителен к масштабу начальной точки.

MTP	val EPA	time
native	0.00210	2810 s
SciPy BFGS	0.00233	2895 s

Вывод: BFGS остаётся главным baseline: для MTP native BFGS немного лучше SciPy BFGS по validation RMSE.

Adam → L-BFGS

Зачем комбинировать методы

- Adam хорошо работает на шумном старте и быстро снижает loss;
- L-BFGS лучше уточняет решение, когда модель уже в разумной области;
- комбинация пытается объединить устойчивый старт и кривизну;
- эксперимент проводился для MTP.

Численные результаты

метод	val EPA	forces	time	loss
Adam	0.0293	0.3386	1914 s	1.75
Adam+L-BFGS	0.01875	0.599	≈555 s	20.30
SciPy BFGS	0.00233	0.0554	2895 s	0.141

Вывод: Adam + L-BFGS остаётся полезной схемой дообучения, но BFGS-варианты дают меньшую ошибку.

MUON: матрично-нормализованный шаг

Мотивация MUON

- Adam нормирует координаты независимо, но не учитывает матричную структуру обновления;
- MUON пытается получить более «сбалансированный» шаг через матричную нормализацию;
- градиент вектора параметров временно рассматривается как матрица;
- Newton–Schulz используется для нормализации направления (ортогонализации матрицы-градиента).

Что сравнивалось

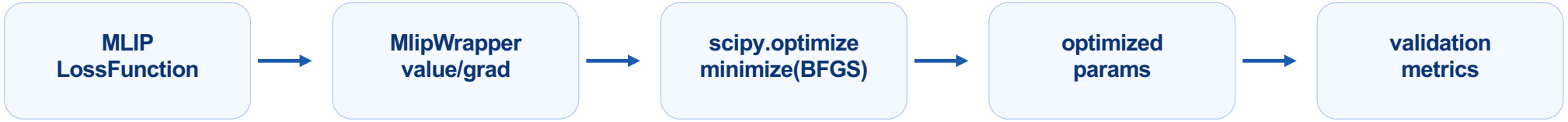
- pad sqrt: матрица близка к квадратной, недостающие элементы дополняются нулями;
- factorization: размеры подбираются по факторизации без padding;
- оба варианта проверялись на MTP, ETN AgPd и ETN MoNbTaVW;
- в итогах Muon показал сильный потенциал для ETN.

Вывод: Muon проверялся как матрично-нормализованный шаг; для ETN он стал конкурентным mini-batch методом.

Task 3

SciPy BFGS и проброс loss/gradient из MLIP

SciPy BFGS: зачем нужен wrapper



Что даёт SciPy

- line search и критерии остановки;
- nit, success и final_loss для диагностики;
- контроль начальной точки;
- готовая реализация BFGS без переписывания оптимизатора.

Что делает MlipWrapper

- записывает x в параметры потенциала;
- вызывает LossFunction.calc;
- получает gradient через accumulate_grads;
- возвращает fun(x) и jac(x) для scipy.optimize.minimize.

Вывод: Wrapper позволил подключить внешний SciPy BFGS без переписывания функции потерь MLIP.

Task 3: начальный интервал SciPy BFGS для MTP

init interval	val EPA mean	val EPA median	val forces mean	train time	comment
[-0.0001,0.0001]	0.00252	0.002542	0.0607	331.1	best scale
[-0.001,0.001]	0.002552	0.002493	0.0603	323.4	best scale
[-0.01,0.01]	0.002844	0.002931	0.0691	335.9	works worse
[-0.1,0.1]	0.002813	0.002782	0.0695	335.8	works worse
[-1,1]	0.9483	0.0254	29.47	394.6	outlier risk

Как читать таблицу

- лучшие интервалы — малые 1e-4 и 1e-3
- крупный интервал [-1, 1] дал катастрофический запуск

Вывод: Для SciPy BFGS качество MTP в первую очередь определяется масштабом начальной точки.

Task 3: MTP native BFGS vs SciPy BFGS

method	val EPA mean	val EPA median	val forces median	train time	comment
native BFGS	0.002104	0.002062	0.0477	2810.3	baseline
Scipy 1e-3	0.002333	0.002327	0.0557	2894.7	close baseline

Как читать таблицу

- native BFGS немного лучше SciPy по EPA/forces
- SciPy BFGS

Вывод: В итоговой серии native BFGS немного лучше SciPy BFGS; SciPy остаётся близким baseline.

Task 3: ETN native BFGS vs SciPy BFGS

method	val EPA mean	val forces mean	train time	nit mean	success
native BFGS	0.0139	0.2771	63.5	—	—
SciPy BFGS	0.0140	0.2773	83.7	100.4	0.20

Как читать таблицу

- native и SciPy почти совпали по EPA/forces
- SciPy в среднем дольше
- выигрыш SciPy для MTP не переносится на ETN

Вывод: Для ETN SciPy BFGS не дал явного выигрыша и работал дольше native BFGS.

Архитектура реализации

Кодовая инфраструктура экспериментов

Архитектура кода

Основные модули

- LossBatcher.py — mini-batch обучение;
- MlipWrapper.py — интерфейс для SciPy;
- mlip_objective.py — loss + gradient;
- optimizers.py — SGD/Momentum/Adam/MUON;
- mlip_trainer.py — общий training loop;
- lr_schedules.py — schedules для learning rate.

Скрипты задач

- task3 — SciPy BFGS;
- task4 — custom optimizers;
- task6 — перенос на MoNbTaVW;
- all_text_results.md — агрегация результатов;
- all_plots.ipynb — графики для презентации.

Вывод: Инфраструктура разделяет модель, функцию потерь, оптимизатор и сохранение метрик.

Task 4

MTP: сначала отдельные методы, затем общее сравнение

Task 4: логика блока MTP

Почему такой порядок

- сначала валидируется собственный mini-batch loop;
- затем отдельно разбираются базовые методы;
- после этого показывается Adam + L-BFGS;
- в конце — единое сравнение всех методов.

Что важно смотреть

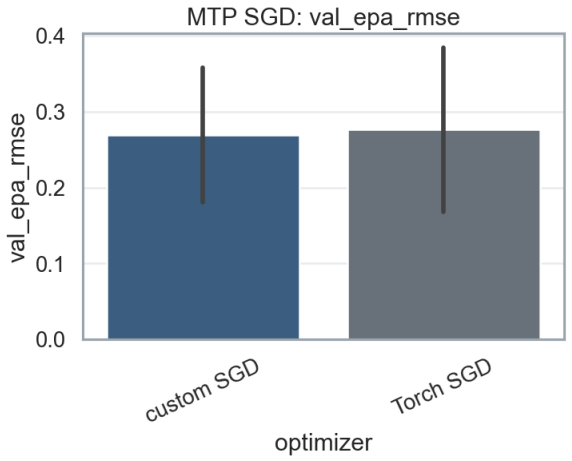
- validation EPA RMSE;
- force RMSE;
- train time;
- loss dynamics;
- разброс по ensemble.

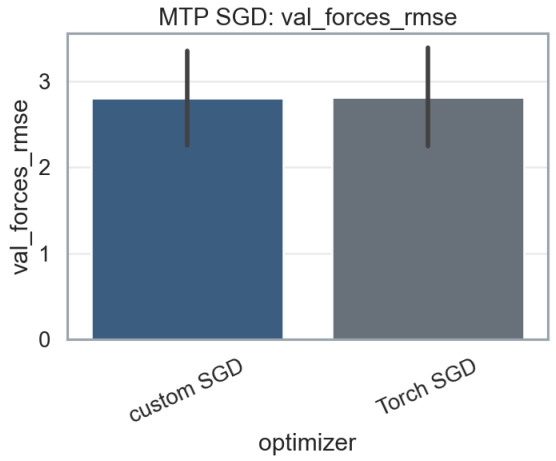
Вывод: Блок Task 4 построен от частных экспериментов к общей сводке.

Task 4: custom SGD vs Torch SGD

Численные результаты

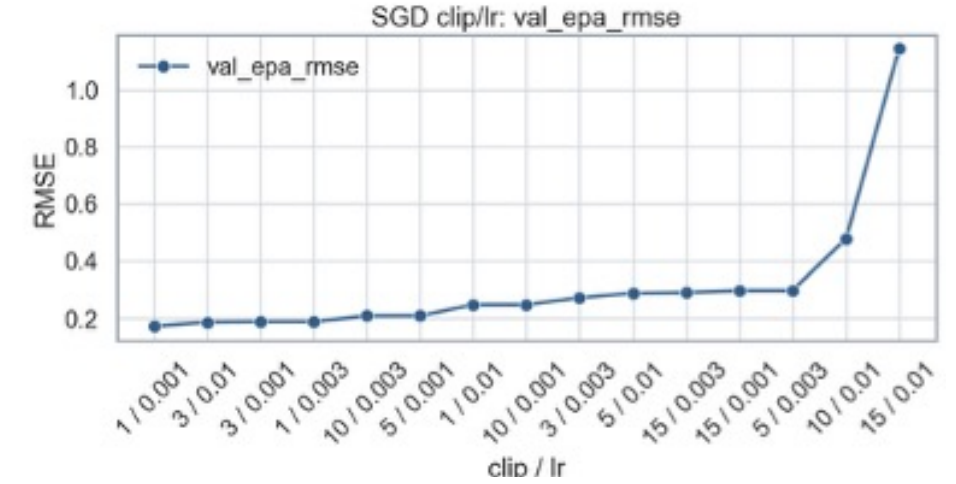
метод	val EPA	forces	time	loss
custom SGD	0.2690±0.0887	2.806±0.551	126.4 s	136.9
Torch SGD	0.2762±0.1079	2.815±0.575	153.2 s	136.9

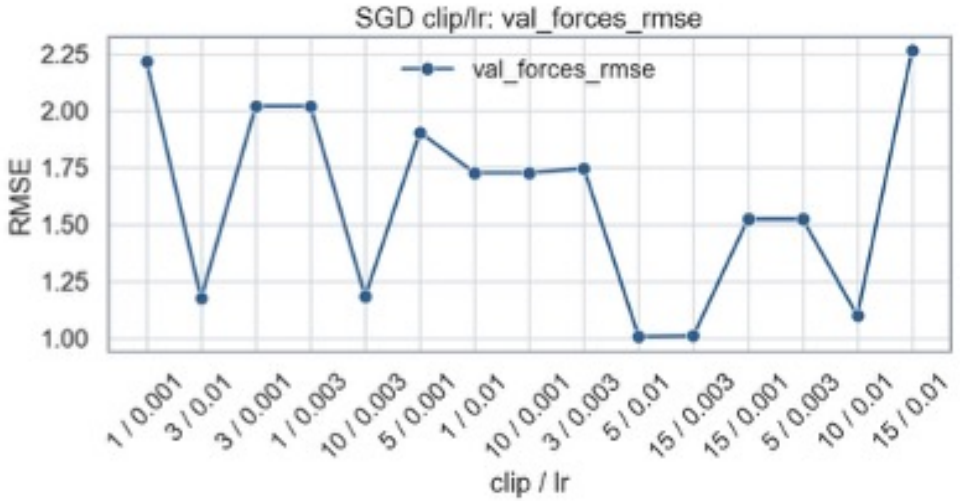




Вывод: Custom SGD сопоставим с Torch SGD, поэтому training loop можно использовать дальше.

Task 4: SGD — подбор clip и lr





Как читать результат

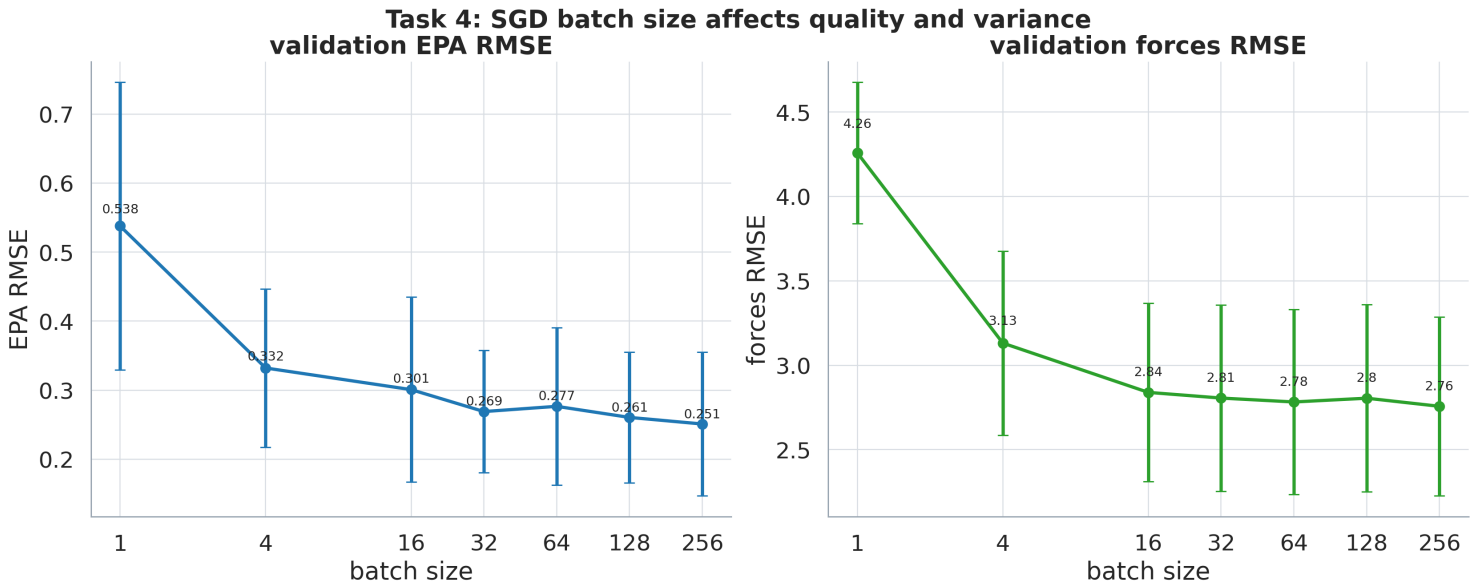
- lr задаёт масштаб шага
- clipping стабилизирует первые итерации
- слишком малый шаг ведёт к плато
- слишком большой — к нестабильности

Численные результаты

clip/lr	val EPA	forces	time
clip=1, lr=.001	0.1729	2.2197	279 s
clip=3, lr=.01	0.1871	1.1755	284 s
clip=3, lr=.001	0.1884	2.0211	285 s
clip=1, lr=.003	0.1884	2.0211	281 s
clip=10, lr=.003	0.2094	1.1854	277 s

Вывод: SGD сильно зависит от learning rate и clipping.

Task 4: SGD — влияние batch size



Как читать результат

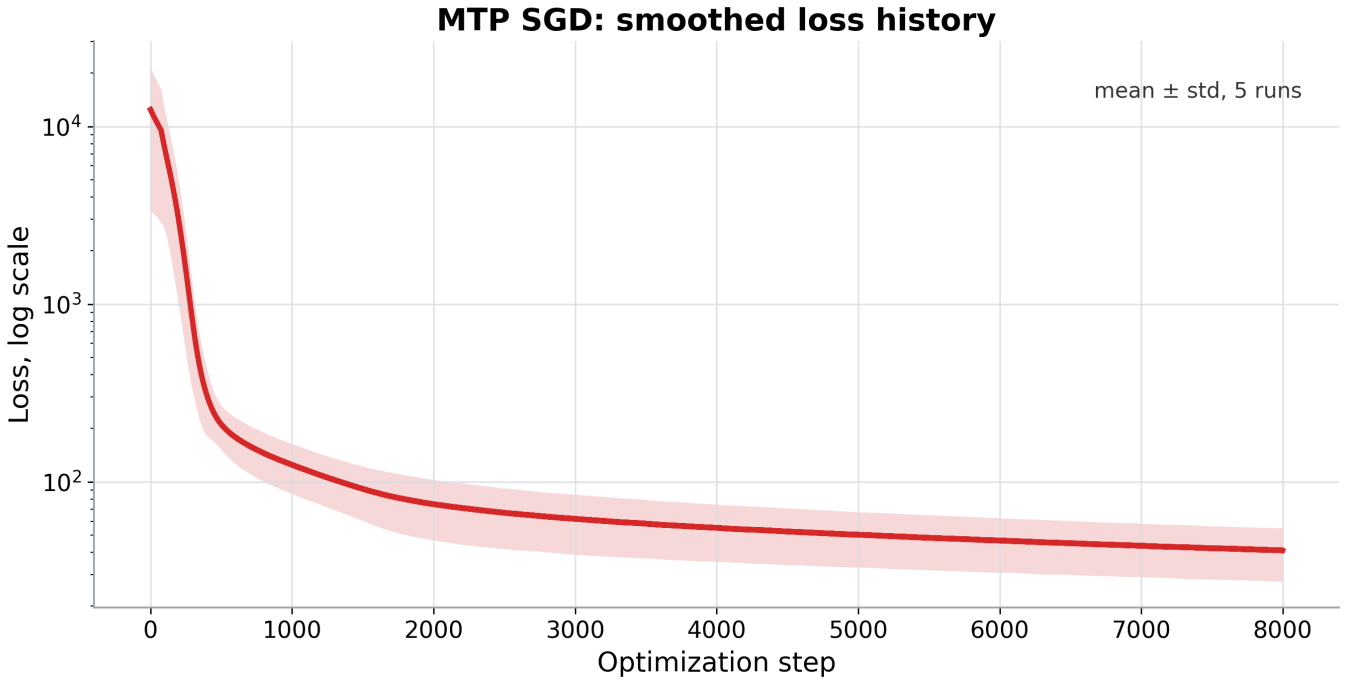
- batch=1 слишком шумный
- рост batch улучшает EPA до плато
- forces почти не улучшаются после batch≈32
- цена большого batch — резкий рост времени

Численные результаты

batch	val EPA	forces	time
1	0.5379	4.257	4.7 s
4	0.3321	3.131	16.5 s
16	0.3008	2.839	60.0 s
32	0.2690	2.806	120.9 s
64	0.2766	2.783	201.4 s
128	0.2605	2.805	301.4 s
256	0.2510	2.757	674.2 s

Вывод: Batch size снижает шум EPA, но быстро упирается в плато качества и рост стоимости.

MTP SGD: динамика обучения



Сравнение с BFGS baseline

metric	SGD	SciPy BFGS	ratio
train EPA RMSE	0.0806	0.001112	72.5×
val EPA RMSE	0.0937	0.002333	40.2×
val forces RMSE	1.425	0.0554	25.7×
train time, s	1958.0	2894.7	0.7×
final loss	32.69	0.1411	231.6×

Вывод: SGD работает как базовый mini-batch метод, но заметно уступает BFGS по validation error.

Task 4: Momentum для MTP

Что добавляет Momentum

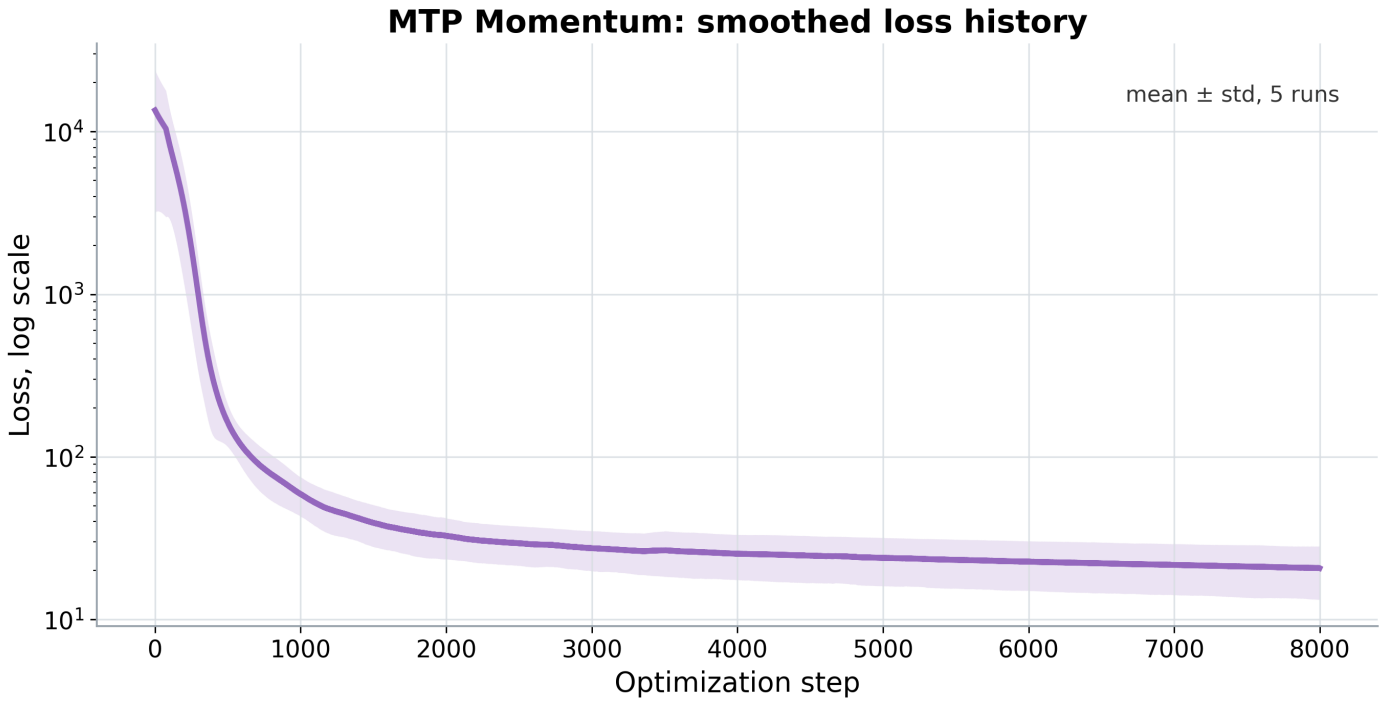
- используется накопленная скорость v_t
- направление шага становится менее шумным
- по loss динамика спокойнее, чем у SGD
- но плато полностью не исчезает

Результат

- validation EPA RMSE ≈ 0.0509
- validation forces RMSE ≈ 1.149
- train time ≈ 1924 s
- лучше базового SGD по EPA
- хуже Adam и BFGS

Вывод: Momentum улучшает SGD, но в этой задаче остаётся заметно слабее Adam и BFGS.

MTP Momentum: динамика обучения

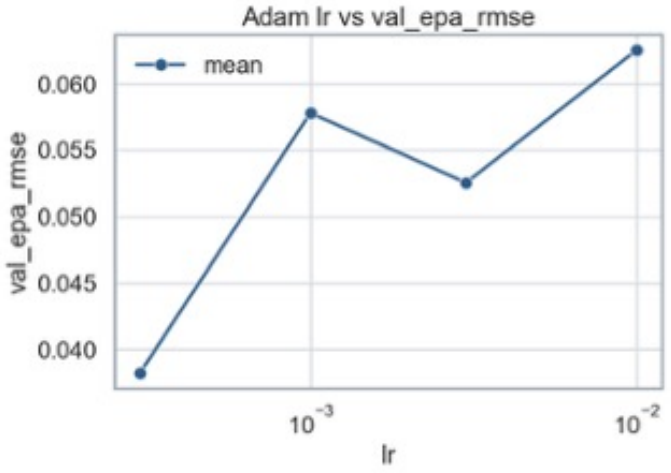


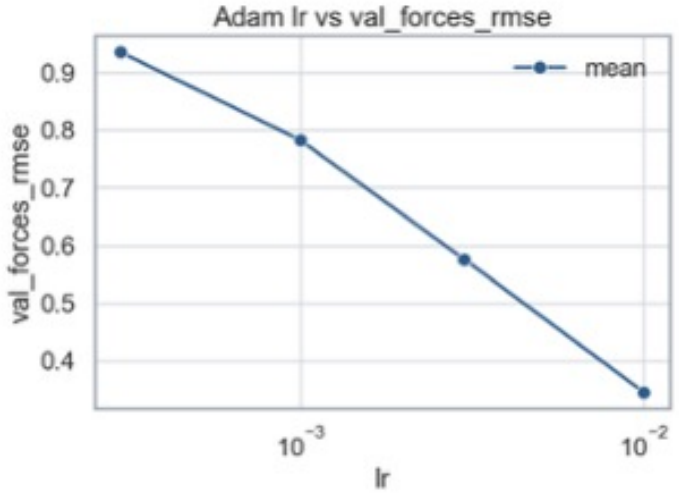
Сравнение с BFGS baseline

metric	Momentum	SciPy BFGS	ratio
train EPA RMSE	0.0429	0.001112	38.6×
val EPA RMSE	0.0509	0.002333	21.8×
val forces RMSE	1.149	0.0554	20.7×
train time, s	1924.5	2894.7	0.7×
final loss	16.09	0.1411	114.0×

Вывод: Momentum сглаживает SGD, но остаётся существенно слабее квазиньютоновского baseline.

Task 4: Adam — подбор learning rate





Как читать результат

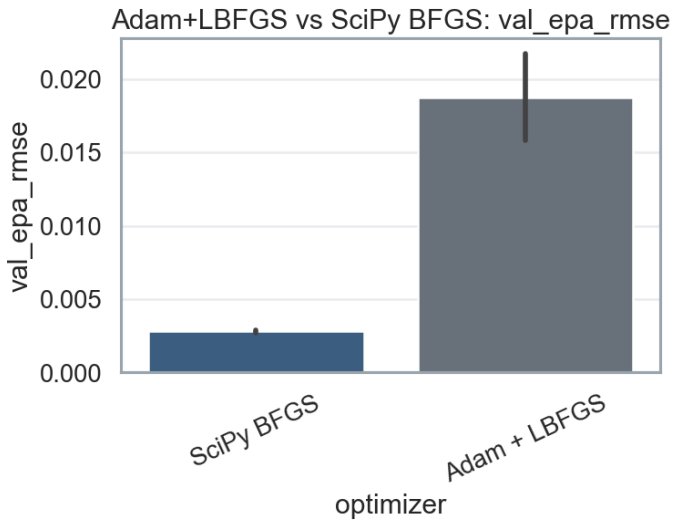
- итоговая конфигурация: lr=0.001
- Adam лучше SGD/Momentum для MTP
- но всё ещё уступает BFGS
- Adam — лучший MTP mini-batch baseline

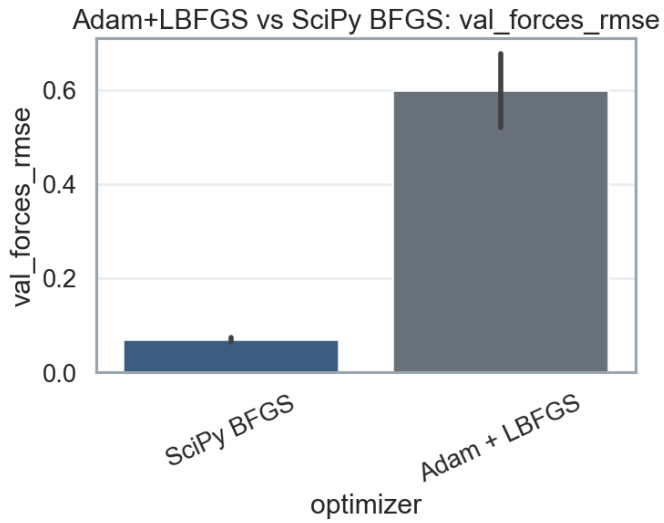
Численные результаты

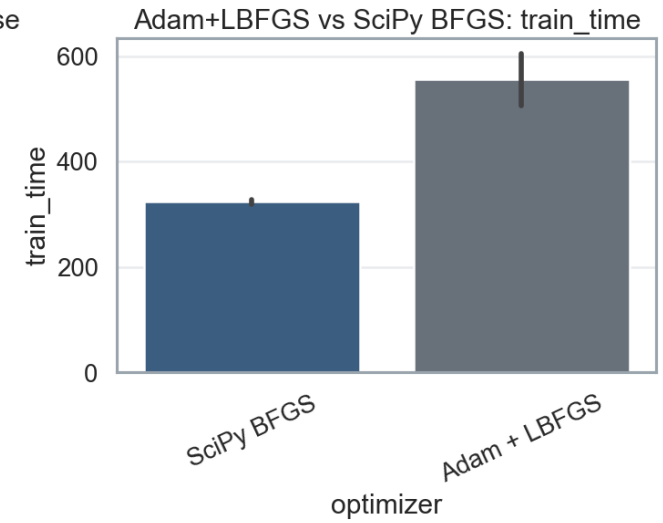
lr	val EPA	forces	time	loss
0.0003	0.0382	0.935	491 s	8.14
0.001	0.0293	0.3386	1914 s	1.75
0.003	0.0525	0.576	492 s	7.02
0.01	0.0625	0.345	493 s	1.75

Вывод: В итоговой серии Adam даёт лучшую EPA среди MTP mini-batch методов.

Task 4: Adam + L-BFGS против SciPy BFGS







Как читать результат

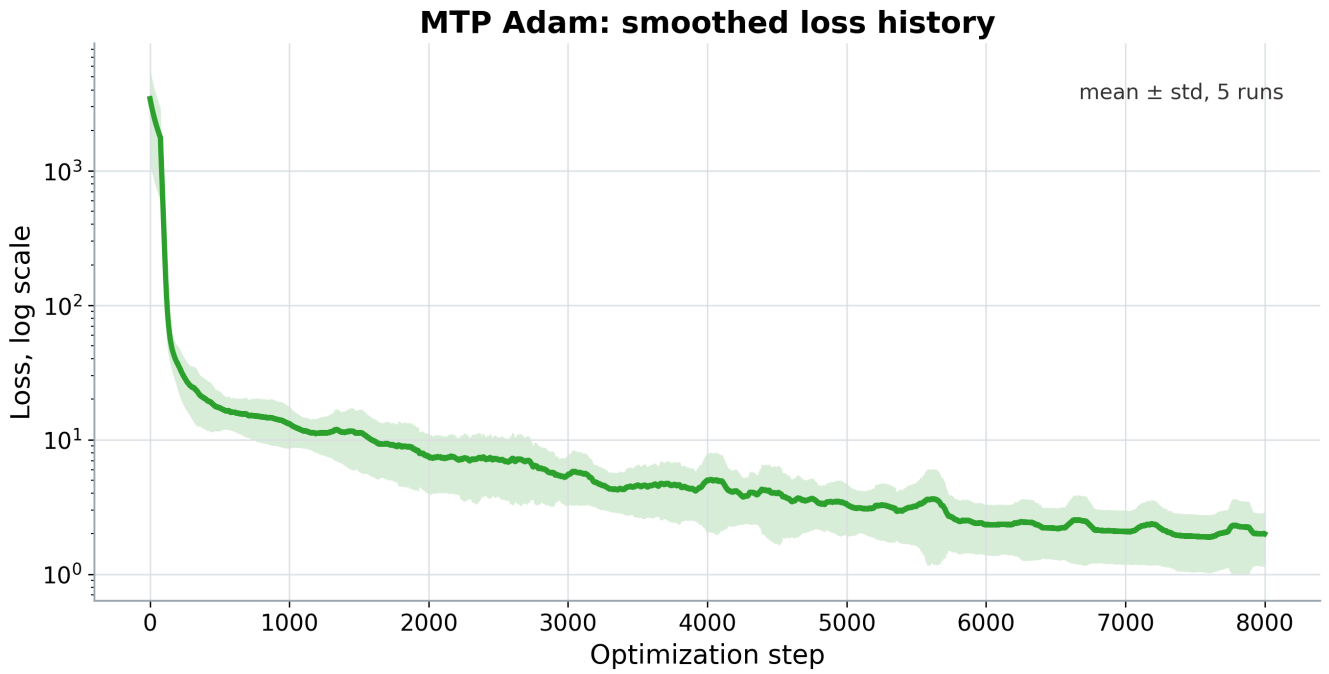
- Adam + L-BFGS: EPA ≈ 0.01875
- SciPy BFGS: EPA ≈ 0.00233
- success=False из-за лимита nit
- главный выигрыш даёт квазиньютоновский этап

Численные результаты

метод	val EPA	forces	time	loss
Adam	0.0293	0.3386	1914 s	1.75
Adam+L-BFGS	0.01875	0.599	≈555 s	20.30
SciPy BFGS	0.00233	0.0554	2895 s	0.141

Вывод: Adam + L-BFGS заметно улучшает Adam, но уступает чистому SciPy BFGS.

MTP Adam: динамика обучения

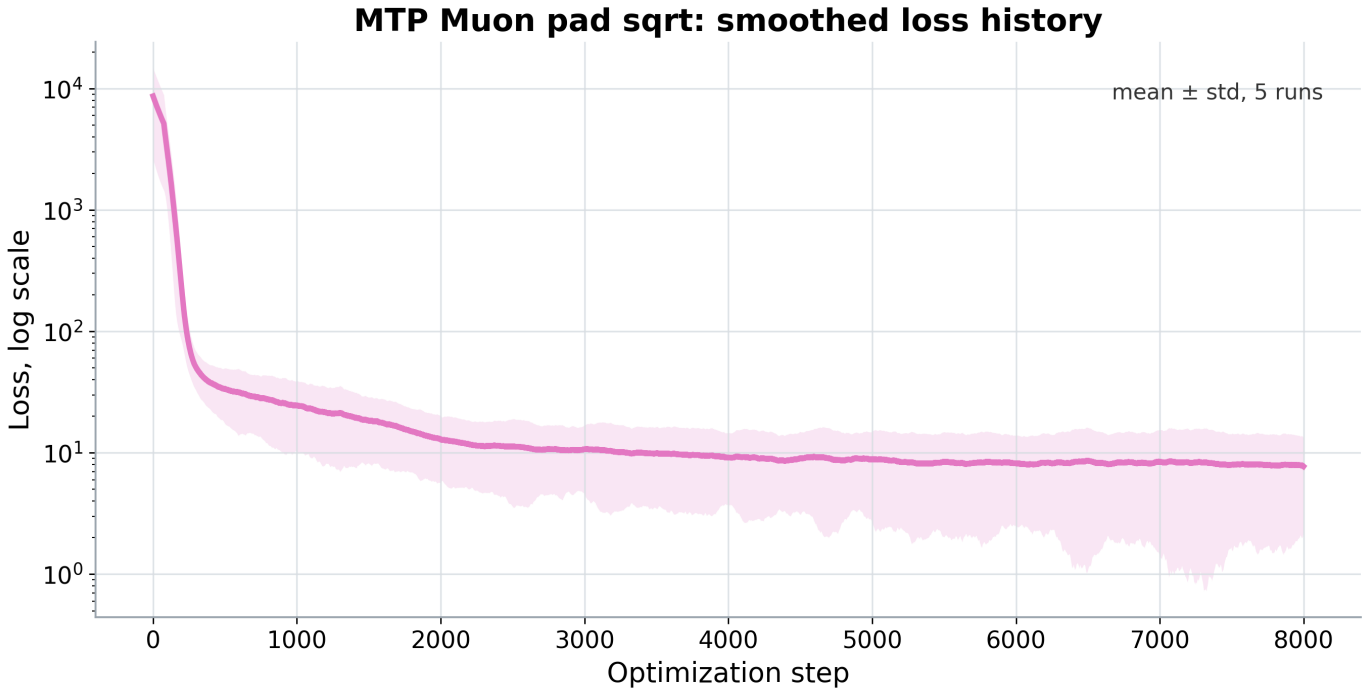


Сравнение с BFGS baseline

metric	Adam	SciPy BFGS	ratio
train EPA RMSE	0.0276	0.001112	24.8×
val EPA RMSE	0.0293	0.002333	12.6×
val forces RMSE	0.3386	0.0554	6.1×
train time, s	1914.0	2894.7	0.7×
final loss	1.748	0.1411	12.4×

Вывод: Adam является лучшим mini-batch baseline для MTP, но по качеству уступает BFGS.

MTP MUON pad sqrt: динамика обучения

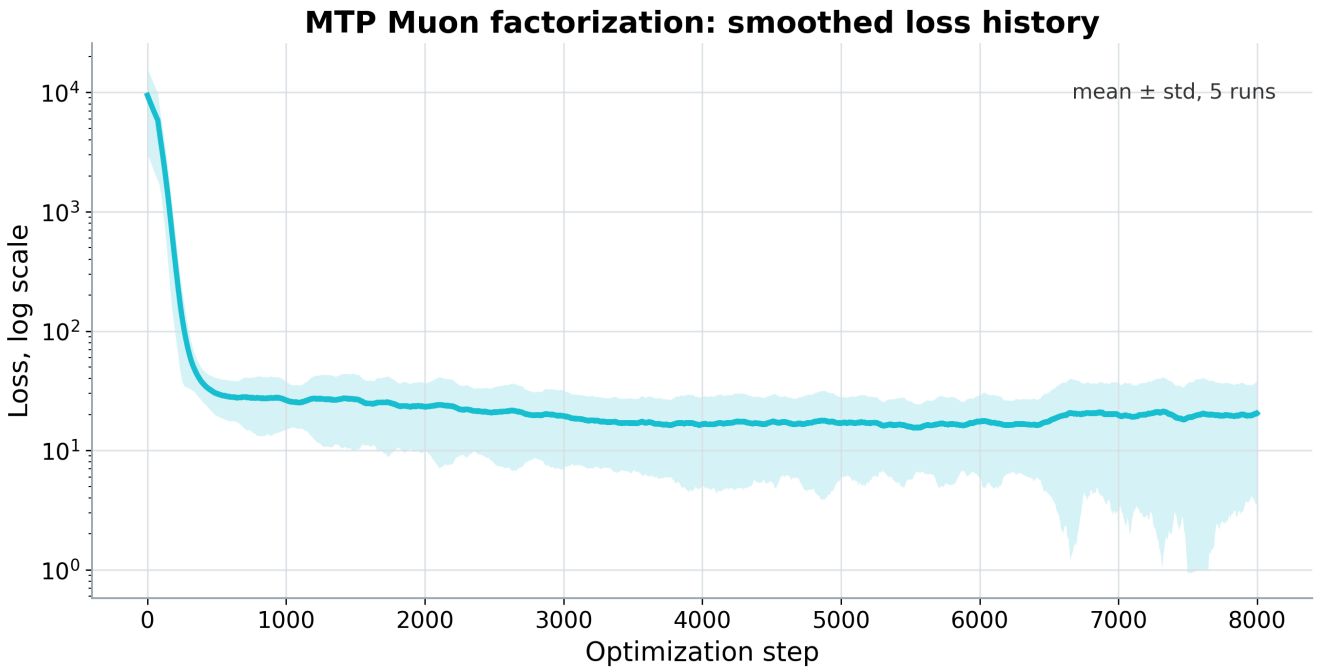


Сравнение с BFGS baseline

metric	MUON pad sqrt	SciPy BFGS	ratio
train EPA RMSE	0.0802	0.001112	72.2×
val EPA RMSE	0.0802	0.002333	34.4×
val forces RMSE	0.5038	0.0554	9.1×
train time, s	2052.8	2894.7	0.7×
final loss	3.514	0.1411	24.9×

Вывод: Для MTP pad sqrt не улучшил качество относительно Adam/BFGS.

MTP MUON factorization: динамика обучения

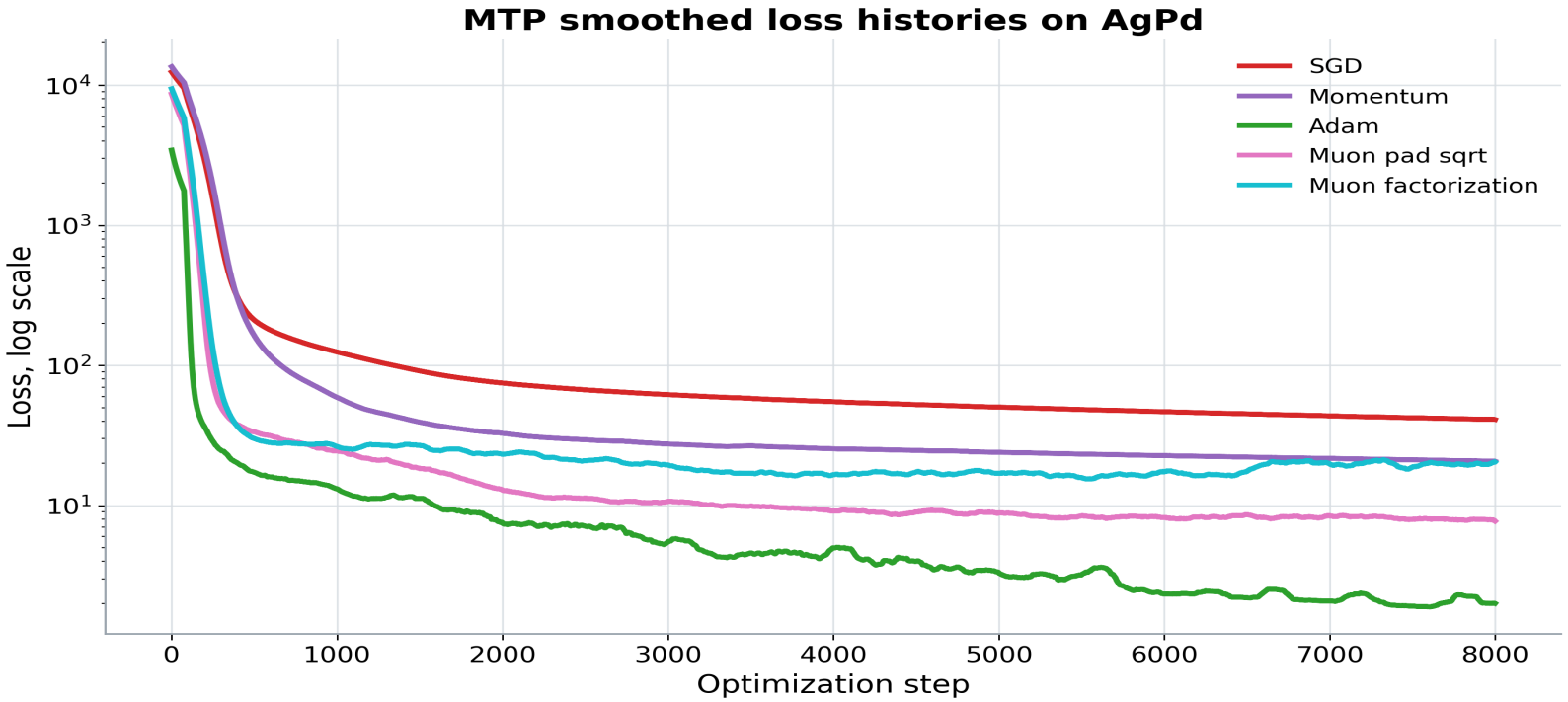


Сравнение с BFGS baseline

metric	MUON factorization	SciPy BFGS	ratio
train EPA RMSE	0.1179	0.001112	106.0×
val EPA RMSE	0.1134	0.002333	48.6×
val forces RMSE	0.5987	0.0554	10.8×
train time, s	1913.0	2894.7	0.7×
final loss	15.41	0.1411	109.2×

Вывод: Factorization оказался хуже pad sqrt на MTP и не приблизился к BFGS baseline.

Task 4: MTP loss histories

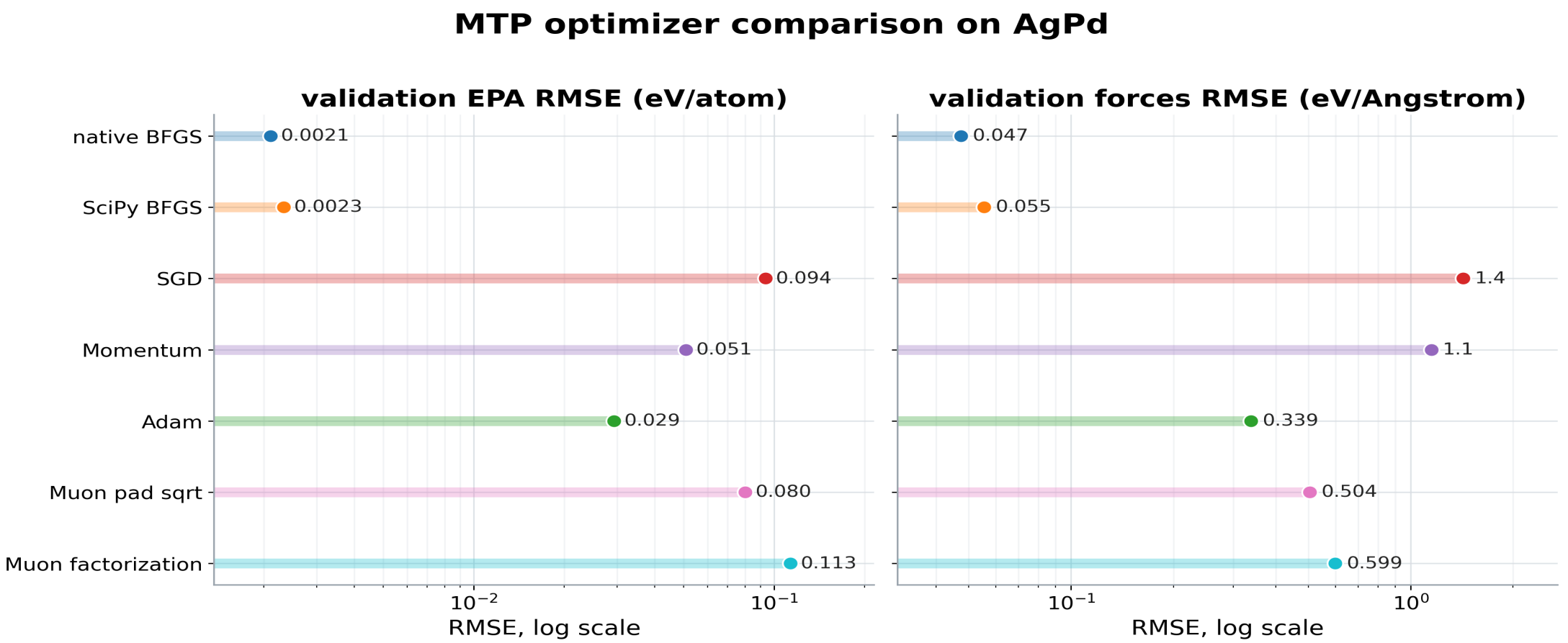


Как читать результат

- истории нужны для интерпретации качества
- SGD более шумный
- Momentum сглаживает, но не решает плато
- Adam даёт наиболее стабильный mini-batch спуск

Вывод: Adam снижает loss стабильнее, но BFGS остаётся точнее по validation RMSE.

Task 4: MTP optimizer comparison



Вывод: Лучший результат для MTP в итоговой серии даёт native BFGS; SciPy BFGS остаётся близким baseline.

Task 4: MTP quality vs train time

Как читать результат

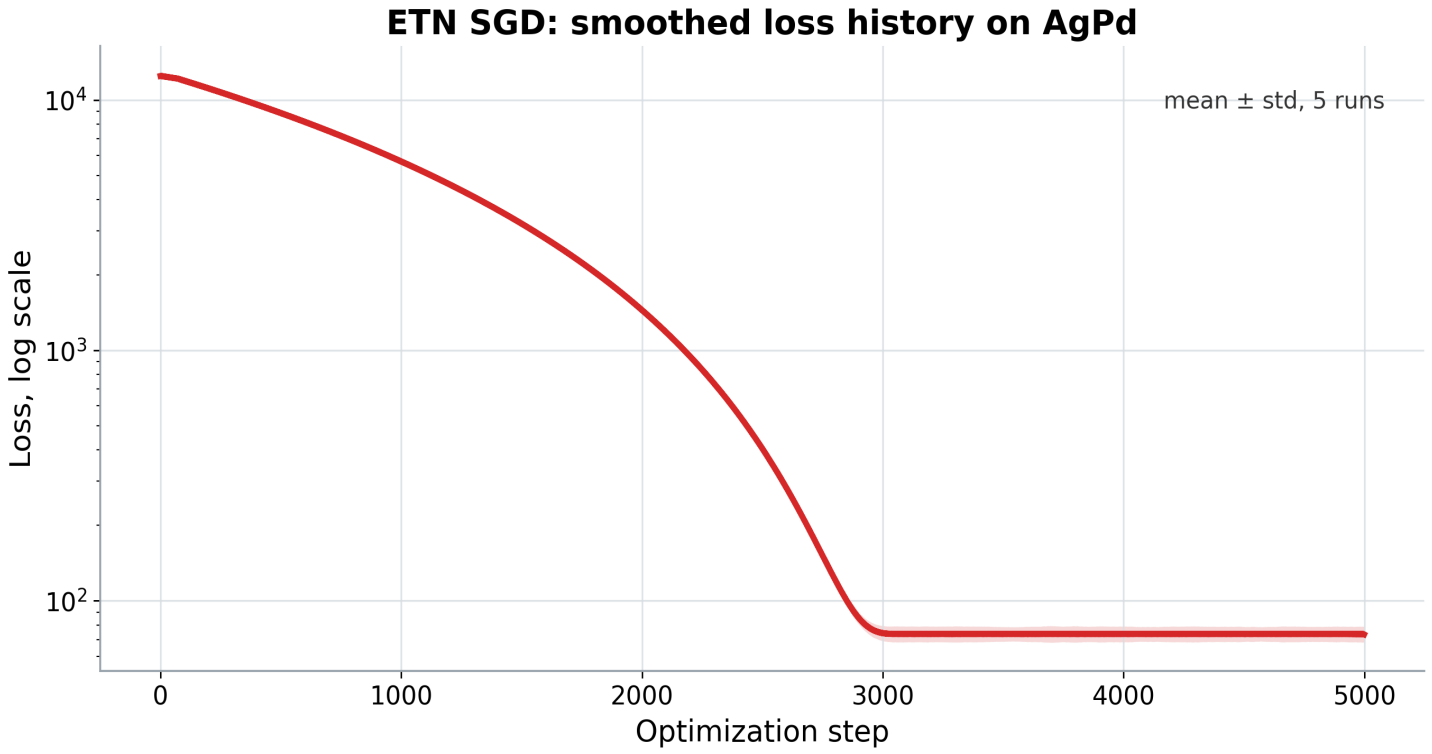
- BFGS-варианты формируют лучший quality cluster
- SGD быстрее, но error на порядок выше
- Adam — лучший mini-batch компромисс
- Muon не дал преимущества над Adam по quality/time

Вывод: По validation error для MTP доминирует BFGS; среди mini-batch методов лучшим остаётся Adam.

Task 4: ETN

Оптимизаторы на AgPd для ETN

ETN AgPd SGD: динамика обучения

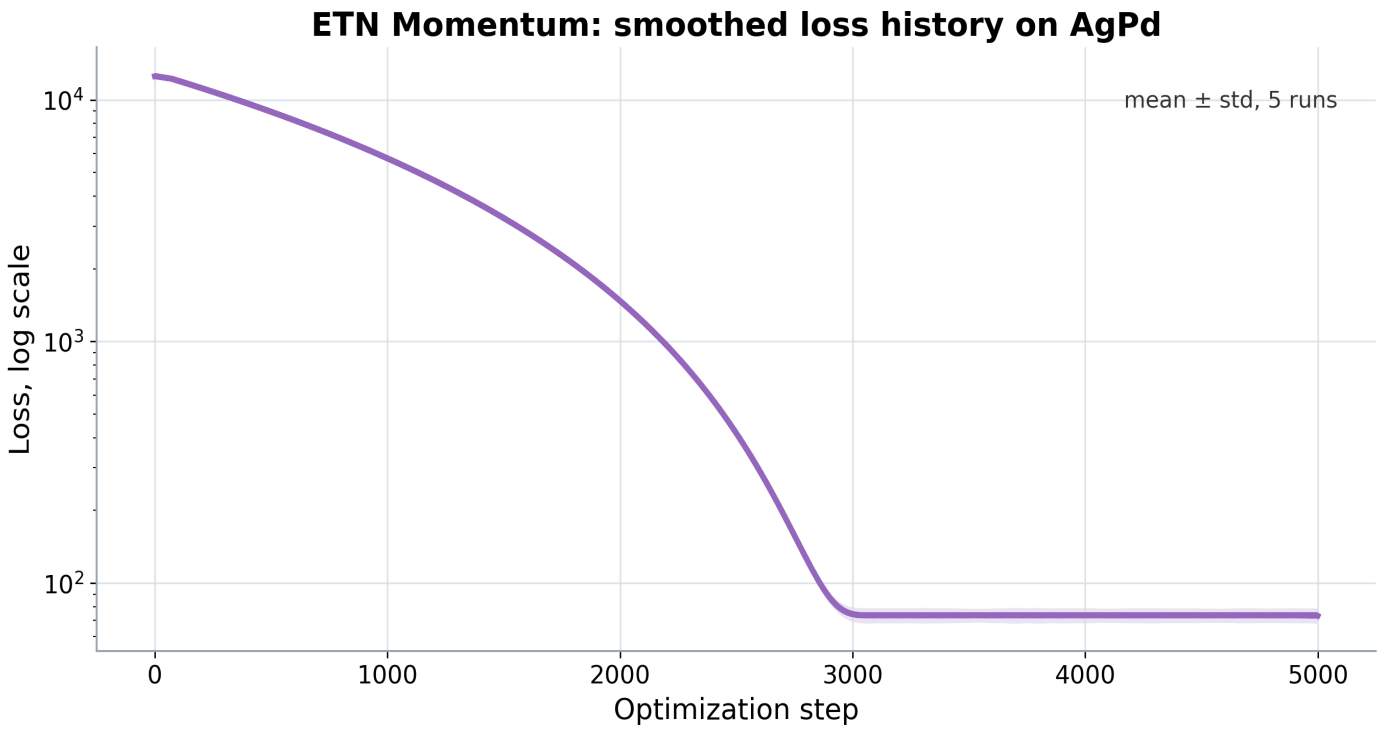


Сравнение с BFGS baseline

metric	SGD	native BFGS	ratio
train EPA RMSE	0.1283	0.008614	14.9×
val EPA RMSE	0.1727	0.0139	12.4×
val forces RMSE	1.730	0.2771	6.2×
train time, s	1061.0	60.5	17.5×

Вывод: На ETN SGD даёт слишком большую energy error и не является сильным кандидатом.

ETN AgPd Momentum: динамика обучения

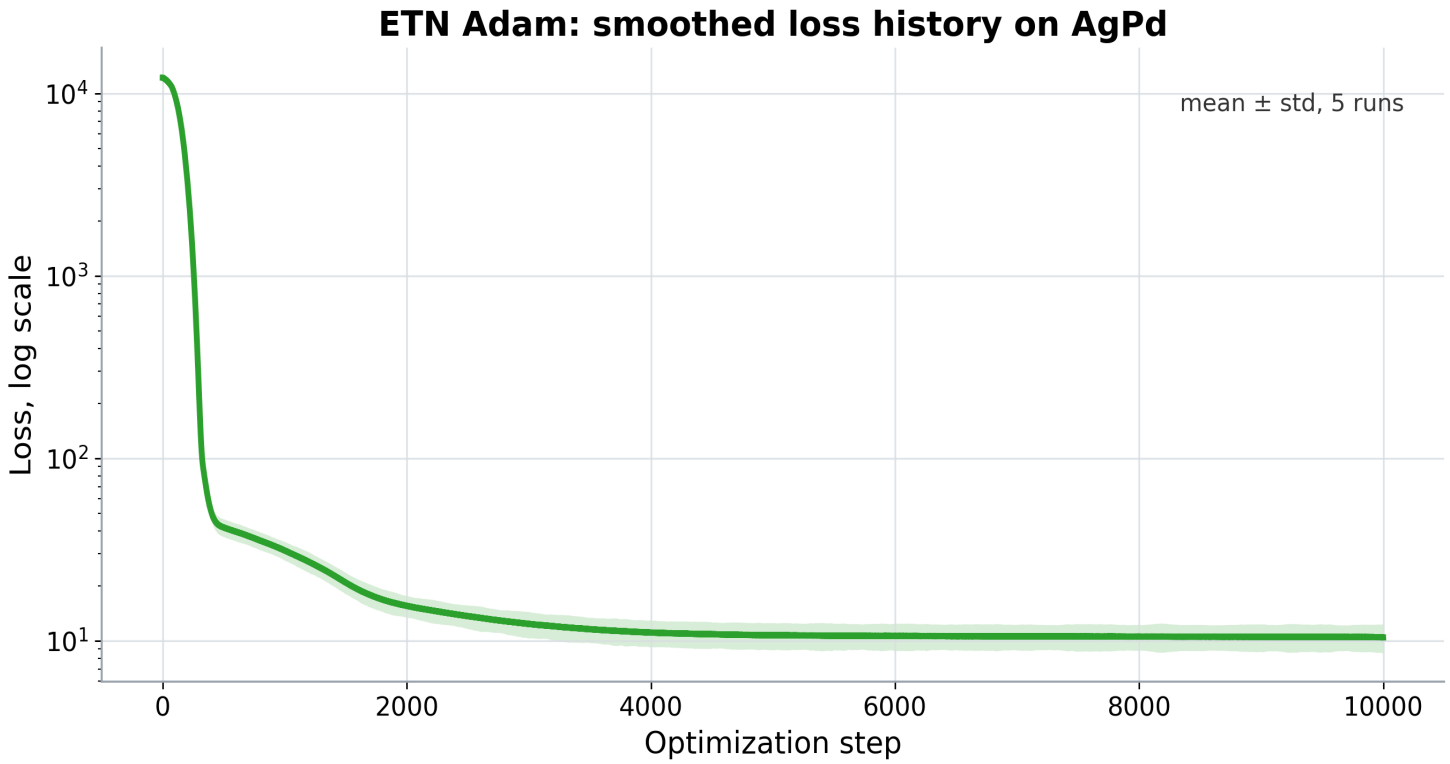


Сравнение с BFGS baseline

metric	Momentum	native BFGS	ratio
train EPA RMSE	0.1283	0.008614	14.9×
val EPA RMSE	0.1727	0.0139	12.4×
val forces RMSE	1.730	0.2771	6.2×
train time, s	1064.0	60.5	17.6×

Вывод: Momentum улучшает SGD по energy error, но остаётся далеко от native BFGS.

ETN AgPd Adam: динамика обучения

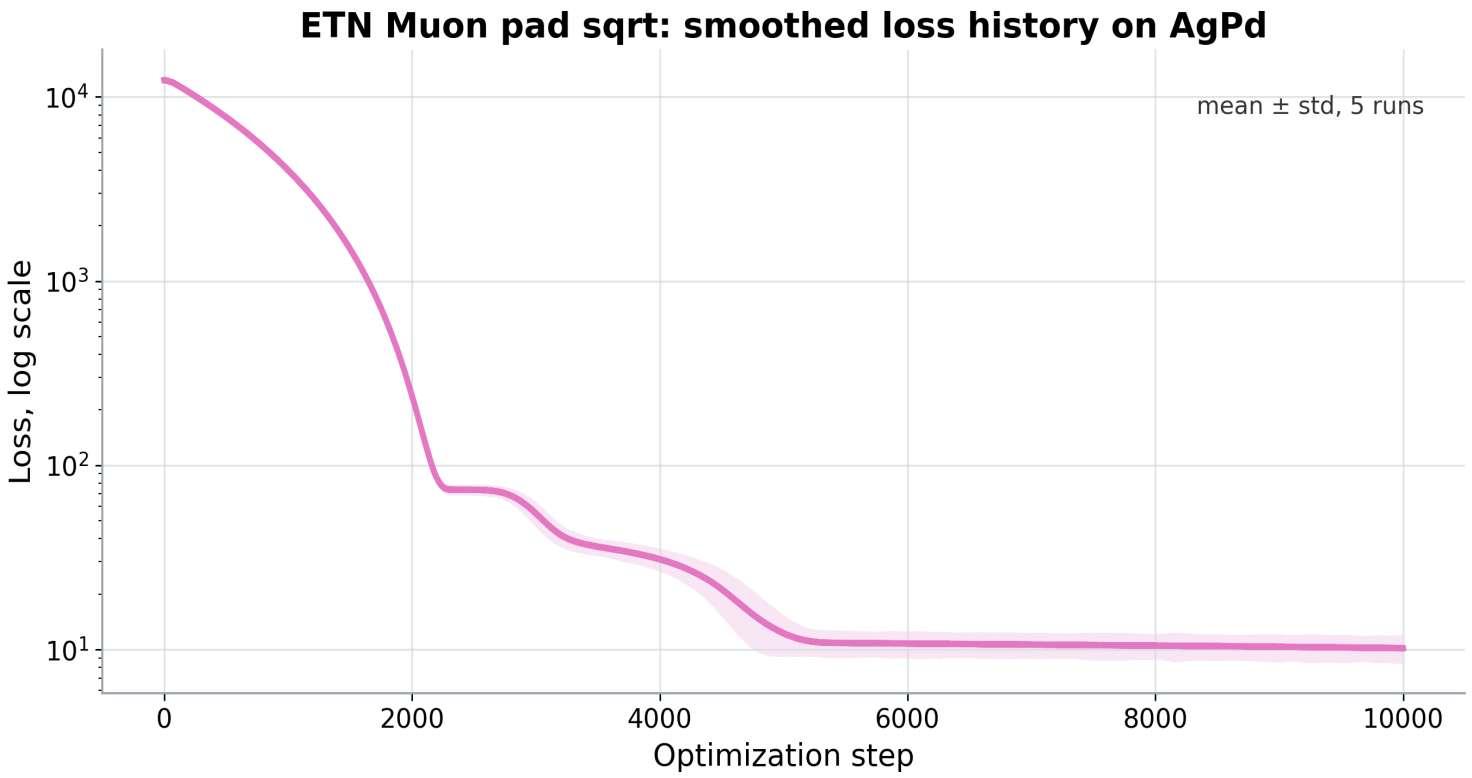


Сравнение с BFGS baseline

metric	Adam	native BFGS	ratio
train EPA RMSE	0.0195	0.008614	2.3×
val EPA RMSE	0.0227	0.0139	1.6×
val forces RMSE	0.8707	0.2771	3.1×
train time, s	2034.9	60.5	33.6×

Вывод: Adam — лучший mini-batch метод для ETN на AgPd, но native BFGS всё равно точнее.

ETN AgPd MUON pad sqrt: динамика обучения

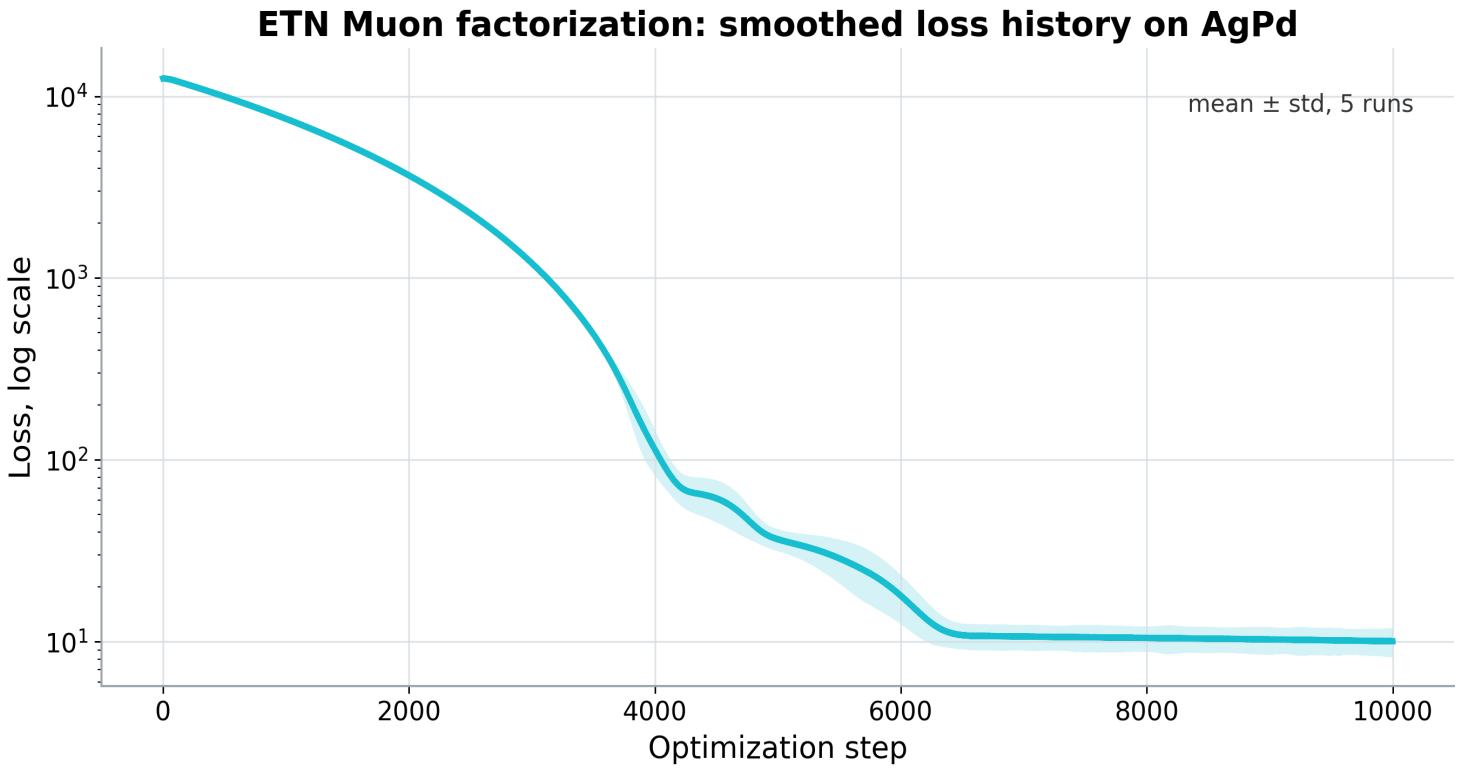


Сравнение с BFGS baseline

metric	MUON pad sqrt	native BFGS	ratio
train EPA RMSE	0.0202	0.008614	2.3×
val EPA RMSE	0.0234	0.0139	1.7×
val forces RMSE	0.8564	0.2771	3.1×
train time, s	2168.0	60.5	35.8×

Вывод: Pad sqrt почти сравнялся с Adam по EPA и немного лучше него по force error.

ETN AgPd MUON factorization: динамика обучения

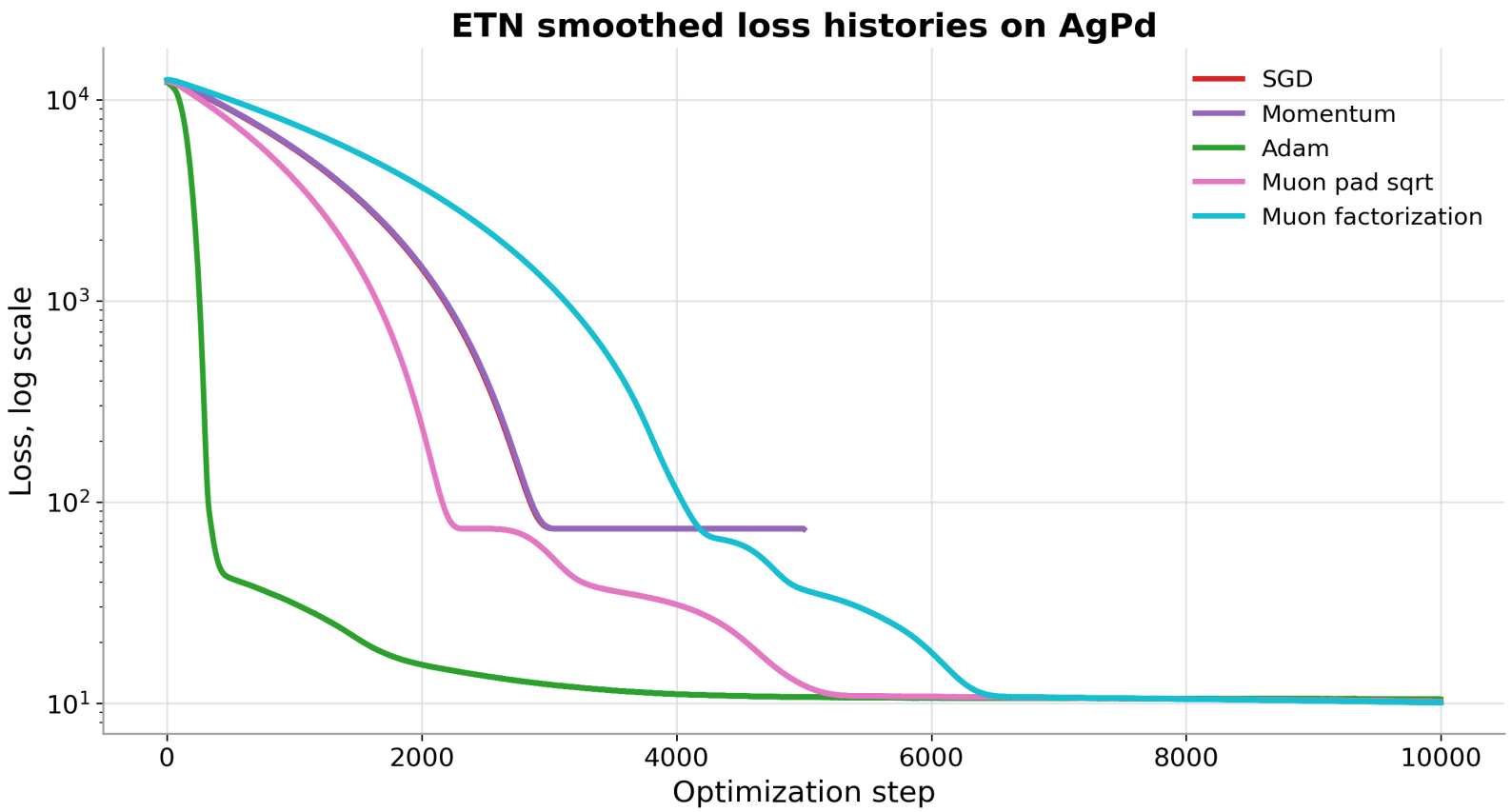


Сравнение с BFGS baseline

metric	MUON factorization	native BFGS	ratio
train EPA RMSE	0.0202	0.008614	2.3×
val EPA RMSE	0.0235	0.0139	1.7×
val forces RMSE	0.8504	0.2771	3.1×
train time, s	2037.9	60.5	33.7×

Вывод: Factorization близок к pad sqrt и даёт минимальную force error среди mini-batch методов.

Task 4: ETN loss histories на AgPd

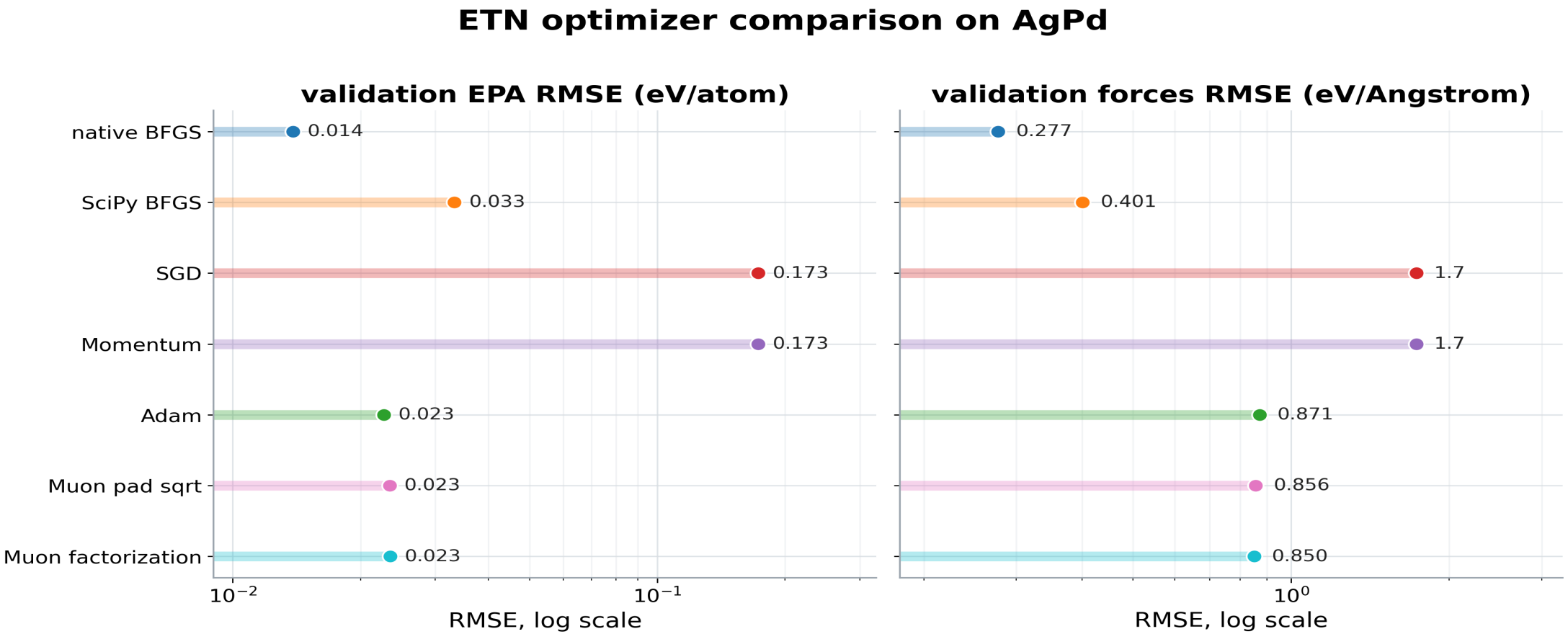


Как читать результат

- Adam лидирует по energy error
- SGD/Momentum дают плохую energy error
- Muon-варианты лучше Adam по forces
- но native BFGS остаётся сильнее

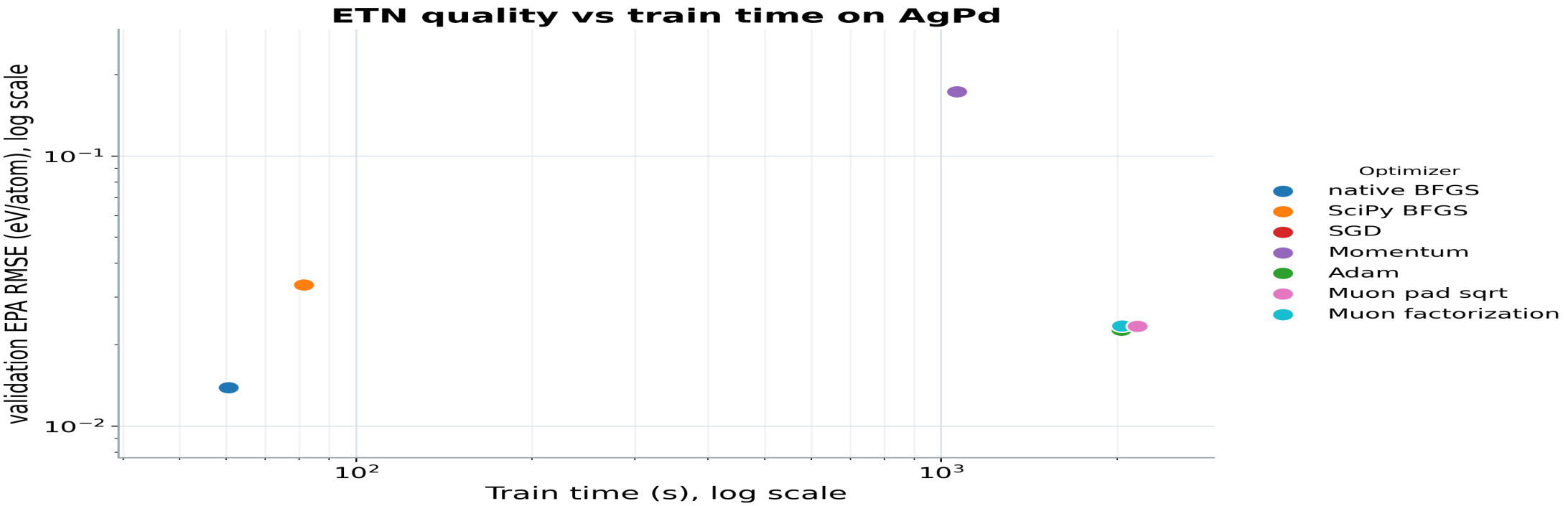
Вывод: Динамика loss показывает сближение Adam и Muon-вариантов среди mini-batch методов.

Task 4: ETN optimizer comparison на AgPd



Вывод: Для ETN native BFGS остаётся сильным baseline; Adam лидирует по EPA, Muon — по forces среди mini-batch методов.

Task 4: MTP и ETN при одинаковых оптимизаторах



Вывод: Выбор оптимизатора зависит от архитектуры потенциала.

Task 5: эксперимент с возмущением параметров

Что проверялось

- периодическое возмущение параметров ETN
- цель — уменьшить разброс ансамбля
- датасет MoNbTaW

Почему эксперимент бессмысленный?

- ETN практически сходился в один минимум

pot num	noise sigma	noise every	train energy	train EPA	train forces	val energy	val EPA	val forces	train time, s
1	0.0001	50	1.399954	0.033380	0.268889	1.437466	0.032859	0.279182	1199.09
2	0.0001	50	1.399954	0.033380	0.268889	1.437466	0.032859	0.279182	1085.92
3	0.0001	50	1.399954	0.033380	0.268889	1.437466	0.032859	0.279182	1127.76
4	0.0001	50	1.399954	0.033380	0.268889	1.437466	0.032859	0.279182	1171.74
5	0.0001	50	1.399954	0.033380	0.268889	1.437466	0.032859	0.279182	1220.26

Вывод: ETN сходился почти в один минимум, поэтому perturbation block оставлен как вспомогательная проверка.

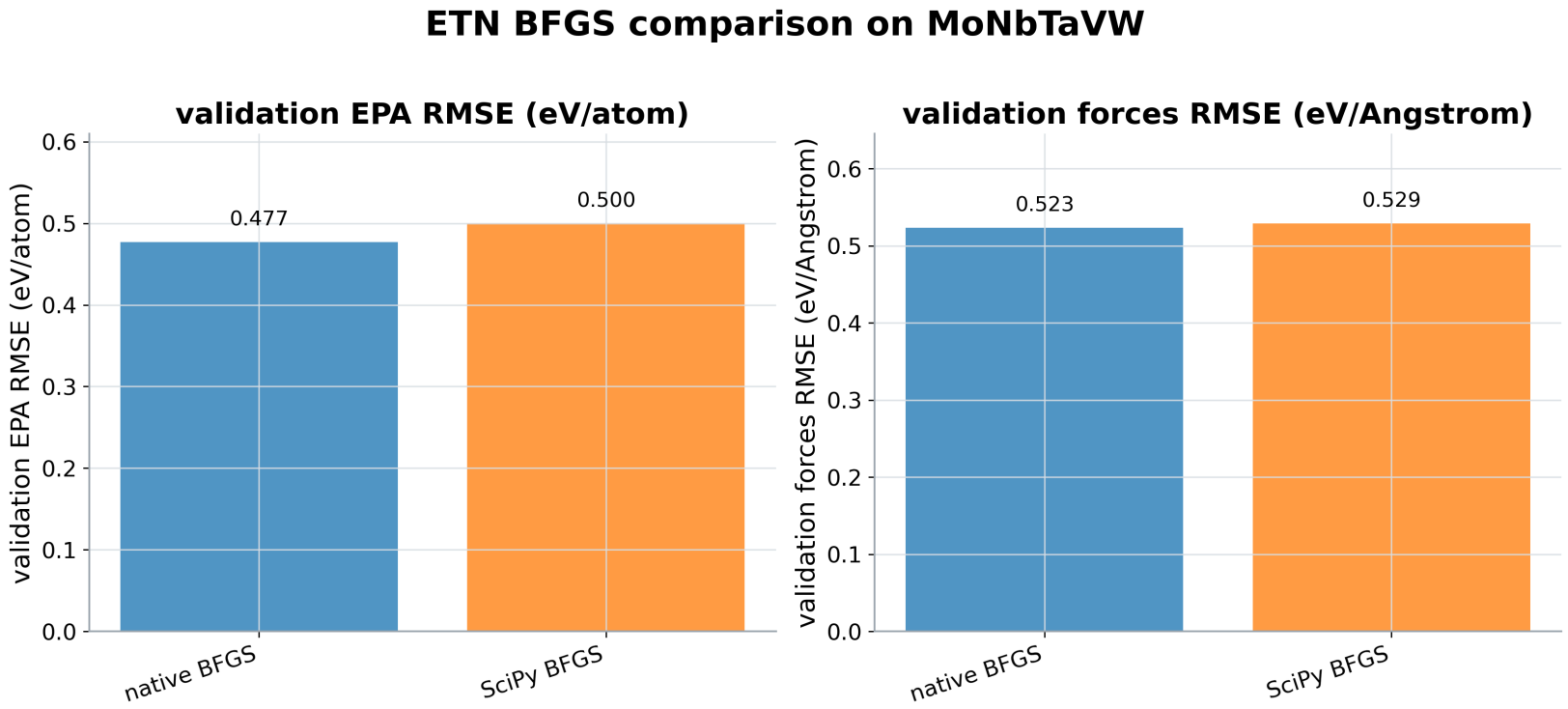
Task 6

Перенос сравнения на MoNbTaVW

Task 6: native BFGS vs SciPy BFGS diagnostics

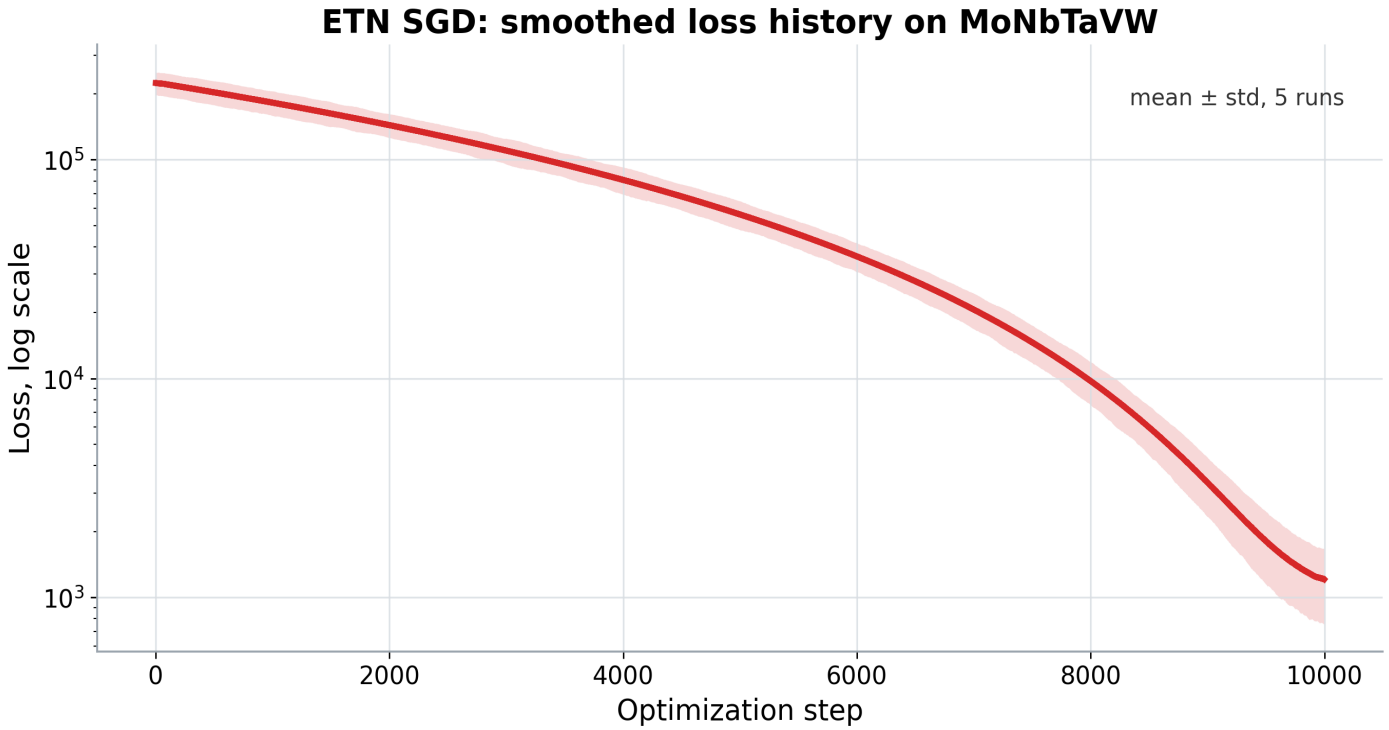
Как читать результат

- важно смотреть validation RMSE
- native немного лучше SciPy по EPA



Вывод: SciPy BFGS близок к native BFGS, но уступает ему по validation RMSE и времени.

ETN MoNbTaVW SGD: динамика обучения

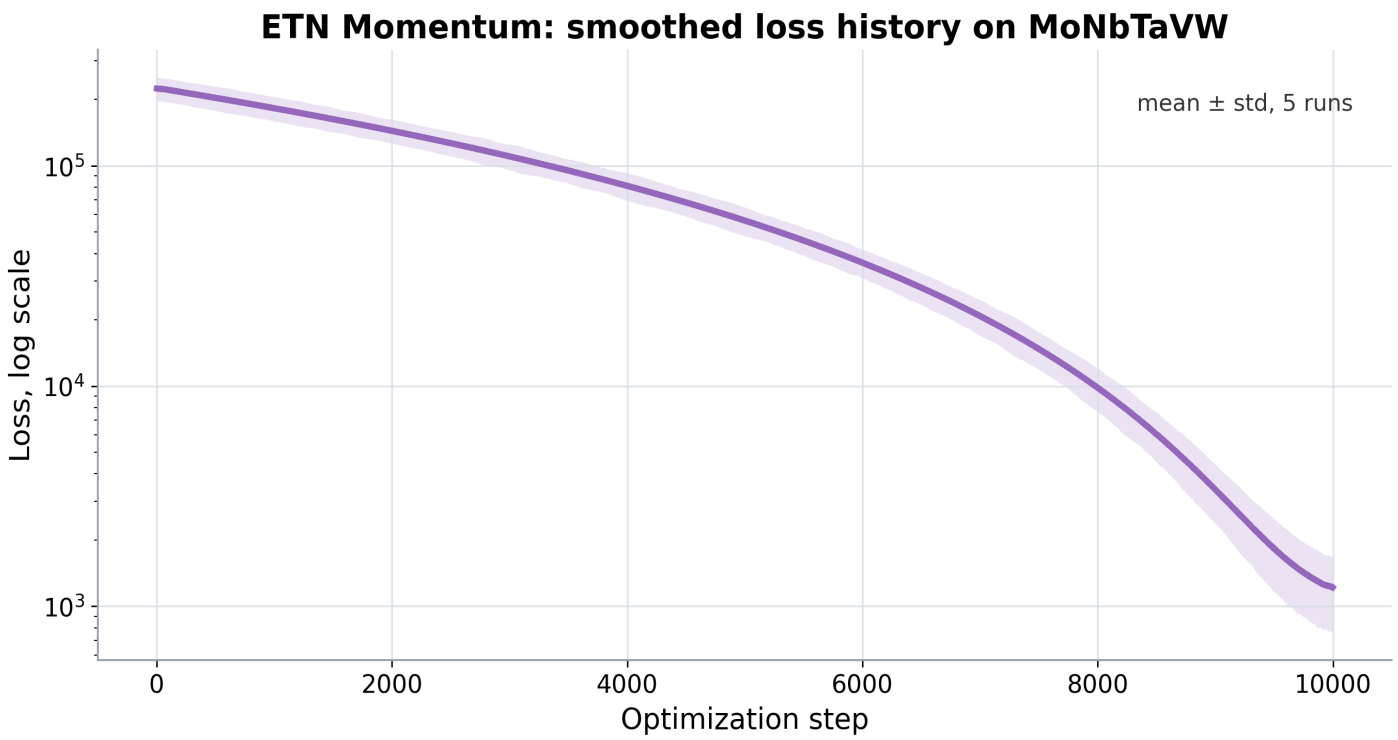


Сравнение с BFGS baseline

metric	SGD	native BFGS	ratio
train EPA RMSE	1.5015	0.4173	3.6×
val EPA RMSE	1.7285	0.4769	3.6×
val forces RMSE	0.8873	0.5235	1.7×
train time, s	5446.4	145.1	37.5×

Вывод: На MoNbTaVW SGD остаётся худшим среди основных методов по validation EPA.

ETN MoNbTaVW Momentum: динамика обучения

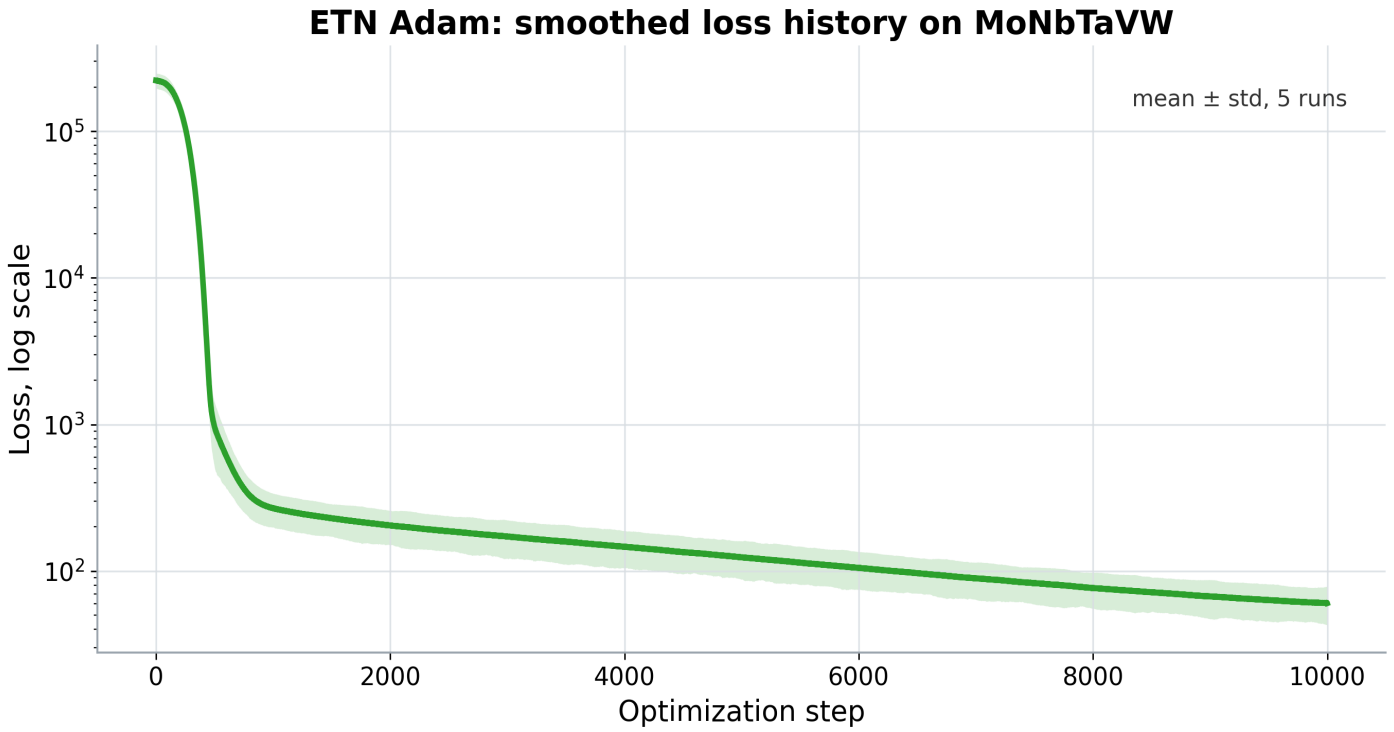


Сравнение с BFGS baseline

metric	Momentum	native BFGS	ratio
train EPA RMSE	1.5031	0.4173	3.6×
val EPA RMSE	1.7301	0.4769	3.6×
val forces RMSE	0.8873	0.5235	1.7×
train time, s	5425.0	145.1	37.4×

Вывод: Momentum снижает loss стабильнее SGD, но итоговая ошибка всё ещё сильно выше BFGS.

ETN MoNbTaVW Adam: динамика обучения

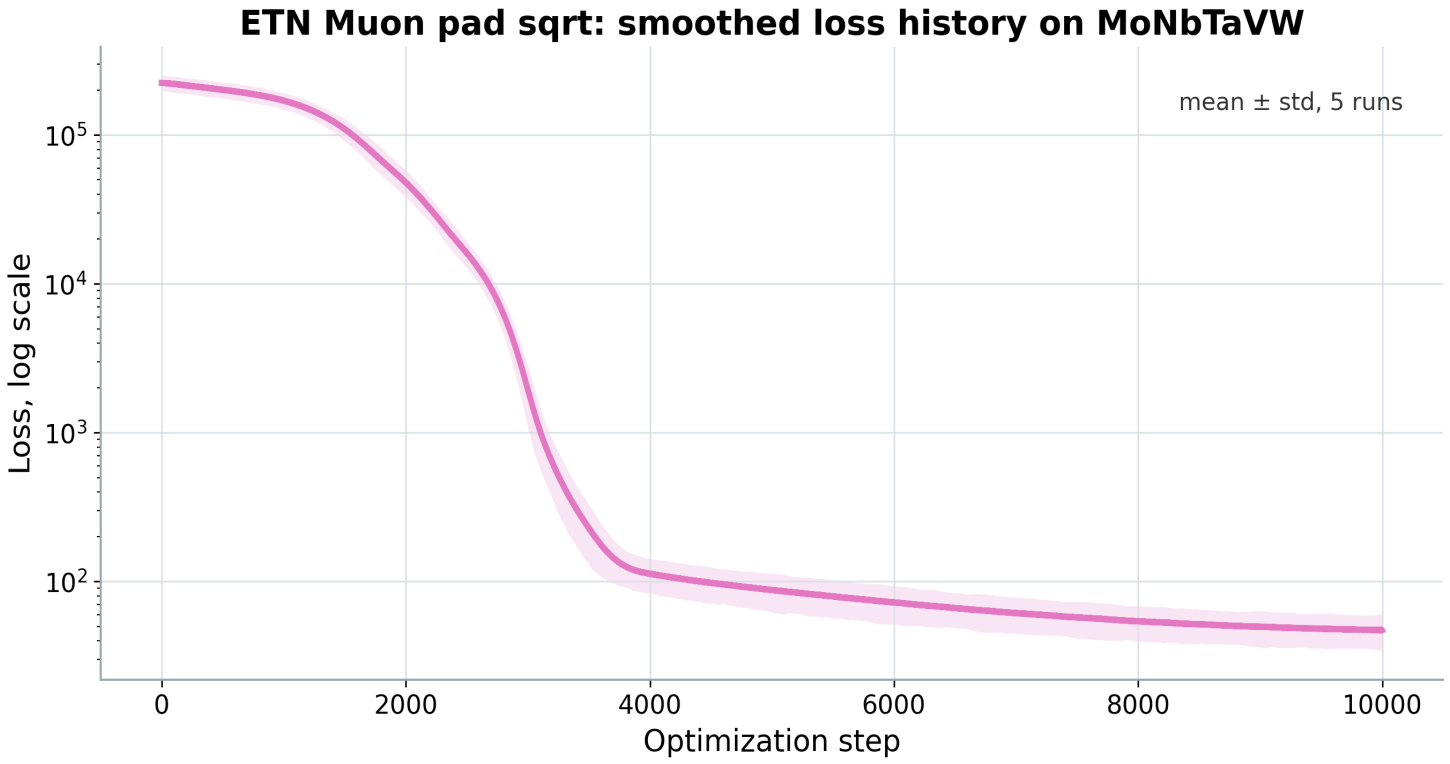


Сравнение с BFGS baseline

metric	Adam	native BFGS	ratio
train EPA RMSE	0.3040	0.4173	0.7×
val EPA RMSE	0.4418	0.4769	0.9×
val forces RMSE	0.6782	0.5235	1.3×
train time, s	5344.0	145.1	36.8×

Вывод: Adam уже превосходит BFGS по validation EPA, но уступает Muon по energy error и BFGS по forces.

ETN MoNbTaVW MUON pad sqrt: динамика обучения

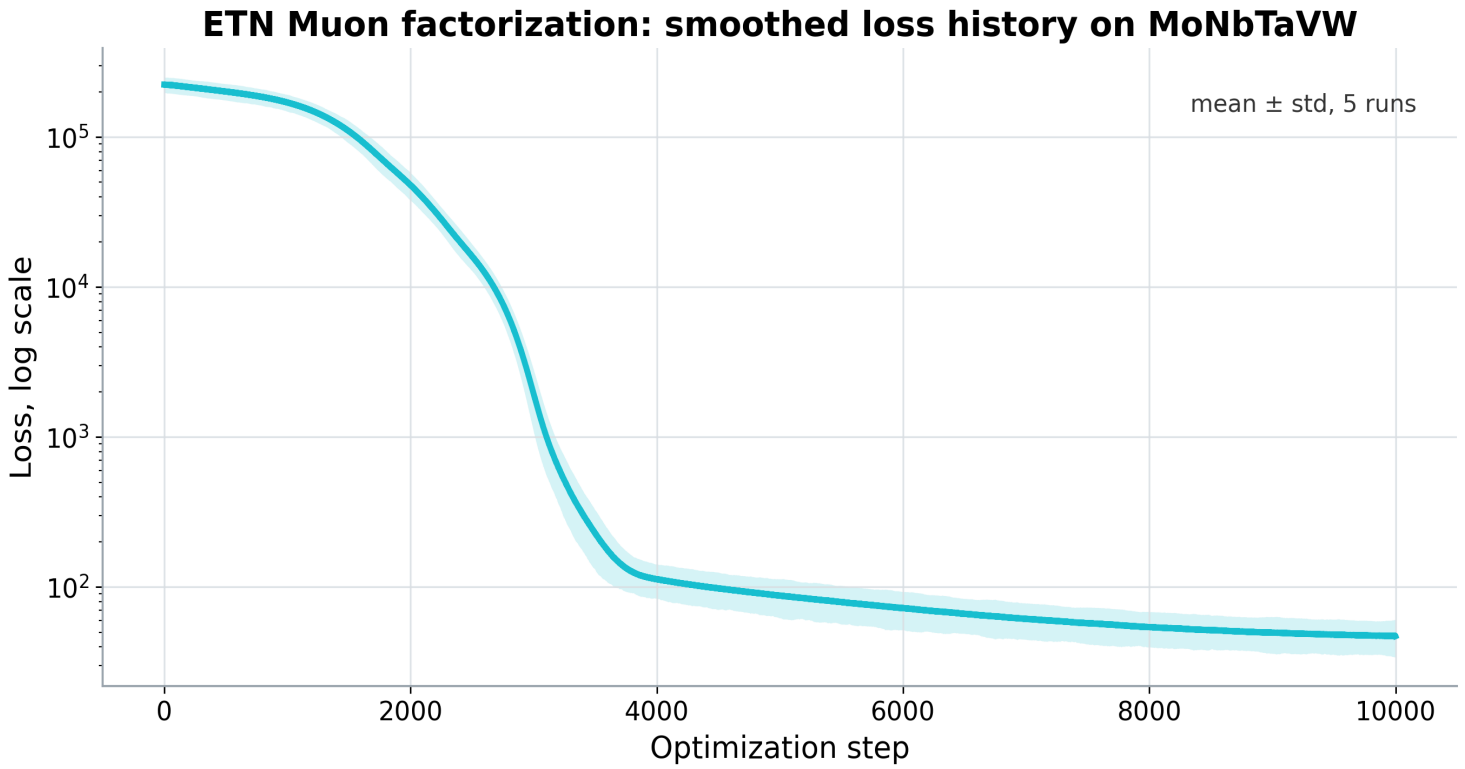


Сравнение с BFGS baseline

metric	MUON pad sqrt	native BFGS	ratio
train EPA RMSE	0.3089	0.4173	0.7×
val EPA RMSE	0.4287	0.4769	0.9×
val forces RMSE	0.6009	0.5235	1.1×
train time, s	5392.0	145.1	37.2×

Вывод: Pad sqrt стал лучшим по validation EPA RMSE, но уступил BFGS по forces и времени.

ETN MoNbTaVW MUON factorization: динамика обучения

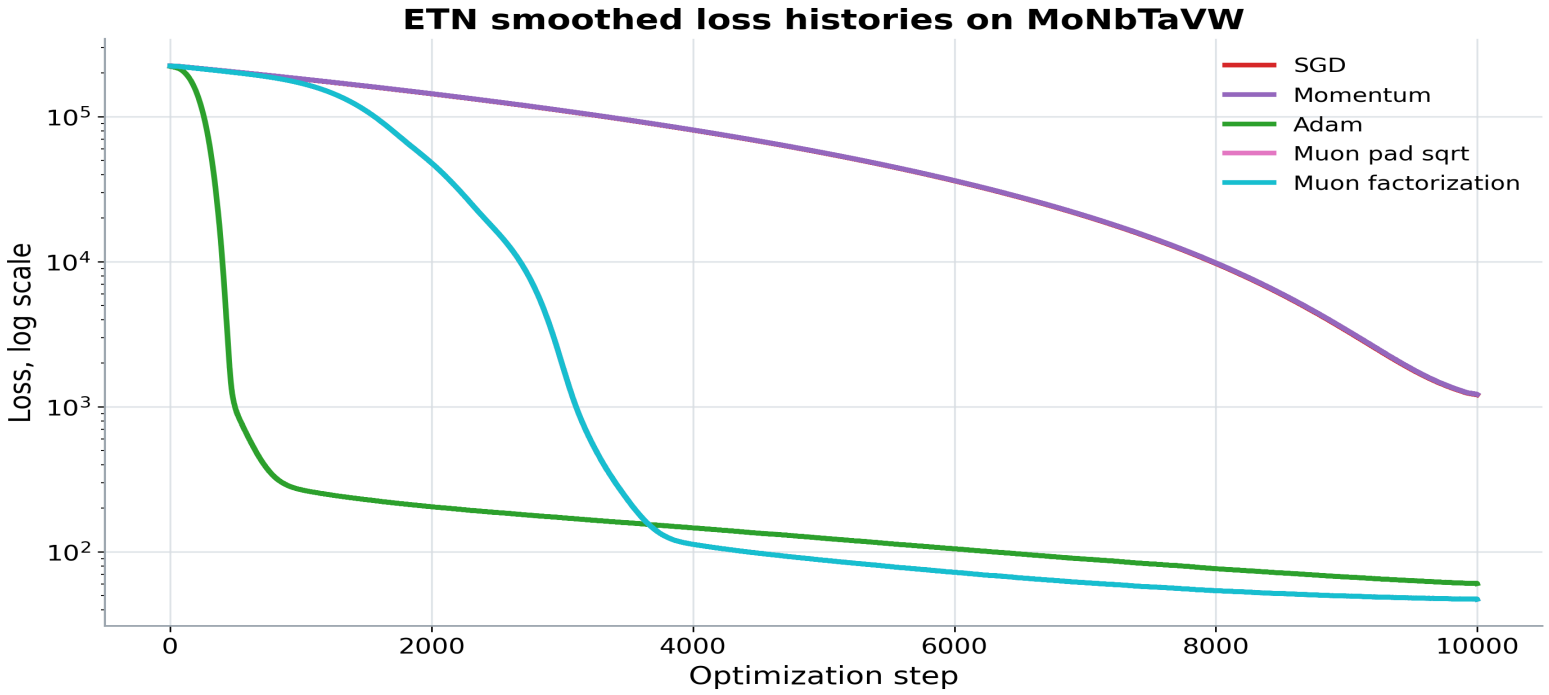


Сравнение с BFGS baseline

metric	MUON factorization	native BFGS	ratio
train EPA RMSE	0.3089	0.4173	0.7×
val EPA RMSE	0.4287	0.4769	0.9×
val forces RMSE	0.6009	0.5235	1.1×
train time, s	5423.3	145.1	37.4×

Вывод: Factorization повторил качество pad sqrt и также превзошёл Adam по validation EPA.

Task 6: ETN loss histories на MoNbTaVW

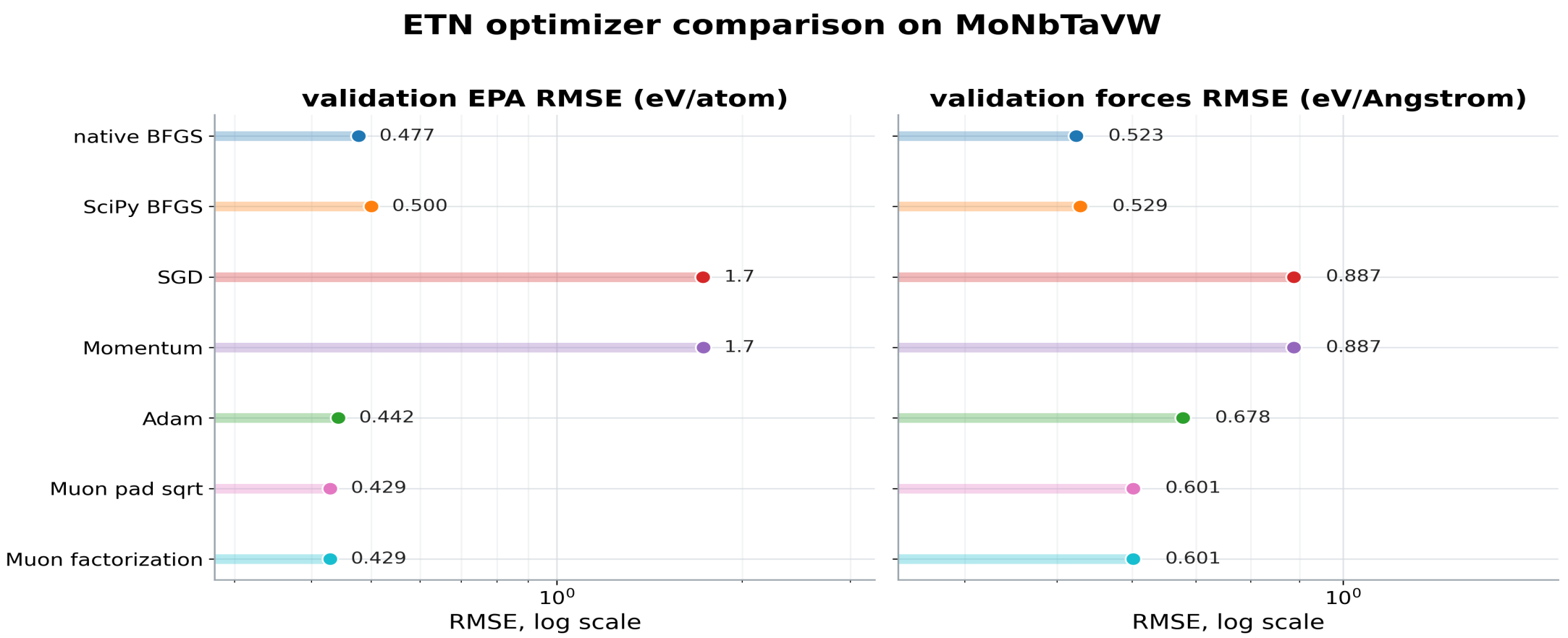


Как читать результат

- SGD/Momentum быстро выходят на плато
- Muon-варианты лучше Adam по EPA
- BFGS лучше по forces и времени
- перенос выводов зависит от метрики

Вывод: На MoNbTaVW Muon даёт лучшую energy error, а BFGS остаётся сильнее по forces и времени.

Task 6: ETN optimizer comparison на MoNbTaVW



Вывод: На MoNbTaVW Muon становится лучшим по validation EPA RMSE, а BFGS остаётся лучшим по forces и времени.

Итоги

Сводные выводы, ограничения и дальнейшая работа

Что получилось для MTP

Ключевые результаты

- лучший метод в итоговой серии — native BFGS
- native BFGS — сильный baseline
- Adam + L-BFGS улучшает Adam
- SGD/Momentum/MUON уступают лучшим BFGS-вариантам

Числа

- native BFGS: EPA RMSE ≈ 0.00210
- SciPy BFGS: EPA RMSE ≈ 0.00233
- Adam + L-BFGS: EPA RMSE ≈ 0.01875
- Adam: EPA RMSE ≈ 0.0293

Вывод: Для MTP наиболее эффективны квазиньютоновские методы.

Что получилось для ETN

AgPd

- native BFGS: ≈ 0.01387
- Adam: ≈ 0.0227
- Muon factorization: ≈ 0.0235
- Muon близок к Adam по EPA и лучше по forces

MoNbTaVW

- native BFGS: ≈ 0.4769
- SciPy BFGS: ≈ 0.5000
- Adam: ≈ 0.4418
- Muon-варианты: ≈ 0.4287

Вывод: Для ETN Muon становится конкурентным mini-batch методом, особенно на MoNbTaVW.

Основные выводы

Финальные тезисы

- BFGS остаётся сильным full-batch baseline, особенно по forces и времени.
- Для MTP итоговая серия даёт минимум EPA у native BFGS; SciPy BFGS близок к нему.
- Для ETN SciPy BFGS не всегда улучшает native BFGS.
- Лучший mini-batch метод зависит от модели, датасета и метрики.
- Для MTP среди mini-batch методов лидирует Adam по EPA.
- Для ETN Muon сравнялся с Adam на AgPd и превзошёл его на MoNbTaVW по EPA.
- На MoNbTaVW разрыв full-batch и mini-batch по energy error стал недраматичным.

Вывод: Универсального лучшего оптимизатора нет; метод нужно подбирать под модель и масштаб параметров.

Спасибо за внимание