Обзор нейронных сетей для компьютерного зрения

План

1. **Введение:** Задачи Computer Vision и почему именно нейронные сети?
2. **Предшественники:** От перцептрона к идее свертки
3. **Эра CNN:** LeNet, AlexNet, VGG, Inception, ResNet
4. **Архитектуры для детекции и сегментации:** R-CNN, YOLO, U-Net, Mask R-CNN
5. **Современные тренды:**
    - Трансформеры в Vision (ViT, DETR)
    - Нейросети без учителя (SSL) и самообучение (Self-Supervised Learning)
    - Диффузионные модели для генерации изображений
6. **Заключение:** Куда движется область?

От NLP к CV

До 2020 года **компьютерное зрение** (CV) было практически **синонимом свёрточных нейронных сетей** (CNN). Архитектуры вроде ResNet, EfficientNet задавали тон.
Они идеально подходили для изображений благодаря индуктивным предпосылкам: локальность и трансляционная инвариантность.

Параллельно, в **обработке естественного языка** (NLP) с 2017 года произошла революция. **Transformer** (статья «Attention is All You Need») предложил **механизм самовнимания (self-attention)**, который:

- **Моделирует глобальный контекст**: Любой элемент последовательности (слово) напрямую взаимодействует с любым другим, в отличие от RNN с их ограниченной памятью.
- **Идеально масштабируется для обучения**: Высокопараллельные вычисления на GPU.

Это привело к созданию гигантов: BERT, GPT, T5, которые доминировали в NLP.

Вопрос

Может ли архитектура, созданная для текста, работать с изображениями?

Вопрос

Может ли архитектура, созданная для текста, работать с изображениями?

Интуиция говорит «нет» — у изображений нет очевидной последовательности, это структурированные 2D-данные.

Вопрос

Может ли архитектура, созданная для текста, работать с изображениями?

Интуиция говорит нет — у изображений нет очевидной последовательности, это структурированные 2D-данные.

Но ключевая идея: **Трансформеры работают не с текстом, а с последовательностями токенов (векторов)**.
Задача — превратить изображение в такую последовательность.

Vision Transformer (ViT): Основной прорыв

Статья 2020 года: «An Image is Worth 16x16 Words» (Дословно: «Изображение стоит 16x16 слов»). Название — ключ к пониманию.

Как превратить изображение в предложение?

1. **Разбиение на патчи (Patches):**
   - Входное изображение (H, W, C) разбивается на N непересекающихся квадратных патчей размера P x P.
   - Пример: Изображение 224x224, патч 16x16 → N = (224/16)^2 = 196 патчей. Это и есть слова нашего предложения.
   - Каждый патч (P, P, C) вытягивается в вектор-строку размерности P*P*C.

2. **Линейное проецирование (Эмбеддинг):**
   - Каждый вытянутый вектор проецируется с помощью обучаемой матрицы **E** в пространство размерности **D** (скрытое состояние трансформера). Получаем Patch Embeddings.
   - К этой последовательности добавляется специальный токен [class] в начало. Его итоговый вектор после обработки трансформером будет использоваться для классификации всего изображения.

Специальный токен [class]

Добавочный, искусственный вектор, который изначально не соответствует ни одному патчу изображения.

Его часто инициализируют нулями или случайными значениями.

Он полностью обучаемый параметр модели.

В разных реализациях его могут называть [CLS], [class] или cls_token.

Обработка трансформером

Вся последовательность (токен [class] + 196 эмбеддингов патчей) проходит через слои энкодера трансформера.

Ключевая особенность архитектуры трансформера — механизм внимания (self-attention), который позволяет каждому элементу последовательности взаимодействовать со всеми остальными.

Токен [class], находясь в начале и проходя через эти слои, аккумулирует информацию со всех остальных токенов-патчей.

Каждый патч, в зависимости от его содержания, вкладывает часть своей информации в обновление вектора [class] на каждом слое.

Именно этот вектор подается на финальный классификационный слой (обычно это один полносвязный слой), который и предсказывает метку всего изображения (например, кошка, собака, автомобиль).

D — это размерность скрытого состояния (hidden dimension)

Это размерность вектора, представляющего каждый элемент последовательности внутри трансформера

Гиперпараметр модели, который задаётся разработчиком перед началом обучения
D явно указывается в коде при создании модели

Когда пропускаются входные данные через первый слой трансформера, они преобразуются в векторы размерности D

D остаётся постоянной на протяжении всех слоёв трансформера. Выход одного слоя (размерности D) является входом для следующего слоя

Где увидеть или узнать D для уже существующей модели?

D не вычисляется из данных — это фундаментальный архитектурный выбор, который определяет ёмкость модели.

Его значение нужно искать:

- В коде создания модели (как гиперпараметр)
- В документации или конфигурации предобученной модели (часто обозначается как d_model, hidden_size, embed_dim).

В известных архитектурах это фиксированные значения:

BERT-base: D = 768
BERT-large: D = 1024
GPT-3: D = 12288 (для самой большой модели)
T5-base: D = 768
Оригинальный Transformer из статьи «Attention Is All You Need»: D = 512

Чем больше D, тем больше потенциал модели к обучению сложным паттернам, но и тем больше требуются вычислительные ресурсы и данные для обучения.

Vision Transformer (ViT): Основной прорыв

Статья 2020 года: «An Image is Worth 16x16 Words» (Дословно: «Изображение стоит 16x16 слов»). Название — ключ к пониманию.

Как превратить изображение в предложение?

3. **Позиционное кодирование:**
   В отличие от CNN, у механизма внимания изначально нет понятия порядка или расположения патчей.
   Чтобы сеть знала, где находится патч, мы добавляем к каждому эмбеддингу патча **позиционное кодирование (Positional Encoding)** — уникальный вектор, который может быть обучаемым или фиксированным (синусоидальным, как в оригинальном Transformer).
   Это критически важно, так как смысл изображения сильно зависит от взаимного расположения частей.

Vision Transformer (ViT): Основной прорыв

Статья 2020 года: «An Image is Worth 16x16 Words» (Дословно: «Изображение стоит 16x16 слов»). Название — ключ к пониманию.

Как превратить изображение в предложение?

4. **Стандартный Transformer-Энкодер:**
    - Полученная последовательность векторов (N+1, D) подается на вход стопке из L идентичных слоев Transformer Encoder.
    - **Каждый слой:**
        - **Multi-Head Self-Attention (MSA):** Позволяет каждому патчу смотреть на все остальные и интегрировать глобальный контекст. Например, патч с головой собаки может усилить внимание к патчам с лапами и хвостом.
        - **MLP-блок (Feed-Forward Network):** Обрабатывает каждый вектор независимо.
        - **Остаточные связи (Skip Connection) и LayerNorm** для стабилизации обучения.

Трансформер

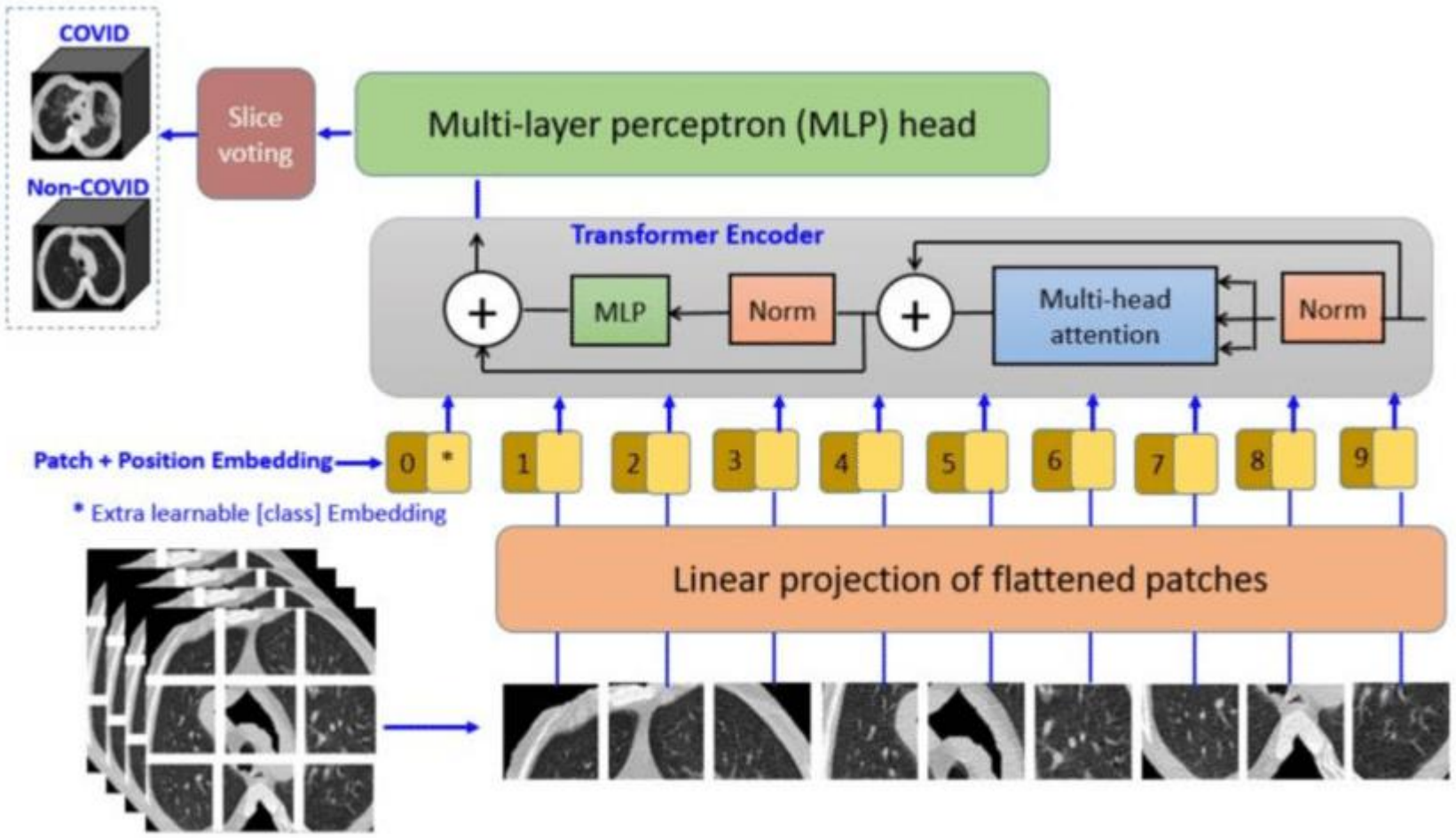
Трансформер состоит из нескольких слоёв, в каждом из которых применяются **механизмы внимания для выявления взаимосвязей между патчами** и агрегации информации, определяя тот самый механизм внимания.

После того как каждый патч изображения преобразуется в векторное представление и проходит через несколько слоёв трансформера, **представления патчей агрегируются**, чтобы получить окончательное представление всего изображения.

Агрегация может быть выполнена различными способами, в зависимости от конкретной задачи и архитектуры модели. Одним из наиболее распространенных **методов агрегации** является **глобальный пулинг** (например, средний пулинг или максимальный пулинг) по всем патчам изображения.

Другим методом агрегации может быть **использование дополнительных слоёв внутри трансформера**, которые **агрегируют информацию из различных частей изображения**. Например, после слоёв трансформера можно добавить дополнительные слои внимания или свёрточные слои для обработки представлений патчей и получения итогового агрегированного представления

Vision Transformer (ViT)



Пример работы ViT для оценки заболевания Covid

Затем агрегированное представление подаётся на вход классификатору, который выдаёт предсказание для конкретной задачи, например, классификации изображений.

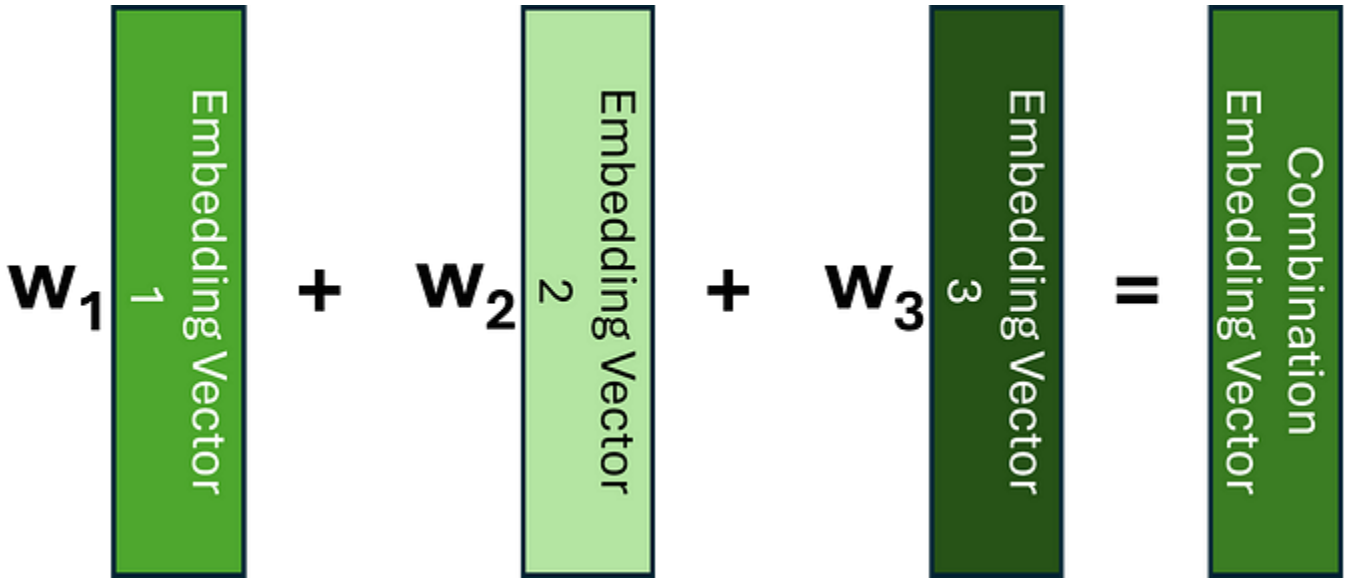
Что такое внимание?

Самый важный механизм в трансформенной архитектуре — это внимание.

Он позволяет нейросети понять, какая часть входной последовательности наиболее релевантна задаче.

Механизм внимания определяет для каждого токена последовательности, какие другие токены необходимы для его понимания в данном контексте.

Механизм внимания



Этот механизм можно представить как метод, который заменяет каждый эмбеддинг токена на эмбеддинг, содержащий информацию о соседних токенах, вместо использования одинакового эмбеддинга для каждого токена вне зависимости от контекста.

Если бы мы знали, какие токены релевантны текущему, то узнать его контекст можно с помощью средневзвешенного — или, в общем случае, линейной комбинации — этих эмбеддингов.

Механизм внимания

До применения механизма внимания у эмбеддингов в последовательности нет контекста их соседей. Поэтому мы можем визуализировать эмбеддинг для слова light как линейную комбинацию:

Token Embedding	It's	Dark	Who	Turned	Off	The	Light
Weight	0	0	0	0	0	0	1

Применив механизм внимания, хотим узнать матрицу весов, которая позволит выразить эмбеддинг для слова light:

Token Embedding	It's	Dark	Who	Turned	Off	The	Light
Weight	0	0.3	0	0.1	0.1	0	0.5

В этот раз эмбеддингам, соответствующим наиболее релевантным для выбранного токена частям последовательности, присвоены веса большего размера.
Это должно обеспечить размещение в новом векторе эмбеддинга самого важного контекста.
Эмбеддинги, содержащие информацию об их текущем контексте, иногда называют контекстуализированными.

Результат и выводы ViT

На **больших наборах** данных (JFT-300M, ImageNet-21K) ViT **превзошел современные CNN** (ResNet, EfficientNet) по точности и вычислительной эффективности.

Ключевой инсайт:

- **Для эффективного обучения** трансформеру в CV нужно **очень много данных**.
- На маленьких наборах (только ImageNet) ViT проигрывал CNN, так как у CNN были более сильные индуктивные предпосылки (принцип локальности).
- Без них трансформеру нужно увидеть больше примеров, чтобы выучить 2D-структуру с нуля.

Открыл путь:
Если трансформер работает для классификации, то его можно адаптировать и для других задач.

DETR (2020) - Трансформер для детекции

DETR: End-to-End Object Detection with Transformers

Задача детекции объектов — не просто классифицировать, но и найти (bounding box) и классифицировать множество объектов на изображении

Проблема классических детекторов (Faster R-CNN, YOLO):

- Anchor Boxes: Требуют ручного выбора якорных рамок разных размеров и пропорций
- NMS (Non-Maximum Suppression): Постобработка для удаления дублирующихся предсказаний
  Эти эвристики сложны для настройки и не являются дифференцируемой частью обучения

Идея DETR:
Избавиться от этих сложностей, сведя задачу к прямой задаче предсказания множества.

Архитектура DETR:

CNN-бэкбон: Изображение пропускается через CNN (например, ResNet) для получения карты признаков меньшего размера, но с высокой семантикой.

Transformer Encoder-Decoder:

Энкодер: Принимает сплющенную карту признаков (последовательность векторов) с позиционными кодированиями. Вычисляет глобальные взаимодействия между всеми частями изображения.

Декодер: Его вход — это набор N обучаемых объектных запросов (object queries). Каждый запрос — это вектор, который ходит по выходу энкодера с помощью кросс-внимания и спрашивает: Где здесь объект?

Каждый из N запросов на выходе декодера должен **предсказать**: 1) Координаты бокса, 2) Класс объекта (или фоновый класс, если запрос не нашел объект). N фиксировано и заведомо больше максимального числа объектов на изображении.

Архитектура DETR:

Бинарное сопоставление и функция потерь: Главная инновация.

Обучение требует сопоставить N предсказаний с M настоящих объектов (где M < N).

DETR находит оптимальное (с наименьшей общей стоимостью потерь) паросочетание между предсказаниями и эталонными данными.

Несопоставленные предсказания считаются фоном.

Функция потерь (Hungarian loss) обучает и классификацию, и регрессию боксов напрямую, в единой дифференцируемой процедуре.

Самообучение (Self-Supervised Learning - SSL)

- **Проблема**: Для обучения современных моделей нужны миллионы размеченных изображений. Разметка — дорого и долго
- **Идея SSL**: Научить модель понимать данные без явных меток
- **Значение**: Позволяет обучать мощные модели на огромных объемах неразмеченных данных из интернета.

Суть проблемы: Голод на метки (The Label Hunger Problem)

Современные глубокие модели (особенно трансформеры, как ViT) имеют огромную емкость — десятки или сотни миллионов параметров. Чтобы раскрыть их потенциал и не допустить переобучения, им нужны огромные, качественно размеченные датасеты (например, ImageNet: 1.2 млн изображений с ручной разметкой по 1000 классов).

Стоимость: Разметка экспертами стоит огромных денег. Разметка одного изображения для сложных задач (семантическая сегментация в медицине, детекция дефектов) может стоить несколько долларов.

Время: Создание датасета размером в миллионы примеров занимает годы.

Масштабируемость: Мир генерирует терабайты неразмеченных данных ежедневно (видео с камер, фото в интернете, научные данные). Ручная разметка просто не поспевает за этим потоком.

Узкая специализация: Модель, обученная на ImageNet, плохо переносит знания на другие домены (например, с обычных фото на рентгеновские снимки или снимки со спутника), где нет больших размеченных датасетов.

Идея SSL: Обучение через понимание данных

Ключевая идея SSL — использовать внутреннюю структуру самих данных для создания обучающего сигнала.
Вместо метки «кошка» или «собака» модель учится отвечать на вопросы, сгенерированные автоматически из данных.
Это похоже на то, как младенец учится понимать мир, наблюдая за ним, а не по учебнику с готовыми ответами.

Фундаментальный принцип: Мы создаем псевдометки или цели непосредственно из данных с помощью заранее заданного правила.
Модель учится решать эту искусственную задачу, и в процессе формирует качественные внутренние представления данных.

Примеры в компьютерном зрении

Пример 1: **Inpainting** (Восстановление). Часть изображения маскируется (закрашивается), модель учится восстанавливать оригинал по контексту.

Пример 2: Rotation Prediction (Предсказание поворота). Изображение поворачивается на случайный угол (0°, 90°, 180°, 270°), модель учится определять этот угол. Чтобы справиться, ей нужно понять, что такое верх и низ у объекта.

Пример 3: Contrastive Learning (Контрастное обучение). Это современный SOTA подход: SimCLR (Simplified Contrastive Learning of Visual Representations) и MoCo (Momentum Contrast).

Контрастивное обучение (Contrastive Learning)

Суть: Похожее — ближе, разное — дальше. Модель учится распознавать, какие примеры являются вариациями одного и того же исходного объекта (позитивные пары), а какие — разными объектами (негативные пары).

Как это работает:

- Берем якорное (anchor) изображение x.
- Создаем из него две аугментированные версии x1 и x2 (изменение цвета, размытие, повороты и т.д.). Это — позитивная пара.
- Все остальные изображения в батче выступают как негативные примеры.
- Модель кодирует все изображения в вектор-представление.
- Цель функции потерь — минимизировать расстояние между x1 и x2 в пространстве представлений и максимизировать расстояние до всех негативных примеров.

Ключевые модификации и их вклад:

SimCLR (Simplified Contrastive Learning of Visual Representations) : Показала критическую важность сильных аугментаций и большой батч-сайз (чтобы иметь много негативов внутри батча). Проста и эффективна, но требует огромных вычислительных ресурсов.

MoCo (Momentum Contrast): Решила проблему большого батча. Использует очередь из представлений прошлых батчей, чтобы иметь много негативов, и momentum encoder для согласованного кодирования примеров в очереди. Это классический и эффективный подход.

BYOL и SimSiam: Пошли еще дальше — обучаются вообще без явных негативных пар. Они используют сложные архитектурные трюки (сиамские сети, stop-gradient, predictor). Это доказало, что негативные пары не всегда обязательны.

Результат

Энкодер обучается видеть смысловое ядро изображения, игнорируя несущественные аугментации. Кот в разных ракурсах и освещении будет иметь похожие представления.

MAE (Masked Autoencoder) и Generative Approaches

Краеугольный камень в CV

Суть: Дополни пропущенное

Прямая аналогия с BERT в NLP

Модель учится восстанавливать исходные данные из их частично замаскированной версии

Как работает MAE

- Изображение делится на патчи (patches), как в Vision Transformer (ViT).
- Большая случайная часть патчей (например, 75%) маскируется (заменяется на learnable [MASK]-токен).
- Видимыми остаются только 25% патчей. Их пропускают через энкодер.
- К представлениям видимых патчей и mask-токенов добавляют позиционные эмбеддинги и пропускают через легкий декодер.
- Декодер учится предсказывать пиксели (точнее, нормализованные значения) замаскированных участков.
- После предобучения декодер отбрасывается, а мощный энкодер используется для downstream-задач.

Плюсы

Высокая скорость обучения:
Энкодер работает только с 25% данных, что сильно экономит память и время.

Мощная семантика:
Чтобы точно восстановить пиксели утенка в пруду, модель должна понять, что такое «утка», «вода», «отражение», «перья» — т.е. выучить контекст и семантику всей сцены.

Масштабируемость: Идеально подходит для трансформеров и гигантских датасетов.

Значение и влияние SSL: парадигмальный сдвиг

Демократизация доступа к большим моделям:
Исследовательские группы и компании, не имеющие бюджета на разметку гигантских датасетов, могут предобучить модель на своих (или публичных) неразмеченных данных (например, на всех фото из Instagram или на всех медицинских снимках больницы).

Основа для Transfer Learning:
Модель, предобученная с помощью SSL на разнообразных данных, приобретает универсальные способности к представлению информации.
Ее затем можно быстро и с малым числом примеров дообучить (fine-tune) на конкретной downstream-задаче (классификация, детекция, сегментация). Это называется «pre-train and fine-tune».

Слияние NLP и CV:
Трансформеры + SSL (BERT/MAE) стерли границы между обработкой текста и изображений.
Единая архитектурная и методологическая парадигма для разных модальностей.

Значение и влияние SSL: парадигмальный сдвиг

Путь к мультимодальности:
SSL — естественный способ учиться на данных разного типа.
Методы вроде CLIP (контрастивное обучение на парах изображение-текст) — прямое развитие идей SSL.
Они учат модель связывать визуальные и текстовые представления без явной пометки того, какое изображение к какому классу относится.

Будущее — самообучающиеся фундаментальные модели (Foundation Models):
GPT, DALL-E, современные мультимодальные системы — все они в своей основе используют принципы SSL (предсказание следующего токена, маскирование, контрастивное обучение), обучаясь на колоссальных объемах неразмеченных данных из интернета.

Диффузионные модели (2022 - н.в.)

Класс мощных генеративных моделей, которые учатся создавать данные (изображения, аудио, видео) путем моделирования процесса их зашумливания и последующего очищения.

Их ключевая философия — разрушить данные, а затем научиться их восстанавливать.

Это привело к революции в качестве и доступности генерации изображений по тексту.

Принцип работы

Модель состоит из двух основных, зеркальных процессов:

- прямого (диффузии) и
- обратного (дениффузии)

Проще всего представить это как плавное превращение четкой картинки в статический шум (как на старом телевизоре), а затем обучение нейросети выполнять этот процесс в обратную сторону.

Прямой процесс (Forward Process / Diffusion)

Это детерминированный (**не обучаемый**) процесс, который происходит за **фиксированное количество шагов** T (обычно 1000 или более).

Цель: Постепенно, шаг за шагом, добавлять к исходному изображению x_0 небольшое количество гауссовского (нормального) шума.

Как работает: На каждом шаге t мы берем изображение с предыдущего шага x_{t-1} и по формуле добавляем новую порцию шума. Этот процесс описывается марковской цепью.

Важное свойство: Благодаря математическим свойствам нормального распределения, мы можем рассчитать зашумленное изображение x_t на любом шаге t напрямую из исходного x_0, без необходимости последовательно проходить все предыдущие шаги. Это критически важно для эффективности обучения.

Конец процесса: После T шагов (например, 1000) исходное изображение x_0 полностью превращается в x_T — чистый гауссовский шум, не содержащий никакой информации о начальной картинке.

Аналогия: Четкий рисунок на песке. Прямой процесс — это медленный ветер, который за T шагов равномерно заносит песчинками рисунок, пока не останется лишь гладкая, бесструктурная поверхность.

Обратный процесс (Reverse Process / Denoising)

Это стохастический и **обучаемый** процесс.

Цель — научиться инвертировать прямой процесс.

Задача: Имея на входе полностью зашумленное изображение x_T (просто шум) и номер шага t, предсказать, как выглядело бы изображение на шаге t-1, то есть убрать немного шума.

Роль нейросети: Эту задачу выполняет нейросеть (чаще всего U-Net).
Ей на вход подаются:

- Зашумленное изображение x_t на текущем шаге t.
- Номер шага t (в виде эмбеддинга).
- Условие (condition), например, текстовый запрос (в моделях типа Stable Diffusion).
- На выходе нейросеть пытается предсказать шум ε, который был добавлен на этом шаге, или, что эквивалентно, само изображение x_0 без шума.

Процесс обучения

- Берем реальное изображение x_0 из обучающей выборки
- Случайно выбираем шаг t (от 1 до T)
- С помощью формулы прямого процесса быстро получаем его зашумленную версию x_t.
- Подаем x_t и t в нейросеть и просим ее предсказать тот исходный шум ε, который мы добавили.
- Сравниваем предсказанный шум с реально добавленным (мы его знаем) через функцию потерь (обычно MSE).
  Градиентным спуском обучаем нейросеть делать это предсказание как можно точнее.

Аналогия: Теперь мы хотим научить художника восстанавливать рисунок. Мы много раз показываем ему случайные этапы занесенного песком рисунка (шаг t) и говорим: "Вот здесь ветер нанес вот такой слой песка (ε)". Художник (нейросеть) учится распознавать, как выглядел рисунок до этого слоя песка.

Процесс генерации (Sampling / Inference)

После обучения нейросети мы можем генерировать совершенно новые изображения.

Старт: Генерируем случайный шум x_T из нормального распределения. Это наша «чистая доска» после прямого процесса.

Итеративное очищение: Для каждого шага от t=T до t=1:

- Подаем текущий шум x_t и шаг t (и текстовый запрос) в обученную нейросеть.
- Нейросеть предсказывает шум ε, содержащийся в x_t.
- По специальной формуле (следующей из байесовских выводов) мы вычисляем x_{t-1} — немного более очищенную версию изображения. В формулу также добавляется небольшой элемент случайности (стохастичность), что повышает разнообразие результатов.

Финиш: После T шагов дениффузии мы получаем x_0 — новое, сгенерированное изображение, соответствующее текстовому запросу.

Ключевые архитектурные инновации и почему они стали State-of-the-Art

Stable Diffusion (2022) — прорыв в эффективности.

Проблема:
Работать в пространстве пикселей (например, 512x512x3 = 786 тыс. измерений) очень затратно.

Решение:
Stable Diffusion работает не с самими изображениями, а с их латентными (скрытыми) представлениями.
Специальный кодировщик (VAE) сжимает изображение в компактное латентное пространство (например, 64x64x4 = 16 тыс. измерений).
Диффузионный процесс идет в этом латентном пространстве, что в разы быстрее. После генерации латентный код декодируется обратно в высококачественное изображение.

Ключевые архитектурные инновации и почему они стали State-of-the-Art

Условная генерация (Conditional Generation) и механизм Cross-Attention.

Задача: Направлять процесс генерации согласно текстовому описанию.

Решение: Текст кодируется большими языковыми моделями (например, CLIP text encoder или T5).
Эти текстовые эмбеддинги подаются в U-Net через механизмы cross-attention.
На каждом шаге дениффузии нейросеть смотрит на текстовый запрос, чтобы решить, какие детали и где проявлять.
Это позволяет точно следовать промпту.

Масштабируемость и качество

Чем больше шагов T и чем лучше архитектура U-Net (например, с остаточными блоками и self-attention), тем качественнее и тем более детализированрезультат.

Обучение на гигантских наборах данных (LAION-5B) позволило моделям освоить невероятно широкие концепции и стили.

Известные реализации и результаты

DALL-E 2 (OpenAI):
Сочетание CLIP-модели для понимания текста и диффузионной модели для генерации изображений в пиксельном пространстве.

Imagen (Google):
Делает ставку на мощные текстовые энкодеры (T5) и каскадные диффузионные модели (генерирует сначала маленькое, затем большое изображение).

Stable Diffusion (Stability AI):
Наиболее влиятельная открытая модель. Ее латентный подход сделал высококачественную генерацию доступной для широкой публики и исследователей на потребительских GPU.
Порождает огромную экосистему моделей-лора (LoRA), контролнетов (ControlNet) и плагинов.

Midjourney:
Проприетарная модель, известная выдающейся художественной эстетикой и кинематографичностью результатов.

Итог

Диффузионные модели обеспечили прорыв в качестве, контроле и разнообразии генерации.

Их принцип «от шума к данным» оказался не только мощным, но и более стабильным в обучении по сравнению с предыдущими генеративными состязательными сетями (GAN).

Именно они лежат в основе текущей революции в генеративном ИИ.

Заключение и Выводы
Эволюция: Путь от простого к сложному

Ручной feature engineering (до ~2012): Эпоха корифеев.
Исследователи вручную проектировали детекторы признаков (SIFT, HOG, SURF) для нахождения углов, границ, текстур.
Проблема: Эти признаки были хрупкими, неадаптивными к специфике задачи и требовали огромных знаний. Система видела то, что в неё закодировал человек.

Автоматическое извлечение иерархических признаков (CNN, с 2012): Революция AlexNet.
Ключевая идея — пусть модель сама научится представлять данные.
CNN с их сверточными слоями автоматически выучивали иерархию:
от краев и цветов -> к текстурам -> к частям объектов (глаз, колесо) -> к целым объектам (лицо, машина).
U-Net (2015) стал пиком этой философии для сегментации, идеально сочетая детальность и контекст через skip-connections.

Заключение и Выводы
Эволюция: Путь от простого к сложному

Преодоление проблем оптимизации (ResNet, 2015):
Когда сети стали очень глубокими, возникла проблема исчезающих градиентов — сигнал обучения затухал.
ResNet ввел остаточные связи (skip connections), которые позволили градиентам перепрыгивать через слои.
Это был не просто прорыв в точности, а инженерный прорыв в обучаемости, позволивший строить сети глубиной в сотни и тысячи слоев.

Новые парадигмы (Transformer, SSL, с 2017):
Трансформеры принесли из NLP **механизм внимания** (attention), который позволяет модели динамически выбирать, каким частям изображения уделять больше внимания в зависимости от контекста, а не ограничиваться локальными окнами свертки.
Self-Supervised Learning (SSL) стал новой философией обучения: модель учит мощные представления без человеческих меток, предсказывая скрытые части данных (например, замазанные участки в MAE — Masked Autoencoder).
Это снимает проклятие дорогой разметки.

Ключевые современные тренды

Гибридные архитектуры (CNN + Transformer): Это не война, а синергия.
Пример: Vision Transformer (ViT) разбивает изображение на патчи и работает с ними как с последовательностью, отлично улавливая глобальный контекст.
Но ему не хватает локальной индуктивной базы (priors), присущей CNN (инвариантность к сдвигу, учёт локальных соседств).
Гибриды вроде **Convolutional Vision Transformer (CvT)** или **CoAtNet** используют CNN на ранних стадиях для эффективного извлечения низкоуровневых признаков, а трансформеры — на поздних для моделирования глобальных зависимостей.
Логика: получить лучшее из двух миров.

Ключевые современные тренды

Эффективность и демократизация (TinyML): Тренд на уменьшение и оптимизацию.
Это уже не только **SOTA-точность** на мощном GPU, а ещё:

- **Квантование**: Перевод весов из float32 в int8 без большой потери качества.
- **Прунинг**: Обрезка неважных нейронов или каналов.
- **Дистилляция знаний (Knowledge Distillation)**: Обучение маленькой студенческой сети на выходе большой учительской.
- **Цель**: Запускать современные модели на смартфонах, микроконтроллерах (IoT), embedded-устройствах. Это ключ к массовому промышленному внедрению.

Ключевые современные тренды

Самообучение (SSL) как новый стандарт:
После успеха моделей вроде BERT в NLP, SSL триумфально пришёл в компьютерное зрение (MoCo, SimCLR, DINO, MAE).
Суть: Модель учит универсальные репрезентации мира, решая претекстовые задачи (например, сравнение, восстановление) на миллиардах непомеченных изображений.
Затем эту предобученную модель можно эффективно дообучить (fine-tune) на небольшом помеченном датасете для конкретной задачи (классификация, детекция).
Это становится default pipeline, как раньше было предобучение на ImageNet.

Ключевые современные тренды

Мультимодальность:
Следующий логичный шаг — модели, которые **понимают связь между разными типами данных**.

CLIP (2021) — эталонный пример: обучается на миллиардах пар изображение-текст, учась сопоставлять их в общем пространстве признаков.
Это позволяет классифицировать изображения по текстовым промптам без дообучения.

DALL-E, Imagen, Stable Diffusion — это генеративные наследники CLIP, которые создают изображения из текста.

Будущее: Единые модели-универсалы, понимающие мир комплексно.

Ключевые современные тренды

Генеративные модели:
Смещение фокуса с распознавания на создание и управление.

Актуальные чемпионы:

- **Диффузионные модели (Stable Diffusion)**:
  Генерируют изображения, итеративно удаляя шум, следуя текстовому описанию.
  Революция в креативных инструментах и синтезе данных.

- **GAN (Generative Adversarial Networks)**:
  Пионеры области, сейчас часто уступают диффузионным моделям в стабильности и качестве, но остаются ключевыми для задач переноса стиля, супер-разрешения, редактирования.

Выводы

Незыблемый фундамент: CNN, ResNet, U-Net — это таблица умножения.
Без глубокого понимания, как работают свертки, пулинги, остаточные блоки, skip-connections и функция потерь, невозможно понять новейшие архитектуры, которые часто используют эти элементы как базовые кирпичики.

CV динамично эволюционирует от узкоспециализированных инструментов к универсальным, мощным и доступным системам.

Будущий специалист должен совмещать прочное знание основ с навыком постоянного, активного обучения по первоисточникам.

Быть в потоке: arXiv.org — ваш главный источник.
Ждать публикаций в журналах — значит отстать на год.