Поведение оптимизаторов на шумных данных: эксперимент с искажениями

Сазонова Ирина 23.Б10

30 июля 2025 г.

1 Спринт 1: Теория и подготовка к эксперименту

1.1 Шум в данных

Шум в данных — это любые искажения, ошибки или неточности в наборе данных. Они встречаются практически во всех датасетах по разным причинам, зависящим от типа данных. Шум на изображениях может появиться из-за сжатий, квантового шума или неисправного сенсора. Текстовый шум появляется, например, из-за орфографических ошибок и неформальной речи.

Существуют различные способы борьбы с шумами, например:

1. предобработка входных данных (очистка, нормализация, удаление выбросов и другие);
2. использование быстрого преобразования Фурье;
3. автоэнкодеры;
4. анализ главных компонент (PCA).

Основные виды шума:

1. Гауссовский шум (Gaussian noise)

   Такой тип шума следует нормальному (гауссовскому) распределению. Большинство шумовых значений близки к среднему (часто к нулю), а сильные отклонения встречаются редко. На изображениях он проявляется как легкая рябь или зернистость.

2. Шум "Соль и перец"(Salt & Pepper noise)

   Характерный для изображений шум, который выглядит как случайно разбросанные черные ("перец") и белые ("соль") пиксели. Обычно он является результатом резких сбоев при передаче сигнала, ошибок в ячейках памяти или проблем в аналого-цифровых преобразователях.

3. Шум в метках (Label noise)

   Представляет собой неправильно присвоенные классам метки. Например, если изображение с кошкой было ошибочно помечено как "собака". Такой шум чаще всего возникает из-за человеческих ошибок на этапе разметки данных.

Кроме того, существуют цветные шумы: розовый и голубой. Вместе с гауссовским они используются для обучения нейросетей, которые используются для предсказаний временных рядов.

Влияние шума на обучение нейросетей:

1. Переобучение (нейросеть начинает запоминать шум вместо общих закономерностей)
2. Затруднение выявления важных корреляций и трендов (могут появляться ложные корреляции)
3. Как следствие, снижение точности предсказаний

Однако в некоторых случаях (особенно для малого объема данных) контролируемое добавление шума, например, гауссовского, может быть полезным. Такой метод (аугментация) помогает сделать модель более устойчивой к небольшим изменениям во входных данных и предотвратить переобучение, заставляя ее концентрироваться на более общих признаках.

1.2 Stochastic Gradient Descent (SGD)

Идея: используется классический градиентный спуск, но обновление весов происходит по градиенту, вычисленному по случайному объекту данных (или по случайному мини-батчу), а не по всему датасету. Почти всегда используется вариант с инерцией (Momentum). Инерция накапливает скорость из предыдущих шагов, помогая преодолевать небольшие локальные минимумы и двигаться более стабильно.

Гиперпараметры:

- lr (learning_rate) — Определяет размер шага обновления весов
- momentum — Коэффициент инерции для ускорения сходимости
- dampening — Ослабление влияния текущего градиента при использовании momentum
- weight_decay — L2-регуляризация, сила штрафа за большие веса
- nesterov — Использование модификации Momentum
- maximize — Максимизация функции потерь

Применение:
Довольно универсальный, но в основном применяется в задачах компьютерного зрения, а также для точной настройки и повышения качества после начального обучения. Кроме того, используется на зашумленных данных, поскольку имеет хорошую обобщающую способность.

1.3 Adam (Adaptive Moment Estimation)

Это адаптивный оптимизатор, который сочетает в себе идеи Momentum и RMSprop.

Идея: поддерживает индивидуальную скорость обучения для каждого параметра (веса) нейросети, вычисляя два момента: скользящее средние градиентов (инерция) и скользящее среднее квадратов градиентов (масштабирование). Затем происходит корректировка их смещений и обновление весов. Таким образом, для параметров с большими или частыми обновлениями скорость обучения автоматически уменьшается, а для параметров с редкими обновлениями — увеличивается.

Основные гиперпараметры:

- lr — learning_rate
- betas — Два коэффициента (β_1, β_2) для вычисления скользящих средних первого и второго моментов
- eps — Малое число для предотвращения деления на ноль
- weight_decay — L2-регуляризация, сила штрафа за большие веса
- amsgrad — Использование улучшенной версии Adam
- maximize — Максимизация функции потерь

Применение:
На данный момент является очень популярным и применяется в задачах разного рода, особенно в NLP и обучении GAN.

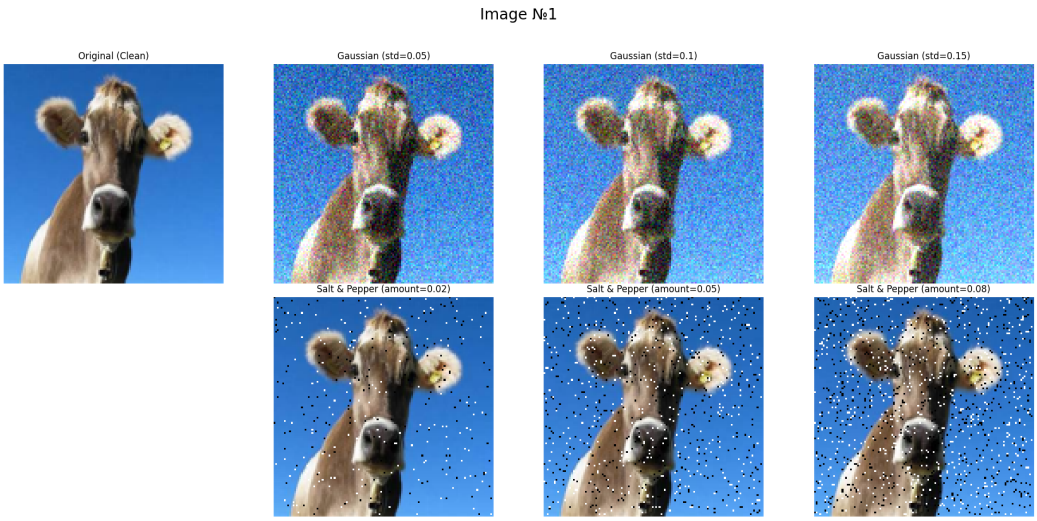

Рис. 1: Виды шумов

1.4 LAMB (Layer-wise Adaptive Moments for Batching)

Это адаптивный оптимизатор, разработанный для эффективного обучения больших моделей (например, BERT и GPT) с использованием очень больших размеров батчей. Он сочетает идеи Adam и нормализации шага обучения для каждого слоя модели, что ускоряет сходимость и улучшает устойчивость.

Идея: обычные оптимизаторы, такие как Adam, требуют уменьшения learning_rate при увеличении размера батча, что замедляет сходимость и приводит к потере точности.

LAMB решает эту проблему, используя послойную нормализацию (layer-wise normalization). Он вычисляет обновления весов так же, как Adam, но перед применением этих обновлений он нормализует их для каждого слоя отдельно. Это происходит путем деления величины обновления на норму весов самого слоя. Такой подход предотвращает затухание обновлений, сохраняя стабильность обучения даже при батчах размером в десятки тысяч примеров.

Основные гиперпараметры такие же, как у Adam.

Применение:

Применяется в задачах NLP и CV, когда доступно большое количество вычислительных ресурсов (GPU/TPU) и требуется максимально сократить время обучения, а также для распределенного обучения.

Научные статьи:

В статье [3] авторы сравнивают обобщающую способность адаптивных оптимизаторов, таких как Adam, AdaGrad и RMSProp, с градиентным спуском и стохастическим градиентным спуском. Исследование показывает, что адаптивные методы менее устойчивы к шуму, чем SGD.

К таким же выводам приходят авторы в статье [1].

В статье [2] авторы показывают, что искусственное добавление градиентного шума улучшает качество модели.

2 Спринт 2: Введение шума и запуск экспериментов

В исходный датасет с 10 видами животных было добавлено 2 вида шума разных уровней. Результат добавления Гауссовского и Salt&Pepper шумов можно увидеть на Рис.1.

Кроме того, были проведены первые эксперименты с разными уровнями шума, результаты которых будут описаны в следующем разделе.

3 Спринт 3: Анализ устойчивости и выводы

В результате подбора гиперпараметров для экспериментов использовались следующие параметры:

1. learning_rate — 0.001
2. batch_size — 64

Кроме того, было выбрано 12 эпох из-за ограничений по времени и ресурсам GPU. Для измерения времени сходимости был установлен целевой лосс = 0.4.

3.1 Точность при разных уровнях шума

Как видно на Рис.2 и в таблице 1, с ростом уровня шума точность модели на тестовых данных становится хуже.

В случае с Гауссовским шумом хуже всего себя показал оптимизатор LAMB: модель с ним имеет худшую итоговую точность и самый резкий график. Оптимизаторы Adam и SGD показали схожую устойчивость к шуму, однако при самом большом уровне шума модель с Adam показала лучшую точность, чем модель с SGD, тогда как при среднем уровне модель с SGD имела большую точность.

В случае с шумом типа Salt&Pepper, то есть с резкими случайными выбросами, лучший итоговый результат показала модель с оптимизатором SGD. Однако график для LAMB выглядит более плавным и с менее резкими перепадами. Модель с Adam показывает лучшую точность с шумом среднего уровня, но эта точность резко падает при высоком уровне шума.

Таблица 1: Сравнение итоговой точности (Accuracy, %) оптимизаторов при разных уровнях шума

Уровень шума	**Оптимизатор**		
	SGD (%)	**Adam (%)**	**LAMB (%)**
Чистые данные (эталон)	**68.41**	**67.59**	**66.73**
Gaussian Noise (уровень = std)			
0.05	61.17	61.61	58.00
0.10	51.28	47.73	41.75
0.15	32.60	39.72	24.14
Salt & Pepper Noise (уровень = amount)			
0.02	66.31	66.62	62.83
0.05	57.30	65.07	59.91
0.08	60.41	51.60	55.73

3.2 Сходимость при разных уровнях шума

Время сходимости определялось как время, за которое оптимизатор достигает целевого значения функции потерь. В таблице 2 представлено время сходимости для каждого оптимизатора и уровня шума. Видно, что SGD достигает целевого значения лосса при любом уровне шума, в то время как адаптивным оптимизаторам это удается не всегда. Визуально результаты можно увидеть на Рис.3. Таким образом, SGD показывает наилучшее время сходимости при любом уровне шума.

3.3 Влияние шума на переобучение

На Рис.4 представлено поведение оптимизаторов на данных с высоким уровнем Гауссовского шума. Видно, что оптимизатор SGD лучше всего подстраивается под зашумленную обучающую выборку: его кривая потерь стабильно находится ниже двух других. Adam хуже всех справился с задачей минимизации ошибки: его кривая потерь находится выше всех.

Однако, как было сказано ранее, на тестовых данных модель с SGD показывает точность хуже, чем модель с Adam, что говорит о том, что Adam меньше всех переобучился на шум.

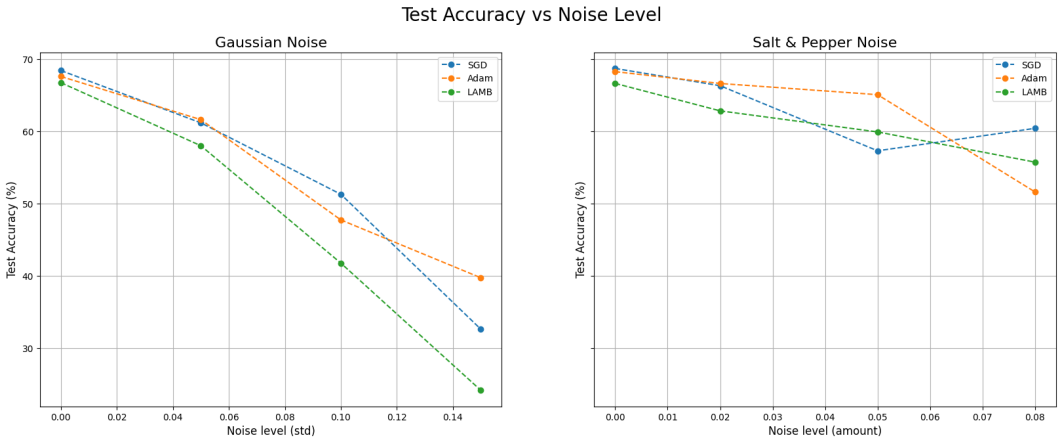

Рис. 2: Точность на тестовых данных

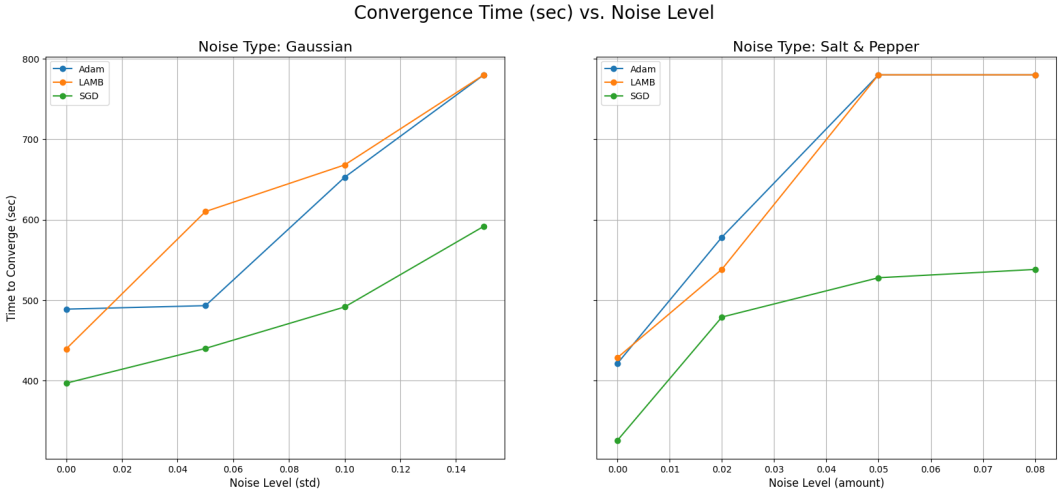

Рис. 3: Зависимость времени сходимости от уровня шума

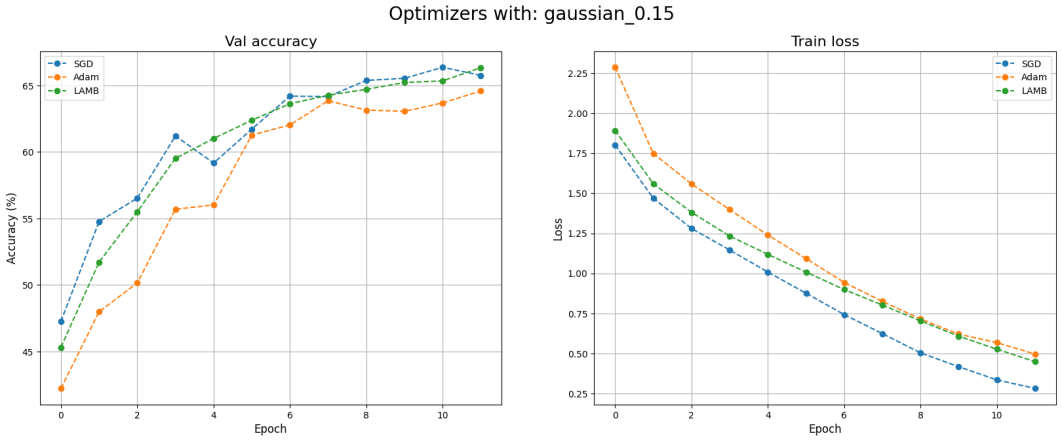

Рис. 4: Поведение оптимизаторов на данных с высоким уровнем Гауссовского шума

Таблица 2: Сравнение времени сходимости (в секундах) до целевого уровня потерь

	Оптимизатор		
Уровень шума	**SGD (сек)**	**Adam (сек)**	**LAMB (сек)**
Чистые данные (эталон)	**396.92**	**488.88**	**439.55**
Gaussian Noise (уровень = std)			
0.05	440.09	493.20	610.14
0.10	491.58	652.72	668.06
0.15	591.82	$>$ 12 эпох*	$>$ 12 эпох*
Salt & Pepper Noise (уровень = amount)			
0.02	479.01	578.24	538.32
0.05	527.87	$>$ 12 эпох*	$>$ 12 эпох*
0.08	538.23	$>$ 12 эпох*	$>$ 12 эпох*

*Модель не достигла целевого уровня потерь за отведенное количество эпох.

4 Выводы

При работе с зашумленными данными модель с Adam показала наилучшие результаты.

SGD показал результаты немного хуже, однако он продемонстрировал более надежную сходимость: время его сходимости значительно лучше, чем у Adam. Также SGD требует тщательной настройки гиперпараметров, поэтому для него результаты на тесте могут измениться в лучшую сторону.

LAMB по поведению близок к Adam и явных приемуществ при работе с зашумленными данными не показал, поскольку он лучше всего работает с батчами больших размеров.

Список литературы

[1] Subhajit Chaudhury и Toshihiko Yamasaki. “Robustness of adaptive neural network optimization under training noise”. В: *IEEE Access* 9 (2021), с. 37039—37053.

[2] Arvind Neelakantan и др. “Adding gradient noise improves learning for very deep networks”. В: *arXiv preprint arXiv:1511.06807* (2015).

[3] Ashia C Wilson и др. “The marginal value of adaptive gradient methods in machine learning”. В: *Advances in neural information processing systems* 30 (2017).