

Трансформеры (II): CLIP, DALL-E, DDPM, etc...

Как использовать?

Большой список доступных моделей и удобный интерфейс - библиотека
HuggingFace Transformers 

- GPT-3.5 - хорошая генерация текста
- BERT, RoBERTa - классификация, использование в качестве эмбеддингов
- T5, BART - seq2seq задачи

Vision Transformer

Идея: применить архитектуру Transformer для CV (image classification)

- нужна последовательность на вход

Как из картинки сделать последовательность?

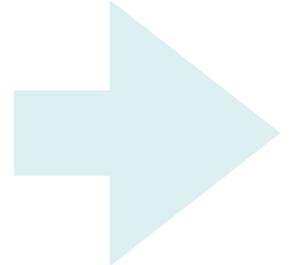


[Image credit](#)

Vision Transformer

Как из картинки сделать последовательность?

- разделим на 2D патчи



Vision Transformer

Как из картинки сделать последовательность?

- разделим на 2D патчи
- вытянем в последовательность 2D картинок



Разделение изображения на 2D патчи — это распространённая операция в глубоких нейронных сетях, особенно для задач, связанных с обработкой изображений, таких как сегментация или трансформеры для изображений (например, Vision Transformers, ViTs).

Предположим, у вас есть изображение размера $H \times W \times H \times W$ (где H — высота, а W — ширина изображения), и вы хотите разделить его на патчи размером $P \times P \times P$:

Размер изображения: $H \times W \times C \times H \times W \times C$, где C — количество каналов (например, 3 для RGB).

Размер патча: $P \times P \times P$.

Количество патчей:

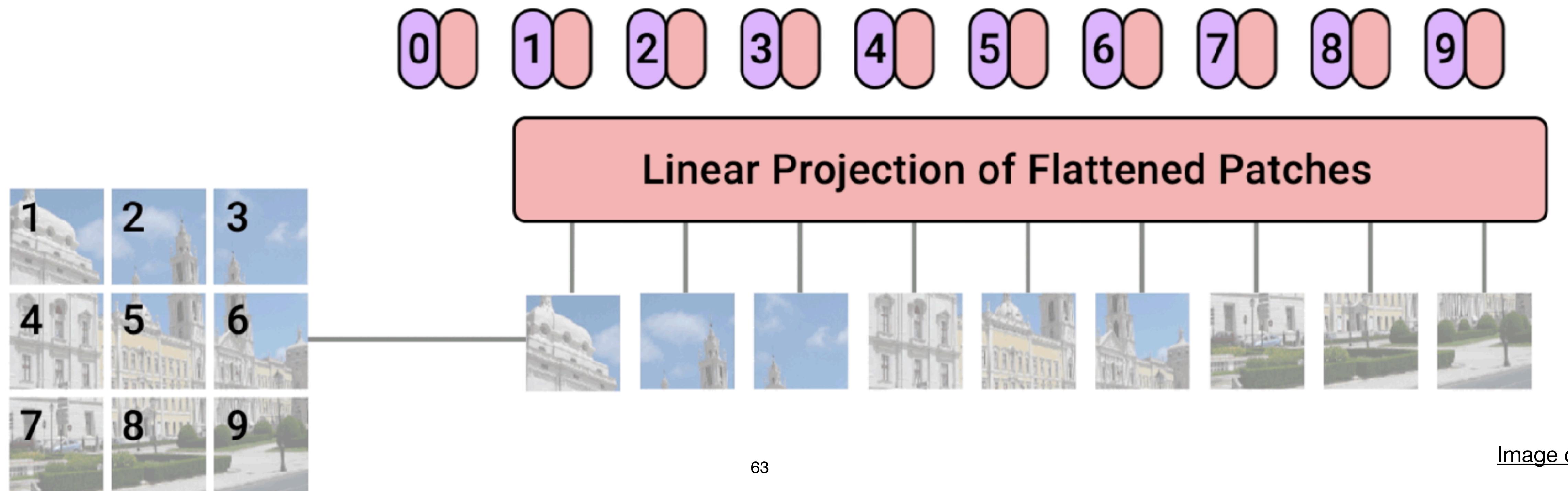
$$N = H \times W \times C$$

$N = P \times P \times P$ При этом важно, чтобы H и W делились на P без остатка.

Vision Transformer

Как из картинки сделать последовательность?

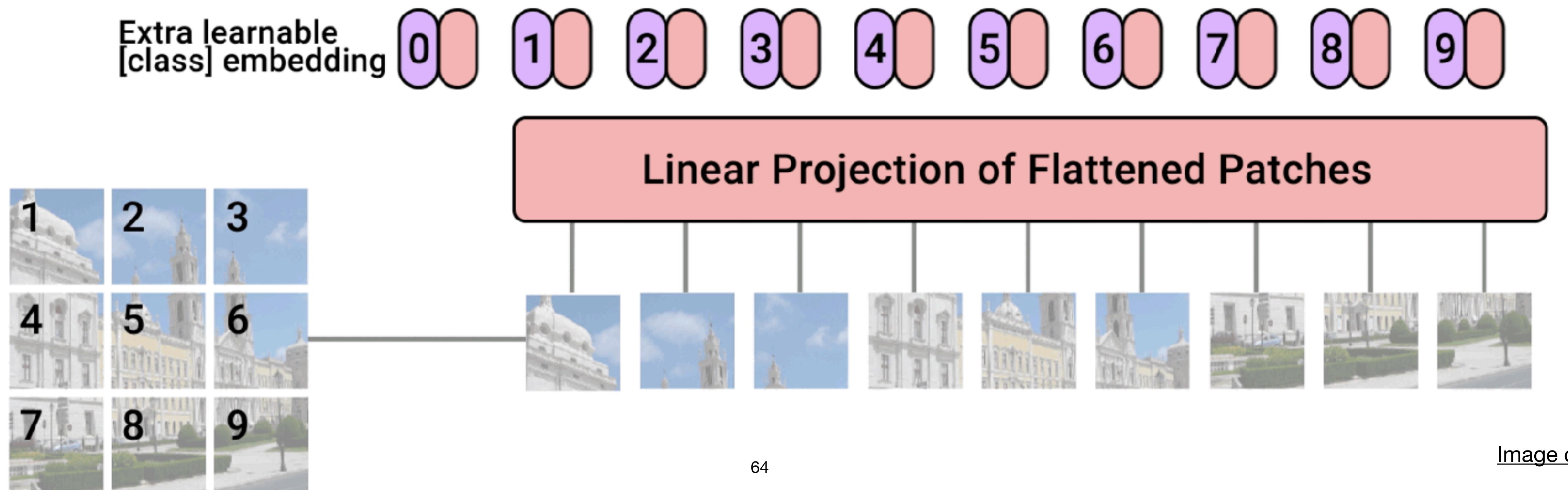
- разделим на 2D патчи
- вытянем в последовательность 2D картинок
- Linear mapping - в меньшую размерность



Vision Transformer

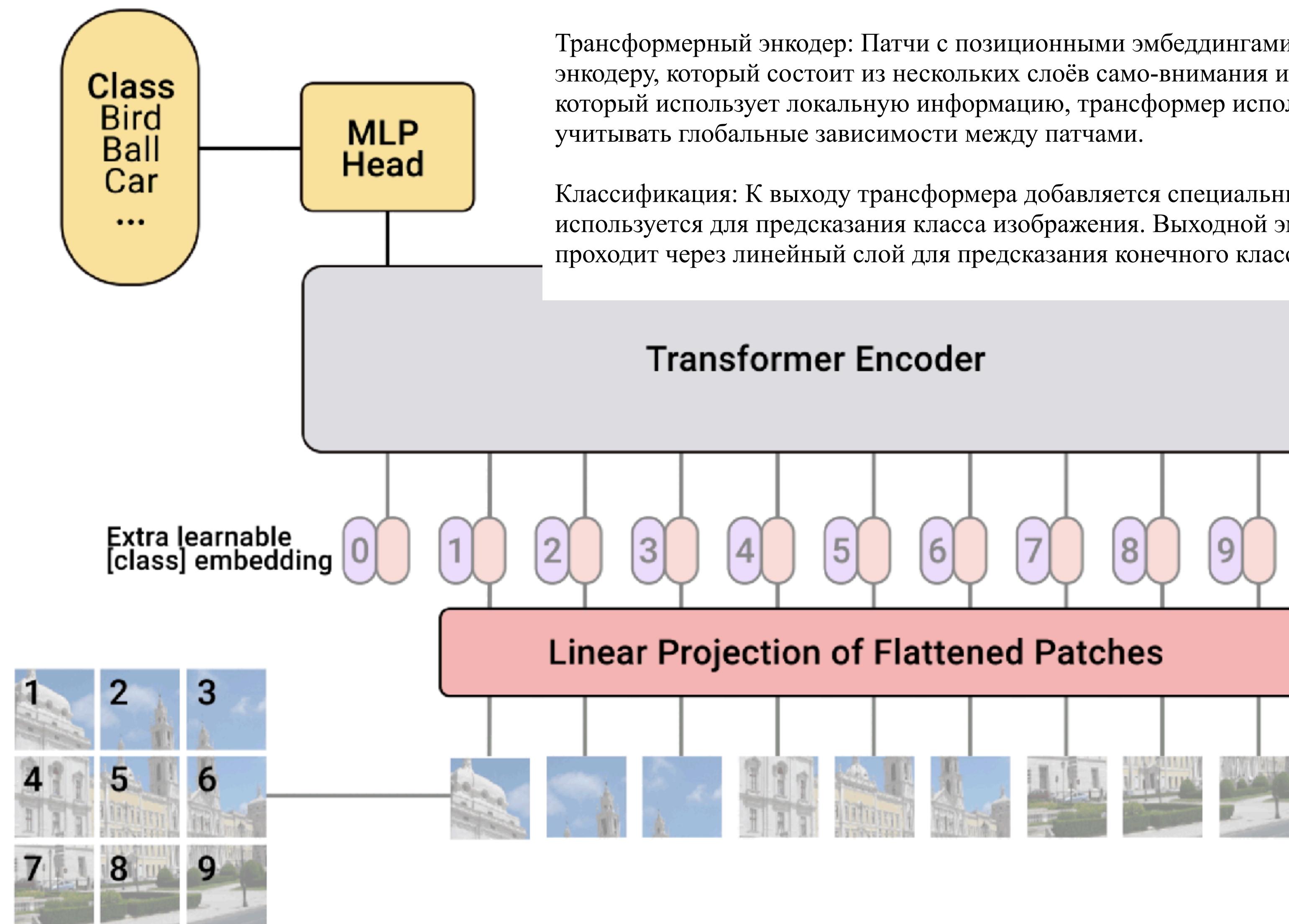
Как из картинки сделать последовательность?

- разделим на 2D патчи
- вытянем в последовательность 2D картинок
- Linear mapping - в меньшую размерность



Преобразование патчей в векторы: Каждый патч разворачивается в одномерный вектор (flatten), что приводит к вектору размером $P^2 \times C P^2 \times C$. Эти векторы затем передаются через линейный слой (проекция), чтобы получить эмбеддинги фиксированного размера DD.

Vision Transformer



Добавление позиционной информации: Поскольку трансформеры не учитывают пространственную информацию, каждому патчу добавляются позиционные эмбеддинги, которые позволяют модели понимать положение патчей в изображении.

Трансформерный энкодер: Патчи с позиционными эмбеддингами подаются на вход трансформерному энкодеру, который состоит из нескольких слоёв само-внимания и нормализаций. В отличие от CNN, который использует локальную информацию, трансформер использует механизм внимания, чтобы учитывать глобальные зависимости между патчами.

Классификация: К выходу трансформера добавляется специальный токен класса ([CLS]), который используется для предсказания класса изображения. Выходной эмбеддинг токена [CLS] затем проходит через линейный слой для предсказания конечного класса.

Vision Transformer

	ViT-H	Previous SOTA
ImageNet	88.55	88.5
ImageNet-ReaL	90.72	90.55
Cifar-10	99.50	99.37
Cifar-100	94.55	93.51
Pets	97.56	96.62
Flowers	99.68	99.63

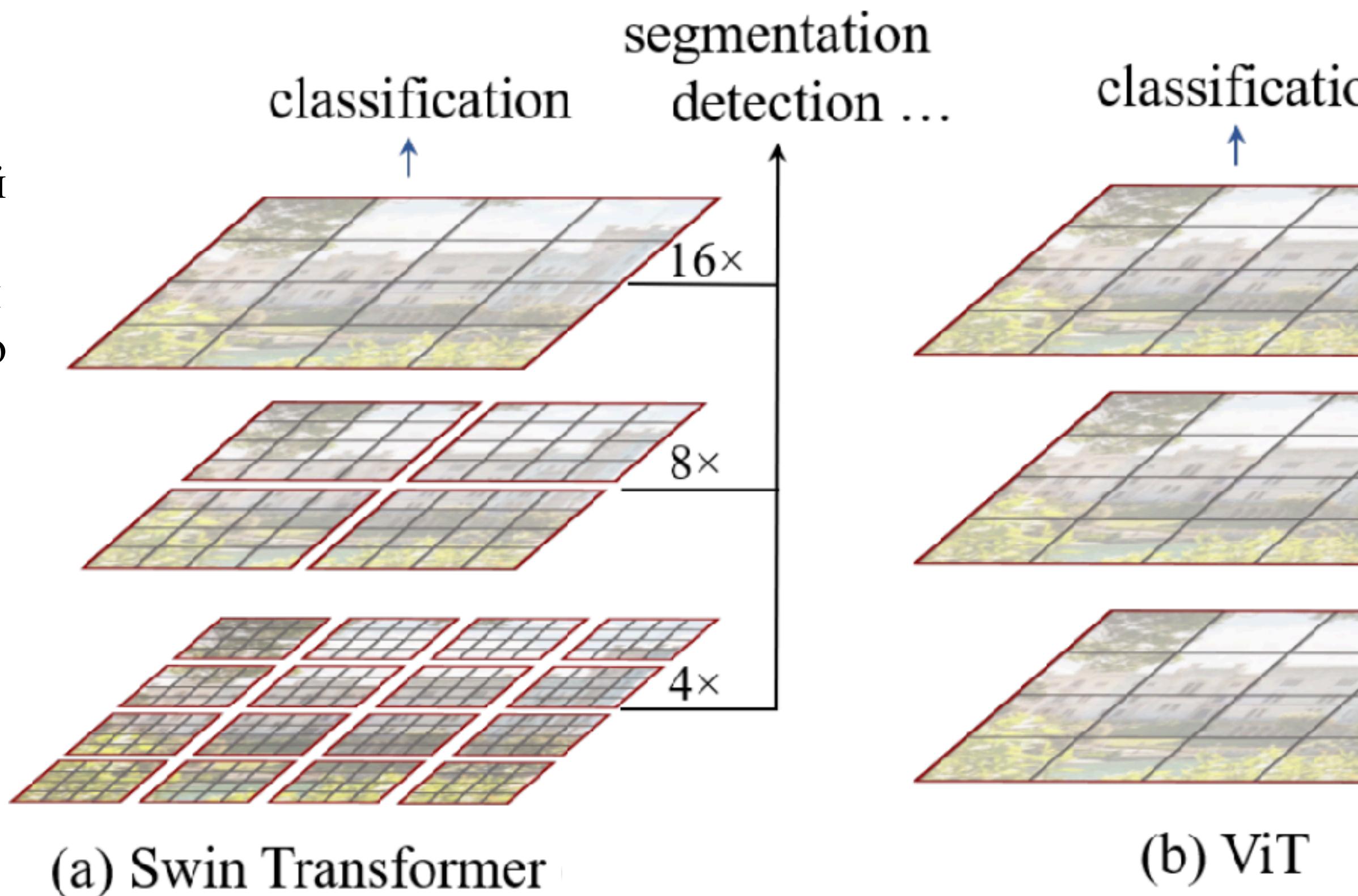


[Image credit](#)

Swin Transformer

Для других задач CV (сегментация, детекторы) фиксированный патчи работают плохо - это исправляет Swin Transformer

Основная идея Swin Transformer — это использование свёрточной структуры, которая оперирует с переменными уровнями разрешения, что делает его более похожим на сверточные нейронные сети (CNN) с учетом глобального контекста благодаря механизмам внимания.



Архитектура Swin Transformer:

Patch Partition: Изображение делится на небольшие патчи, например, 4x4 пикселя.

Linear Embedding: Каждый патч разворачивается и преобразуется в вектор фиксированного размера с помощью линейного слоя.

Swin Transformer блоки:

W-MSA (Window-based Multi-Head Self-Attention): Механизм внимания, который работает в пределах фиксированных окон.

SW-MSA (Shifted Window Multi-Head Self-Attention): Сдвиг окон на половину их размера между блоками для захвата глобального контекста.

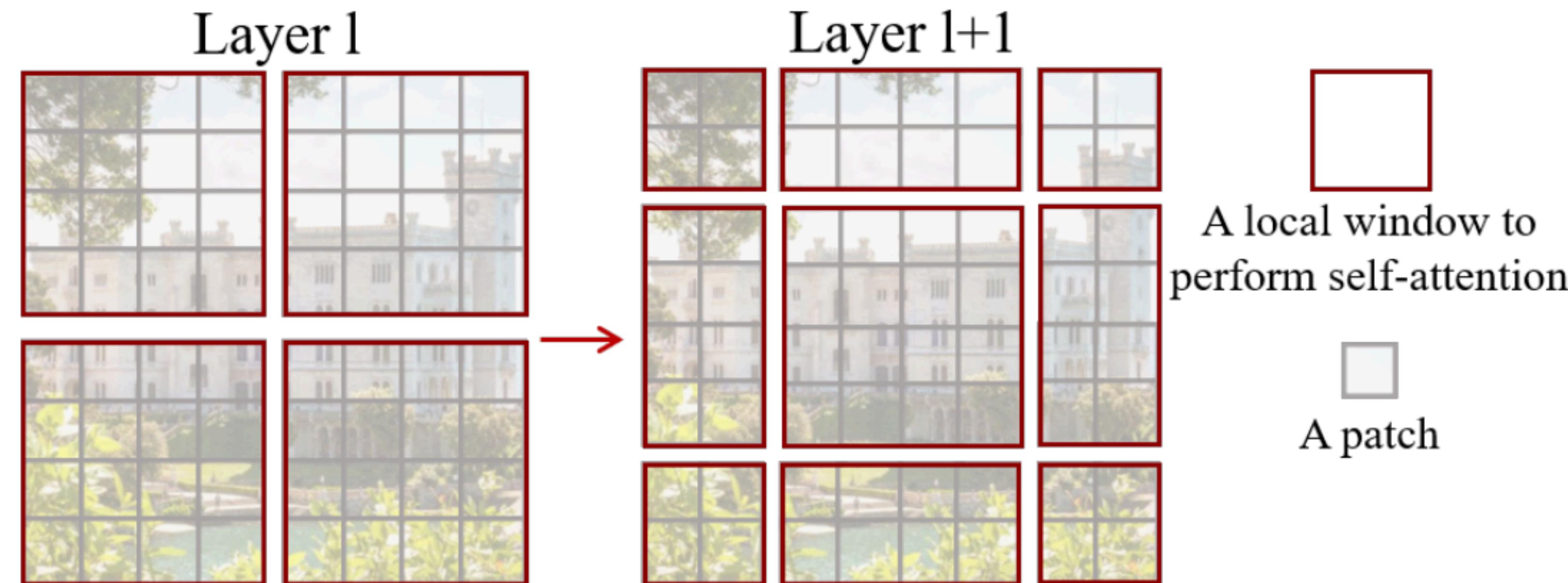
Patch Merging: Слияние патчей, чтобы уменьшить пространственное разрешение и увеличить количество каналов (аналогично операции пулинга в CNN).

Head: Полносвязная классификационная голова для предсказания класса.

[Image credit](#)

Swin Transformer

Shifted Window: self-attention между блоками патчей,
сдвигаем окна на следующем слое



CLIP: Contrastive Language– Image Pre-training

CLIP (Contrastive Language–Image Pre-training)

BERT-like модели предобучаются на неразмеченных данных (тексты)

Image classifiers учат на размеченных людьми данных (ImageNet, etc.)

Основные идеи CLIP:

- Контрастное обучение (Contrastive Learning): Модель обучается на парах "изображение-текст". Позитивные пары — это соответствующие друг другу изображение и описание, негативные пары — случайные комбинации изображения и текста. Цель обучения — максимизировать сходство между эмбеддингами изображений и текстов для позитивных пар и минимизировать для негативных. Это достигается с помощью контрастной функции потерь (contrastive loss), например, cross-entropy loss для сходства между парами.
- Два отдельных энкодера: Image Encoder: Для обработки изображений используется CNN или Vision Transformer, который извлекает эмбеддинг изображения. Text Encoder: Для обработки текста применяется трансформер, который извлекает эмбеддинг текста. Оба энкодера обучаются одновременно, чтобы эмбеддинги изображения и текста соответствовали друг другу.
- Обучение на огромных объемах данных: CLIP обучен на массивном датасете, состоящем из изображений и их текстовых описаний, взятых из интернета. Это позволяет модели развивать понимание объектов и их связей, не требуя явной аннотации.

CLIP (Contrastive Language–Image Pre-training)

- Собрали 400 миллионов пар (картишка, подпись) - из интернета

**a train traveling down a track
next to a forest.**



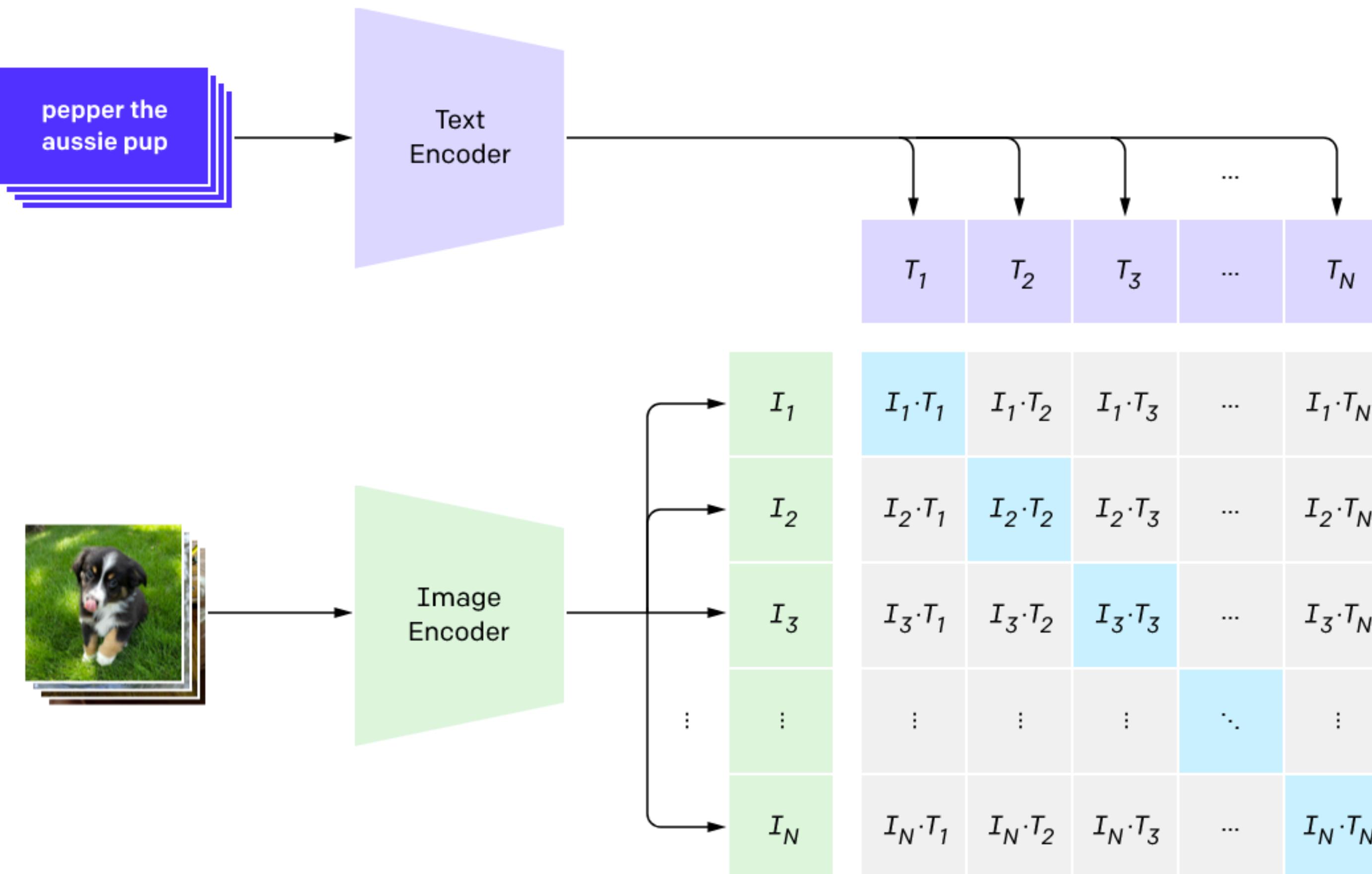
**a group of young boys playing
soccer on a field.**



© WALTHER.SIKSMA.NL

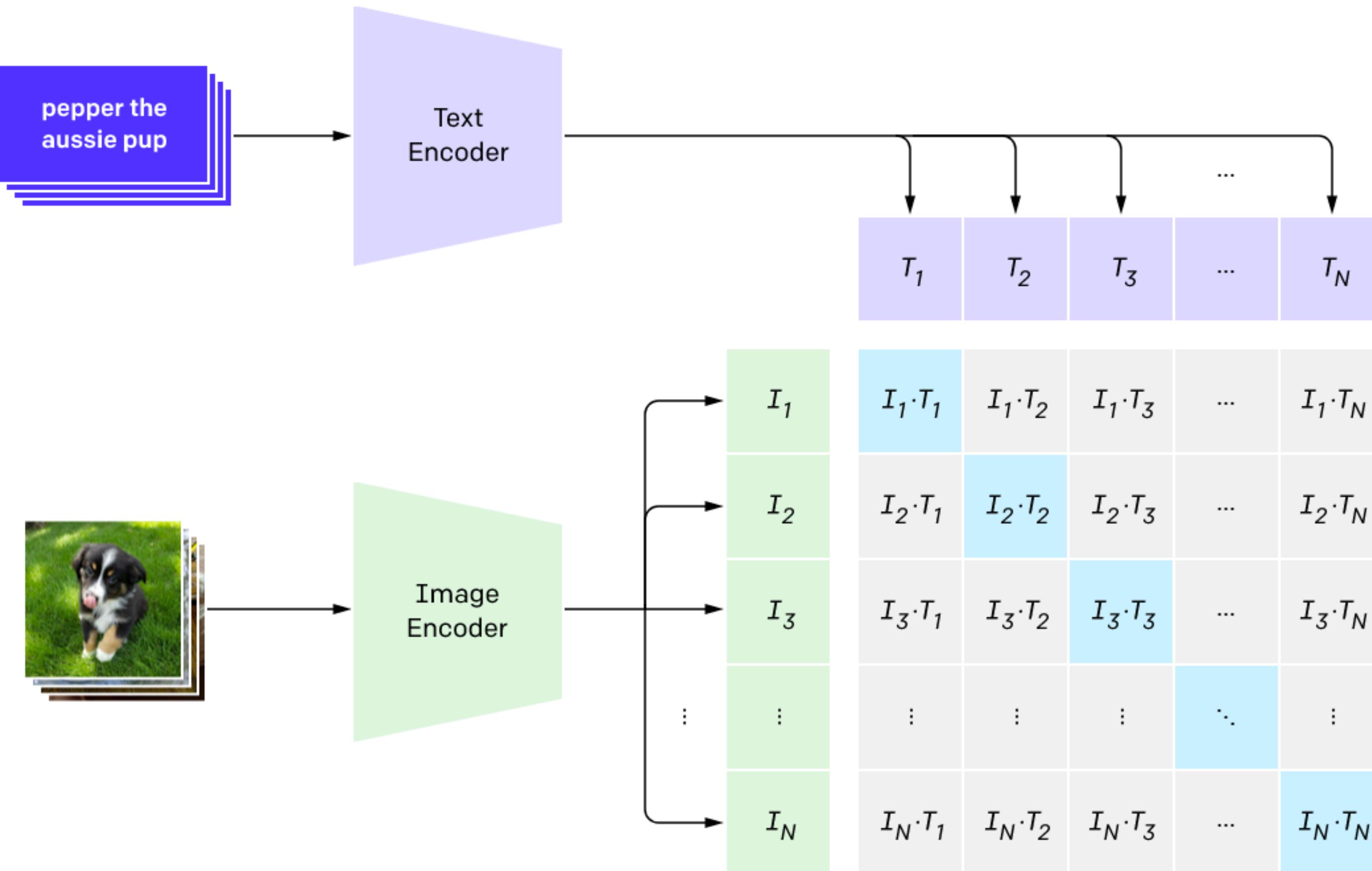
CLIP (Contrastive Language–Image Pre-training)

- Предобучение: энкодим N картинок (ViT) и N подписей к ним (Transformer)



CLIP (Contrastive Language–Image Pre-training)

- Предобучение: энкодим N картинок (ViT) и N подписей к ним (Transformer)

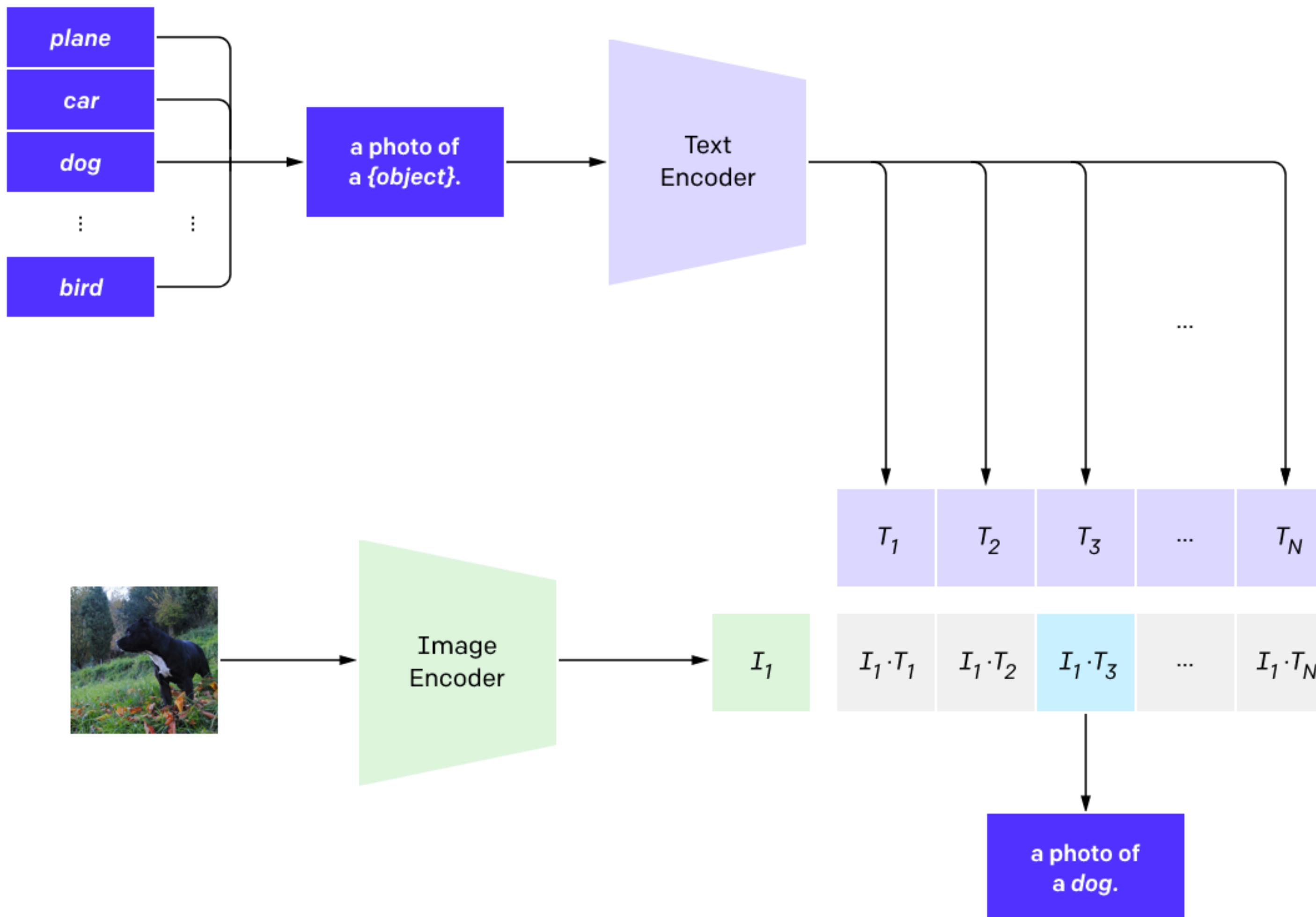


Лосс:

```
# symmetric loss function
labels = np.arange(n)
loss_i = cross_entropy_loss(logits, labels, axis=0)
loss_t = cross_entropy_loss(logits, labels, axis=1)
loss  = (loss_i + loss_t)/2
```

CLIP (Contrastive Language–Image Pre-training)

- Применение для Image Classification: меняем метку на текст “the photo of ...”



CLIP (Contrastive Language–Image Pre-training)

FOOD101

guacamole (90.1%) Ranked 1 out of 101 labels



- ✓ a photo of **guacamole**, a type of food.
- ✗ a photo of **ceviche**, a type of food.
- ✗ a photo of **edamame**, a type of food.
- ✗ a photo of **tuna tartare**, a type of food.
- ✗ a photo of **hummus**, a type of food.

SUN397

television studio (90.2%) Ranked 1 out of 397



- ✓ a photo of a **television studio**.
- ✗ a photo of a **podium indoor**.
- ✗ a photo of a **conference room**.
- ✗ a photo of a **lecture room**.
- ✗ a photo of a **control room**.

YOUTUBE-BB

airplane, person (89.0%) Ranked 1 out of 23



- ✓ a photo of a **airplane**.
- ✗ a photo of a **bird**.
- ✗ a photo of a **bear**.
- ✗ a photo of a **giraffe**.
- ✗ a photo of a **car**.

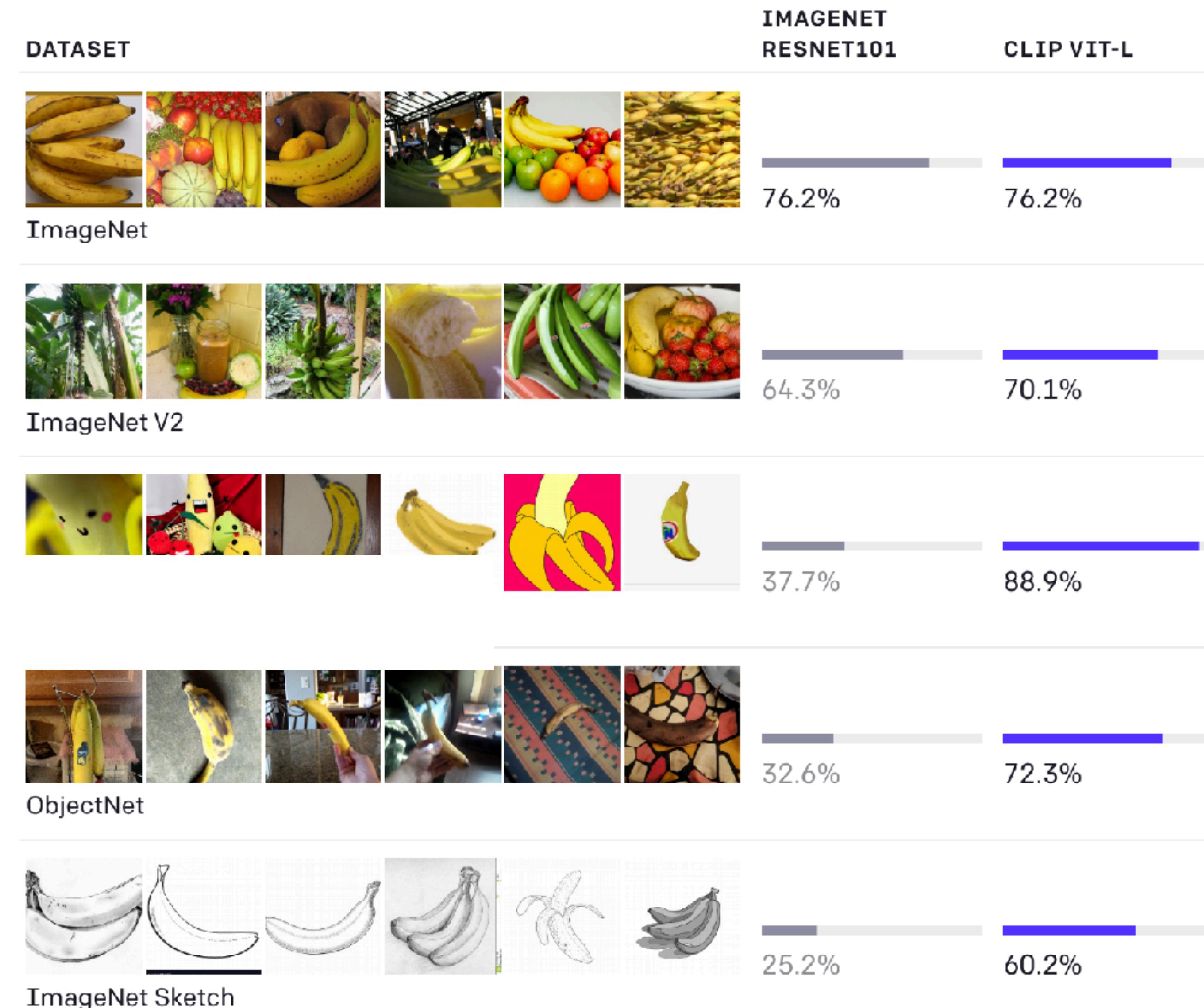
EUROSAT

annual crop land (12.9%) Ranked 4 out of 10



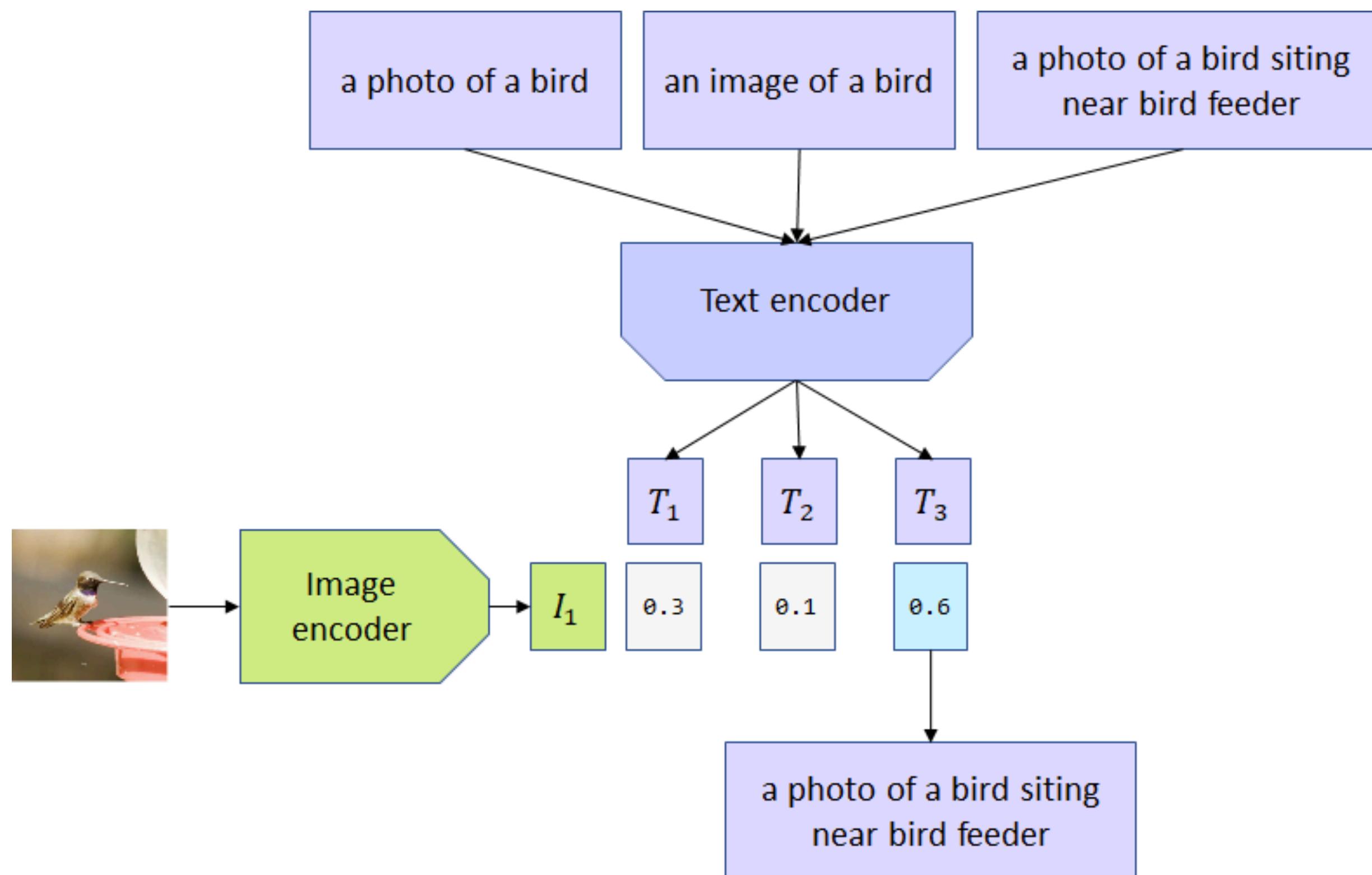
- ✗ a centered satellite photo of **permanent crop land**.
- ✗ a centered satellite photo of **pasture land**.
- ✗ a centered satellite photo of **highway or road**.
- ✓ a centered satellite photo of **annual crop land**.
- ✗ a centered satellite photo of **brushland or shrubland**.

CLIP (Contrastive Language–Image Pre-training)



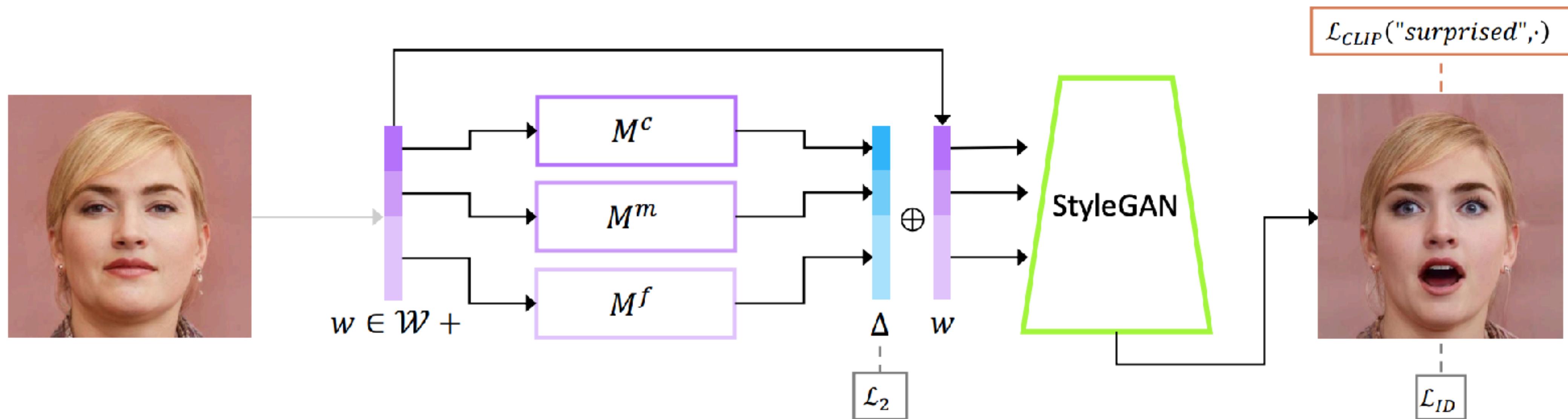
CLIP (Contrastive Language–Image Pre-training)

- Чувствителен к описанию



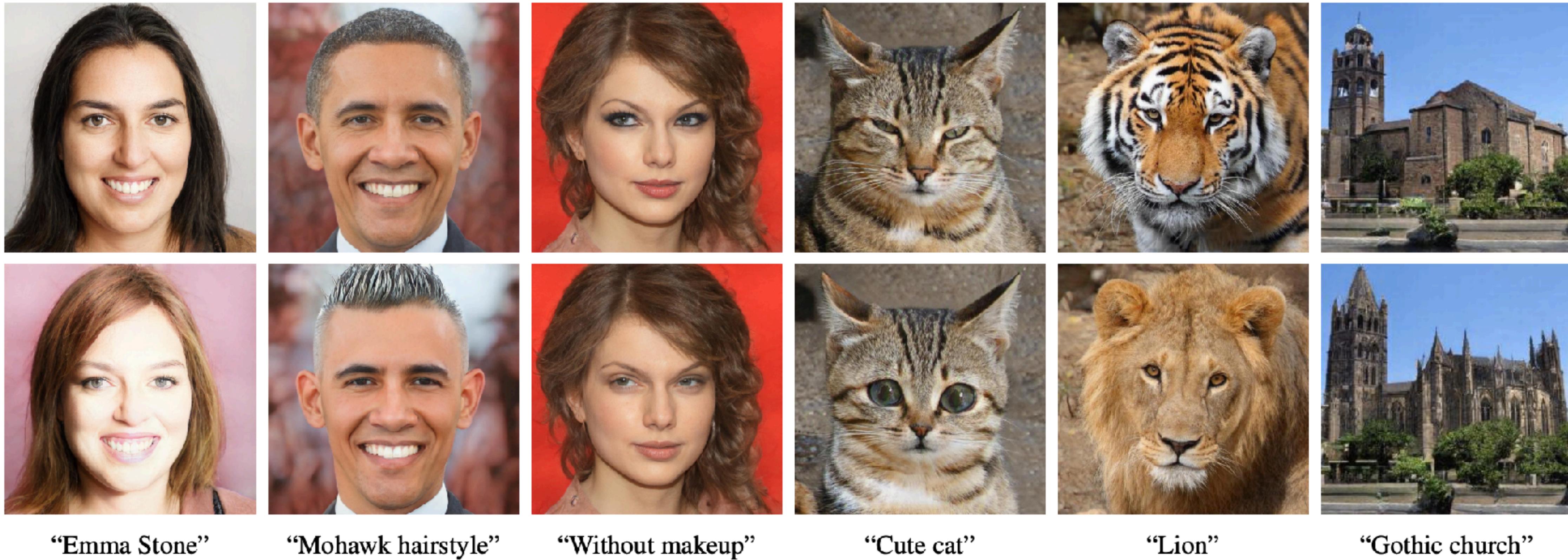
CLIP + StyleGAN

- Используют преодобученные StyleGAN и CLIP
- Цель - выучить трансформации над исходной картинкой, чтобы результат генерации StyleGAN был близок к текстовому описанию (эмбеддингам CLIP)



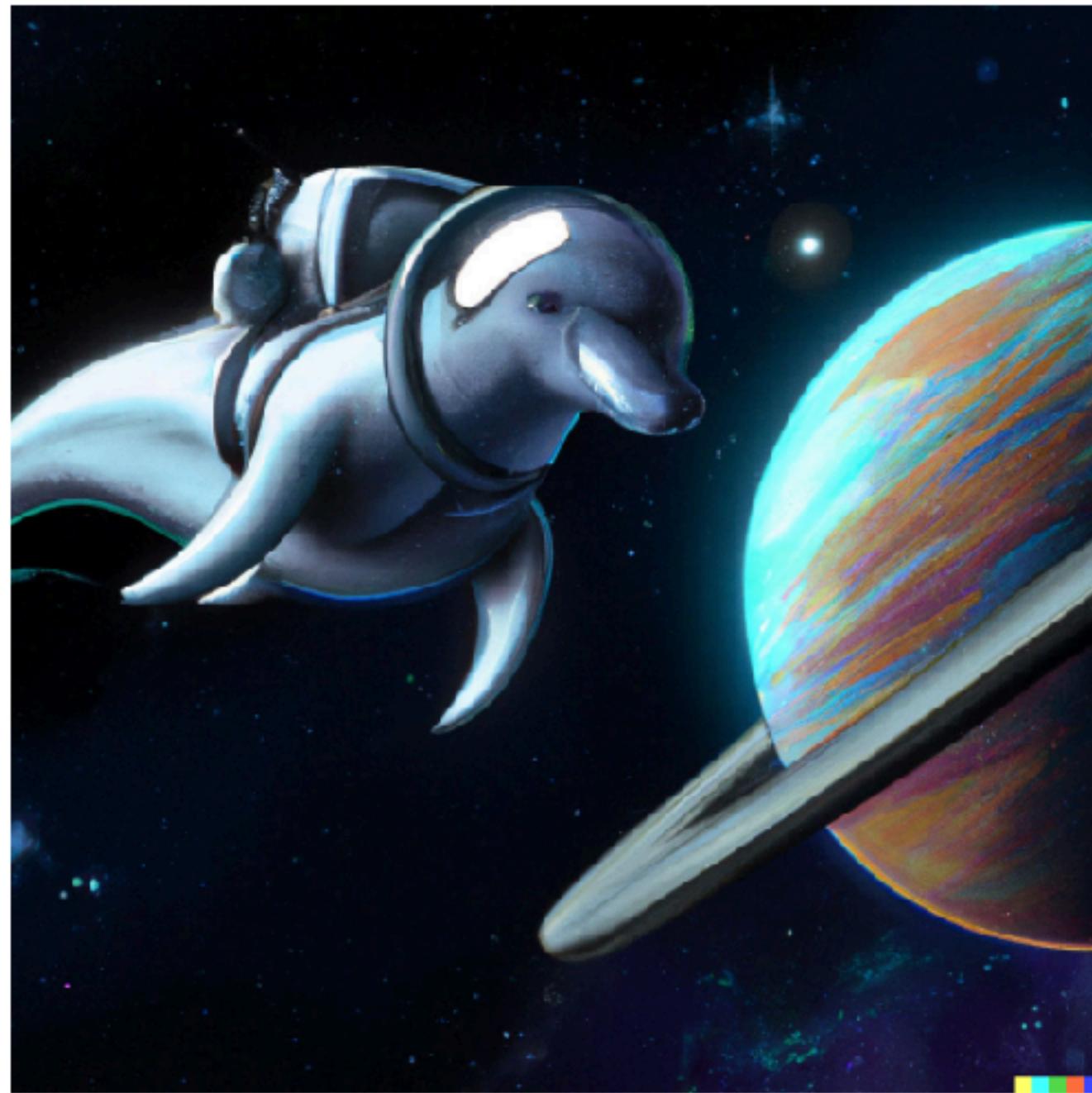
CLIP + StyleGAN

CLIP + StyleGAN — это мощное сочетание, которое позволяет использовать семантическую информацию из текста для управления генерацией изображений. Основная идея заключается в том, чтобы использовать возможности CLIP по интерпретации текста и изображений, чтобы направлять генеративную модель StyleGAN. Это открывает возможности для создания изображений, соответствующих текстовому описанию, без необходимости использования вручную размеченных данных.



DALL-E 2

Модель: CLIP + GLIDE (Denoising Diffusion Probabilistic Model)



a dolphin in an astronaut suit on saturn, artstation



a propaganda poster depicting a cat dressed as french emperor napoleon holding a piece of cheese



a teddy bear on a skateboard in times square

Диффузионная модель (Diffusion Model): GLIDE основан на идее диффузионных моделей, которые представляют собой тип вероятностных моделей. Эти модели обучаются пошагово добавлять шум к изображению и затем обучаются восстанавливать изображение из зашумленной версии. Генерация начинается с случайного шума, который затем очищается шаг за шагом на основе обученной модели. В каждом шаге используются текстовые подсказки, чтобы направить процесс создания изображения. Управление генерацией на основе текста: GLIDE использует текстовые описания для управления генерацией на каждом шаге денойзинга, чтобы изображение соответствовало заданному описанию. Это достигается с помощью модели, которая кодирует текстовые инструкции и интегрирует их в процесс денойзинга, влияя на результат генерации.

Редактирование изображений с помощью текста:

GLIDE позволяет не только генерировать изображения с нуля, но и редактировать существующие. Это достигается добавлением текстовых указаний, которые уточняют, что необходимо изменить в изображении (например, изменить цвет или добавить объект).

Модель корректирует только те части изображения, которые соответствуют описанным изменениям.

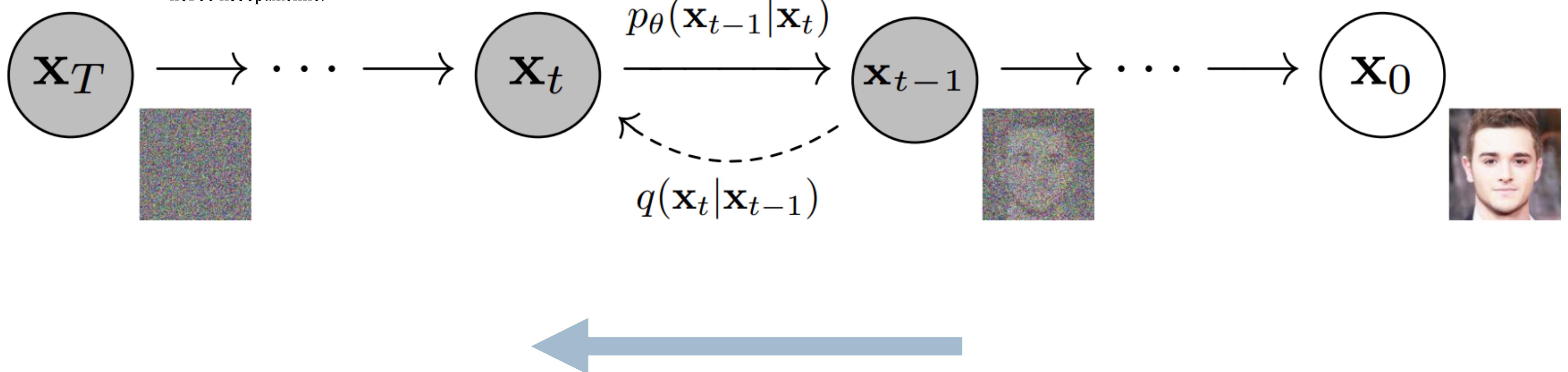
DDPM: overview

Denoising Diffusion Probabilistic Models
Процесс добавления шума (Forward Diffusion Process): В ходе обучения изображение на каждом шаге постепенно зашумляется. На первом шаге изображение еще близко к исходному, а на последних шагах оно становится почти случайным шумом. В результате этого процесса, в изображение добавляется гауссов шум, и оно постепенно становится неузнаваемым.

DDPM:
Процесс удаления шума (Reverse Diffusion Process): Во время генерации модель обучается удалять шум с изображений, начиная с зашумленного изображения и возвращаясь к исходному состоянию. Это задача восстановления исходного изображения из шума, где модель должна предсказать, как устраниить шум и восстановить детали изображения.

Обучение модели: Модель обучается предсказывать шум, добавленный на каждом шаге. Обучение заключается в минимизации ошибки между предсказанным шумом и реальным шумом, добавленным на соответствующем шаге.

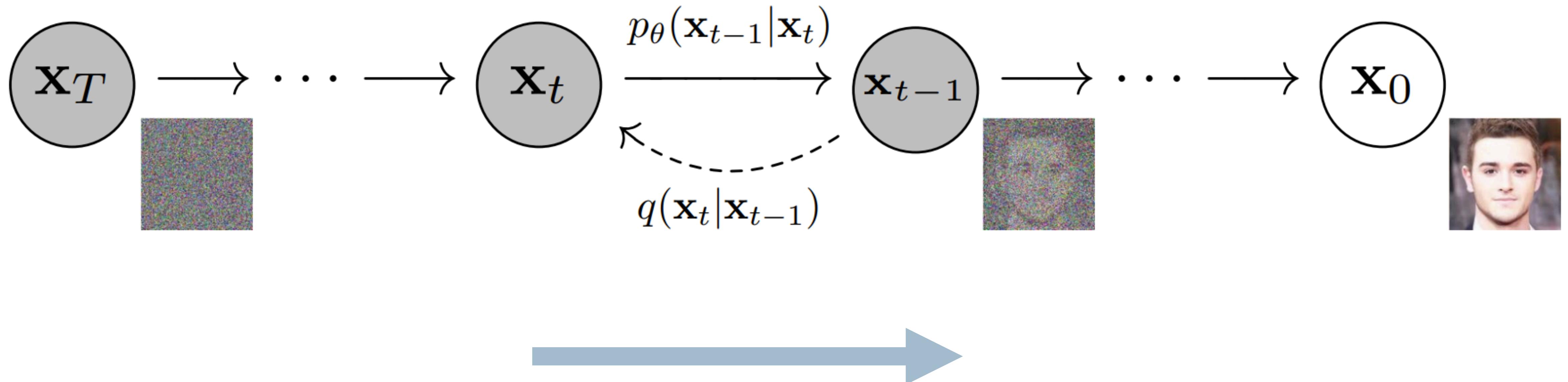
Генерация изображения: Для генерации изображения процесс начинается с чистого шума, который постепенно очищается с помощью предсказаний модели, пока не получится новое изображение.



Добавляем случайный шум на каждом шаге

DDPM: overview

DDPM:

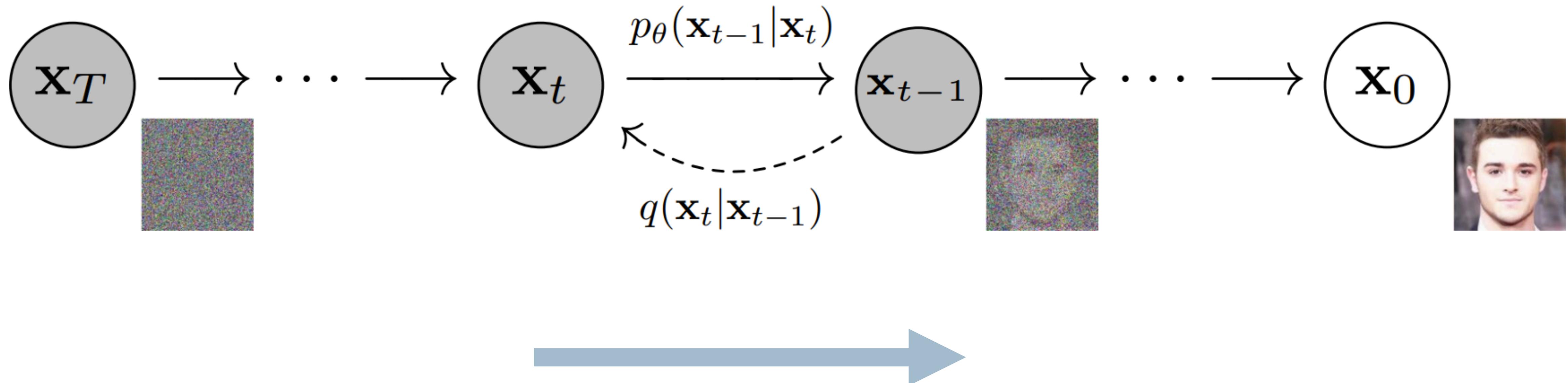


Восстанавливаем (убираем добавленный шум)

DDPM: overview

DDPM:

Markov chain



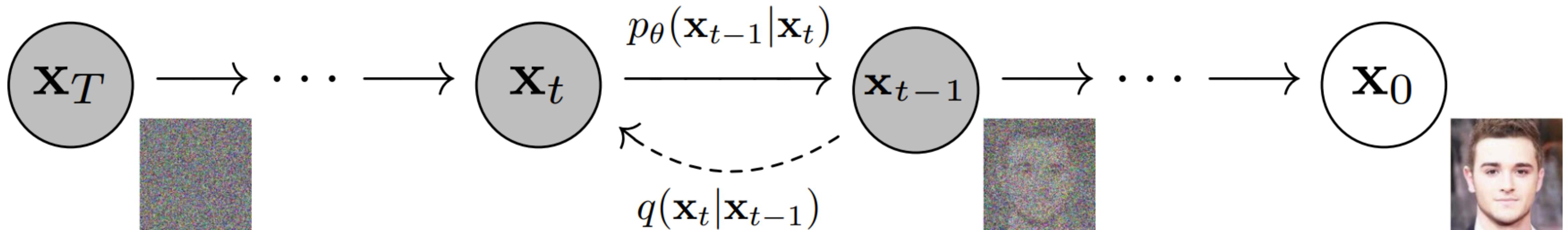
Восстанавливаем (убираем добавленный шум)

Каждый раз применяем модель типа UNet

DDPM: overview

DDPM:

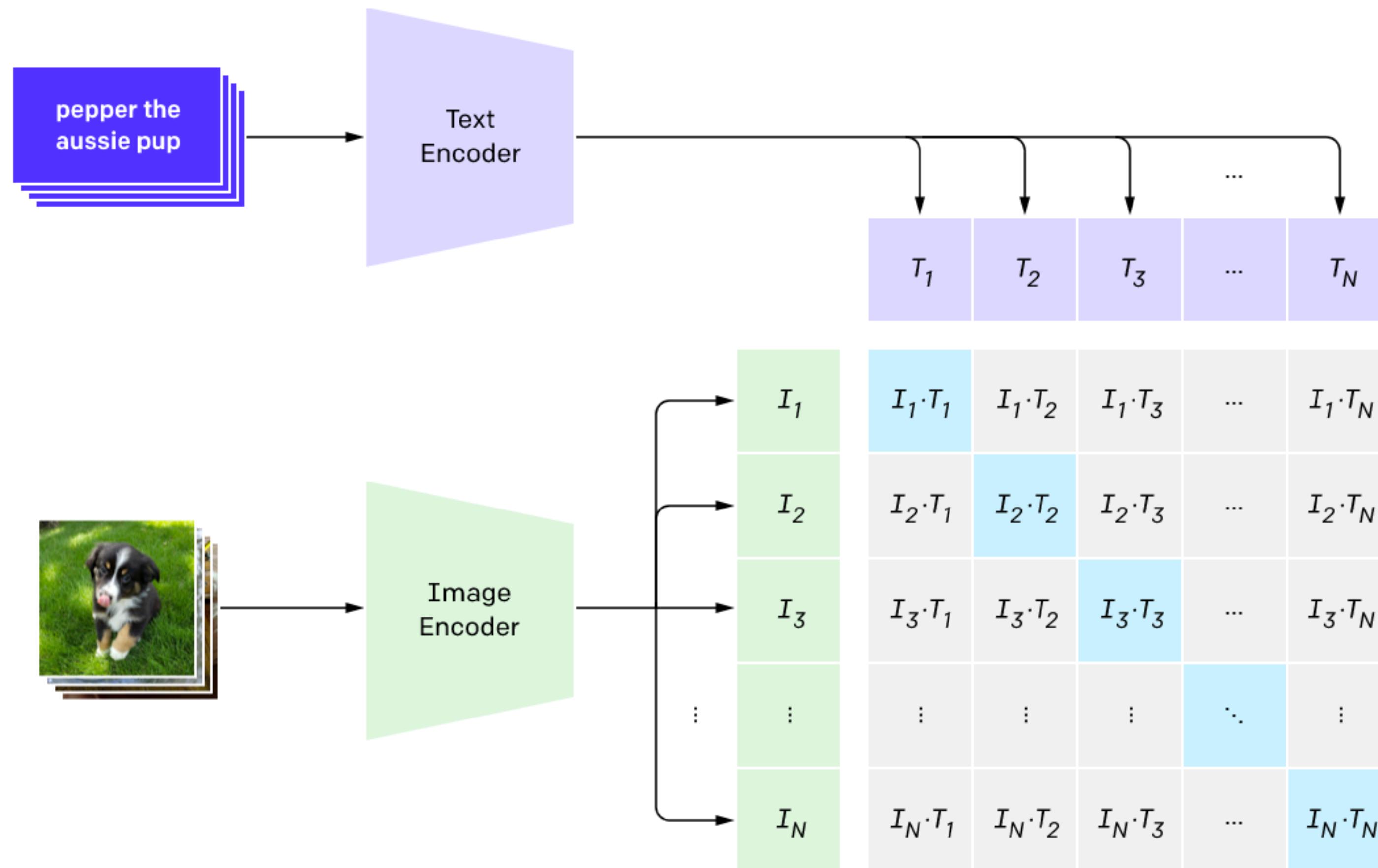
Markov chain



$$L_{\text{simple}} := E_{t \sim [1, T], x_0 \sim q(x_0), \epsilon \sim \mathcal{N}(0, \mathbf{I})} [||\epsilon - \epsilon_\theta(x_t, t)||^2]$$

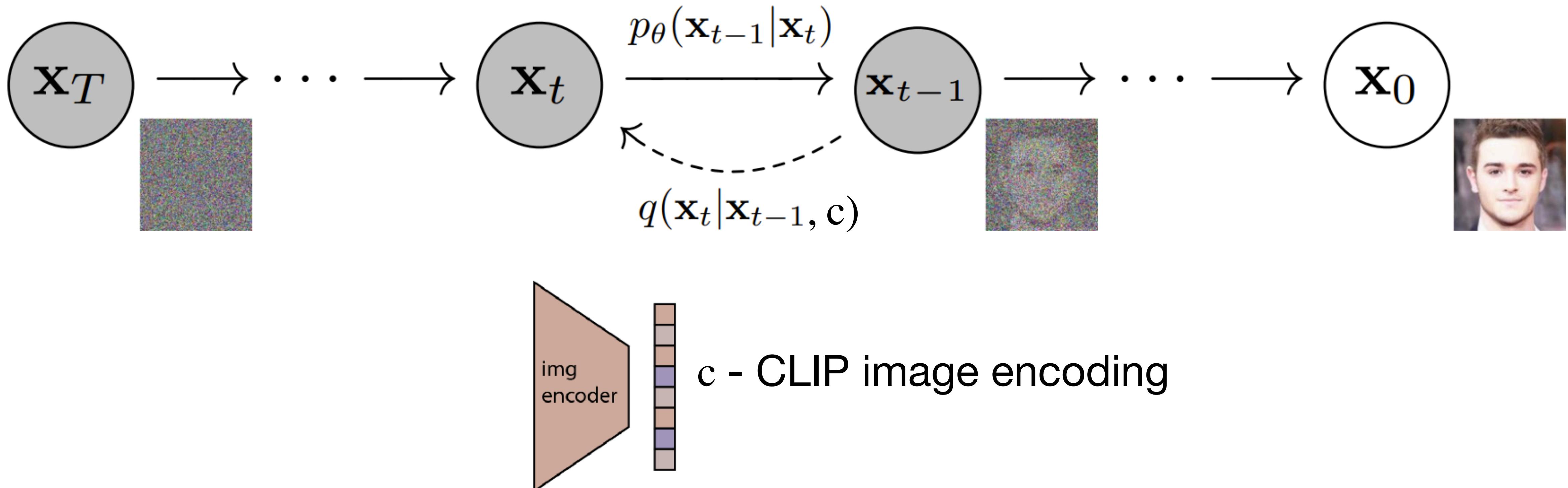
DALL-E 2

1. Обучаем CLIP



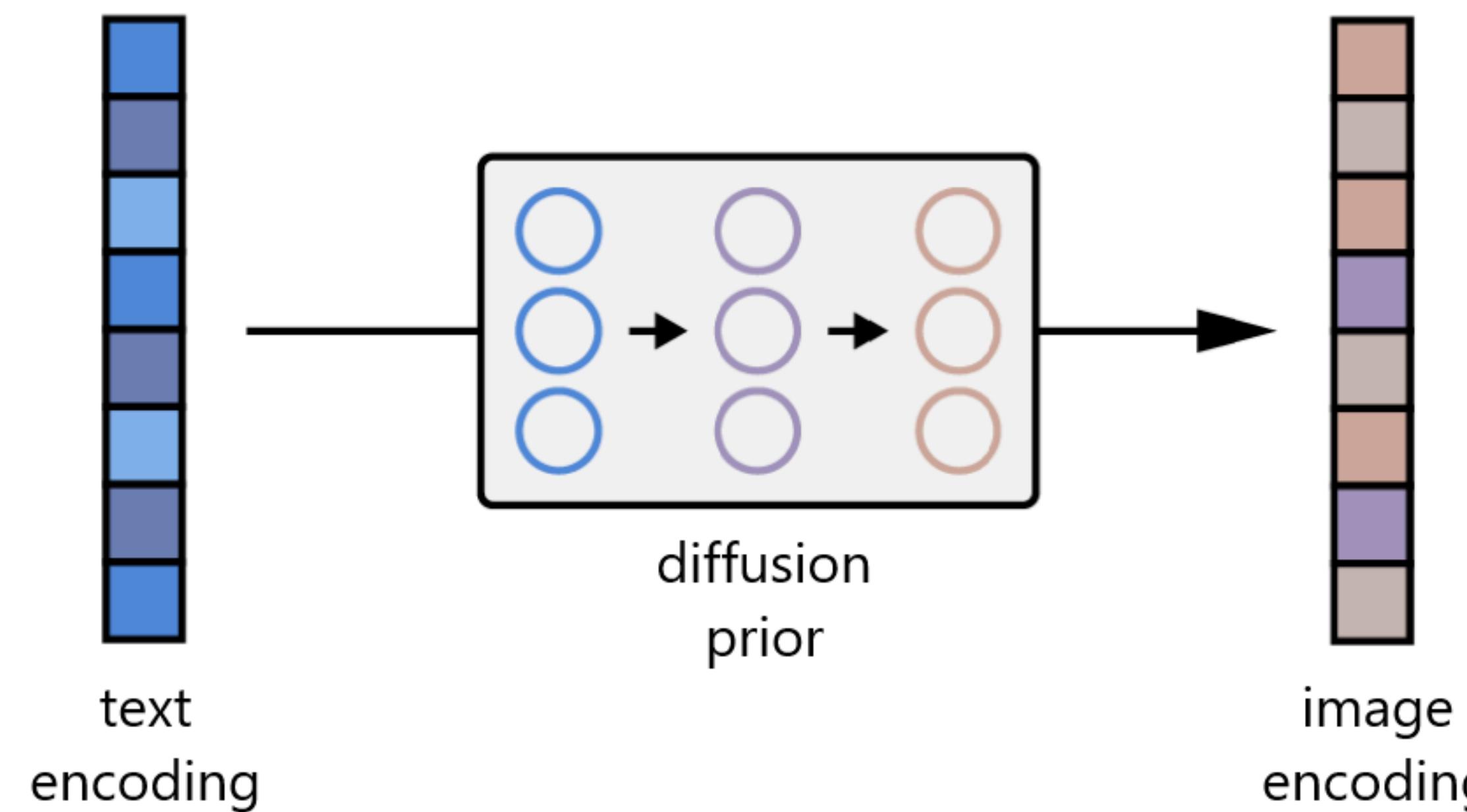
DALL-E 2

1. Обучаем CLIP
2. Обучаем DDPM обусловленную на CLIP image encoding (GLIDE)



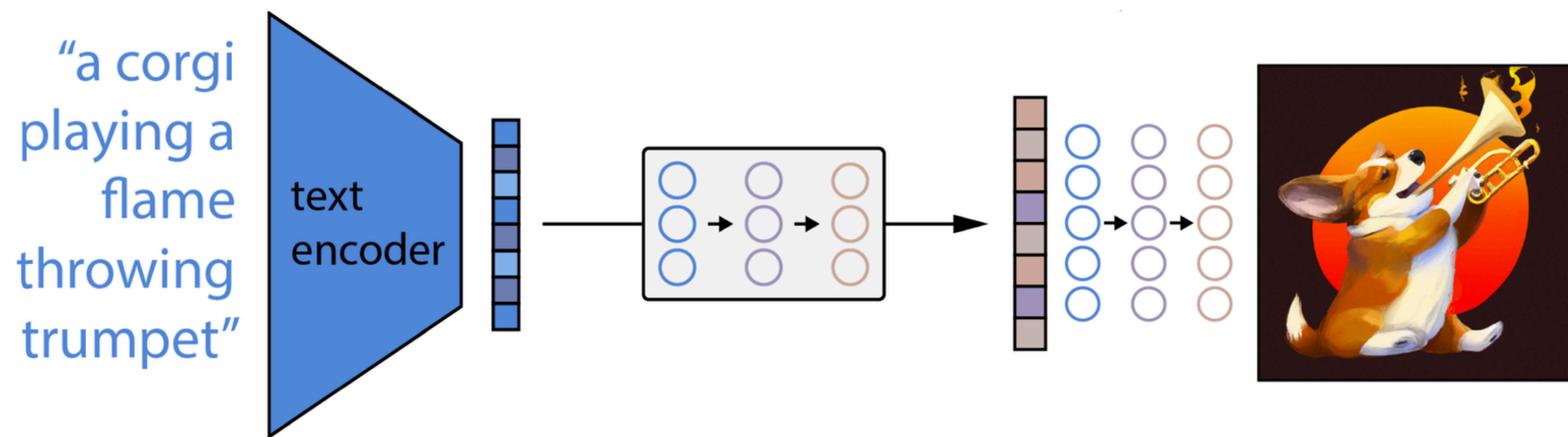
DALL-E 2

1. Обучаем CLIP
2. Обучаем DDPM обусловленную на CLIP image encoding (GLIDE)
3. Обучаем другую DDPM предсказывать CLIP image encoding (по text encoding)



DALL-E 2

Генерация:



DALL-E 2



an espresso machine that makes coffee from human souls, artstation



panda mad scientist mixing sparkling chemicals, artstation



a corgi's head depicted as an explosion of a nebula

Методы дообучения LLM

Методы дообучения LLM

Fine-tuning (тонкая настройка) — это процесс дообучения уже существующей большой языковой модели (LLM) на специфических данных, чтобы адаптировать её к выполнению конкретной задачи или улучшить её точность в определённой области. Вместо того чтобы обучать модель с нуля (что требует огромных вычислительных ресурсов и данных), fine-tuning позволяет взять предобученную модель и адаптировать её под свои задачи с меньшими затратами.

Методы дообучения LLM

Fine-tuning (тонкая настройка) — это процесс дообучения уже существующей большой языковой модели (LLM) на специфических данных, чтобы адаптировать её к выполнению конкретной задачи или улучшить её точность в определённой области. Вместо того чтобы обучать модель с нуля (что требует огромных вычислительных ресурсов и данных), fine-tuning позволяет взять предобученную модель и адаптировать её под свои задачи с меньшими затратами.

Как это работает?

Предобученная модель: Модель уже обучена на большом корпусе текстов (например, модель GPT-2, GPT-3).

Данные для тонкой настройки: Составляется датасет, содержащий текст, релевантный задаче (например, вопросы и ответы, техническая документация и т.д.).

Процесс обучения: Модель обучается на новом наборе данных, используя предварительно выученные представления. Это ускоряет процесс и снижает требования к вычислительным ресурсам.

Оценка и корректировка: После обучения модель тестируется на отдельном наборе данных, чтобы убедиться, что она выполняет задачу на требуемом уровне.

Методы дообучения LLM

Традиционный метод **Fine-tuning**, при котором настраиваются все параметры предварительно обученной модели, становится непрактичным и вычислительно дорогостоящим при работе с современными моделями LLM

Методы дообучения LLM

Традиционный метод **Fine-tuning**, при котором настраиваются все параметры предварительно обученной модели, становится непрактичным и вычислительно дорогостоящим при работе с современными моделями LLM

PEFT Parameter-Efficient Fine-Tuning

PEFT (Parameter-Efficient Fine-Tuning) представляет собой эффективный подход, позволяющий не терять производительность при тонкой настройке модели, снижая при этом требования к памяти и вычислительным мощностям.

PEFT - это метод fine-tune, который позволяет улучшить результаты предварительно обученных языковых моделей при выполнении определенных задач. Его идея заключается в том, чтобы обучить небольшое подмножество параметров предварительно обученной модели LLM, оставляя большую часть замороженными.

Аддитивный метод в контексте машинного обучения и адаптации моделей означает подход, при котором изменения или адаптации к существующей модели добавляются **дополнительно**, а не заменяют оригинальные параметры модели. Это позволяет сохранить исходные знания модели и аккуратно добавлять новые, минимизируя вероятность утраты старых знаний.

Методы дообучения LLM

Преимущества PEFT

Ускорение времени обучения: PEFT позволяет сократить количество времени, затраченное на обучение, благодаря fine-tune небольшого количества параметров, а не всей модели.

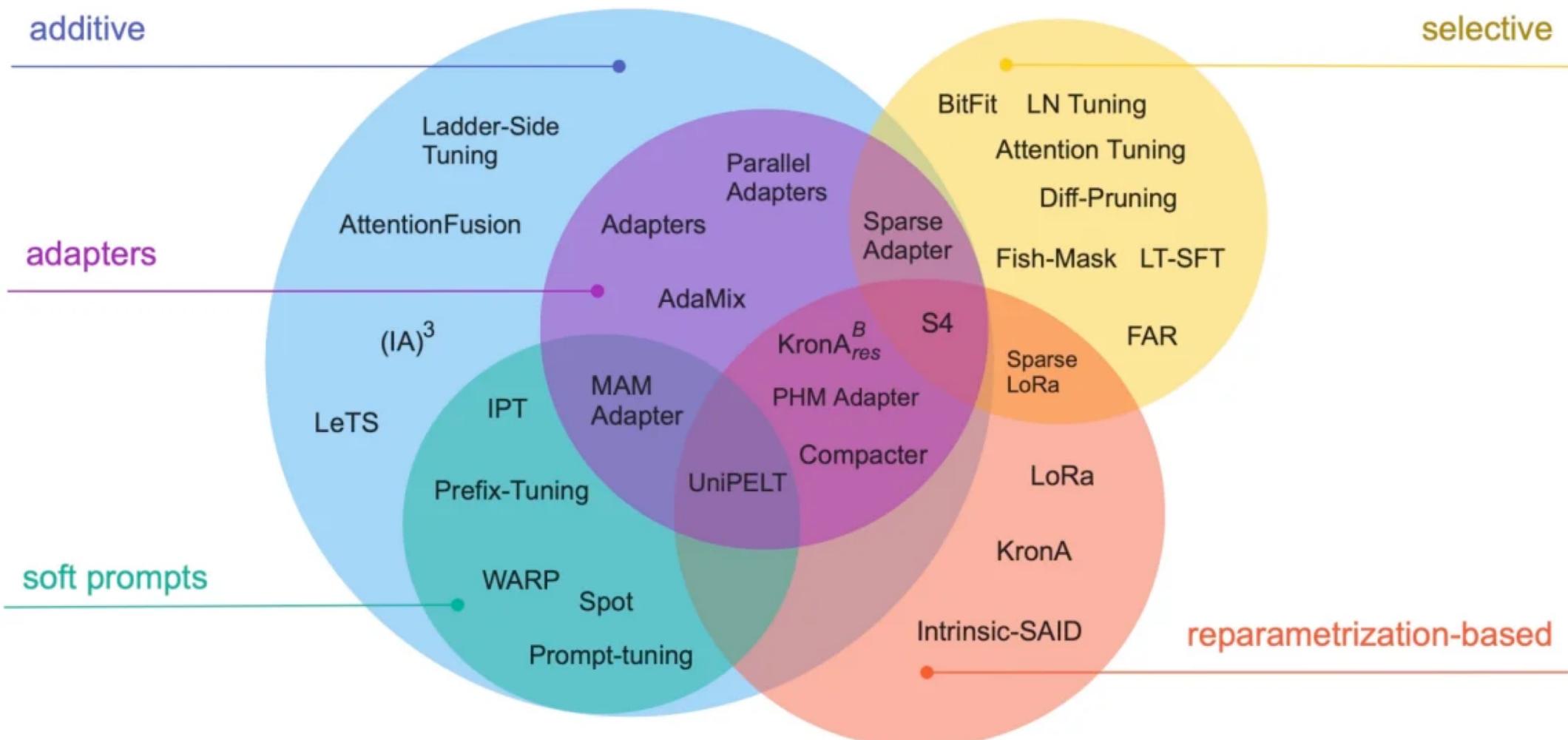
Снижение затрат на вычисления и хранение: PEFT выполняет тонкую настройку только небольшого подмножества параметров, что значительно уменьшает затраты на вычисления и хранение и снижает требования к оборудованию.

Меньший риск переобучения: Благодаря замораживанию большей части параметров предварительно обученной модели мы можем избежать переобучения на новых данных.

Преодоление катастрофического забывания: С помощью PEFT модель может адаптироваться к новым задачам, сохраняя ранее полученные знания благодаря заморозке большинства параметров.

Удобство развертывания: Контрольные точки, созданные при PEFT, компактнее, чем при традиционной тонкой настройке, что делает их развертывание и перенос на другие устройства более простым, поскольку они требуют меньше места для хранения и могут загружаться быстрее.

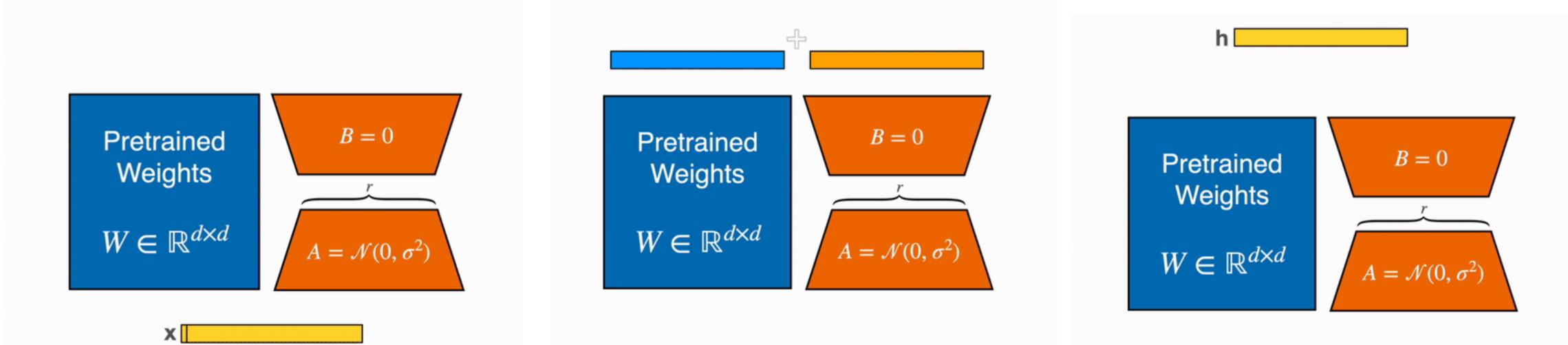
Методы дообучения LLM



Методы дообучения LLM

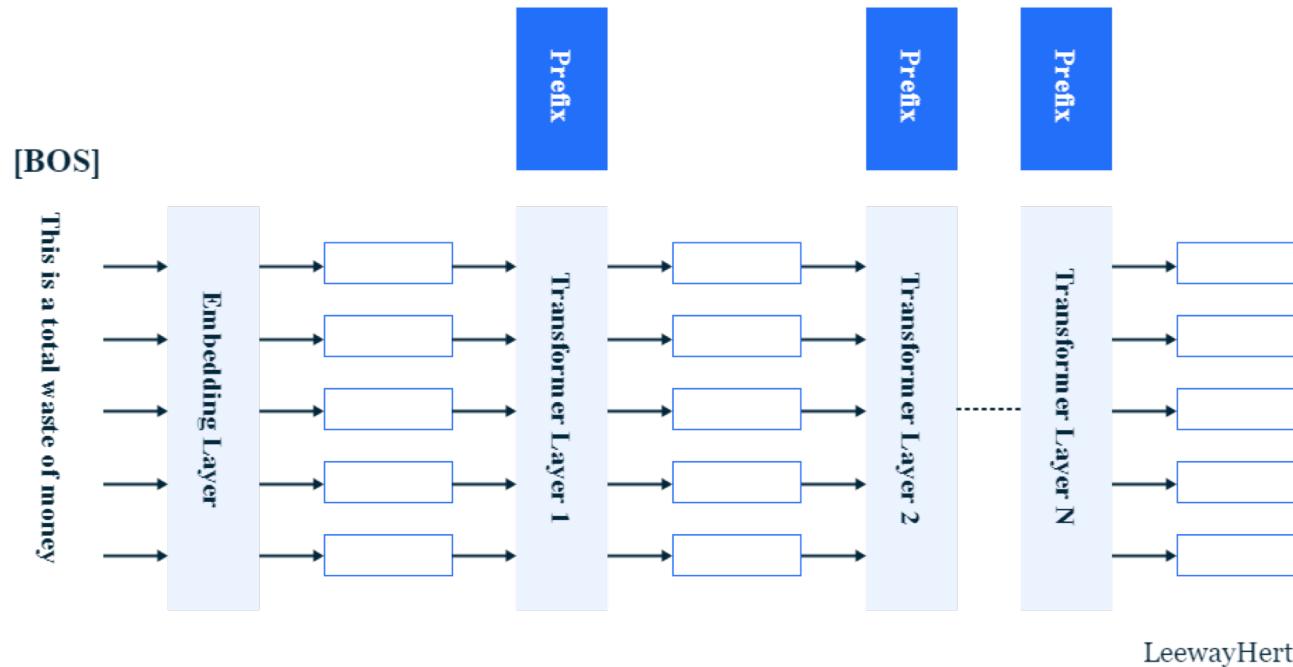
LoRA

Метод LoRA (Low-Rank Adaptation) был разработан в 2021 году и представлен в данной [статье](#). Его создатели были вдохновлены данной научной [работой](#), в которой авторы отмечают, что, хотя LLM имеют миллионы или даже миллиарды параметров, они имеют низкую "внутреннюю размерность" (intrinsic dimension) при адаптации к новой задаче. Т. е. большинство параметров являются избыточными. Из чего можно сделать вывод, что матрицы можно представить пространством меньшей размерности, сохраняя при этом большую часть важной информации.



Создатели LoRA предположили, что изменение весов при fine-tuning модели имеет низкий "внутренний ранг" (intrinsic rank). Идея данного метода заключается в том, что для предварительно обученной матрицы весов мы представляем её обновление двумя меньшими матрицами, полученными путем низкоранговой аппроксимации. Эти матрицы мы тренируем при обучении, а исходную матрицу весов замораживаем. Затем для получения окончательного результата мы объединяем исходные и обученные веса.

Методы дообучения LLM

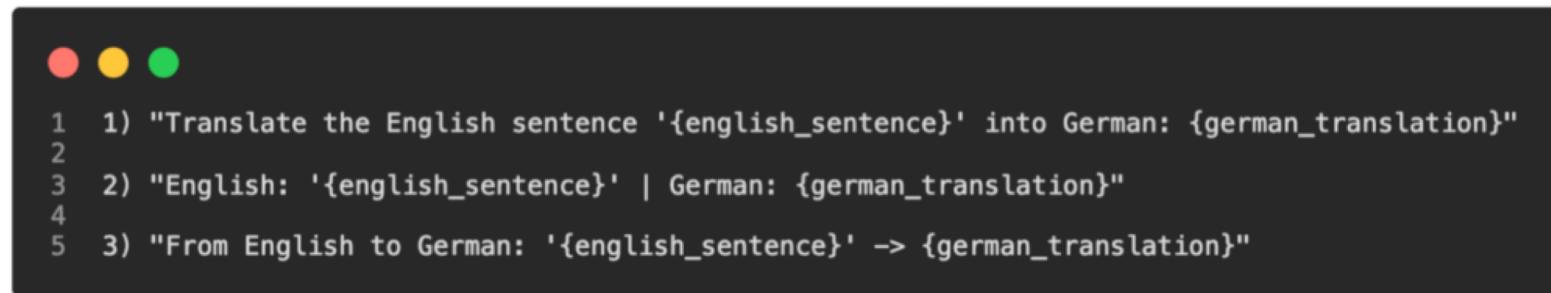


Prefix tuning является аддитивным методом, который обворачивает исходные входные данные в дополнительный контекст для конкретной задачи, однако мы не сами задаем его, а находим с помощью обучения модели. Суть данного метода заключается в том, что мы добавляем последовательность обучающих векторов (continuous task-specific vectors), называемым префиксом, к каждому блоку трансформера и обучаем только её, не трогая остальные данные.

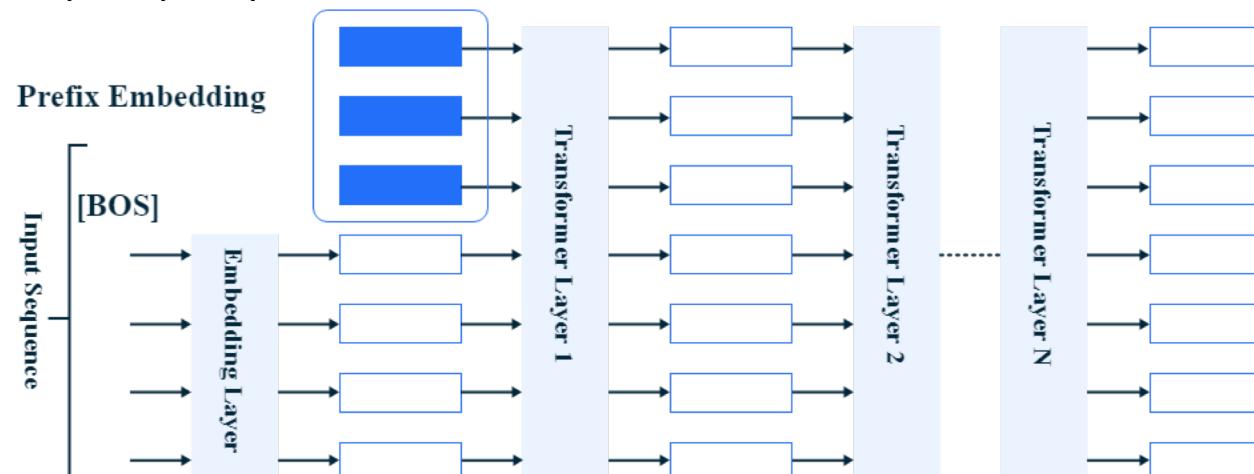
Методы дообучения LLM

Prompt tuning - это аддитивный метод, который является упрощенной версией Prefix tuning.

- 1) **Hard prompts:** Такой тип промпта можно рассматривать, как некий шаблон. Он создается вручную людьми и является статичным.



- 2) **Soft prompts:** Даный вид создается в процессе использования метода Prompt tuning. Мы объединяем входные векторные представления предобученной модели с обучаемым тензором, который затем оптимизируем с помощью обратного распространения ошибки.



Методы дообучения LLM

Основные отличия

	Hard Prompts	Soft Prompts
Тип	Обычный текст	Обучаемые векторы
Гибкость	Ограничены естественным языком	Высокая гибкость в обучении
Интерпретируемость	Легко интерпретируются человеком	Не имеют интерпретации
Эффективность	Эффективны для простых задач	Более эффективны для сложных задач
Обучение	Не требуют обучения	Требуют обучения на задаче

Системный промптинг

Что это? Системный промпт — это скрытая инструкция, которая задается модели перед началом общения. Он определяет начальные настройки, включая тон, стиль, ограничения и контекст. Пользователь обычно не видит системный промпт, так как он находится "за кулисами".

Когда используется? Системные промпты часто используются для настройки моделей на выполнение определённых задач или для ограничения поведения, например, чтобы модель оставалась в рамках определенной темы или говорила определённым стилем.

Пример: Системный промпт может быть: "Ты эксперт в области медицины, отвечай на вопросы с научной точностью, но простым языком."

Основные отличия между системным промптом и hard prompt:

Характеристика	Системный промпт	Hard Prompt
Цель	Настроить начальные условия, поведение и контекст модели	Дать конкретный запрос или инструкцию модели
Видимость пользователю	Невидим для пользователя	Видим и создаётся пользователем
Использование	Определяет глобальное поведение модели	Запросы или инструкции для конкретных задач
Влияние	Действует на всё взаимодействие	Влияет только на текущий запрос
Тип	Скрытый текст, который задаётся в начале сессии	Явный текст, обычно на естественном языке

Методы дообучения LLM

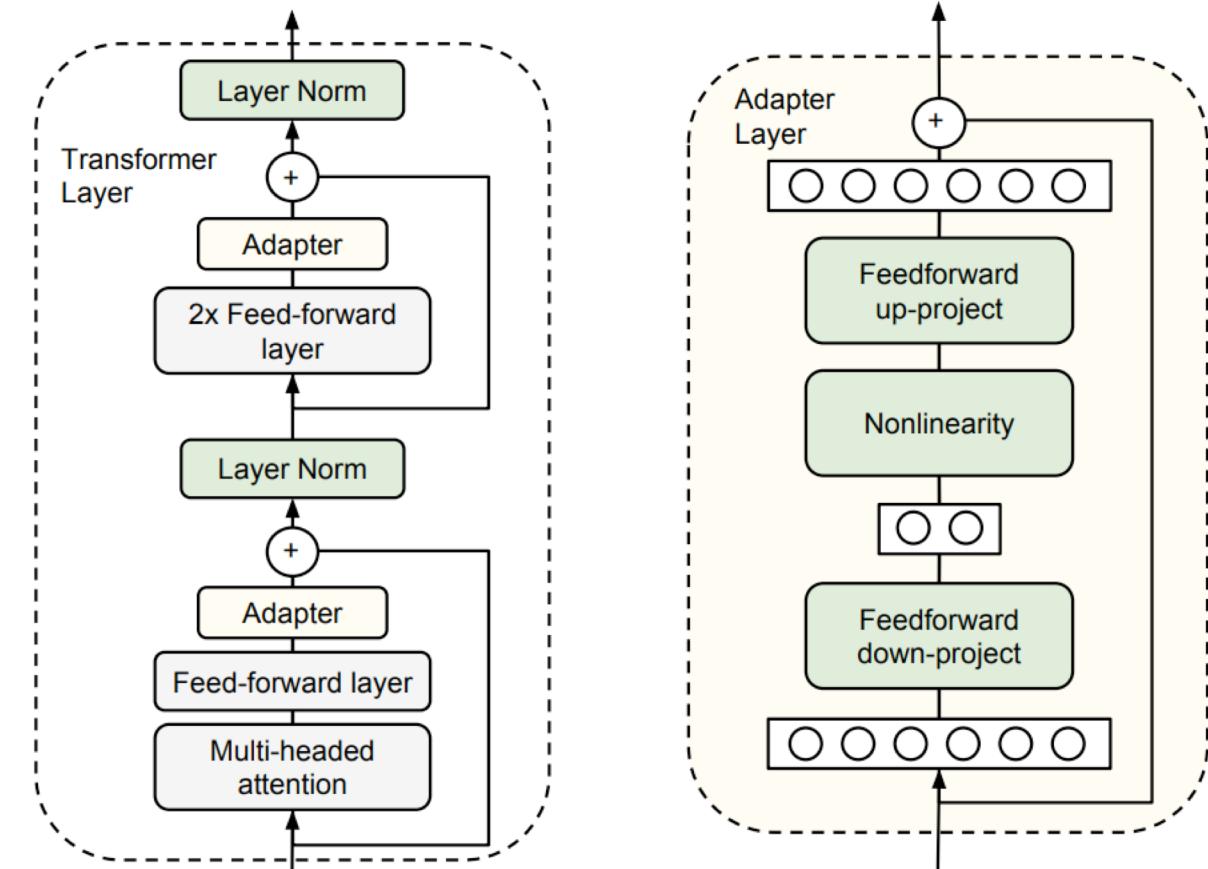
Adapters также является аддитивным методом. Он похож на метод Prefix tuning, только в этом случае мы добавляем не префикс, а адаптеры. Авторы данного метода ([статья](#)) предложили следующую структуру:

Внутри адаптера исходные d -мерные признаки сначала проецируются в меньшее измерение m , затем применяется нелинейность, а после снова проецируются в d -мерное измерение. Также здесь присутствует skip connection.

Как работает Adapter Tuning

Добавление адаптеров: Адаптеры — это небольшие обучаемые слои, которые добавляются между слоями предобученной модели. Эти слои обычно имеют небольшое количество параметров по сравнению с весами самой модели. Адаптеры обычно работают путем выполнения незначительных трансформаций входных данных на каждом слое, что позволяет модели адаптироваться к новой задаче или домену.

Параллельное использование: Основная модель (например, GPT-3 или BERT) остается фиксированной, и её веса не обновляются. Вместо этого адаптеры добавляются в виде дополнительных слоёв, которые обучаются на данных для конкретной задачи. Эти адаптеры могут быть разных типов, например, они могут работать как сети с пропуском (skip networks), которые вносят минимальные изменения в оригинальные слои.



Методы дообучения LLM

Сравнительная таблица

Метод	Модификация	Ресурсоёмкость	Производительность	Применение
Fine-Tuning	Полное изменение модели	Низкая (высокие затраты)	Высокая (оптимизация для конкретной задачи)	Специализированные задачи (классификация, перевод)
Prefix Tuning	Изменение только префикса	Высокая (замораживает модель)	Умеренная	Генерация текста, перевод
Prompt Tuning	Обучение подсказок	Очень высокая	Умеренная	Ответы на вопросы, завершение текста
Adapter Tuning	Добавление адаптеров между слоями	Высокая (необходимы дополнительные параметры)	Высокая (сравнительно с prompt tuning)	Многозадачность, доменная адаптация
LoRA	Использование низкоранговых матриц	Очень высокая	Высокая (в сочетании с другими методами)	Адаптация при ограниченных ресурсах

Fine-Tuning дает максимальную производительность, но требует много ресурсов.

Prefix Tuning и **Prompt Tuning** эффективны в экономии ресурсов и подходят для задач, где важно минимальное вмешательство в модель.

Adapter Tuning предоставляет хорошее соотношение между производительностью и ресурсами, позволяя адаптировать модель для разных задач.

LoRA предлагает эффективный способ адаптации при ограниченных вычислительных мощностях, идеально подходящий для быстрого изменения модели без затрат на полную настройку.