Семинар "Педагогика ИИ"

Дата и время: 28.01.2026, 10.00

Место проведения: НИУ «МЭИ», г. Москва, ул. Красноказарменная, д. 13с3, аудитория М-101.

Участники: Олег Вадимович Глухов (НИУ «МЭИ»), Павел Юрьевич Анучин (НИУ «МЭИ»), Шамиль Алиевич Оцоков (НИУ «МЭИ»), Эдуард Артурович Челышев (НИУ «МЭИ»), Константин Евгеньевич Бошков (НИУ «МЭИ»), Вадим Игоревич Лазарев (НИУ «МЭИ»), Евгений Александрович Волошин (НИУ «МЭИ»).

Повестка:

1) Тема основной дискуссии «Генеративный ИИ»:
- Доклад «Принципы, заложенные в основу генеративных моделей, и подходы к их проектированию»;
2) Выбор тематики следующего семинара.

Олег Вадимович Глухов:
Всех приветствую. Это первый в новом году семинар — одиннадцатый. Мы выложили десять семинаров на сайт МЭИ в качестве стенограмм — можно ознакомиться. Когда мы только начинали наши первые дискуссии, мы много говорили о генеративном ИИ. Это генерация различных типов данных: изображения, звукозаписи, видео и так далее. Но на самом деле за всем этим стоит сложная математика, различные подходы, которые активно развивались в последние десять лет. И сегодня Вадим Лазарев расскажет нам о принципах, заложенных в основу генеративных моделей. Фактически все используемые сейчас сервисы — ChatGPT, DeepSeek и так далее — это тоже своего рода генеративные модели. Интересно также затронуть вопросы обработки не только текстовой информации. Мы часто в разговорах касались вопросов обработки различной документации. Там есть типы данных, которые представлены не векторно, а в формате растровых картинок и различных схем. Где-то это старые документы, которые нужно оцифровать и как-то с ними работать. Думаю, если мы сейчас коснёмся вопросов генеративных моделей, то, возможно, ответим и на вопрос, что с этим делать и возможно ли это вообще.

Вадим Игоревич Лазарев:
Здравствуйте, уважаемые коллеги. Сразу перейду к архитектурам. Здесь представлены все основные на данный момент архитектуры, которые используются в том же ChatGPT. Сюда ещё можно добавить трансформеры, но я их не указал по простой причине — это модель, направленная не на генерацию, а на внимание, то есть на определение ключевых моментов в данных и ключевых объектов. Как раз

гибриды трансформера с одной из этих четырёх архитектур и используются. Что касается более подробного рассмотрения вариантов этих моделей, какие модификации на данный момент используются. То, что представлено в главе, — это лишь базовая архитектура. Всё, что ниже, — это различные доработки, которые позволяют избавиться от некоторых недостатков этих моделей, либо направлены на решение конкретных задач. Прежде чем мы перейдём непосредственно к самим генеративным сетям, надо проанализировать следующую архитектуру— автоэнкодер. Он преобразовывает данные, которые у нас есть, в те же самые данные, за исключением того, что посредине у нас есть скрытое векторное пространство Z, которое является сжатым образом наших данных на входе. То есть, например, есть какой-то элемент датасета. Эта двойка из датасета с рукописными цифрами MNIST-784. Там квадрат 28 на 28 пикселей, в котором присутствуют пиксели цифр от 0 до 9. Мы это посылаем на вход в наш энкодер, который закодирует наше изображение в более сжатое пространство. То есть вместо 784 пикселей у нас, например, получается 32 значения, которые затем при помощи декодера будут восстановлены обратно в изображение, которое было изначально на входе. В чём здесь идея? Зачем нам вообще этот автоэнкодер понадобился? Потому что у него появляется новый термин — скрытое пространство. Оно используется для того, чтобы так или иначе увеличить скорость работы модели и обучения, поскольку мы работаем не с большими изображениями, у которых высокое разрешение, а с подпространством, которое является сжатым прообразом.

Олег Вадимович Глухов:
Это применимо к другим типам данных?

Вадим Игоревич Лазарев:
Ко всему применимо: к текстовым данным, к изображениям, к музыке, к звукам, к чему угодно. Энкодер — это многослойный персептрон, обыкновенная глубокая нейронная сеть прямого распространения, которая обучается сжимать наше изображение в пространство с меньшим количеством измерений. А декодер — это развёртываемая в другую сторону нейросеть, которая, наоборот, из сжатого вектора получает обратно наши пиксели. Сам процесс обучения достаточно прост. Здесь используется следующая функция потерь — это наименьшая квадратическая ошибка. Мы можем обозначить это так: энкодер — это некоторая функция с параметрами φ, декодер — функция с параметрами θ. Энкодер, получая на вход наше изображение X, проходит через него. Мы получаем вектор в пространстве Z и сравниваем, насколько хорошо получилось восстановленное изображение по сравнению с тем, которое было на входе. Вот эту ошибку мы и пытаемся уменьшить, сделать потери как можно меньше. Мы стремимся к тому, чтобы это векторное пространство Z, а вместе с ним энкодер и декодер, были способны идеально восстанавливать наше изображение, звук, музыку — всё что угодно. Смысл в чём? Мы вместо того, чтобы подавать на вход наше идеальное изображение, мы его зашумляем. Добавляем туда какие-то шумы или, например, удаляем просто нейроны, которые получают информацию от этого изображения, и делаем так, чтобы либо энкодер у нас не зацикливался на том, что у него все нейроны есть и всегда есть вся информация, либо делаем так, чтобы на входе у энкодера у нас никогда не было идеального изображения, а было изображение с шумом.

Олег Вадимович Глухов:
Это техника регуляризации, да?

Вадим Игоревич Лазарев:
Да, это через дропаут, например, если мы работаем внутри нейронной сети.

Олег Вадимович Глухов:
А вот есть, допустим, регуляризация Тихонова — про то же самое в данном случае, когда мы ограничиваем размеры весов?

Вадим Игоревич Лазарев:
Скорее всего, если мы ограничиваем размеры весов L2, L1 норма. Это тихоновская, по-моему, L2 называется как раз-таки.

Олег Вадимович Глухов:
У нас есть гребневая регуляризация, а есть лассо. Когда мы ограничиваем размеры весов по манхэттенской норме L1 — это получается лассо, а гребневая — это по L2. То есть мы в функцию потерь включаем дополнительный член, который ограничивает размерность и влияет на расчёт самой функции потерь.

Вадим Игоревич Лазарев:
Я не знаю, насколько это здесь релевантно, но в целом в самом чистом автоэнкодере это вообще не используется - достаточно функции потерь. А именно в автоэнкодере, который избавлялся бы от шума, есть либо такой вариант с добавлением регуляризации через откидывание нейронов, либо ограничение весов, либо мы зашумляем само изображение.

Олег Вадимович Глухов:
Допустим, у нас есть растровая картинка, и мы хотим получить вектор. Для задачи обработки технической документации, потому что векторные данные проще обрабатывать. Автоэнкодер мне поможет из растрового изображения получить векторное? Векторное — это когда мы конкретными числами описываем изображение.

Шамиль Алиевич Оцоков:
Изображение так и так числами описывается.

Вадим Игоревич Лазарев:
Именно как векторное, как SVG, например?

Олег Вадимович Глухов:
Да, такого плана.

Вадим Игоревич Лазарев:
Здесь нет — находим только растровое. То есть будут просто числа. Мы не сможем получить пункты, которые связывают наше изображение в виде пикселей.

Олег Вадимович Глухов:
Смысл вектора — это когда мы не элементарными пикселями оперируем, а уже какими-то образами: линии, полилинии и так далее. Для этого нужно уравнение.

Вадим Игоревич Лазарев:
Да, это надо уже тогда делать через ту же диффузионную модель - связку. Или через трансформер. Мы можем сделать обучающий датасет, в котором у нас слева есть картинки в формате PNG, а справа — SVG — векторные. Соответственно, заставить трансформер вычленять ключевые элементы из нашего изображения и сопоставлять их с вектором.

Олег Вадимович Глухов:
Я думаю, решение не будет актуально.

Вадим Игоревич Лазарев:
Трансформер здесь подойдёт. Точнее, не именно трансформер, а visual transformer, который работает с изображениями.

Олег Вадимович Глухов:
Это не языковая модель?

Вадим Игоревич Лазарев:
Нет, не языковая.

Олег Вадимович Глухов:
Именно базовый трансформер?

Вадим Игоревич Лазарев:
Это модификация трансформера. Итак, про сам автоэнкодер. Он используется для извлечения признаков. Мы об этом уже поговорили. То есть он создаёт более сжатое пространство, которое содержит ключевые элементы из нашего пространства более высокой размерности.

Олег Вадимович Глухов:
Обнаружение аномалий. Когда в языковых моделях используются эмбеддинги — это то же самое в данном случае?

Вадим Игоревич Лазарев:
Эмбеддинг — это кодирование. Про RAG слышали? Это дополненная генерация к контекстам, когда вам нужно связать большую модель со справочником, который у вас в базе данных присутствует. Вы должны представить весь свой контекст, который передаёте, в том числе запрос. То есть закодировать эти предложения так, чтобы они по смыслу близко друг к другу находились. Мы из многомерности такого большого объёма, когда у нас есть абзац или целая страница текста, превращаем это в некие вектора, между которыми чётко математически можно посчитать нормы и понять схожесть. Эмбеддинг — это когда мы кодируем по смыслу. То есть у нас заранее есть какая-то база данных, в которой мы сделали кодировщик по смыслу. Либо можно применять более простые варианты, когда мы просто кодируем наши слова.

Олег Вадимович Глухов:
Это исходя из контекста?

Вадим Игоревич Лазарев:
Да, то есть мы можем учитывать контекст, можем не учитывать контекст. Можем учитывать смысл или можем не учитывать смысл. Тут зависит от того, какая у нас задача стоит, и, можем ли мы все наши данные на входе считать равнозначными. Это если мы тренируем наш автоэнкодер на каких-то конкретных данных — какие-нибудь банковские транзакции или, например, какой-то стандартный видеопоток. Мы транслируем туда только новости. Как только у нас появляется на ходе блокбастер, экшен, мы сразу же должны обнаружить эту аномалию и сказать: нет, здесь что-то непонятное. Каким образом это обнаруживается? Когда мы работаем с нашим пространством, мы его заранее делаем таким, чтобы оно генерировало оттуда данные только из нашего набора. Например, цифры. То есть мы сделали специально такое Z-пространство, которое генерирует цифры. Берём случайный вектор из него. Достаточно низкая вероятность того, что мы реально получим цифру, но аномалию мы тоже обнаружить сможем, потому что на выходе мы точную цифру не получим, если наш вектор Z, который был получен при кодировании — точнее наши данные, которые закодированы, — с очень низкой вероятностью попадут под пространство цифр. Ну и то же самое с банковскими транзакциями. Как только обнаруживаются какие-то аномальные переводы и так далее, декодер не сможет восстановить это в меняемое значение. То есть банально по числам мы можем это определять.

Олег Вадимович Глухов:
У нас есть только энкодер и декодер. То есть их как-то можно уже в других задачах использовать?

Вадим Игоревич Лазарев:
Да, в этом и суть.

Шамиль Алиевич Оцоков:
Несбалансированное количество классов: допустим, много представителей одного класса и мало — другого. Это очень удобно использовать.

Вадим Игоревич Лазарев:
Средства балансировки: мы можем случайно отбирать одинаковое, равномерное количество всех классов. Это можно сделать ещё на этапе процессинга. Но да, автоэнкодеры устойчивы к неравномерности.

Шамиль Алиевич Оцоков:
Вот те же самые на атомных электростанциях: предсказания аварийных ситуаций. Датасеты дадут, где много нормальных ситуаций, а где аварийных — нет.

Вадим Игоревич Лазарев:
Это как раз аномалия.

Шамиль Алиевич Оцоков:
Мы детектим аномалию. Это как раз можно тоже применить.

Олег Вадимович Глухов:
Ещё надо заметить: по сути, обучение без учителя, потому что у нас только входные данные присутствуют, нам не нужно разметку делать.

Вадим Игоревич Лазарев:
Да, это достаточно удобно.

Олег Вадимович Глухов:
Потому что, если мы сравниваем, просто результат, то получим, что она выдала.

Вадим Игоревич Лазарев:
А дальше уже просто пользуемся обычными многослойными персептронами, энкодером, декодером, по отдельности.

Константин Евгеньевич Бошков:
Мы можем в таком случае классифицировать информацию? То есть мы должны какую-то метку задать.

Олег Вадимович Глухов:
Это другая задача.

Вадим Игоревич Лазарев:
У нас здесь не стоит задача делать метки и классифицирование. Надо быть в том пространстве, которое мы задаём. То есть оставаться в цифрах.

Олег Вадимович Глухов:
Другой обнаруживающий подход — тут просто можно подумать, как его конкретно задать.

Вадим Игоревич Лазарев:
Суть этой цифры — это и есть метка.

Константин Евгеньевич Бошков:
Да. Если мы хотим сгенерировать двойку, то мы должны находиться в пространстве двоек. А если в этом пространстве все цифры?

Вадим Игоревич Лазарев:
Это уже другой вопрос. Если мы хотим что-то конкретное сгенерировать, это уже называется проблемой условной генерации. И это уже лучше к диффузионным моделям обращаться, потому что там мы получим более точные значения высокого разрешения. В случае с автоэнкодером — нет. Сразу скажу: мы генерируем уже новое изображение. Это изображение, которого никогда не было.

Олег Вадимович Глухов:
Какая-то линейная или нелинейная комбинация таких векторных пространств.

Константин Евгеньевич Бошков:
Можем применить, если какой-нибудь датасет хотим увеличить?

Вадим Игоревич Лазарев:
Да, его как раз используют для того, чтобы синтетические данные генерировать, например, в медицинских сферах. Таким образом, можно и датасет сбалансировать. Это всего лишь основа, которая вообще не привязана никак к генеративному ИИ. На

самом деле, его для генерации не используют. Вот основные направления, где его можно применять сейчас.

Олег Вадимович Глухов:
В технических системах в целом довольно интересно что-то такое попробовать.

Вадим Игоревич Лазарев:
Да, его на самом деле достаточно просто реализовать. Здесь нет каких-то сложных функций потерь, он обучается достаточно быстро. Вот пример, я его обучал на 10 эпох. То есть 10 эпох, эпоха датасета на 60 тысяч цифр всего.

Олег Вадимович Глухов:
А тут что решалось? Какая задача? Просто восстановление?

Вадим Игоревич Лазарев:
Да, восстановление цифр. Теперь уже непосредственно мы переходим к генеративному ИИ. В общем, вариационный автоэнкодер. Как будто бы те же самые элементы: у нас есть входные данные, энкодер, скрытое пространство, декодер и данные, которые нам нужно восстановить. Но помимо всего того, что было в автоэнкодере, мы добавляем следующие вещи: мы добавляем вероятности, начинаем с ними работать.

Олег Вадимович Глухов:
Смысл такой: я, допустим, вам объяснял тогда, что вероятность правдоподобия означает, что мы, имея это измерение, когда оно в данной ситуации подходит, имеем большую вероятность. Пример: допустим, если мы сейчас дальномером будем мерить около стенки, дальность до стены в 20 сантиметров — такое измерение будет иметь большую вероятность. Если мы получим 2 метра — там будет низкая вероятность. То есть такой принцип: насколько правдоподобно твоё измерение текущей ситуации, твоему состоянию?

Вадим Игоревич Лазарев:
Итак, собственно, почему вероятности? Я лично могу это описать следующим образом. В общем, в автоэнкодере у нас тоже так или иначе присутствует вероятность, плотность вероятности для нашего вектора Z, просто они не используются там в принципе ни при обучении, ни при непосредственно уже генерации каких-то изображений. Здесь же мы разграничиваем энкодер и декодер как раз с целью того, чтобы у нас была система для генерации чего-либо, например тех же самых цифр. Обучение делится на два этапа. У нас задача — обучить энкодер точно так же, как и в автоэнкодере, но у нас появятся две функции потерь. То есть нам нужно обучить энкодер определённым образом. У нас есть плотность вероятности наших векторов Z в пространстве Z, которую мы как раз и пытаемся воссоздать.

Олег Вадимович Глухов:
A Z — это состояние в данном случае?
Вадим Игоревич Лазарев:
Случайная величина.

Олег Вадимович Глухов:
Тут в данном случае Z — это как раз наше состояние, которое мы хотим получить, а X — это измерение, да?

Вадим Игоревич Лазарев:
X — это что-то на входе, какие-то данные.

Олег Вадимович Глухов:
То есть фактически это то, что мы хотим получить, как априорную вероятность?

Шамиль Алиевич Оцоков:
Да, именно. Если у нас пространство Z двумерное, то есть у нас каждая цифра, допустим, изображения, преобразуется в точку на координатной оси, на некоторые точки. Если мы возьмём какую-нибудь точку рядом лежащую и от неё декодером получим изображение, оно будет похоже на цифру 2?

Вадим Игоревич Лазарев:
Вероятность того, что будет похоже, нулевая.

Шамиль Алиевич Оцоков:
В вариационном то же самое.

Вадим Игоревич Лазарев:
То есть и в автоэнкодере, что у нас был до этого, и здесь: если мы берём какой-то случайный вектор в автоэнкодере, то вероятность того, что мы получим именно цифру на выходе, очень низка.

Шамиль Алиевич Оцоков:
Нет, посмотрите. Я про другое. У нас цифра 2 преобразовалась в какую-то точку с координатами 10, 10. Вот я беру точку с координатами 10,1, 10,1.

Вадим Игоревич Лазарев:
Если мы работаем уже с нашим вероятностным пространством, которое мы построили, то есть это кусок нашего пространства Z, то да, так и будет работать.

Олег Вадимович Глухов:
Ну вот это похоже на то, про что мы говорили с эмбеддингами. То есть вектор смыслов и здесь присутствует. Про смыслы — не в плане текста, а в плане картинок.

Павел Юрьевич Анучин:
Ну и по итогу всё равно в этом векторе, в этом пространстве будут однотипно?

Шамиль Алиевич Оцоков:
Обычно в автокодировщиках — нет. В вариационном — да.

Вадим Игоревич Лазарев:
Может быть, я не прав. В вариационном у него достаточно высока вариативность данных, которые он генерирует. То есть у нас не будут генерироваться одни и те же картинки - они будут разные. Но общий смысл будет сохраняться.

Олег Вадимович Глухов:
Как раз было исследование, когда взяли вот этот датасет MNIST и попробовали кластеризовать все примеры в двумерном пространстве. Там получилось так, что мы сформировали 10 кластеров — цифр от 0 до 9, и они все друг другу очень близки были. По смыслу, наверное, то же самое должно получаться: мы формируем некий кластер, в котором эта точка, допустим, может быть, край, центр и так далее. Если мы пойдём влево-вправо, то получаем очень похожие примеры.

Вадим Игоревич Лазарев:
Потому что вот эта двумерная картинка, которая там как раз генерировалась, она была получена через снижение размерности. И поэтому насколько там оно их в итоге преобразовало и переконвертировало так, что они так выглядят, нельзя с точностью сказать. Но смысл тот, о котором мы говорили: смогу ли я, взяв одно случайное число, один случайный вектор Z из нашего уже готового вероятностного распределения, и взяв очень близкий к нему, получить ту же самую цифру? Да, смогу. Вот это как раз параметры нашего распределения, которые должны будут быть выучены нашим энкодером. То есть наш энкодер должен на выходе своём дать два значения — сигма и мю. Если мы вскроем этот вариационный автоэнкодер, что мы получим? В общем, у нас есть вероятностное распределение наших цифр и чего угодно. Это огромное количество измерений обычно, сумасшедшая размерность. И зачастую, мы его не знаем и узнать не сможем никогда. Например, я хочу генерировать котиков. У меня нет вероятностного распределения всех картинок котиков. Я его никак не получу, мне его ниоткуда не взять. Потому что все картинки с котиками, которые возможны, я не могу найти.

Олег Вадимович Глухов:
Как генеральная совокупность и выборка.

Вадим Игоревич Лазарев:
Да. У нас есть только выборка, только конкретные примеры. На основе этих примеров мы делаем следующее: мы через эту связь пытаемся создать другое вероятностное распределение, которое более низкой размерности, но при этом мы его считаем нормальным — так нам проще просто будет. Мы его считаем нормальным с параметрами 0, 1 — среднее, стандартное распределение. И на основе этих предположений мы говорим: нам нужно получить вот это вероятностное распределение Q(Z|X), которое будет похоже на это P(Z) и при этом будет нормальным. То есть мы тоже его считаем нормальным распределением. Именно этот кусочек с параметрами μ и σ.

Олег Вадимович Глухов:
Что оно конкретно делает над этим нашим распределением? Как оно преобразовывает?

Вадим Игоревич Лазарев:
Это условная вероятность — получить из нашего вектора Z пример из X.

Олег Вадимович Глухов:
Я к тому, что мы делаем в сочетании математики с примерами.

Вадим Игоревич Лазарев:
Это идёт в функцию потери, и работает уже там. Здесь это условная вероятность получения из нашего изображения.

Олег Вадимович Глухов:
В первом случае получается функция правдоподобия, а в обратном случае — это апостериорная плотность вероятности.

Вадим Игоревич Лазарев:
Да, как раз вероятность получить именно из нашего вектора Z, который мы взяли из этого вероятностного распределения P(Z), обратно что-то из X. Что касается обучения: у нас есть уже два компонента у функции потерь. Первый компонент — это как раз то, насколько это наше распределение Q(Z|X) будет похоже на P(Z). У нас задача, чтобы это распределение было как можно более похожим на то, которое мы аппроксимируем. Имеем стремящуюся к ней плотность вероятности Q. Здесь используется дивергенция Кульбака - Лейблера. Просто функция для того, чтобы сравнить два вероятностных распределения.

Олег Вадимович Глухов:
Что за операция этих параллельных линий?

Вадим Игоревич Лазарев:
Это такое обозначение, чтобы не путаться с условными вероятностями. То есть у нас есть два распределения. У нас, во-первых, оно не симметричное. Во-вторых, у него нет никаких пределов снизу и сверху.

Евгений Александрович Волошин:
На самом деле есть дивергенция Дженсена - Шеннона, и она как раз лучше с точки зрения применимости в вычислениях, потому что она, во-первых, ограничена от нуля до единицы, во-вторых, она симметричная.

Олег Вадимович Глухов:
Это всё не из теории информации? Это энтропия?

Вадим Игоревич Лазарев:
Да, это оттуда полезно.

Олег Вадимович Глухов:
Почему там не применимы базовые MSE, как раз МНК-постановки?

Вадим Игоревич Лазарев:
Потому что мы работаем с вероятностным распределением.

Вадим Игоревич Лазарев:
Мы его к нормальному пытаемся привести, то есть создать P(Z), который будет нормальный. Но само P(X) — оно вообще никакое, неизвестно, что это.

Евгений Александрович Волошин:
Эта дивергенция или расстояние Кульбака - Лейблера — это фактически то, насколько два вероятностных пространства похожи друг на друга.

Вадим Игоревич Лазарев:
Сравнение понятно. Чем, соответственно, они будут более похожи друг на друга, тем оно будет ближе к нулю.

Евгений Александрович Волошин:
Соответственно, эта метрика Дженсена – Шеннона. На самом деле, дивергенцию Кульбака - Лейблера нельзя назвать именно метрикой в полном понимании, потому что он не ограничен и не симметричен.

Вадим Игоревич Лазарев:
То есть если мы, например, здесь поменяем местами P(Z) и Q(Z|X), мы получим другую совершенно функцию, она по-другому будет выглядеть. И соответственно, мы получим уже другие значения.

Евгений Александрович Волошин:
Но это уже качественные показатели.

Вадим Игоревич Лазарев:
Это как раз один из вопросов, который я пытался выяснить, но так и не выяснил: почему в статье, где создавался этот вариационный автоэнкодер не использовали дивергенцию Дженсена - Шеннона? Разница в том, что тот же Дженсен - Шеннон скорее всего даже будет лучше в плане обучаемости нейросети, чем вот это.

Олег Вадимович Глухов:
Я думаю, здесь не принципиально.

Вадим Игоревич Лазарев:
То есть насколько наше значение здесь сходится условно с тем, что могло бы быть на входе здесь. Между вот этими двумя картинками. Если это проецировать на конкретное значение вероятности. Похоже чем-то нижнее на метод максимального правдоподобия.

Олег Вадимович Глухов:
Потому что там тоже апостериорная плотность. Наверное, это и есть решение для апостериорной плотности.

Вадим Игоревич Лазарев:
Я точно понял, что это сравнение. Точно такое же, как было в автоэнкодере. Того, что может быть на входе, и того, что будет на выходе.

Олег Вадимович Глухов:
В чём здесь принципиальное отличие от автоэнкодера?

Вадим Игоревич Лазарев:
Автоэнкодер — детерминистический, а это — вероятностный.

Олег Вадимович Глухов:
А с точки зрения результата на что это повлияет?

Вадим Игоревич Лазарев:
Разнообразие, более долгое обучение и, соответственно, другие направления, где его можно использовать. В общем, суть в чём: где мы его можем применять? Вот это уже точно генеративная нейросеть, которая будет генерировать что-то похожее, в каком-то пространстве, но аналогов, идентичных тем картинкам, которые будут создаваться, не будет. Где её применять? Соответственно, везде, где применяются та же диффузионные модели, GPT и так далее.

Олег Вадимович Глухов:
Интересно, кстати, был ли уже какой-нибудь иск, когда использовали модель генерации лиц и сделали реального человека или очень похожего? То есть технически это возможно всё-таки?

Вадим Игоревич Лазарев:
Здесь как раз есть один способ для детекции того, что было сгенерировано нейросетью. Это связано с кодированием JPEG. И по сути, если у нас есть изображение, сгенерированное тем же ChatGPT, то смысл следующий: если мы посмотрим на границу нашего изображения, то мы увидим там непостоянство границ. То есть в каком-то месте оно обязательно чуть-чуть будет куда-то выезжать, где-то будет похоже на способы сжатия JPEG. Такие прям отдельные квадраты, которыми сжимает JPEG, и мы их будем видеть в одном сегменте изображения, в другом их может и не быть. Потому что таким образом происходит генерация. И не забывайте, что здесь шум ещё есть.

Олег Вадимович Глухов:
Я имею в виду: сейчас появляются разные ассистенты или реклама, клеят это на баннер, где угодно. И вот представим себе, что всё-таки из этого скрытого пространства восстановили что-то.

Вадим Игоревич Лазарев:
Это очень низкая вероятность на самом деле, потому что даже если мы чуть-чуть сдвинемся, мы уже ту девятку не получим в этом пространстве. Если мы чуть-чуть наш вектор Z, который мы как раз генерируем из P(Z) чуть-чуть поменяем, то там на самом деле есть вероятность получить вообще не девятку, а какой-то непонятный набор пикселей. Это тоже может присутствовать, но с большей вероятностью мы всё-таки девятку получим. В общем, теперь непосредственно про GAN — генеративно-состязательные нейросети. Здесь подход уже совершенно другой используется. Но мы тоже имеем дело с вероятностными распределениями, которые в явном виде, как в автоэнкодере, не присутствуют в самой функции потерь, но, тем не менее, здесь на всякий случай прописывают то, что это всё-таки математическое ожидание, потому что мы проходимся по огромному количеству различных значений. Как он работает? У нас есть точно так же две нейросети. Как декодер и энкодер. Только декодер теперь у нас называется генератором, а энкодер, в свою очередь, является дискриминатором. То есть, каждый раз, когда мы получаем какой-то выкат здесь, мы отправляем результат, просчитанный для функции потерь, в дискриминатор и в генератор. И они одновременно всё время обучаются. Задача генератора — сделать функцию потерь как можно меньше. Это будет означать, что у нас, во-первых, наши данные неотличимы от реальных, к которым есть доступ у дискриминатора, он знает это распределение. Соответственно, это означает, что наш генератор сделал достаточно правдоподобный экземпляр. Задача дискриминатора —сделать эту

функцию потерь как можно больше, чтобы у нас одновременно и реальные значения здесь были, которые дискриминатор видит. И чтобы то, что генерирует ему генератор, он отбраковывал, говорил, что это фейк. То есть количество ответов должно быть правильным у дискриминатора.

Шамиль Алиевич Оцоков:
Эти слагаемые — какая из них ошибка дискриминатора, какая из них — генератора?

Вадим Игоревич Лазарев:
На самом деле и то, и то относится к дискриминатору и генератору. Это относится к тому, чтобы у нас дискриминатор реальное изображение принимал как реальное. Потому что к нему же на вход могут и реальные поступить, и он должен сказать: «Да, это реальное изображение». А изображения, которые были сгенерированы через генератор, отбраковывал. Поэтому здесь как раз обратное значение 1 минус D от того, что было на выходе генератора.

Шамиль Алиевич Оцоков:
То есть для него важно, чтобы второе слагаемое было большое. То есть ошибка была большая.

Вадим Игоревич Лазарев:
Чтобы как можно больше он отбраковывал от генератора и как можно больше он воспринимал реальное изображение как реальное. И здесь, на самом деле, у меня только один слайд даже. У GAN есть одна большая проблема: у нас, во-первых, здесь есть две нейросети, которые обучаются одновременно. На что это может повлиять? Из-за того, что у нас генератор должен уменьшать функцию ошибки, а дискриминатор, наоборот, стремится к тому, чтобы её увеличить, у нас получается, что обучение может резко стать нестабильным. В каком случае? Например, если генератор обнаружит такую комбинацию из нашего скрытого пространства, которая будет эффективно дурить дискриминатора. Ту самую купюру он найдёт, одну единственную, и каждый раз будет её подсовывать и ксерокопировать. К чему это приводит? Данные на выходе у нас будут совершенно одинаковыми и идентичными. Я ещё играл с генерацией лиц, и у меня та же самая ситуация произошла. Лица у большинства людей стали примерно одинаковыми с улыбками. Хотя по факту они до этого были все разные: крутые, счастливые и несчастливые.

Шамиль Алиевич Оцоков:
В OpenSpace есть кто-то? Это то же самое векторное пространство Z, которое и было указано ещё в автоэнкодере и в вариационном энкодере, просто оно здесь у меня по-другому обозначено.

Вадим Игоревич Лазарев:
Берём из нормального распределения.

Олег Вадимович Глухов:
Да.

Вадим Игоревич Лазарев:
То есть из нормального распределения будут браться какие-то случайные вектора, подаваться на генератор. Здесь нет у нас опять же задачи классификации, сделать

какие-то контекстно правдоподобные изображения, например, лица только женщин, лица только мужчин. Здесь просто генерация чего-то случайного из нужного нам набора реальных изображений.

Олег Вадимович Глухов:
Например, для этого вариативно только дерево, которое будет от латентного пространства получать и использовать.

Вадим Игоревич Лазарев:
Да, случайно генерируем что-то.

Олег Вадимович Глухов:
Нет, мы всё-таки сначала его создали в латентном пространстве, потом уже в этой задаче. Это просто нормальное распределение, мы его не создаём никак.

Вадим Игоревич Лазарев:
Это вообще случайный вектор.

Шамиль Алиевич Оцоков:
Да. Если у нас два плохих генератора и дискриминатора, и два хороших генератора и дискриминатора, тогда у них одинаковое качество GAN будет, потому что тот плохо находит реальное изображение, а этот плохо генерирует. И во втором случае первое очень хорошо находит, а второй очень хорошо генерирует.

Вадим Игоревич Лазарев:
Получается, что у нас не может получиться такого, что дискриминатор будет плохим, потому что у нас постоянно идёт обучение как его, так и его.

Шамиль Алиевич Оцоков:
Может быть, мы там прервали такую слабую модель дискриминатора.

Олег Вадимович Глухов:
В принципе, как будто бы дискриминатор раньше обучен.

Вадим Игоревич Лазарев:
Это правда, потому что с этой проблемой сталкиваются. Дискриминатор может резко стать суперкрутым, а генератор не сможет его догнать и не будет понимать, что он делает не так, уже потеряет этот момент. Либо наоборот: генератор крутым станет, дискриминатор, соответственно, станет хуже. Это породит у нас просто генерацию каких-то картинок, которые слабо напоминают реальные изображения.

Евгений Александрович Волошин:
У меня вопрос о дискриминаторах. Есть ли проблема холодного старта? То есть, генератор — он что-то генерировал и получилось вообще непохоже.

Вадим Игоревич Лазарев:
Там изначально будет непохоже.

Евгений Александрович Волошин:
Но дискриминатор, он же тоже чистый, ещё не обученный. Он не обучился. Я ему даю реальное изображение, неизвестное.

Вадим Игоревич Лазарев:
Мы можем его предобучить.

Евгений Александрович Волошин:
Это вариант.

Вадим Игоревич Лазарев:
Но у тебя всё равно внутри функции потерь присутствует.

Олег Вадимович Глухов:
Всё равно есть какая-то обратная связь о том, что правильно, а что неправильно. Вот реальное изображение — это то, что генерируется.

Павел Юрьевич Анучин:
Как оно тогда стартует? В какой-то момент, если оно стартанёт неправильно изначально, это же проблема GAN.

Олег Вадимович Глухов:
Каким образом происходит обратная связь? Она вообще случайна. То есть там нельзя даже сказать: правильно он это сделал или неправильно.

Павел Юрьевич Анучин:
Да, то есть пока не протестируешь решение, которое выделил, уже непонятно.

Вадим Игоревич Лазарев:
Смотрите, мы же знаем, что тут реальные изображения. Мы знаем про блок реальных изображений. Если реальное изображение пришло сюда, и при этом он выдал то, что это фейк, — мы знаем, что он не прав. Мы можем ему на это указать.

Олег Вадимович Глухов:
Система классификации.

Вадим Игоревич Лазарев:
Она просто в неявном виде.

Павел Юрьевич Анучин:
Так она всё-таки есть.

Вадим Игоревич Лазарев:
Я думал, что вы имеете в виду разметку реальных данных и нереальных данных.

Олег Вадимович Глухов:
Хотя бы метку дать, что это реальное изображение.

Вадим Игоревич Лазарев:
Да, это есть. Это не в виде метки, опять же.

Павел Юрьевич Анучин:
Но оно же всё равно на обучении никак не скажется. Даже если он сопоставит и ответит неправильно, это никак не отразится.

Вадим Игоревич Лазарев:
У нас это значение опустится ниже. Задача дискриминатора — сделать функцию потерь как можно больше. Он увидит то, что у него тут ноль условно, он поймёт: я делаю не так. Здесь те же самые задачи решает он, которые решает генеративный ИИ: синтетических данных, улучшение изображения, обнаружение новых каких-то аномалий. Ну, опять же, скажу сразу: у нас отдельный дискриминатор и отдельный генератор могут использоваться. Мы можем разъединить условно, если это предусмотрено архитектурой в программном смысле.

Константин Евгеньевич Бошков:
Для генерации синтетических данных всё-таки какой тип лучше выбрать?

Шамиль Алиевич Оцоков:
Диффузионные.

Вадим Игоревич Лазарев:
Диффузионные модели, очевидно. Потому что они генерируют, во-первых, вариативные изображения, они будут разные, во-вторых, высокого разрешения. В-третьих, они с большей вероятностью придут к тому, что мы у неё запросили, чем если бы мы это делали через вариационный автоэнкодер или же тот же GAN. У нас здесь сразу стоит выделить два направления работы с ними. Это прямой и обратный процессы. В прямом процессе мы зашумляем наше изображение, добавляя каждый раз в него чуть-чуть шума. Таких шагов может быть несколько тысяч. Задача нейросети — убрать этот шум, то есть узнать, сколько его, как он распределён, и, соответственно, выдав, сколько там у нас шума, уже через отдельные программные методы мы этот шум убираем. То есть создаём на основе того распределения, которое нам говорит нейросеть — сколько тут шума и какой он, — вычитаем эти значения, получаем картинку, которая становится менее шумной. Что касается математики, которая здесь присутствует - вероятности повсюду. У нас есть вероятность на распределение нашего шума и на распределение на каждом этапе для того шума, который у нас был добавлен, и того шума, который накоплен. Задача нашей нейросети — сделать это распределение, но не для конкретного каждого промежутка времени и не для конкретной итерации, а для всех.

Олег Вадимович Глухов:
Диффузионная модель называется диффузионной, потому что цепь Маркова используется?

Вадим Игоревич Лазарев:
Здесь на самом деле присутствует название потому, что человек, который создал диффузионные модели, изначально занимался динамикой жидкости, газа. И как-то получилось то, что он привнёс какие-то идеи и в генеративные. Диффузия здесь имеется в виду броуновское движение, связанное с тем, что у нас добавляется шум.

Олег Вадимович Глухов:
У нас в теории фильтрации тоже есть диффузионные марковские процессы. Когда получаем винеровский шум за счёт того, что белый шум подсовываем на интегратор, интегрируем — получаем винеровский процесс. Это, соответственно, то же самое, броуновское движение.

Вадим Игоревич Лазарев:
Тут тоже броуновское движение, только пиксели. У нас нейросеть создаёт пространство изображений, в котором она работает. И на основе этих маленьких шажочков, которые предпринимаются нами уже непосредственно на основе обратной связи от нейросети, мы постепенно убираем шум из изображения. Делается опять же это маленькими шагами, потому что если бы мы сразу посчитали весь шум, который потенциально может быть в этом изображении, мы рисковали бы попасть куда-то совершенно не туда, не в то распределение.

Олег Вадимович Глухов:
Как идёт цепочка?

Вадим Игоревич Лазарев:
Вот это уже обратный процесс.

Олег Вадимович Глухов:
Это обратный представлен. Просто тут по поводу этих вероятностей.

Вадим Игоревич Лазарев:
Это связывающее как раз то, что было до добавления шума и после.

Олег Вадимович Глухов:
То есть Q известен?

Вадим Игоревич Лазарев:
Да, мы его знаем. Нам надо узнать P – это тета. Всё, что снизу, — это истина. А сверху мы аппроксимировали.

Олег Вадимович Глухов:
Это более простая модель.

Вадим Игоревич Лазарев:
Да, тоже нормальное распределение. Только у него там ещё один параметр добавляется — сколько шума добавлено. Тета — это сами параметры нейросети — весовые коэффициенты. Здесь стоит отметить то, что изображение получается достаточно высокого разрешения, потому что мы сразу работаем с нужным нам разрешением. И с нужного нам разрешения убираем шум. Но здесь присутствуют и дополнительные варианты: как работа с более сжатыми изображениями, а затем тоже через диффузионную модель мы можем интерполировать в более высокое разрешение.

Олег Вадимович Глухов:
X это левая часть? Что является исходником? То, что снизу?

Вадим Игоревич Лазарев:
Нет. Это как раз в зависимости от количества шума, который у меня есть.

Олег Вадимович Глухов:
Уже натренированные модели.

Вадим Игоревич Лазарев:
Да. Я пытаюсь получить цифры.

Олег Вадимович Глухов:
А насколько вся цифра, которую я могу предъявить, соответствует действительности - верхней грани?

Вадим Игоревич Лазарев:
Мы можем посмотреть вот здесь.

Олег Вадимович Глухов:
Только это сильно зашумленное опять же.

Вадим Игоревич Лазарев:
В заключении хочу ещё раз отметить ключевые особенности всех трёх моделей. Если мы хотим получать высококачественные изображения, при этом разные, — нам нужно использовать диффузионные модели. Хотим получать высокое качество, но при этом быстро их генерировать — нам надо GAN как-то обучить, чтобы они это всё хорошо генерировали. Потому что GAN, опять же, у нас работает с высоким разрешением. И если хотим быстрое создание, при этом чтобы были они разные, мы будем вводить, потому что там есть сжатие.

Олег Вадимович Глухов:
Там есть выбор определённого типа шума, который используется. Если стоит достаточно тривиальная задача, тогда можно чем-то одним воспользоваться.

Вадим Игоревич Лазарев:
Если нет, то опять же то же видео, например, генерировать. Тогда уже используется Diffusion Transformer. Там огромная, гигантская архитектура и последовательность действий. Текст есть какой-то, его надо эмбеддингом закодировать, потом добавить в диффузионную модель, которая, в свою очередь, несколько раз вставлена в трансформер, который должен в каждом кадре определять ключевые элементы, чтобы у человека третья рука не появилась или пятая нога.

Олег Вадимович Глухов:
Да. Трансформеры тоже встречаются периодически, потому что в процессе обучения тех диффузионных моделей, которые есть, получают, условно говоря, задавая теги, изображения на английском языке, которые в процессе обучения подсунули. Причём эти теги как-то были размечены. И потом для того, чтобы преобразовать тот запрос, который сделал человек, в понятные диффузионной модели теги, которые она знает, отдельно используется трансформер.

Вадим Игоревич Лазарев:
И сможет направить генерацию диффузионной модели в нужную сторону, а дальше уже трансформером поддерживать, чтобы кадры были синхронизированы и шли к тому, что мы хотим увидеть.

Олег Вадимович Глухов:
Из опыта использования этих моделей локальной генерации при работе с диффузионными моделями, когда модель уже обученная, мы можем использовать разные виды шума. Добавление шума на каждом этапе — мы просто по шагам это делаем, чтобы получить разнообразие. Это пока постановка, так сказать, сцены. Вид и интенсивность шума на этапе обучения — она никак не зависит от того, какой шум мы используем на этапе генерации для того, чтобы это изменять. Мы же в латентное пространство добавляем шум. Или мы используем этот шум для того, чтобы из латентного пространства влиять на распределение - брать значение. Как именно эта часть работает? Почему здесь эти шумы совершенно независимы друг от друга?

Вадим Игоревич Лазарев:
Про шумы, которые при генерации используются, я вот тут точно сказать не могу, потому что я помню только один момент: он используется для того, чтобы у нас не были идентичные изображения, например, или чтобы разнообразить как-то картинку. Потому что, когда мы пытаемся на основе того шума, который был добавлен, пройти по этим траекториям, вернуться к пикселям, которые на самом деле там были, у нас может произойти свёртывание в одну точку. У нас была какая-то там спираль нарисована, и чтобы у нас всё не схлопнулось в какую-то одну центральную точку, что является по сути чем-то средневзвешенным всех пикселей, мы добавляем шум, чтобы растолкать все эти точки, чтобы они остановились на = других местах нашего изображения. Что касается разницы между шумом, который добавляем, и который у нас используется при генерации, здесь я видел только про нормальное распределение, то есть обычный гауссов шум, который добавляется. Просто мы его ещё дополнительно параметризуем, чтобы подсчитать, сколько мы шума добавили. Но это нужно исключительно для обучения.

Олег Вадимович Глухов:
Вопрос был в том, что скрытое пространство — оно всё сходится. Есть разные вариации использования, когда мы изменяем только часть изображения, остальное не зашумляем. Почему мы можем использовать разные виды шумов при генерации, не тот, на котором мы обучали, а какие-то другие шумы? Тут всё сходится всегда к нормальному, потому что там есть аппроксимационные формулы, которые как раз-таки из любого распределения могут нормальное получить. У нас размер входного изображения и размер выходного изображения — он был одинаковый. Можем ли мы в случае с автоэнкодером увеличить количество выходных слоёв и получить изображение большего размера?

Вадим Игоревич Лазарев:
Сравнивать не сможем.

Олег Вадимович Глухов:
Можно какие-то программные прокладки сделать.

Вадим Игоревич Лазарев:
Можно, конечно. Мы можем просить его генерировать в более высоком разрешении, но сравнивать мы всё равно будем два сжатых изображения. Но я боюсь, что любую функцию потери надо немного поменять.

Олег Вадимович Глухов:
В случае с диффузионными моделями – там размерность изображения достаточно гибко настраивается: там и квадратный, и не квадратный.

Вадим Игоревич Лазарев:
В диффузионных моделях его в принципе нет. У него есть просто одно единственное пространство, в котором расположены все шумы.

Олег Вадимович Глухов:
И это такие отдельные полки.

Вадим Игоревич Лазарев:
Они просто проходят траектории и переходят к нужному вероятностному распределению, если это, опять же, условная генерация. То есть с каким-то промптом.

Шамиль Алиевич Оцоков:
Вот такая задача: анонимизация изображения лиц людей. То есть я, допустим, не хочу своё изображение в социальных сетях выставлять. Хочу сделать изображение, сгенерировать похожее на меня, но с какими-то отклонениями. Можно ли такую задачу решить и какую модель использовать?

Вадим Игоревич Лазарев:
Во-первых, здесь надо сохранить черты лица. Нам надо определить, где лицо расположено, прямоугольник, например, нарисовать. И как раз на этом месте нам надо получить сумму. Из этого шума уже просто генерация лица на основе диффузионного трансформера, потому что нам надо учитывать пропорции тела человека, чтобы там огромное лицо не получилось, чтобы у нас был контекст нашего изображения. То есть тут, опять же, две задачи: детекшн и генерация изображений на основе диффузионной модели в том квадрате, который мы выбрали.

Шамиль Алиевич Оцоков:
Это диффузионный трансформер.

Вадим Игоревич Лазарев:
Да. Или вариационный автокодировщик. Вариационный автокодировщик плохо подойдёт, потому что изображения разные бывают, разного качества и разрешения. Поэтому проще всего диффузионную модель использовать. Сейчас мощностей хватает для того, чтобы ими воспользоваться, как теми же самыми трансформерами.

Шамиль Алиевич Оцоков:
То есть я могу задавать своё фото и задавать меру сходства, если я хочу черты лица сохранить?

Вадим Игоревич Лазарев:
Тогда точно Diffusion Transformer нужно. Потому что мы должны выделить какие-то ключевые аспекты вашего лица, чтобы он их запомнил: это брови, это глаза и так далее. А затем мы уже будем по отдельности: отдельно брови, отдельно глаза.

Олег Вадимович Глухов:
Есть готовые инструменты для того, чтобы осуществлять object detection на сгенерированном или на фотографии. Частая проблема — генерация рук и лица, мелких деталей. И там есть уже готовые узлы, которые детектируют руку, причём попиксельно, то есть не рот, а квадрат. Потом делается размытие этой части. И потом возвращается на этап генерации или заново генерируется в этом месте, но только не с нулевого шага начинается, когда полностью шум, а начинается с пятого-шестого шага работы определённой модели и на основе размытого изображения, условно зашумленного, добавляя шум туда, генерируется заново какая-то часть. И уже она проверяется на соответствие, насколько адекватно получилось. Надо, чтобы они ещё друг с другом были плавно согласованы. А вот это размытие как раз и даёт вот эту плавность, что он совмещается с другими частями изображения. А уже степень размытия и на какой шаг в генерации откатываемся — она позволяет как раз изменять, насколько количество фич будет сохранено, а сколько нет. В принципе, такие вещи и делаются. Что касается темпоральных данных - там, где у нас ещё и время завязано. Я видел, что диффузионные модели в том числе используются для синтеза музыки. На новогодних праздниках вышла модель для синтеза анимаций: промпт пишется и синтезируется анимация движения персонажа — танцы, сел, встал, лёг. Генерация 3D-изображений. Но там не полностью сразу генерится 3D. Это генерится набор кадров с разных сторон. И уже с точки зрения фотограмметрии генерится. Но вот сам факт наличия возможности генерации темпорального типа — музыки или анимации — вот как это сделать с точки зрения представления данных? То есть тоже нужен изначально какой-то датасет?

Вадим Игоревич Лазарев:
Опять трансформер. Потому что идея трансформера изначально, чтобы он последовательность смог запомнить. Чтобы у нас приходили какие-то данные, он из каждых вычленял какие-то ключевые значения, но при этом помнил, что у него до этого ещё было. Его архитектура именно для этого и создана. То есть каждый раз с выхода своего он получает на один свой вход соответственно то, что он сгенерировал, с другого входа — новые данные. И каждый раз на основе всего того, что он перед этим видел, он генерирует опять что-то новое уже.

Олег Вадимович Глухов:
То есть в случае с генерацией видео у нас есть распределение вероятностей, которое мы тренировали один раз, и мы должны сделать какой-то набор из ключевых кадров разных распределений вероятностей для этих видео и как-то их совмещать. Мы же не можем один и тот же использовать?

Вадим Игоревич Лазарев:
Тут, к сожалению, не могу точно сказать, потому что я ещё немножко в этой теме плаваю.

Олег Вадимович Глухов:
Я просто видел, что эти модели сильно по размеру отличаются от моделей для генерации изображений. Применение диффузионных моделей для синтеза текста. Мне попалась небольшая демонстрация работы такой модели для синтеза кода. Выглядит очень интересно, когда промпт написали на Python, и он генерирует сразу весь блок кода. Потом раз — где-то синтаксис больше на питоновский похож и там три-пять шагов, и quicksort получился уже.

Вадим Игоревич Лазарев:
Но в середине это просто какой-то набор букв, потому что он постепенно шум удаляет из того текста, который изначально был сгенерирован. Как он генерирует шум? У нас есть зашумленный текст — это какие-то случайные буквы. Понятно, как это сделать с числами. Понятно даже, как это с эмбеддингами сделать, потому что там просто вектор- значение. Мы можем зашуметь его, и он будет направлен в другую сторону.

Олег Вадимович Глухов:
Обучение таких моделей занимает достаточно большое количество времени. Выпускаются такие базовые модели. Вопрос файн-тюнинга - прививания этой модели какой-то дополнительной информации решается с помощью low-rank адаптеров и ещё так далее. С точки зрения обучения из трёх типов технологий — какие из них легче дообучать под свою конкретную задачу?

Вадим Игоревич Лазарев:
Мне кажется, что дообучить легче GAN. Потому что у нас есть блок реальных изображений, мы его докидываем и заново прогоняем через новый датасет с учётом того, что у нас уже было. В случае диффузионных моделей, это, мне кажется, надо устраивать распределение. Потому что тут мы не можем просто взять коэффициенты, которые были бы согласованы с тем, что мы обучали. Ну а про вариационный энкодер — тут ещё более грустная ситуация, потому что тут придётся переобучать его. Я не знаю, как модели могут дополнить своё внутреннее вероятностное распределение, которое они выучили, с учётом новых данных. Тут вопрос — насколько это хорошо будет сделано.

Олег Вадимович Глухов:
VAE или AE — маленькие модели.

Вадим Игоревич Лазарев:
Да, поэтому они такие фигурненькие.

Олег Вадимович Глухов:
Может, есть смысл их задать?

Вадим Игоревич Лазарев:
Диффузионные модели достаточно обширные и мощные, и ситуация с их дообучением, мне кажется, что там изначально фреймворк используется, который предполагает её дообучение. Возможно, она какие-нибудь ключевые моменты может запоминать, и мы их можем дополнять. Но именно, если у нас есть обученная диффузионная модель, как её дообучить — это вопрос совершенно другой.

Олег Вадимович Глухов:
Допустим, использование диффузионных моделей позволяет взять первые три шага генерации, сделать на одной модели, потом это же изображение взять и остальные шаги делать на другой модели и так далее.

Вадим Игоревич Лазарев:
Им без разницы: они с любого шага могут продолжать. Они просто определяют, сколько тут шума и сколько шума мы можем предположить, сколько можно вычесть, чтобы продвинуться чуть дальше.

Олег Вадимович Глухов:
Как определяется, сколько шума? Есть параметры? Как раз мы параметризовали это распределение. Помимо этой теты, которая сидит внутри нейросети, она в себя включает ещё и дополнительный коэффициент в нормальном распределении, который говорит, сколько здесь шума.

Вадим Игоревич Лазарев:
То есть там - как матрица весов, так и плюс какой-то параметр?

Олег Вадимович Глухов:
Внутри этой матрицы весов он уже присутствует, и мы обыгрываем её.

Вадим Игоревич Лазарев:
Я понял, модель.

Олег Вадимович Глухов:
Спасибо за содержательный доклад. Напрашивается следующая тема — про трансформеры. Мы подумаем насчёт доклада. Спасибо.