Интерпретация Нейросетей

Наумов Антон (@anyo019)

План лекции

1. Интерпретация
2. В общем случае
3. От ML к DL
4. CV
5. NLP

1. Интерпретация

1. Интерпретируемость
2. Почему важно?
3. Всегда ли нужна?
4. Объяснимость

1.1. Интерпретируемость

Нет строгого математического определения..., но к примеру:

© Miller (2017): **”Interpretability is the degree to which a human can understand the cause of a decision”**

© Kim (2016): **”Interpretability is the degree to which a human can consistently predict the model’s result”**

Неформально – “Как хорошо мы понимаем работу нашего алгоритма в различных условиях”

Так, к вам могут прийти с вопросами:

- Почему в конкретном примере ответ такой?
- Что больше влияет на принятие решения?
- Соотносится ли с человеческой логикой?
- Какой алгоритм принятия решений?

Всё это и есть *интерпретируемость* и *объяснимость*!

1.2. Почему важно?

I. Исследование:
   a) Как связаны данные и результаты (**извлечение знания из данных**)?
   b) Какие странности и проблемы есть в данных (**очистка и сбор данных**)?
   c) Почему модель не справляется так, как мне бы хотелось (**отладка обучения**)?

II. Безопасность:
   a) Как злоумышленник может обмануть модель (**защита от атак**)?
   b) Нет ли в данных bias-ов, не соответствующих реальности, а порождённые качеством сбора данных или методом обучения (**bias-ы и дискриминация**)?
   c) Не допустит ли модель ошибку в зоне, где нельзя совершать ошибок (**безопасность и репутация**)?
   d) Можем ли мы объяснить ответ при обращении к нам с вопросами (**понятность**)?

1.3. Всегда ли нужна?

Конечно же, нет.

Не нужна или даже противопоказана, когда:

- Влияние и риски вокруг модели низки
- Проблема хорошо изучена и разработана
- Класс моделей хорошо изучен, разработан и широко применяется
- Требуется высокая степень защиты от атак или внешнего понимания работы алгоритма

1.4. Объяснимость

Способность дать ответы на различные вопросы про модель:

- Что она делает?
- Почему она так решила?
- Кто ее этому научил?
- Надежна ли она?
- На чем она сломается?
- Как повлиять на решение?

2. В общем случае

1. Свойства методов объяснения
2. Свойства отдельных объяснений
3. Что хотим объяснять?
4. Как будем объяснять?

2.1. Свойства методов объяснения

- **Expressive Power** – на каком “языке” или с использованием какой структуры выдаётся объяснение
- **Translucency** – как много информации требуется методу о внутренностях модели
- **Portability** – на каком количестве различных моделей можно использовать данный подход
- **Algorithmic Complexity** – насколько дорого вычислительно посчитать

2.2. Свойства отдельных объяснений

- **Accuracy** – насколько точны объяснения на новых примерах
- **Fidelity** – насколько хорошо объясняется результат black-box модели
- **Consistency** – насколько близки объяснения двух похожих моделей на одной задаче с похожими результатами
- **Stability** – насколько близки объяснения на двух похожих конкретных примерах
- **Comprehensibility** – насколько объяснения понятны человеку (в т.ч. уместность)
- **Certainty** – использует ли объяснение уверенность модели в ответах, если таковые тоже есть

2.2. Свойства отдельных объяснений (ещё)

- **Degree of Importance** – даёт ли объяснение информацию о важности отдельных фичей
- **Novelty** – даёт ли объяснение информацию о новизне данных (регион удалённый от основного распределения)
- **Representativeness** – как много покрывает данное объяснение (один пример vs ... vs вся модель)
- **Selectivity** – не выдаёт ли объяснение нам слишком много информации
- **Contrastiveness** – даёт ли объяснение понимание отличия данного результата от другого

2.3. Что хотим объяснять?

- Входные данные (как алгоритм порождает модель из данных?)
- Модель как белый ящик (как части модели влияют на предсказания?)
- Модель как чёрный ящик (как обученная модель делает предсказания?)
- Результат работы модели (почему было сделано конкретное предсказание?)

Так же стоит вопрос вида подхода:

- **Model-specific** – с пониманием конкретной используемой модели
- **Model-agnostic** – общий подход, не привязанный к виду модели

2.4. Как будем объяснять?

- **Статистика или визуализация фичей** – к пр. важность различных фичей относительно результатов
- **Внутренности модели** – к пр. обученные веса из интерпретируемой модели
- **Точка данных** (новая или ранее существующая) – смотрите как надо или как не надо (в таком случае сами примеры должны быть интерпретируемы)
- **Интерпретируемая модель** – более простая и понятная модель, которая ведёт себя так же или похоже

3. От ML к DL

1. Интерпретируемые модели
2. Как быть с ML?
3. ML vs DL
4. Как быть с DL?
5. Линейные модели

3.1. Интерпретируемые модели

- **Линейные модели** – сравниваем эффекты, а не веса (вес * значение), т.к. признаки могут иметь разный масштаб, значимость – p-value. В случае LogReg – логарифм шанса.
- **Деревья и списки правил** – разбиение данных на группы и если не слишком большой глубины, то легко интерпретируемо.
- **KNN** – если сами примеры интерпретируемы, то и модель интерпретируема (ответ такой, потому что вот похожие примеры с таким же ответом).
- **Naive Bayes Classifier** – предполагается независимость признаков, соответственно легко из формулы получаются значимости относительно каждого из классов.

3.2. Как быть с ML?

Большая часть подходов к интерпретации классических ML методов опирается на:

- Оценку влияния одной или нескольких фичей на предсказание
- Взаимодействие между отдельными фичами
- Влияние модификации или фиксации части признаков
- Упрощение до более простых и интерпретируемых моделей
- Интерпретация относительно отдельных примеров из данных (позитивных или негативных)
- Создание близких данных и оценка изменения предсказаний

3.3. ML vs DL

Многие подходы интерпретации из классического ML могут ограниченно работать для DL.

Однако есть несколько ключевых отличий между ML и DL:

- Входы, как правило, имеют структуру (изображения, тексты, звук, ...)
- Тысячи или даже миллионы фичей (к пр. цветное фото 1920 x 1080)
- Значимость отдельных фичей маленькая
- Модели имеют огромное количество параметров
- Task-specific архитектуры моделей

3.4. Как быть с DL?

Т.к. многие данные имеют структуру того или иного вида, то часто сами данные для нас более интерпретируемы, чем в классическом ML (100 показателей датчиков vs картинка из 6.220.800 фичей [1920 x 1080 x 3]) – можем интерпретировать примерами.

Однако интерпретация одной отдельной фичи сложнее.

Промежуточные представления, как правило, так же имеют структуру – можем это использовать.

Нейросети большие, но мы не пытаемся оценить один конкретный вес, а всю модель целиком через активации на данных и градиенты (нужно уметь доставать из нейросети всё нам необходимое – больше на семинаре).

Создание синтетических примеров возможно, но не очень просто.

3.5. Линейные модели

Хотим оценивать влияние входов на выходы.

Как и в ML – эффекты (вес * значение) очень легко интерпретируемы.

Несколько линейных слоёв через активации:

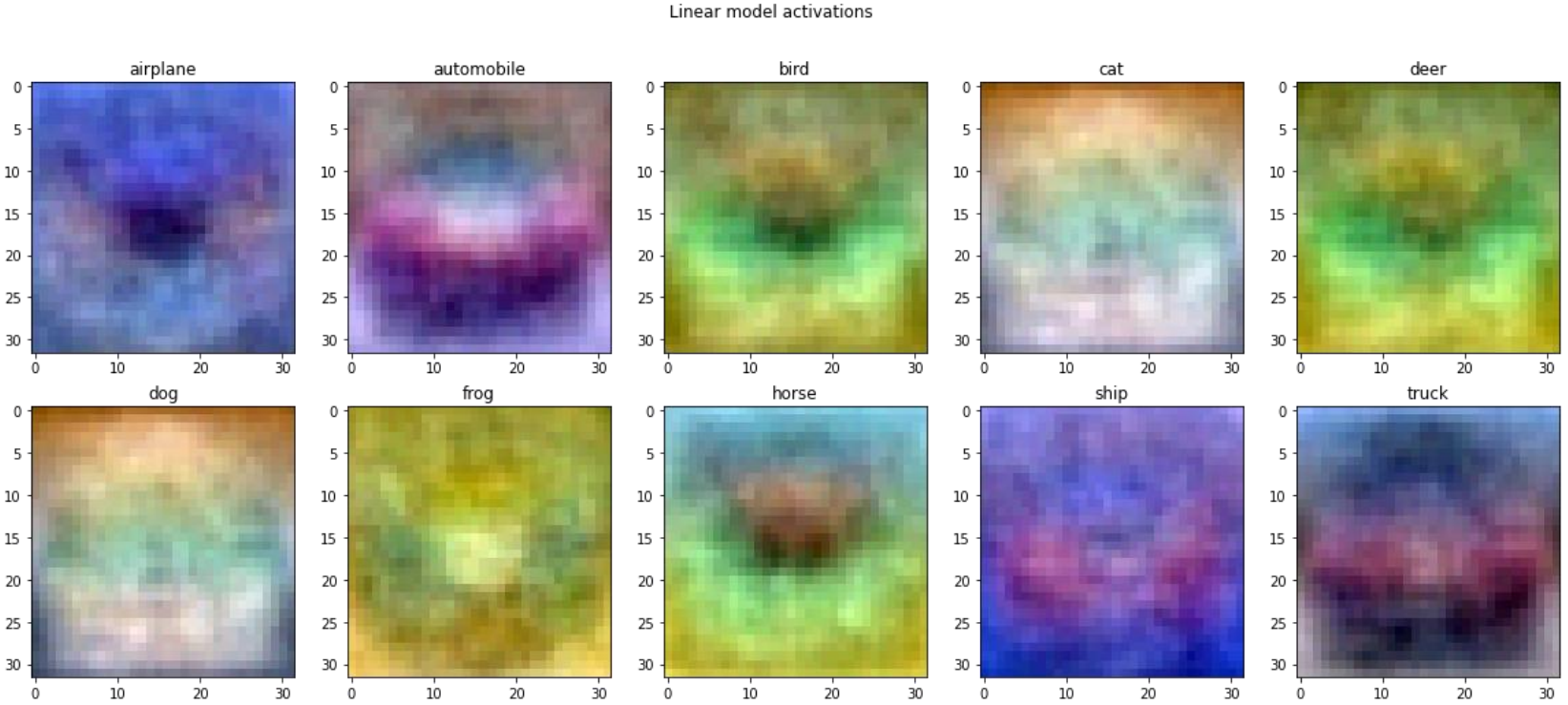
- Можем честно посчитать вклад каждого входа (сложно, не всегда адекватно возможно)
- Можем аппроксимировать меняя/подбирая входы и оценивая влияние на выходы (не очень грубая, но не всегда легко интерпретируемая аппроксимация)
- Можем проигнорировать активации (грубая, но более легко интерпретируемая аппроксимация)

3.5. Линейные модели

Пример грубой аппроксимации:

Датасет – Cifar 10 (картинки 32 x 32 x 3 из 10 классов)

Модель – 4-слойная линейная нейросеть над всеми пикселями как фичами (без учёта структуры)



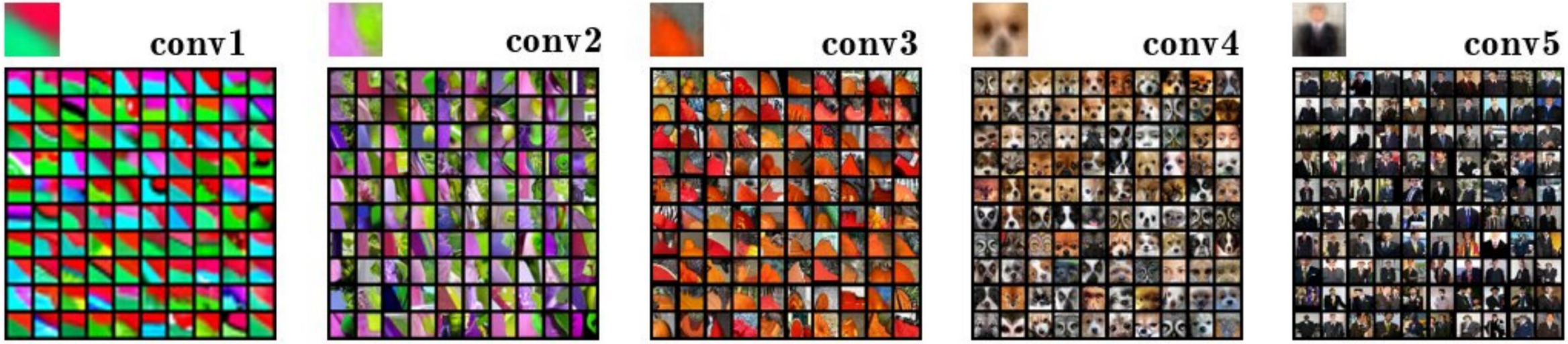
4. CV

1. Receptive field
2. Deconvolutional network
3. Gradient-based
4. Guided Backpropagation
5. Class Activation Mapping (CAM)
6. Grad-CAM

4.1. Receptive field

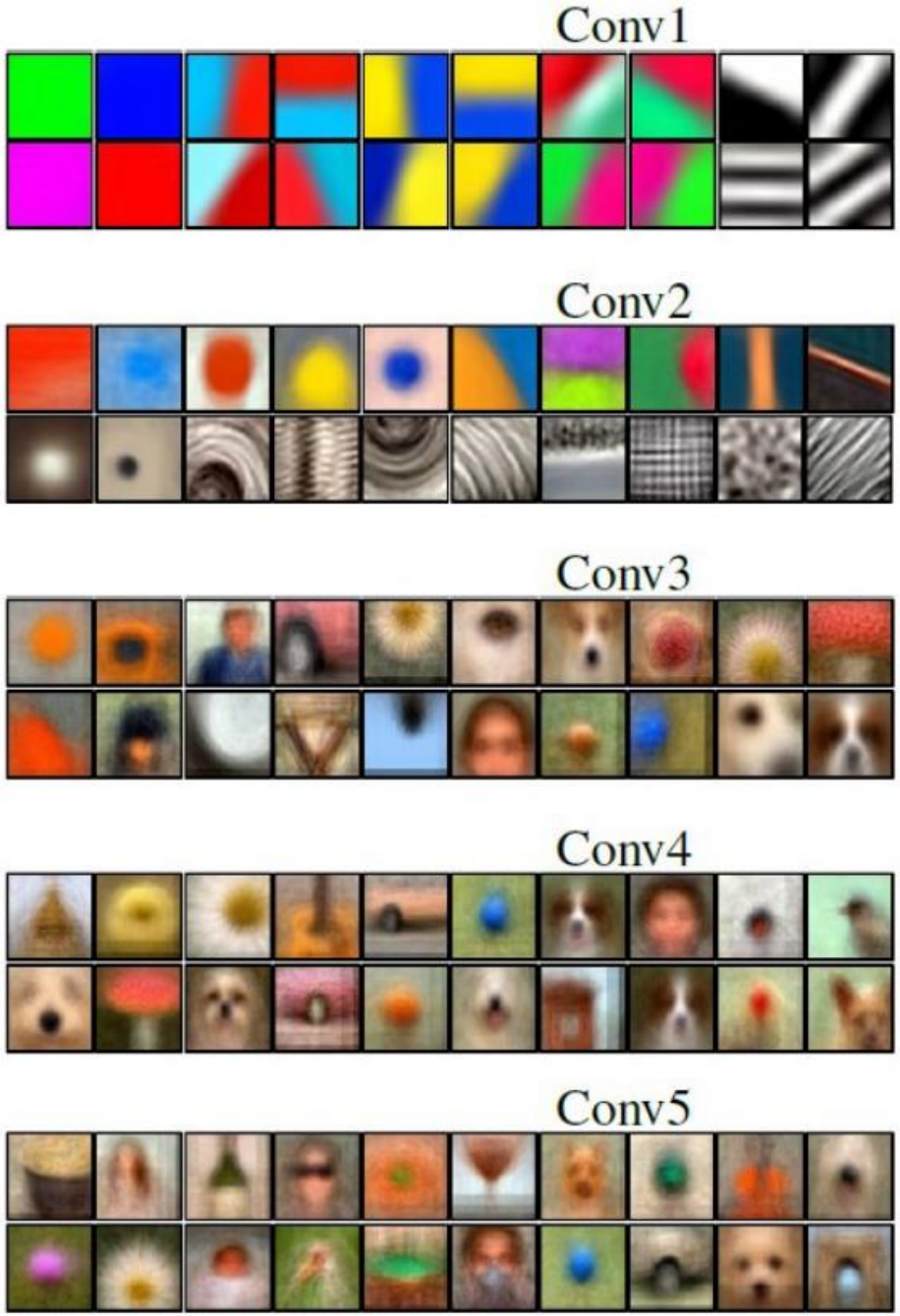
Имея представление о receptive field нашей модели на каждом слое – можем оценивать какие кусочки изображения привели к активации тех или иных фильтров.

Не даст нам полной картины, но может дать инсайд на внутренности модельки.



4.1. Receptive field

Примеры картинок, полученных как прямое усреднение первых N кропов изображений, максимально активировавших соответствующее ядро



4.2. Deconvolutional network

Хотим по выходу свёртки восстанавливать кусочки изображения, наиболее активировавшие данное ядро.

Операция *deconv*, которую позже переименовали в *conv2d_transpose* (более честное название).

Pooling инвертируем, используя позицию, на которой нашли максимум (если MaxPooling).

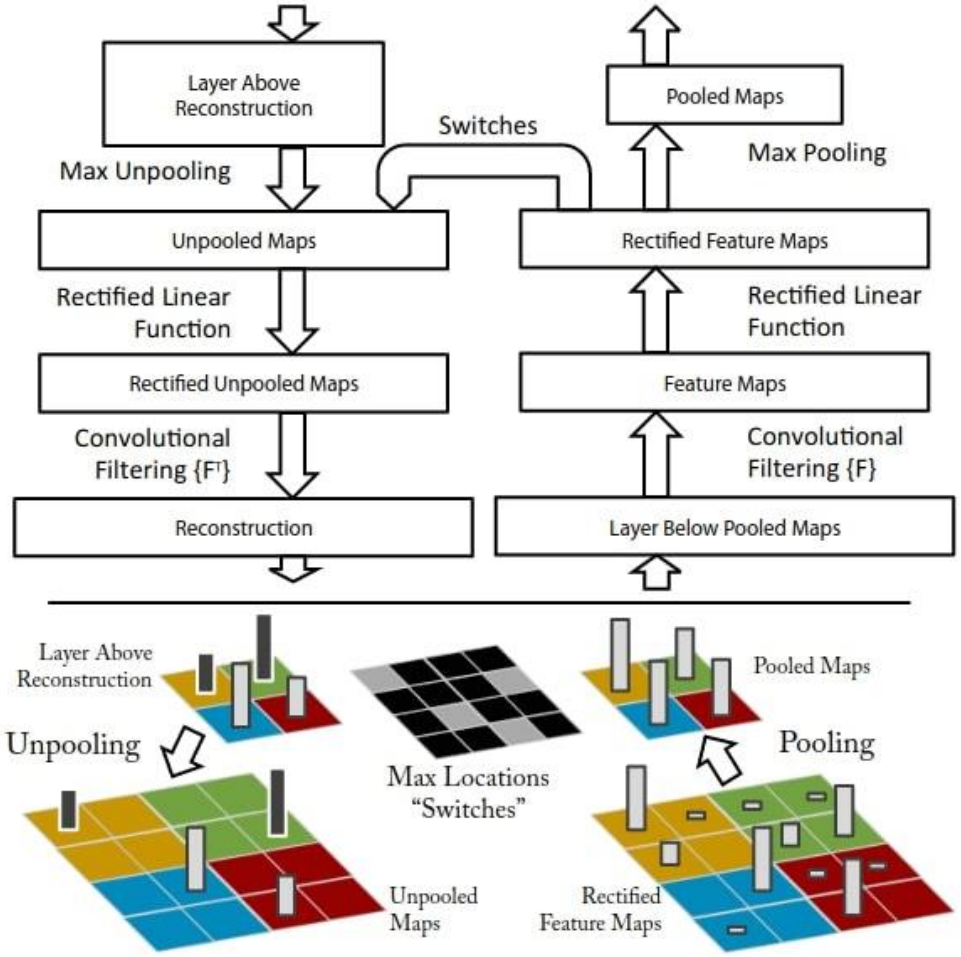
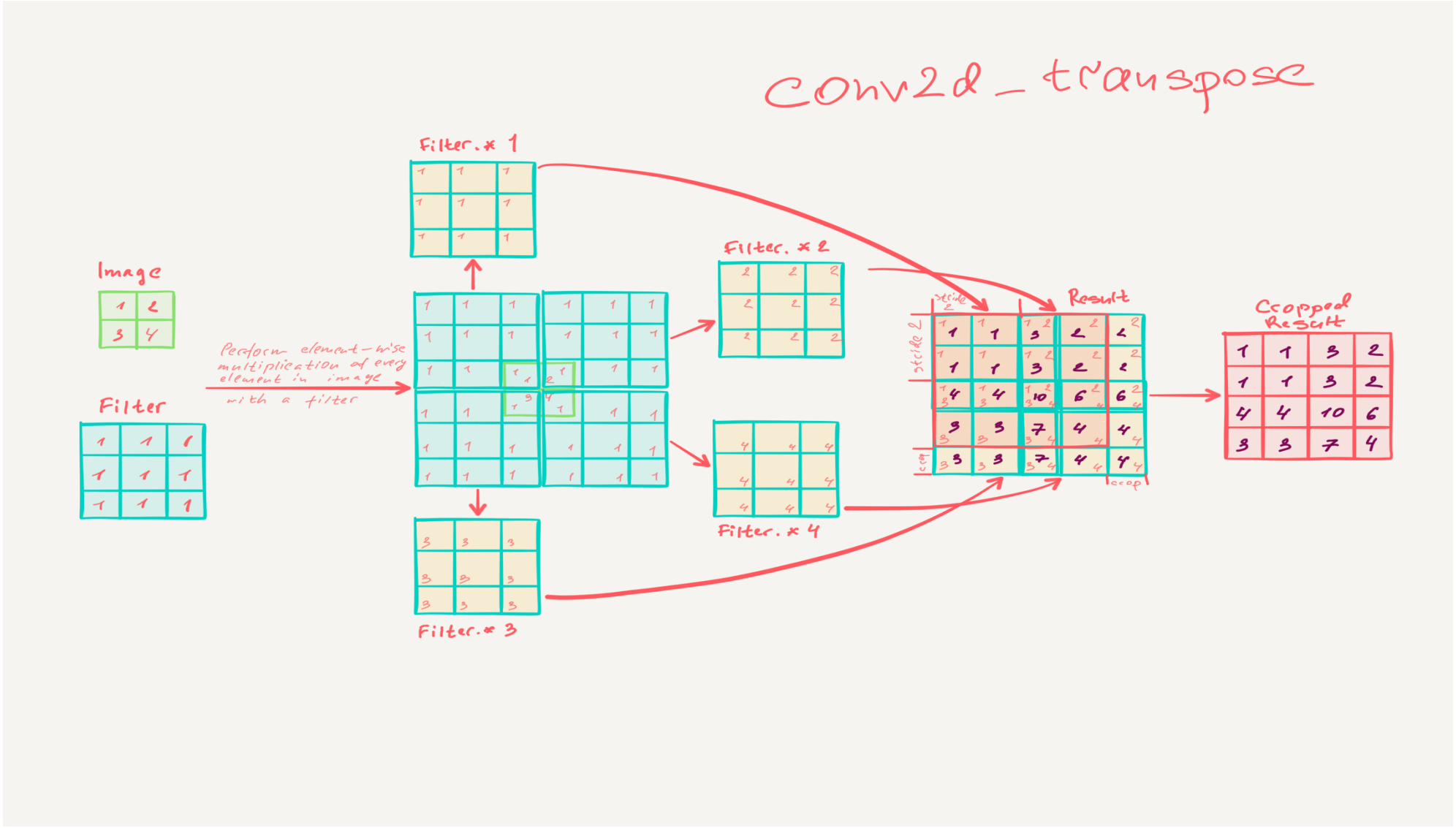


Figure 1. Top: A deconvnet layer (left) attached to a convnet layer (right). The deconvnet will reconstruct an approximate version of the convnet features from the layer beneath. Bottom: An illustration of the unpooling operation in the deconvnet, using *switches* which record the location of the local max in each pooling region (colored zones) during pooling in the convnet.

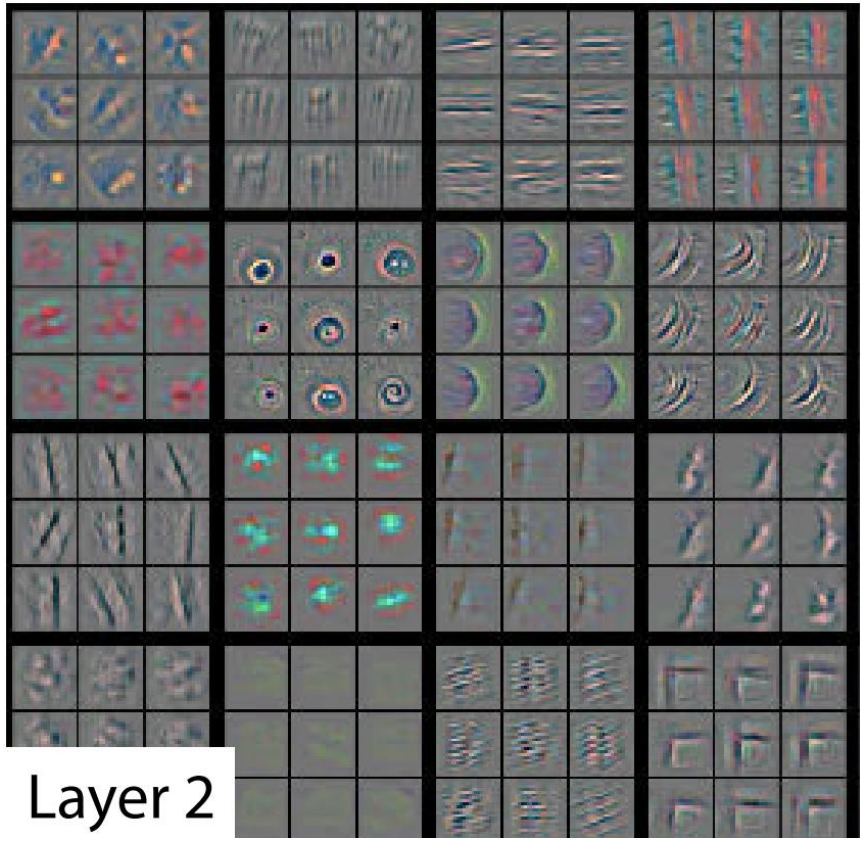
4.2. Deconvolutional network



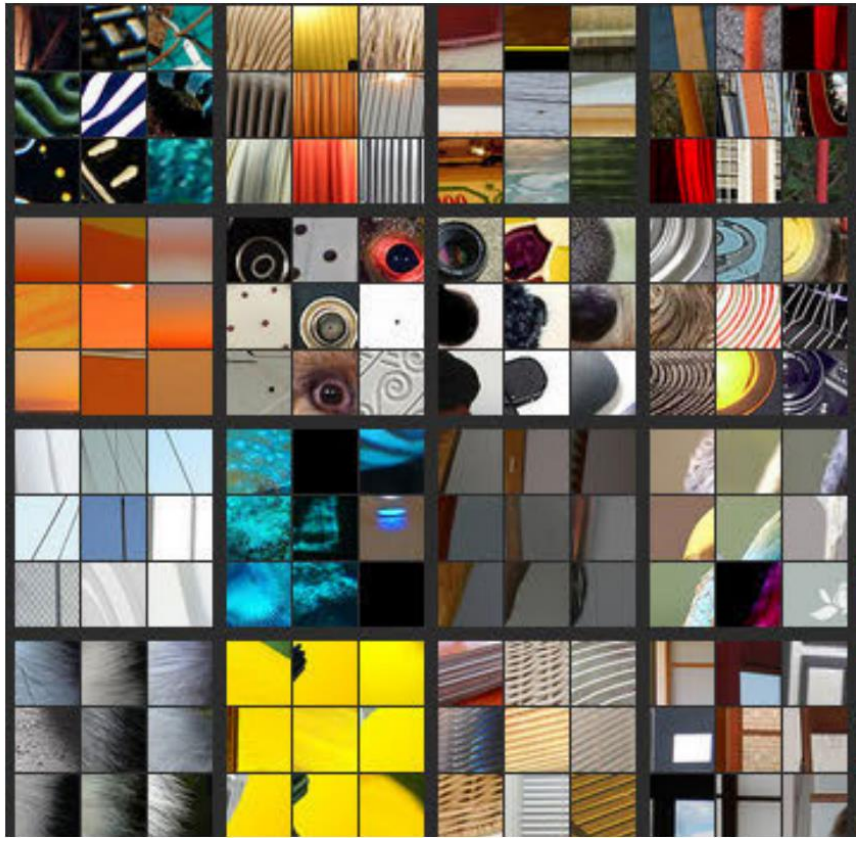
4.2. Deconvolutional network

[Visualizing and Understanding Convolutional Networks]

Layer 1



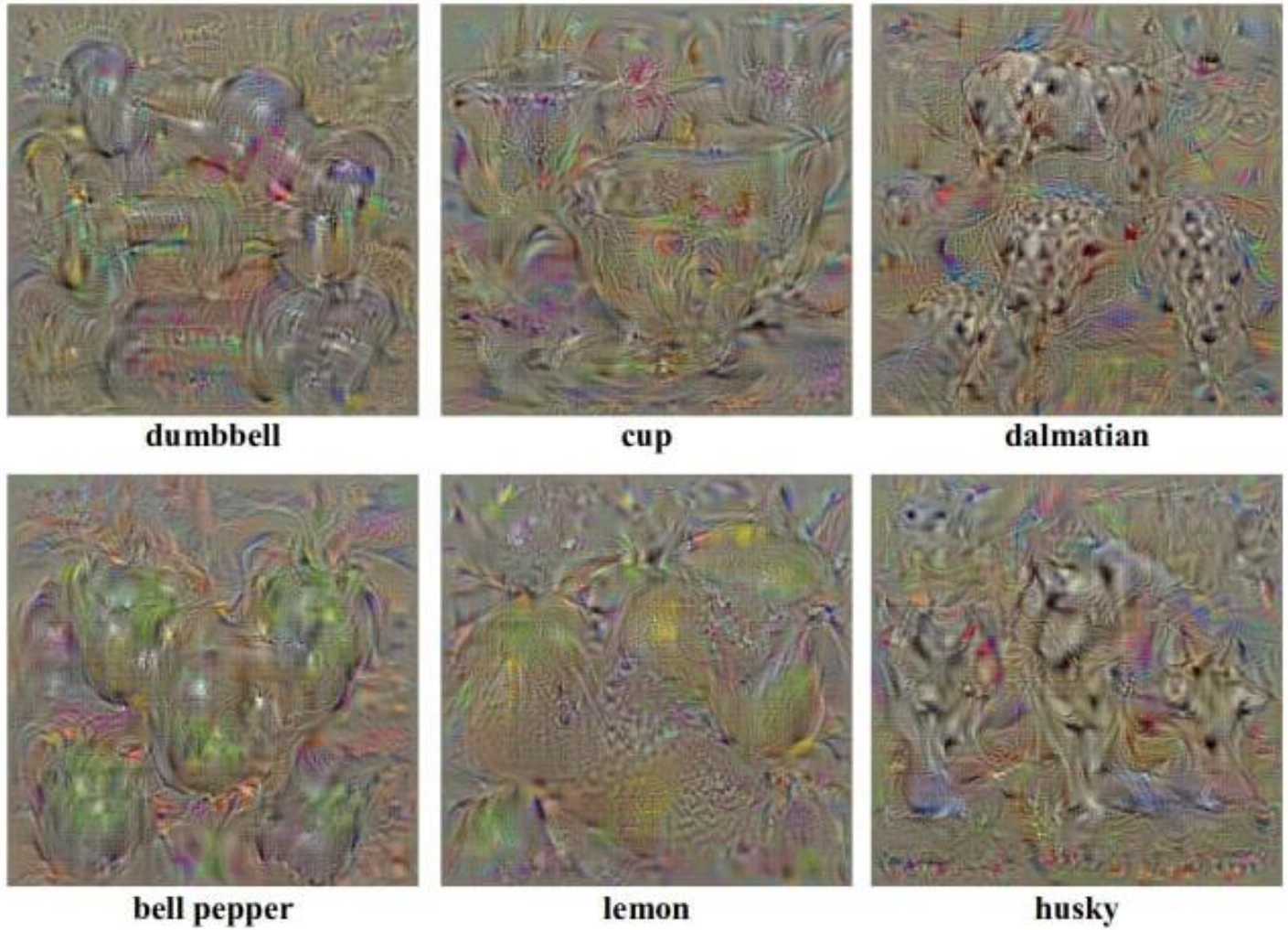
Layer 2



4.3. Gradient-based

Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps

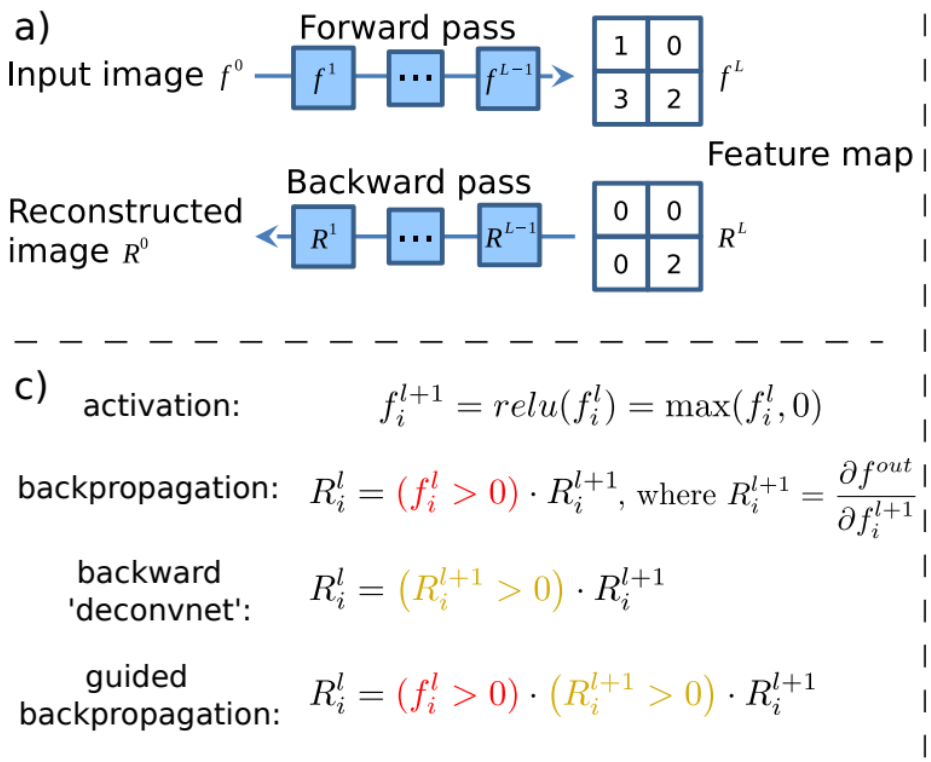
Возьмём предобученную нейросеть, заморозим все слои и будем “обучать” изображение для максимизации конкретного класса.

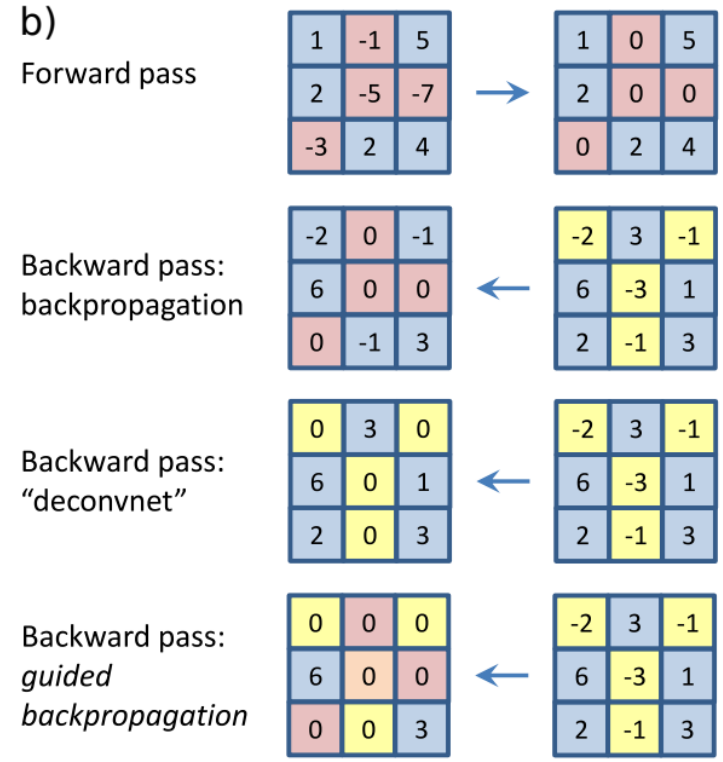


4.4. Guided Backpropagation

Делаем прямой проход через нейросеть, затем зануляем не интересующие нас нейроны.

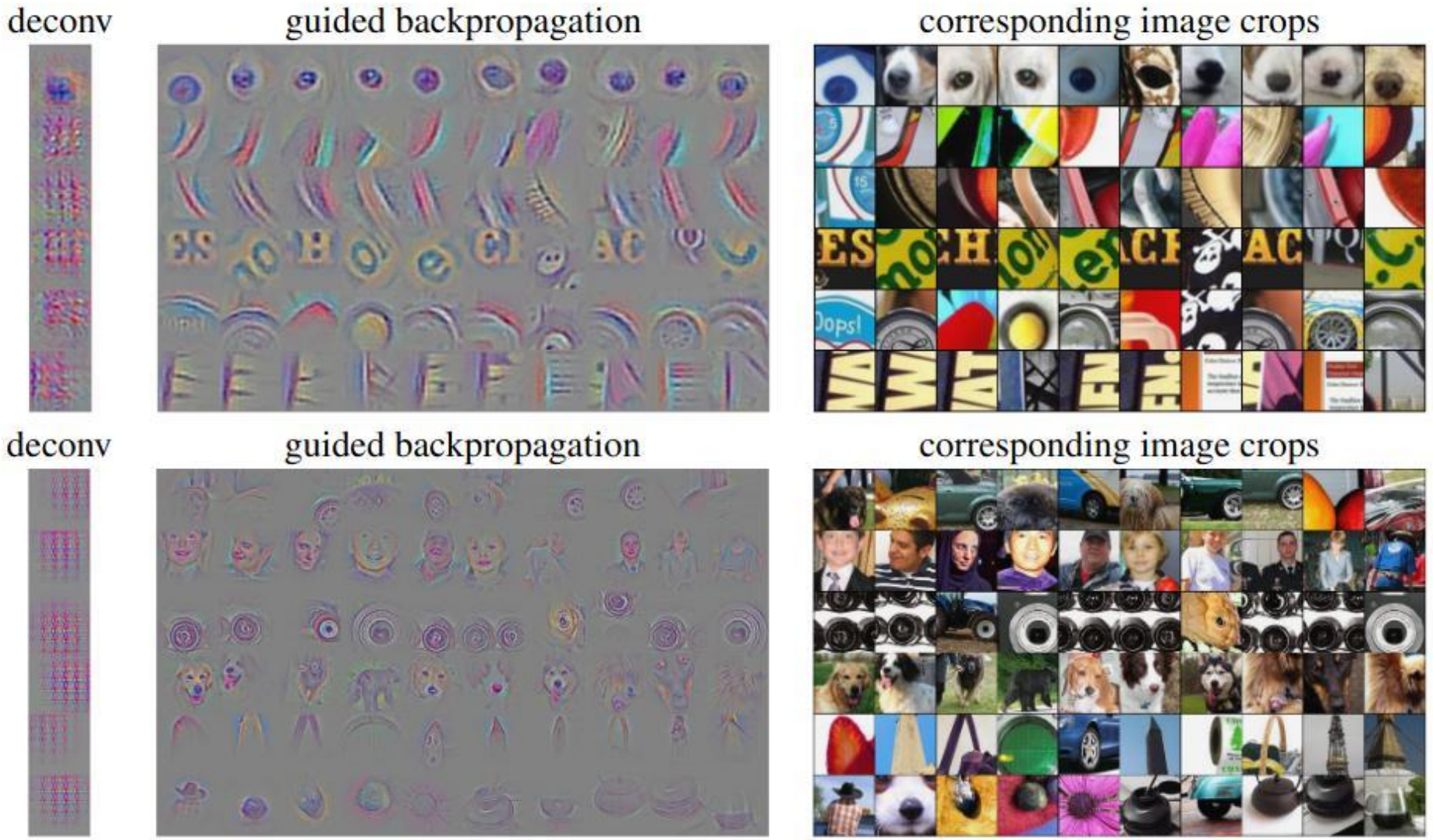
Делаем обратный проход от карты активаций со всеми нулями, кроме одного.





4.4. Guided Backpropagation

Striving for Simplicity: The All Convolutional Net



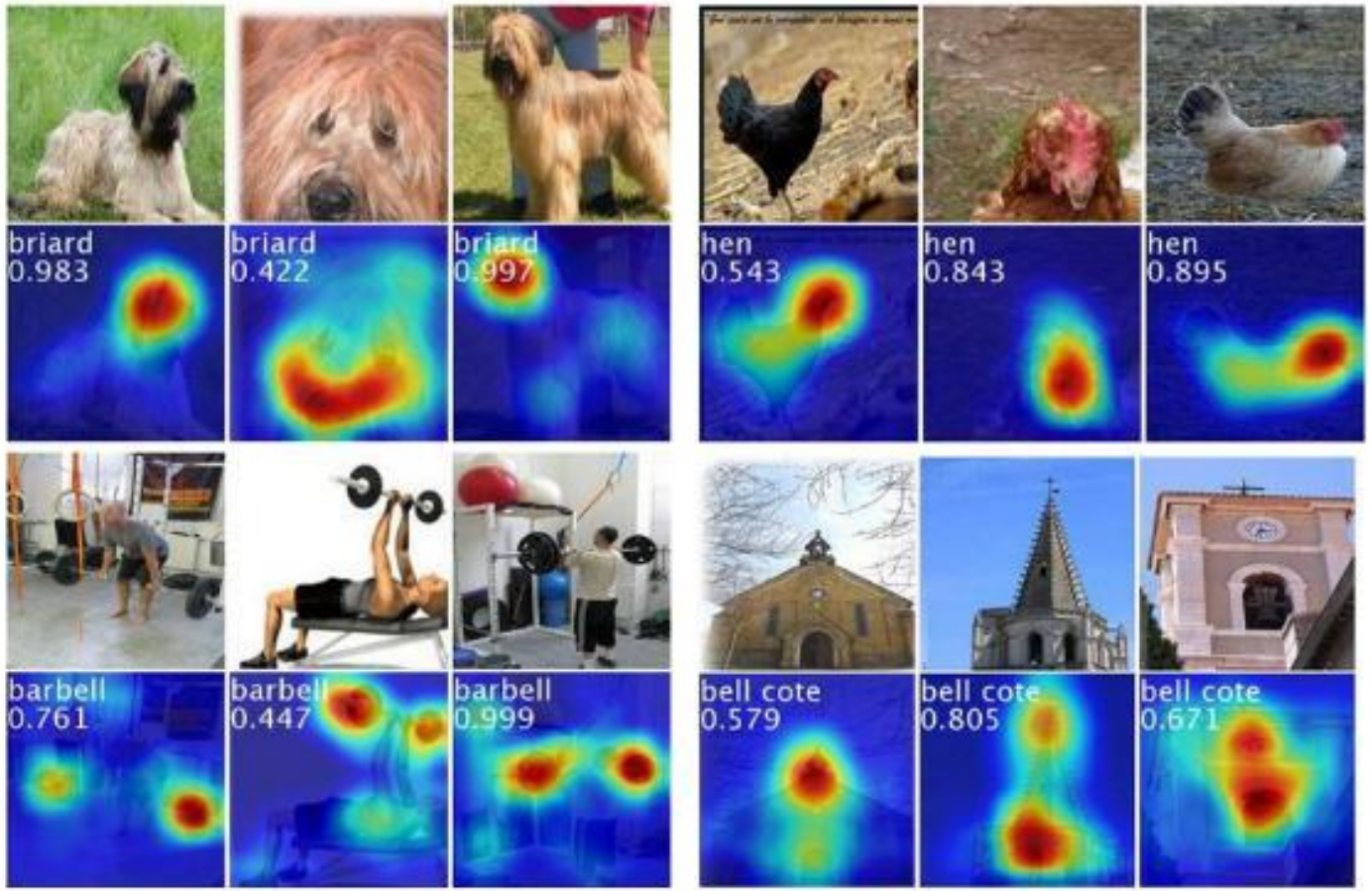
4.5. Class Activation Mapping (CAM)

Learning Deep Features for Discriminative Localization

Смотрим какие части изображения активируют соответствующий нейрон.

Усредняем feature-map-ы с весами, соответствующими их вклад в финальное изображение.

Визуализируем через heat-map-ы поверх картинок.

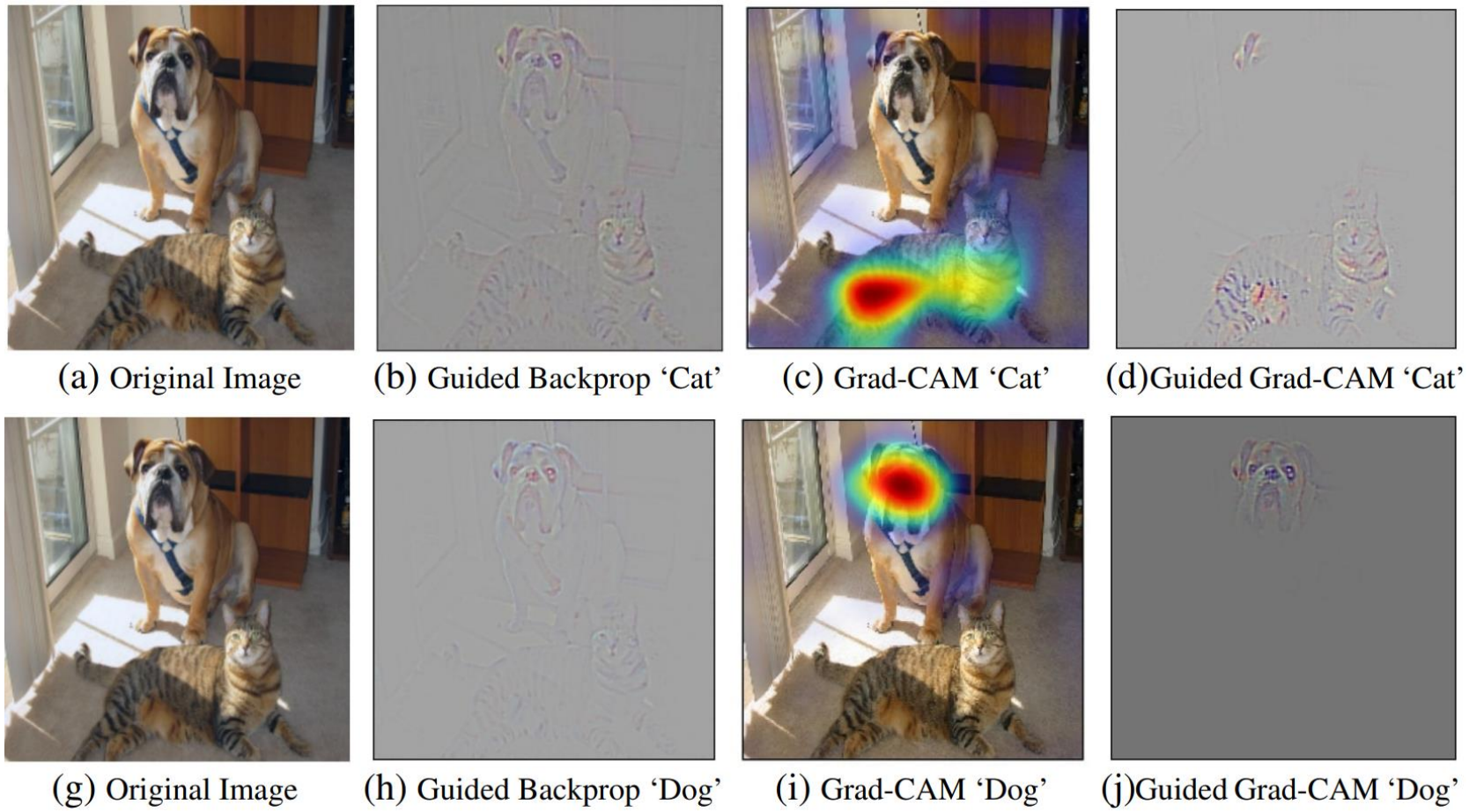


4.6. Grad-CAM

Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization

Вместо forward прохода как в CAM используют градиенты относительно одного класса выхода.

Усредняют feature-map-ы, используя усреднённые по H, W градиенты, а не чисто веса.

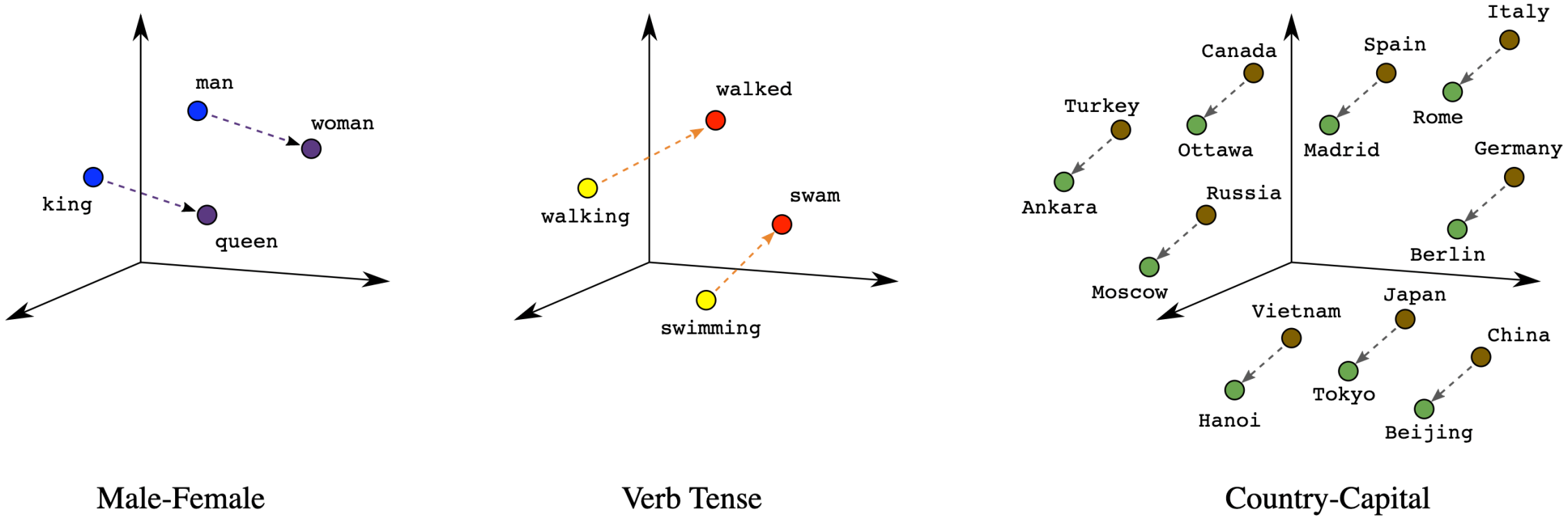


5. NLP

1. Интерпретируемые эмбеддинги
2. RNN-like
3. Attention и Transformer

5.1. Интерпретируемые эмбеддинги

Часто получаемые эмбеддинги текстов хорошо кластеризуются, отвечают базовым математическим операциям.



Male-Female | Verb Tense | Country-Capital

5.2. RNN-like

По большому счёту – два линейных слоя (перенос информации от входа к скрытому и сквозь время).

Т.к. линейные слои легко интерпретируемы, то и здесь получается довольно лёгкая интерпретация, просто требующая много шагов (как для многослойной линейной нейросети).

LSTM и GRU так же легко интерпретируемы относительно интуиции архитектуры (забываем что-то, даём на выход что-то, записываем в память что-то).

Т.к. фичами для RNN-like сети являются скорее элементы эмбеддингов, то для интерпретации RNN-like сетей важна интерпретируемость эмбеддингов.

5.3. Attention и Transformer

В силу архитектуры Attention-механизма у вас есть явная компонента, отвечающая за вес того или иного токена входа.

На каждом шаге токен агрегирует в себе информацию из нескольких токенов с весами.