Эссе
«Визуализация нейронных сетей»

Сенин Александр

14 июня 2020 г.

Аннотация

В последние годы произошел значительный прорыв в обучении глубоких нейронных сетей (deep neural networks). Например, успехи в обучении сверточных сетей (convolutional networks) для работы с изображениями. При этом несколько отстает прогресс в глубоком понимании, как работают нейронные сети, например, какие вычисления они осуществляют на промежуточных слоях. Успехи в применении глубоких сетей можно усилить, если разобраться в сути их работы, а для этого необходимо научиться их визуализировать и интерпретировать.

Введение

Опишем сперва преимущества, которые можно получить, если научиться «наблюдать» за нейронными сетями:

- Визуализация позволяет получить понимание, как работают нейронные сети, почему они так работают.
- Известно, что нейронная сеть осуществляет преобразование из одного признакового пространства в другое. Можно считать, что каждый слой в сети преобразует одни признаки в другие. Зачастую бывает полезно «посмотреть» на эти преобразования, а затем может быть использовать промежуточные признаки, которых нет ни во входном, ни в выходном описании. Другими словами, хочется понимать, что происходит на промежуточных слоях сети, иметь интуицию, с какими абстракциями взаимодействуют промежуточные слои.
- Визуализация позволяет увидеть проблемы в данных и моделях. Например, можно понять, когда сеть неправильно обучена, или почему сеть ошибается на конкретных объектах.

Что можно визуализировать в нейронной сети? Перечислим (детали позднее):

- Параметры. Например, можно «взглянуть» на фильтры в сверточных слоях (convolutional layers), другими словами, на свертки, рассмотрев их как изображения.

- Активации нейронов. Например, можно найти изображения в данных, на которые функция активации нейрона выдает максимальное значение. Можно отследить, на какую именно часть изображения активировался нейрон.

- Качество «кодов», которые сеть строит для конкретного входа. Например, можно построить по датасету карту изображений, двумерное представление в карте для каждого изображения можно получить с помощью методов понижения размерности (например, t-SNE) из признакового описания, полученного как выход одного из последних слоев сети.

- Производные по входу. Напомним, что можно считать производные не только по весам сети, но и по входам. Тогда, например, для каждого входа (пикселя) можно вычислить градиент и построить новое изображение.

- Входы, максимизирующие ответ. Например, можно, умея вычислять градиенты по входам, построить изображение, на котором будет достигаться максимум ответа (например, максимум вероятности какого-нибудь класса для задачи классификации).

Визуализация параметров

Напомним общую логику построения сверточных нейронных сетей.

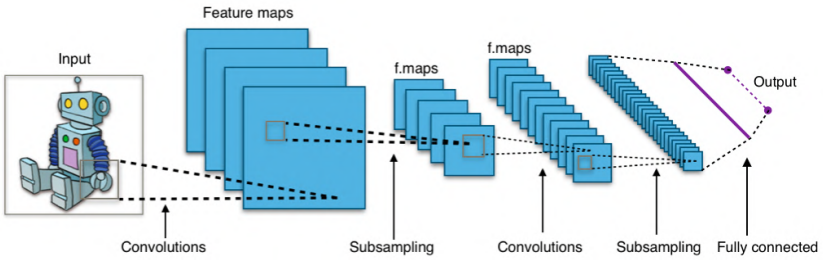

Рис. 1: Типовая архитектура сверточной нейронной сети. Источник [1]

Основные блоки такой сети:

- Свертка (convolution). Слой свёртки содержит для каждого канала свой фильтр, по сути результат работы этого слоя эквивалентен применению к входу операции свертки с некоторым ядром. Веса этого ядра свёртки неизвестны и являются параметрами модели. Обычно выходы слоя свертки называют картами признаков (feature maps). Интуиция применения слоя свертки та же, что

и применение свертки для классических методов обработки и распознавания изображений — для каждого входа из карты признаков учесть информацию о его соседях. Ключевое преимущество — сверточный слой содержит сравнительно небольшое количество параметров, по сравнению с потенциально возможными размерами входого изображения или входной карты признаков. Обычно входная карта признаков состоит из нескольких каналов (три цветовых канала в случае исходного входного цветного изображения), поэтому используют многоканальные свертки, причем обычно число каналов растет с глубиной слоя, а значит одному слою соответствует несколько сверток.

- Пуллинг (pooling) или слой подвыборки (subsampling layer). Идея этого слоя — добавить в модель нелинейности, при этом сжав карту признаков. Например, можно группу пикселей (обычно размера 2×2) уплотнить до одного пикселя, взяв максимальное значение среди пикселей в группе. Интуиция применения этого слоя — уплотнить карту признаков, избавившись от избыточности информации. Другими словами, если свертка с некоторым центром приняла большое значение, то знание значений соседних сверток избыточно для модели.

- Полносвязная нейронная сеть (fully connected neural network). Идея этого блока — перейти от пространственной структуры изображения к новым уровням абстракции, чтобы получить компактное признаковое описание исходного входного изображения (часто такое описание называют кодом).

Первый сверточный слой преобразует именно входное изображение, поэтому визуализация обученных сверток первого слоя легко интерпретируема — можно представить результат применения каждой свертки к изображению и понять, что она выделяет на изображении. Для второго сверточного слоя такая интепретация уже теряется, но его визуализация все равно позволяет судить о том, насколько хорошо обучена сеть. Для хорошо обученных сетей на изображении сверток должна прослеживаться структура, а не случайный шум.

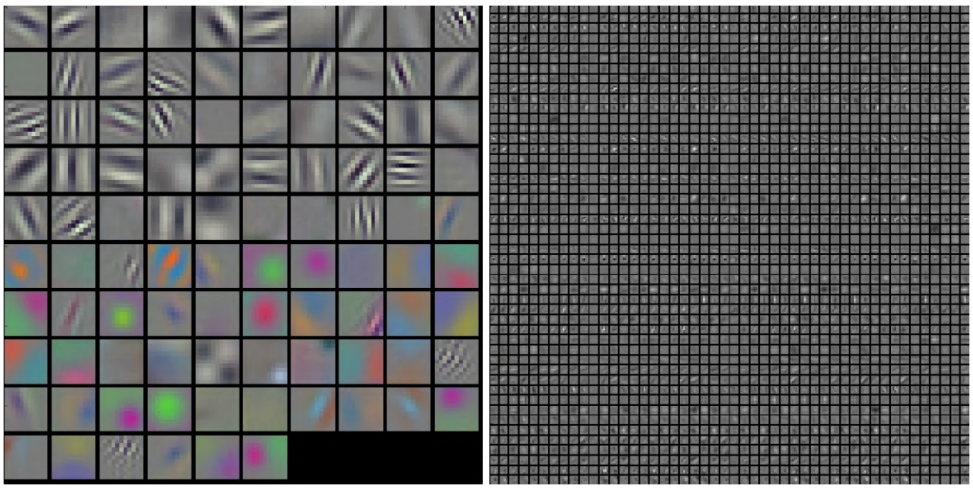

Рис. 2: Слева типичные изображения сверток первого сверточного слоя, справа второго сверточного слоя обученной сети AlexNet. Источник [2]

Для примера на Рис.2 разграничение сверток на почти монохромные и цветные является следуствием ее архитектуры: AlexNet содержит два раздельных «потока».

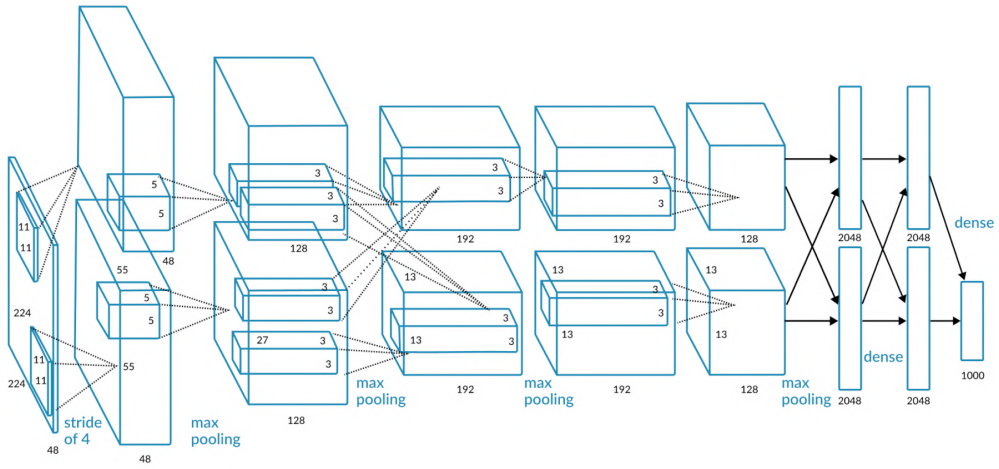

Рис. 3: Архитектура AlexNet. Обратите внимание на два параллельных «потока». Источник [3]

Еще пример визуализации сверток первых слоев:

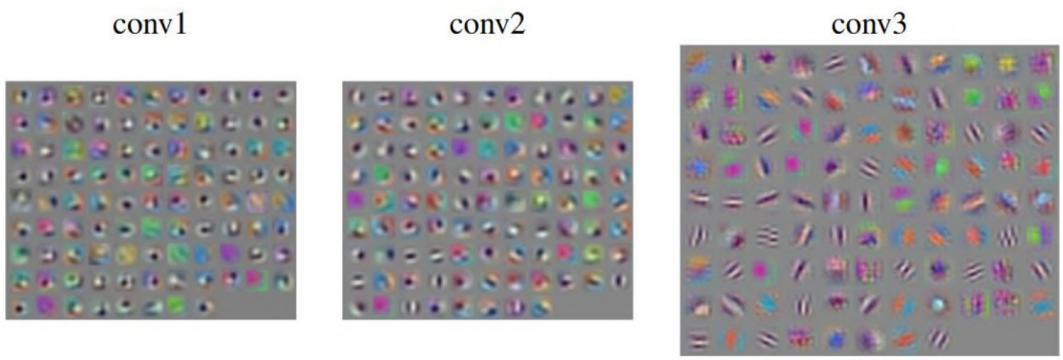

Рис. 4: Сеть обученная на датасете ImageNet. В этом случае фильтры, похожие на фильтры Габора появляются лишь на третьем слое (в случае обученной сети AlexNet их можно заметить уже на первом слое). Источник [4]

Основное, что можно понять по такой визуализации — насколько хорошо обучена сеть. Если изображения сверток недостаточно четкие, значит сеть недообучена. Если же изображения слишком шумные, значит сеть либо недообучена, либо переобучена (нужно обратить внимание на регуляризацию). Ключевой недостаток — низкая интерпретируемость для глубоких слоев. Глубокие слои работают с некими более высокими абстракциями, такая визуализация вряд ли поможет лучше понять, что происходит на глубоких слоях.

Стандартные средства в признаковых пространствах

Известно, что выход последнего слоя сверточных сетей (обычно полносвязный) можно рассматривать, как компактное признаковое описание изображения. К таким кодам изображений можно применить стандартные методы машинного обучения, таким образом визуализировать работу последнего слоя.

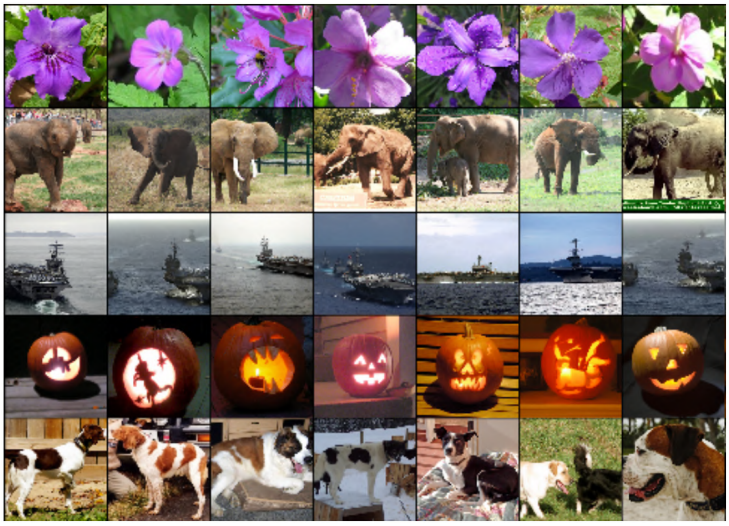

Рис. 5: В первом столбце изображения из датасета ILSVRC-2010, в остальных столбцах ближайшие соседи к этим изображениям. Под близкими понимаются изображения с наименьшим евклидовым расстоянием между признаковыми представлениями, полученными из последнего слоя сети. Если получаются действительно близкие по смыслу изображения (изображения одного цветка, слонов и т.д.), значит сеть хорошо «кодирует» изображения. Источник [5]

Рис. 6: Карта объектов, полученная с помощью метода понижения размерности для визуализации t-SNE из признаковых описаний изображений, полученных из последнего слоя сверточной сети. Источник [6]

Анализ активации нейронов

Напомним, что сверточные слои преобразуют одни карты признаков в другие, поэтому можем рассматривать эти карты, когда через них передается сигнал, как изображения. Такие изображения часто называют картами активаций. Для ReLU сетей, активации обычно сначала выглядят плотно и кучковато, но при дальнейшем обучении они становятся более разреженными и локализованными. С помощью визуализации можно заметить один из опасных подводных камней функции активации ReLU — некоторые области после активации состоят полностью из нулей, что может указывать на «мертвые» фильтры (фильтры суть свертки) и констатирует, что скорость обучения (learning rate) слишком велика.

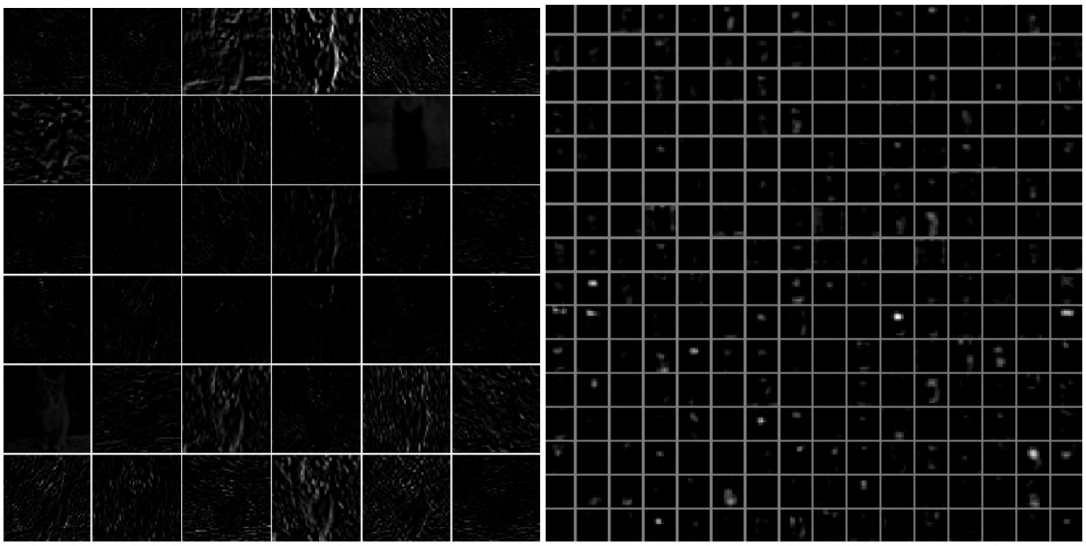

Рис. 7: Типичная картина активаций на первом свертночном слое (слева) и на 5 сверточном слое (справа) сети AlexNet, которая анализирует изображение кошки. Каждый квадрат показывает карту активаций, соответствующую какому-либо фильтру. Заметьте, что активации на глубоких слоях разрежены (большинство значений нули, на картинке это соответствует черному цвету) и достаточно локализированы. Интуиция здесь заключается в том, что первые слои «улавливают» примитивные паттерны, например, горизонтальные линии, а более глубокие слои «ищут» абстракции, которые появляются на изображении более точечно, например, лицо человека. Источник [2]

Проблема здесь та же — если первые слои при такой визуализации можно интерпретировать (так, например, на Рис.7 слева можно даже «разглядеть» очертания кошки и понять, что слой «ищет» что-то похожее на горизонтальные и вертикальные линии), то визулизация активаций более глубоких слоев не позволяет лучше понять, что на них происходит. Лучшее, что можно увидеть на такой визуализации — качество обучения: насколько четкие карты мы видим, насколько мало «мертвых» фильтров, для которых карта активации состоит из нулей (см. Рис.7 справа).

Работу нейронов в промежуточных, средних слоях можно проанализировать косвенно. Например, можно подать на вход нейросети большой датасет изображений, а затем вычленить изображения, максимизирующие активацию какого-нибудь нейрона (напомним, что под активацией нейрона понимаем значение функции активации,

например, ReLU или сигмоида). Таким образом, можно понять на какой шаблон активируется нейрон.

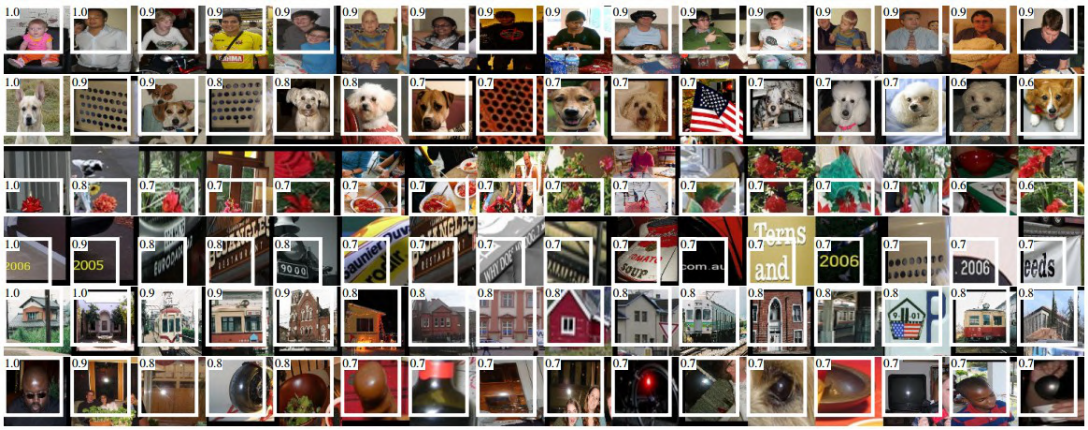

Рис. 8: Изображения, которые имеют наибольшие значения функции активации для некоторых нейронов в пятом сверточном слое (после пуллинга) в AlexNet. Если прослеживается логика, значит нейрон хорошо выделяет шаблон. Например, нейрон, соответствующий второму ряду, активируется на шаблон крупных черных пятен на светлом фоне (например, глаза и нос у собаки). Белые рамки на изображениях в данном случае — границы объектов, нейросеть изначально решала задачу детекции. Источник [2]

Рассмотрим другой подход к анализу активации нейронов. Мы помним логику работы сверточной сети — постепенно преобразовать огромной размерности признаковые описания изображений к признаковым описаниям все меньшей размерности с помощью операций свертки и пуллинга. Предлагается (см. [7]) придумать способ по компактной признаковой карте получить изображение в исходном признаковом пространстве, другими словами, научиться восстанавливать исходное изображение по его сжатому описанию, полученному сетью с помощью сверток и пуллинга. Для этого нужно решить, какие операции будем считать «обратными» для сверток и пуллинга (формально эти операции не будут являться обратными). Для анпуллинга (unpooling) можно просто запомнить позиции, в которых находился максимум, а затем построить признаковую карту нужной размерности, положив в эти позиции максимум. Для «обращения» свертки (deconvolution) можно просто взять транспонированную матрицу весов соответствующего сверточного слоя (мы помним, что свертка — линейная операция, применение свертки к изображению можно представить в виде умножения соответствующей ей матрицы весов на вектор изображения (изображения в сеть мы подаем в вытянутом в вектор виде)), тогда умножение на такую матрицу даст признаковую карту нужной размерности.

Теперь, если мы знаем, как «обращать» ход работы сверточной сети, можем выбрать интересующий нас нейрон в сети, выбрать изображения, на которые нейрон дает максимальную активацию (в оригинальной статье [7] берется топ-9 таких изображений), получить признаковую карту входного изображения на слое, содержащем выбранный нейрон, занулить активации неинтересующих нас нейронов, а затем «обратить» работу сети, в результате получить на выходе изображение, характери-

зующее активацию выбранного нейрона.

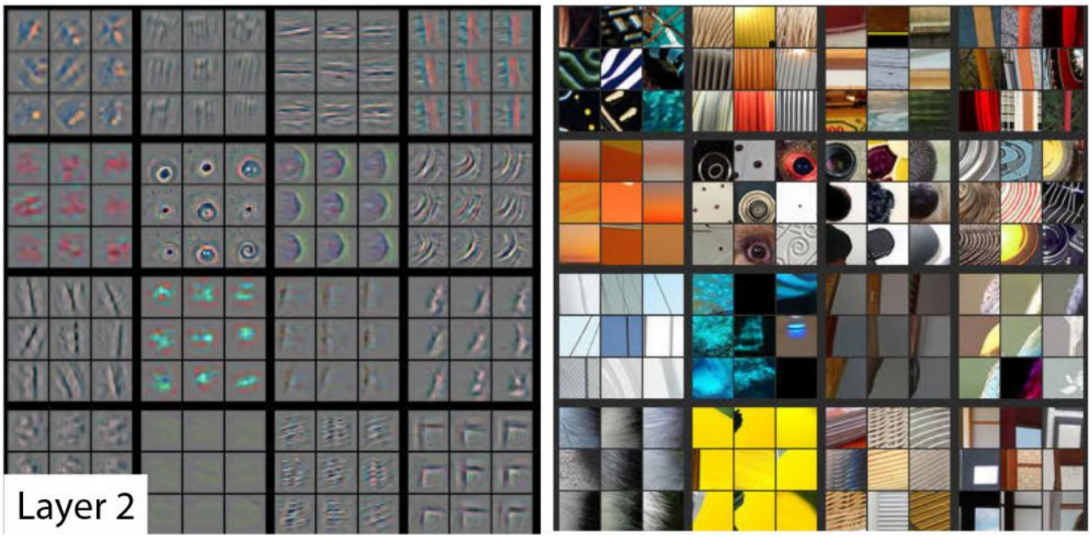

Рис. 9: Здесь был взят второй слой сети, выбрано 16 нейронов, для каждого выбрали 9 изображений, на которых у нейрона наибольшая активация, затем «обратили» работу сети по принципу, описанному выше, получили выход «в терминах» изображений. Источник [7]

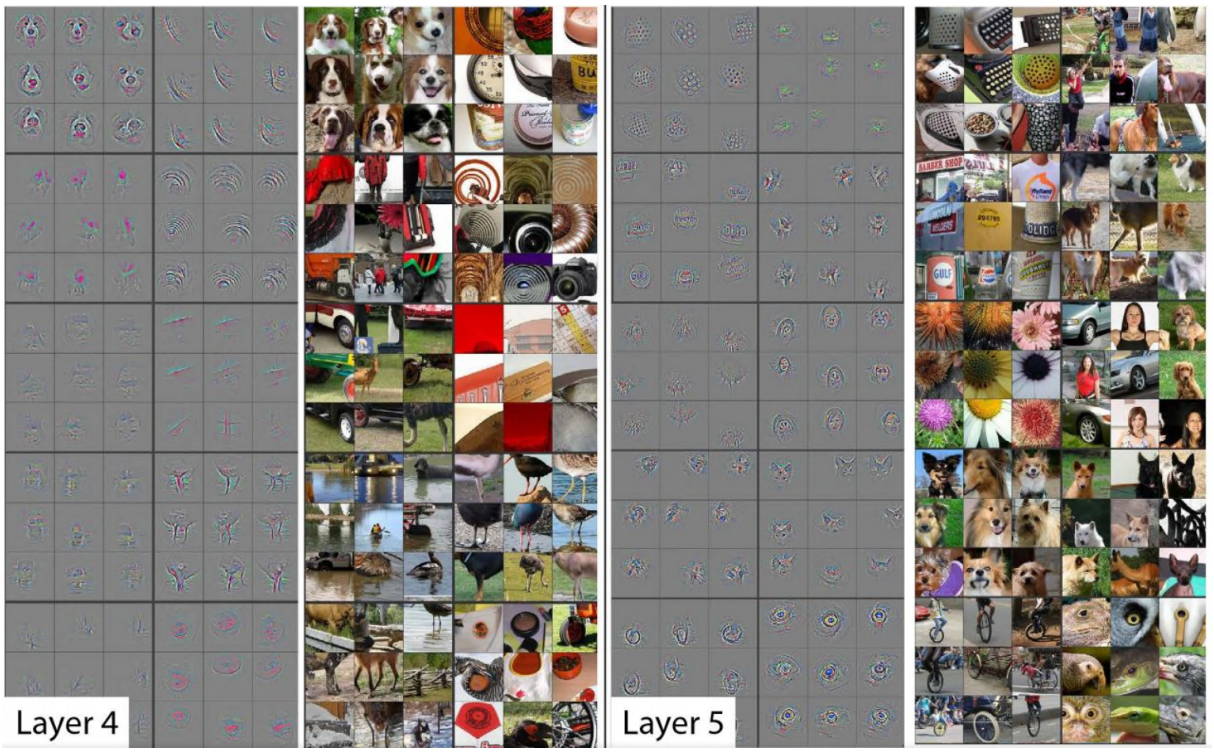

Рис. 10: Описанный подход хорошо визуализирует «на что» активируются нейроны не только первых слоев, но и более глубоких слоев, например, четвертого и пятого слоев. Источник [7]

Чувствительность к удалению

Можно визуализировать работу сети косвенно, не вникая в ее архитектуру, зная задачу, которую она решает, и меняя входные данные. Простая идея — закрыть на изображении какую-нибудь область и посмотреть, как сеть на это отреагирует. Другими словами, анализируем чувствительность к удалению (occlusion sensitivity). Будем последовательно сдвигать серый квадрат по изображению, тем самым скрывая от сети часть изображения, и наблюдать за изменениями в ее ответах.

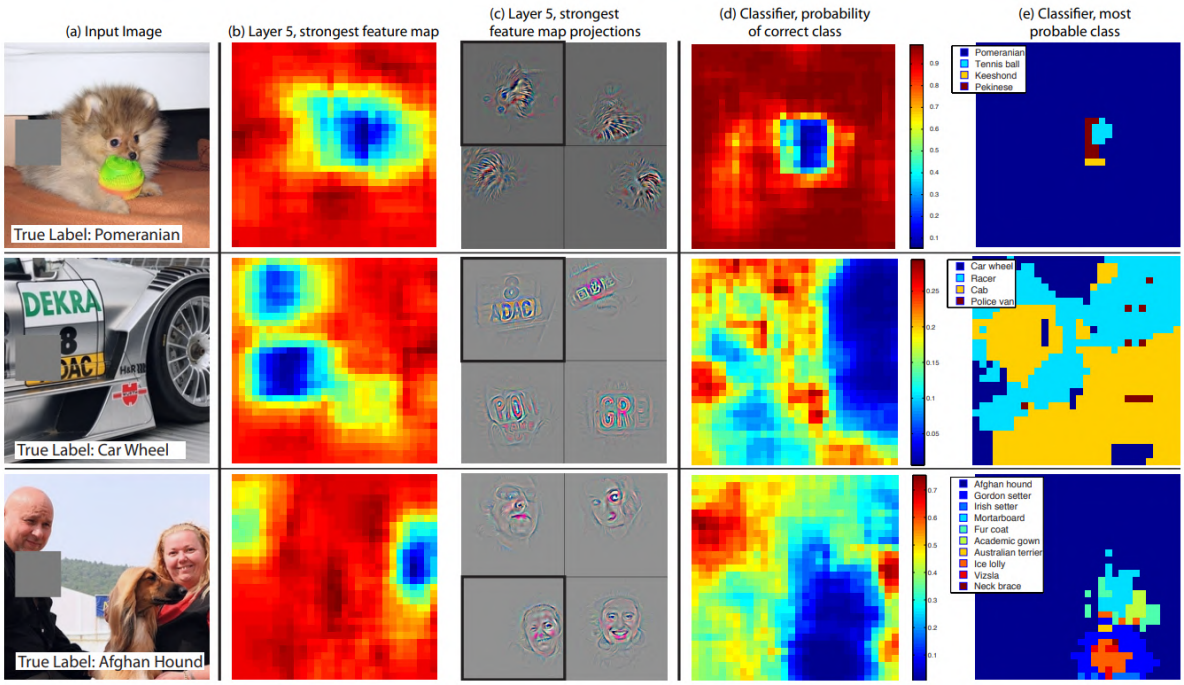

Рис. 11: Последовательно разберем, что можно визуализировать таким анализом. Нейросеть решает задачу классификации по изображению. В выборке были выбраны три изображения (a), затем серым квадратом «прошлись» по изображению, каждый раз проанализировав реакцию сети. Для пятого слоя сети (конкретно в этом случае был выбран пятый слой, потому что это последний слой, сохраняющий пространственную логику изображения, далее в сети идут полносвязные слои) каждый раз запоминали суммарную активацию, получили карту (b). В столбце (c) визуализирована активация пятого слоя (по принципу «обращения» работы сети, описанному выше — только в этот раз нас интересуют все нейроны, которые были активированы): черным обведена активация оригинального изображения, для сравнения представлены активации других похожих изображений. В столбце (d) представлена карта вероятностей истинного класса. В столбце (e) карта наиболее вероятного класса. Например, если закрыть от сети серым квадратом морду собаки (первая строка), то снизится общая активация пятого слоя (b), значительно упадет вероятность истинного класса (d), сеть будет выбирать другой класс (например, теннисный мяч) (e), что подтверждается визуализацией активаций пятого слоя (c) — слой чувствителен к шерсти на морде собак. Источник [7]

Визуализация градиентов

Мы помним ключевой принцип в обучении нейронных сетей — метод обратного распространения ошибки (backpropagation). По сути, сеть — очень сложная функция от входа (например, от изображения), и мы можем, подав на вход некоторое изображение, произведя прямой проход (forward pass), вычислить градиенты по входам (например, по пикселям). Тогда у нас будет набор значений соответствующий изображению (для каждого пикселя будет свое значение градиента). Разумеется, градиенты могут быть отрицательными, поэтому можно взять их модули и отмасштабировать их, чтобы получить полноценное изображение. Полученные изображения часто называют картами заметности (saliency maps).

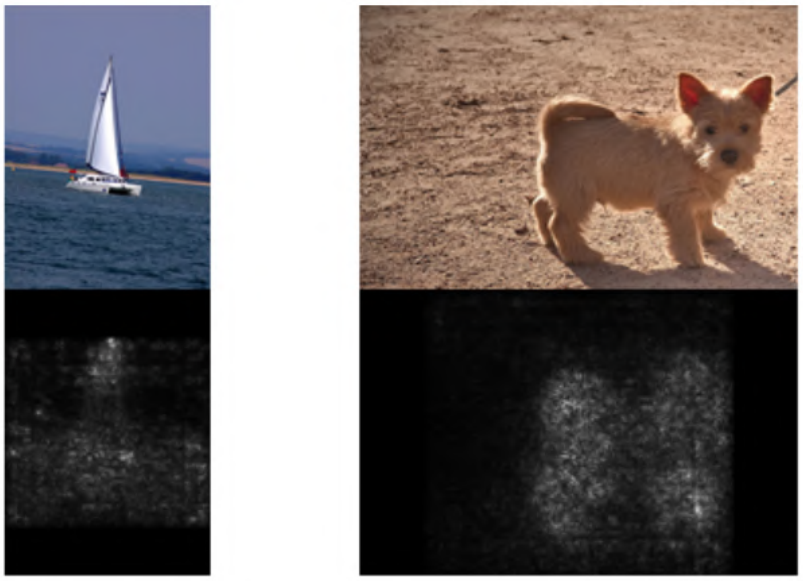

Рис. 12: «Saliency maps», полученные с помощью обратного распространения ошибки. Интуиция следующая: если какой-то пиксель важен для сети, то модуль градиента для этого пикселя будет большим. Интересное наблюдение: для изображения яхты (слева) для сети важна не только яхта, но и вода. Таким образом, сеть может ошибаться, если ей подать на вход изображение с объектом в нетипичных для объекта условиях. Еще одно подтверждение, почему визуализация важна для понимания устройства нейросетей. Источник [8]

У такой визуализации есть явный недостаток — карты получаются шумными. Существуют подходы, позволяющие снизить шум на картах, улучшив качество визуализации. Например, в статье [9] предлагается избавиться от шума на карте с помощью добавления шума к изображению. К входным пикселям применялся случайный Гауссов шум с нулевым матожиданием и дисперсией σ^2. Для конкретного изображения эта операция повторялась пятьдесят раз, а затем пятьдесят полученных карт усреднялись.

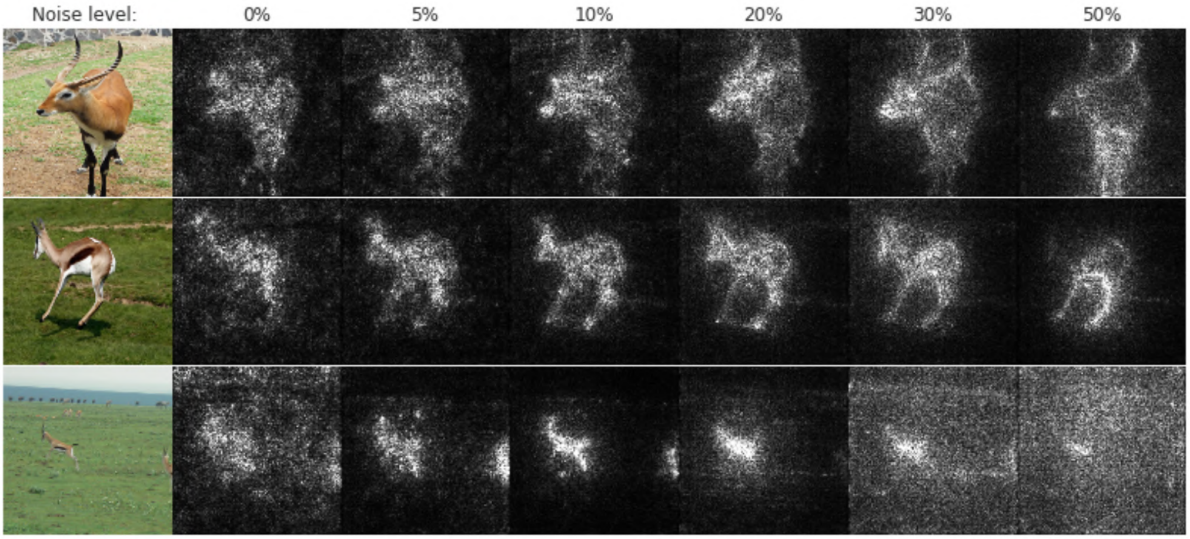

Рис. 13: «Saliency maps», построенные для трех изображений газелей из датасета ImageNet. Под уровнем шума (noise level) здесь понимается величина $\sigma/(x_{\max} - x_{\min})$. Явно видно, как добавление шума к входам позволяет получить менее шумные карты при некотором уровне шума. Источник [9]

Можно пойти еще дальше и объединить идею с визуализацией градиентов и с «обращением» свертки (см. [4]).

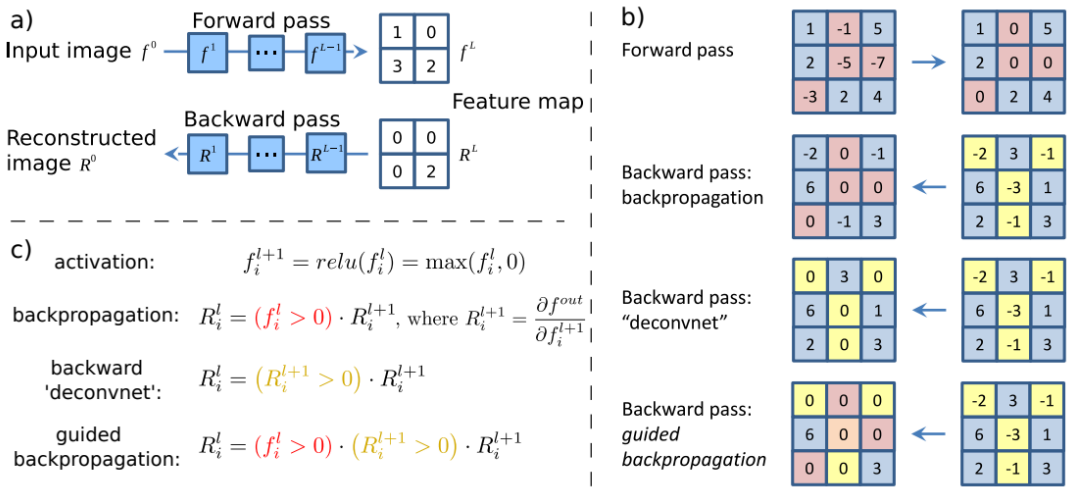

Рис. 14: Вспомним принцип работы сети с функцией активации ReLU. В случае классического обратного распространения ошибки: при прямом проходе (b, Forward pass) просто зануляем отрицательные значения, при обратном проходе, обнуляем позиции, где были отрицательные при прямом проходе (b, Backward pass: backpropagation). В случае «deconvnet» сетей (тех, где «обращаем свертку») в обратном проходе обычно просто применяется ReLU — зануляем отрицательные значения (b, Backward pass: "deconvnet"). Можно объединить эти подходы, и распространять градиенты ошибок назад, зануляя и отрицательные значения, и те позиции, где были отрицательные значения при прямом проходе (b, Backward pass: guided backpropagation). Источник [4]

Подход на Рис.14 называется «guided backpropagation». Тогда можем исполь-

зовать такое распространение градиентов назад для визуализации, по аналогичному принципу, что и при построении карт заметности (saliency maps): можем распространить назад активацию одного нейрона после прямого прохода с некоторым входным изображением, причем обратное распространение будем осуществлять по правилам с Рис.14.

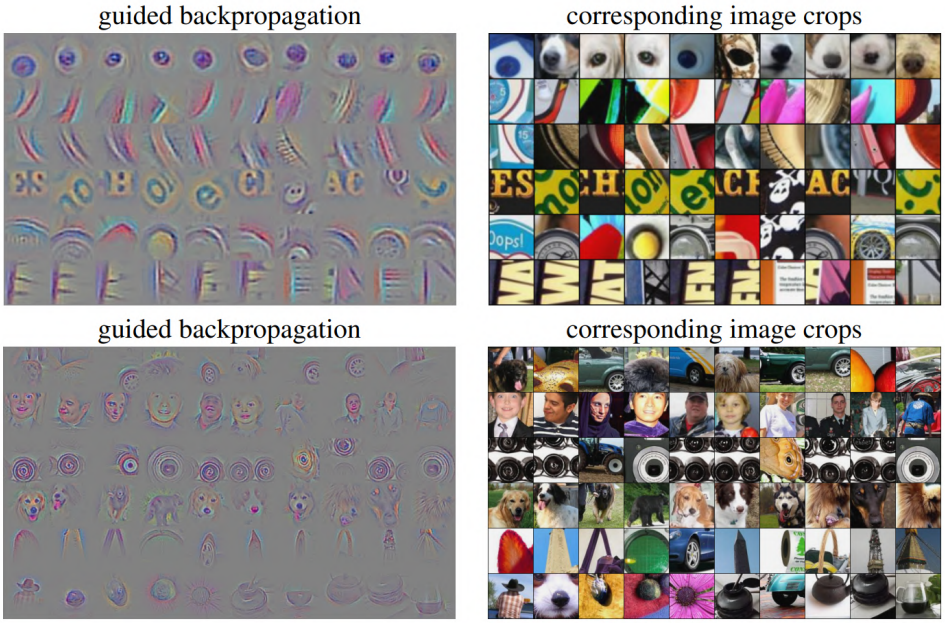

Рис. 15: Были выбраны шестой (вверху) и девятый (внизу) сверточные слои сети, обученной на ImageNet. Каждая строка здесь соответствует своему фильтру. Входные изображения были выбраны, как топ-10 изображений с наибольшей активацией фильтра. Изображения справа получены из исходных изображений вырезанием фрагментов, соответствующих построенным картам. Получили достаточно четкие карты, по которым можно судить, на какие паттерны активируется конкретный фильтр. Источник [4]

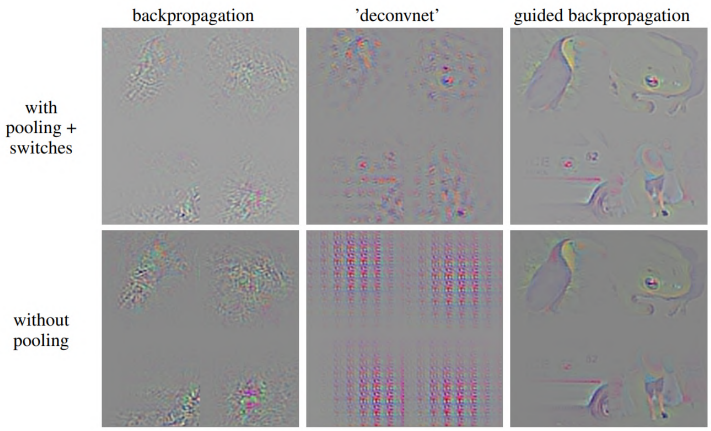

Рис. 16: Сравнение описанных методов визуализации. Выбрали четыре изображения, построили карту градиента, использовали «обращение» сверточной сети и метод «guided backpropagation». Авторы оригинальной статьи использовали сеть без пуллинга, поэтому для честности сравнения они демонстрируют работу и с использованием пуллинга (первая строка), и без него (вторая строка). Превосходство «guided backpropagation» явно видно. Источник [4]

Генерация изображений, максимизирующих активацию нейрона

Мы знаем, как считать производные по входам. Можно решать следующую задачу: сгенерировать, учитывая эти производные, изображения, максимизирующие активацию какого-нибудь выделенного нейрона.

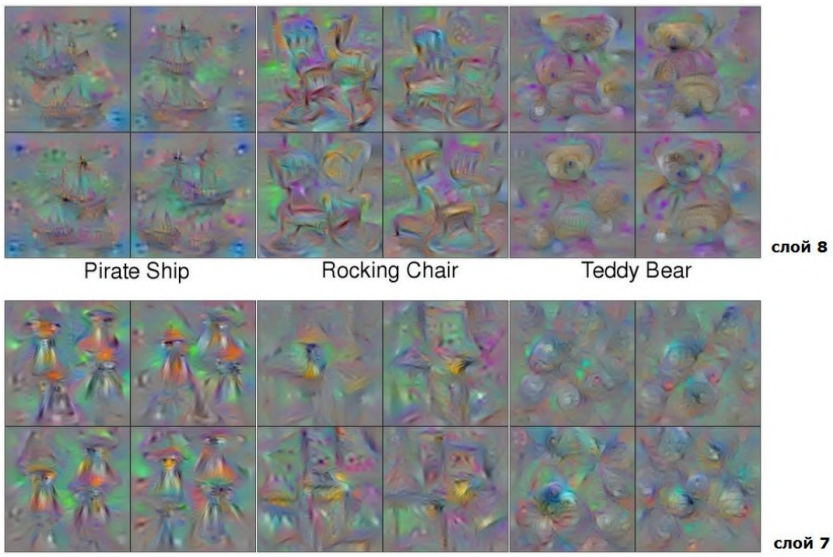

Рис. 17: Результат генерации изображений, максимизирующих активацию нейрона. Интересное наблюдение: восьмой слой — последний, нейроны с этого слоя непосредственно соответствуют классам, причем на таких сгенерированных изображениях можно увидеть паттерны соответствующего класса (например, корабли, стулья, медведи). Источник [10]

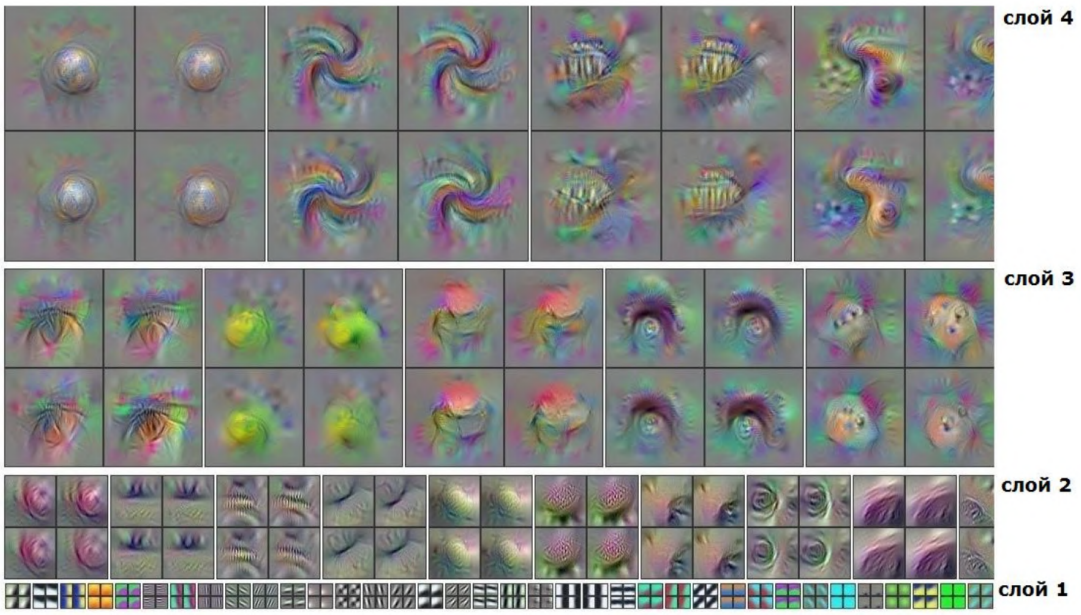

Рис. 18: Аналогично сгенерированные изображения для остальных слоев. Источник [10]

Можно улучшать постановку этой задачи, например, добавить регуляризацию.

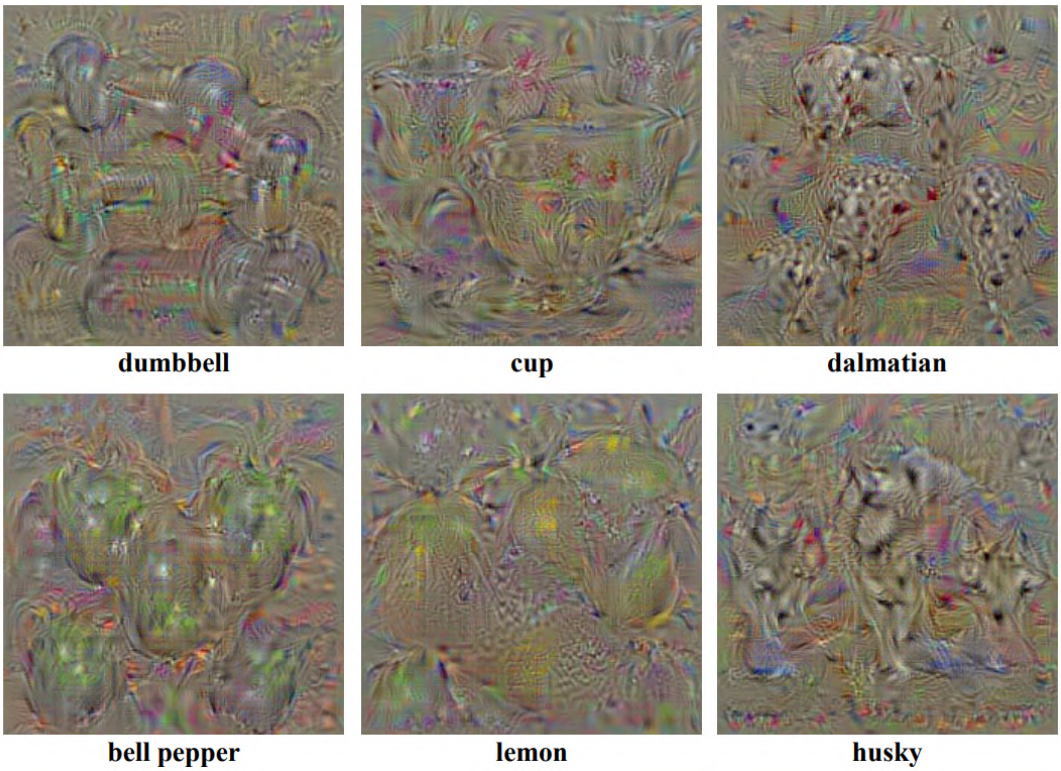

Рис. 19: Сгенерированные изображения с использованием регуляризации. Источник [11]

Существуют еще более продвинутые подходы к решению нашей оптимизационной задачи, как, например, в статье [12]. Если коротко, то нужно грамотно работать с регуляризацией, штрафовать за разницу соседних пикселей, можно размывать изображение через k шагов, можно построить априорное распределение на генерируемое изображение, можно решать оптимизационную задачу градиентным спуском в другом, декоррелируемом пространстве. Однако, ключевой принцип здесь не меняется — по-прежнему смотрим, на чем активируется нейрон, используя обратное распространение.

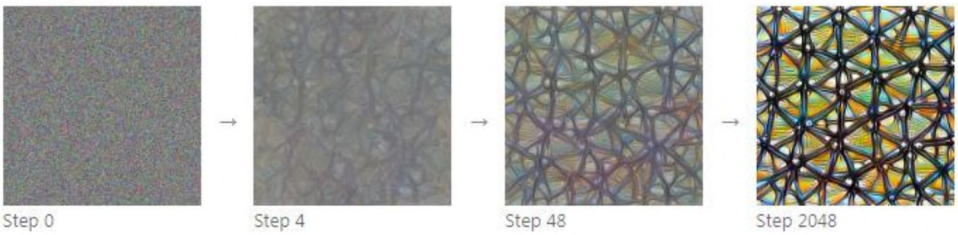

Рис. 20: Генерация изображений, на которых максимально активируется нейрон, с продвинутой оптимизацией. Источник [12]

В той же статье [12] авторы обобщают оптимизационную задачу, чтобы можно было визуализировать «на что» максимально активируется не только один нейрон,

но канал, слой, нейроны последнего слоя до активации (обычно такие выходы до активации называют «class logits»), нейроны последнего слоя после активации, то есть непосредственно вероятности классов (обычно, после применения softmax в качестве функции активации).

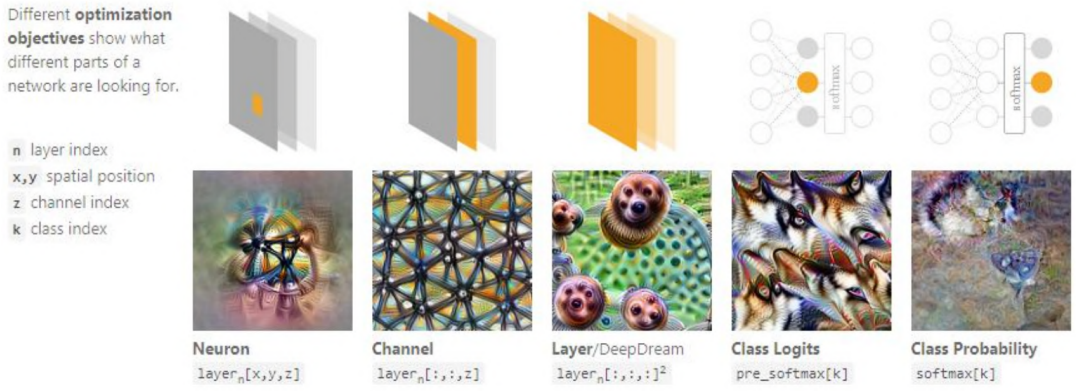

Рис. 21: Генерация изображений, на которых максимально активируется нейрон, канал, слой, выходы последнего слоя до активации и после активации, с продвинутой оптимизацией. Интересное наблюдение: активация последнего слоя с помощью softmax, другими словами, получение вероятностей классов, заметно ухудшает качество визуализации. Источник [12]

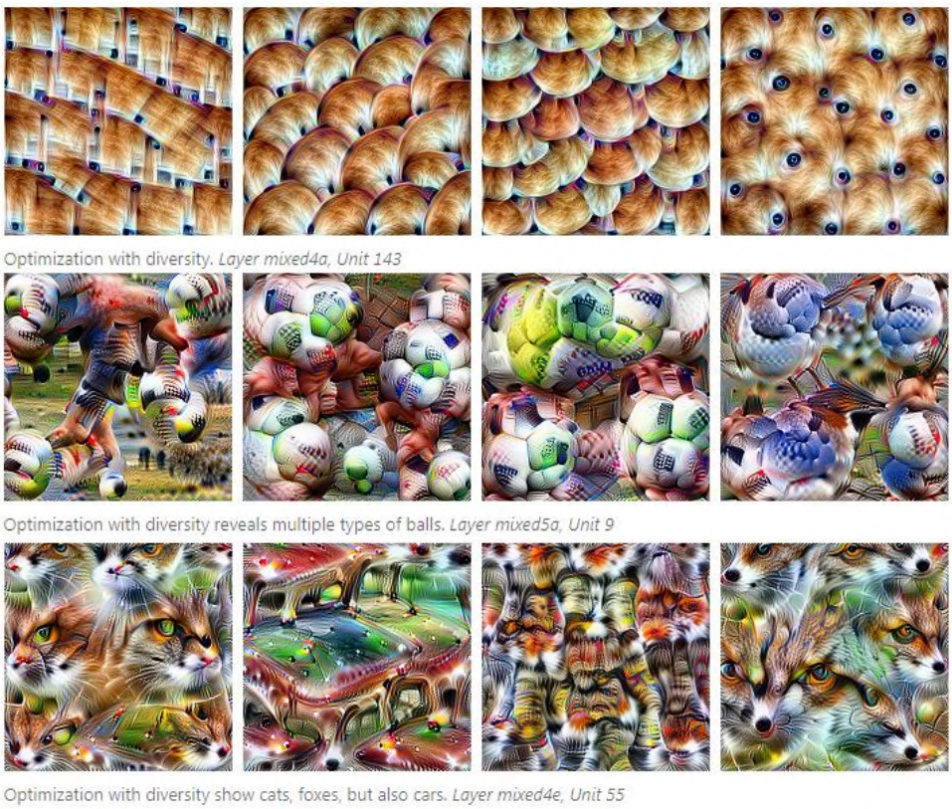

Рис. 22: К замечанию о необходимости генерации изображения несколько раз. Например, в третьей строке визуализируется активация слоя, из которой ясно, что слой активируется на кошек, лис, но еще и на машины. Источник [12]

Обратим внимание, что в случае решения нашей задачи максимизации активации итерационно, например, градиентным спуском, стартовое изображение задается случайным шумом (см. Рис. 20). Понятно, что для разных начальных изображений мы будем получать разные конечные решения. Поэтому для более глубоко понимания, на что активируется нейрон или слой, стоит повторить визуализацию несколько раз (см. Рис. 22).

Все в той же статье [12] авторы объединяют оптимизационные задачи сразу для нескольких нейронов. Другими словами, они строят изображение, на которое больше всего активируется и один, и второй нейрон.

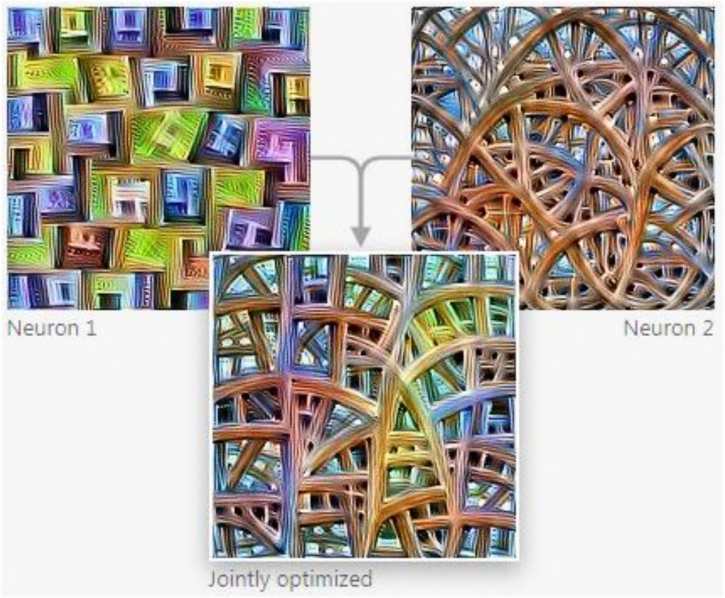

Рис. 23: Совместная оптимизация для двух нейронов. Источник [12]

Следующая идея (см. [13]): мы помним, что обычно в сверточной сети каждый слой соответствует не одному, а нескольким фильтрам (сверткам), так, что с прохождением сигнала по сети, увеличивается число каналов, с одновременным уменьшением размерности карты признаков (см. типовую архитектуру на Рис. 1). Значит мы можем выбрать какой-нибудь фрагмент на изображении, а затем отсортировать соответствующие фрагменту нейроны из разных фильтров (каждый фильтр дает свой канал) по невозрастанию активаций. Просто значения этих актитваций не слишком информативны, но можно построить, например, «семантический словарь». В этом словаре мы визуализируем активацию нейрона с помощью сгенерированного изображения (например, как на Рис. 20), полученную картинку положим ключом, а значением будет величина активации нейрона.

Рис. 24: Семантический словарь. В оригинальной статье [13] можно самому выбрать фрагмент на изображении, по которому будет строиться словарь.

Современные методы

Современные методы представляют собой развитие идей, которые мы перечислили до этого. Например, некоторые авторы предлагают для построения визуализации агрегировать информацию сразу со всех слоев. Мы умеем с помощью обратного распространения ошибки вычислять градиенты по весам во всех слоях. Можно агрегировать такие градиенты для всех слоев. Например, авторы [14] предлагают такой способ агрегирования, который они называют «full-gradient».

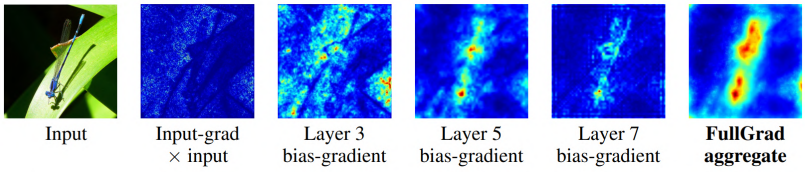

Рис. 25: Агрегируем градиенты сразу со всех слоев. Получили своеобразную «карту внимания», которая выглядит гораздо информативнее, чем обычная «saliency map». Источник [14]

Другое направление — сочетать визуализацию с динамикой обучения сети. Например, можно «наблюдать» за работой сети с каждой последующей эпохой, как это предлагают авторы [15]

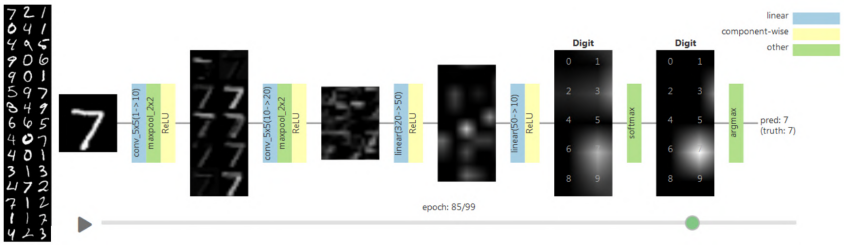

Рис. 26: Визуализация сети по ходу ее обучения. В оригинальной статье [15] можно увидеть ее в динамике.

Заключение

Визуализация сетей важна для понимания принципов их работы, получения интуиции о том, как они функционируют. С помощью визуализации можно понять, насколько хорошо обучилась сеть, выявить проблемы в данных, обнаружить потенциальные ошибки при ее применении (вспомните пример с яхтой на Рис. 12), увидеть даже фундаментальные «подводные камни», связанные с ее архитектурой (например, проблема «мертвых» фильтров при использовании ReLU, см. Рис. 7). Современные подходы к визуализации сетей позволяют понять, что происходит внутри сети, в том числе на промежуточных и последних слоях. Визуализация дает возможность вычленить, на что смотрит нейронная сеть на изображении, даже построить карту ее внимания (Рис.11 и Рис. 25).

Все это значительно приближает нас к глубоком пониманию, почему нейронные сети показывают такие выдающиеся результаты в задачах машинного обучения. Сети перестают быть «черными ящиками», мы можем разобрать их по «винтикам» и понять работу каждого нейрона, слоя, всей модели.

Список литературы

[1] Свёрточная нейронная сеть, https://ru.wikipedia.org/?oldid=106516242.

[2] CS231n: Convolutional Neural Networks for Visual Recognition, https://cs231n.github.io/understanding-cnn/.

[3] Convolutional Neural Network Architecture: Forging Pathways to the Future, https://missinglink.ai/guides/convolutional-neural-networks/convolutional-neural-network-architecture-forging-pathways-future/.

[4] Striving forsimplicity: the all convolutional net, https://arxiv.org/pdf/1412.6806.pdf

[5] ImageNet Classification with Deep Convolutional Neural Networks, https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

[6] t-SNE visualization of CNN codes, https://cs.stanford.edu/people/karpathy/cnnembed

[7] Visualizing and Understanding Convolutional Networks, https://arxiv.org/pdf/1311.2901.pdf

[8] Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps, https://arxiv.org/pdf/1312.6034.pdf

[9] SmoothGrad: removing noise by adding noise, https://arxiv.org/pdf/1706.03825.pdf

[10] Understanding Neural Networks Through Deep Visualization, https://arxiv.org/pdf/1506.06579.pdf

[11] Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency MapsKaren Simonyan et al 2014, https://arxiv.org/pdf/1312.6034.pdf

[12] Feature Visualization, https://distill.pub/2017/feature-visualization/

[13] The Building Blocks of Interpretability, https://distill.pub/2018/building-blocks/

[14] Full-Gradient Representation for Neural Network Visualization, https://arxiv.org/pdf/1905.00780.pdf

[15] Visualizing Neural Networks with the Grand Tour, https://distill.pub/2020/grand-tour/