

Компьютерное зрение

Лекция 10.

Similarity learning. CNN visualization. Style transfer

02.07.2020

Руслан Алиев

Задачи DL

Классификация



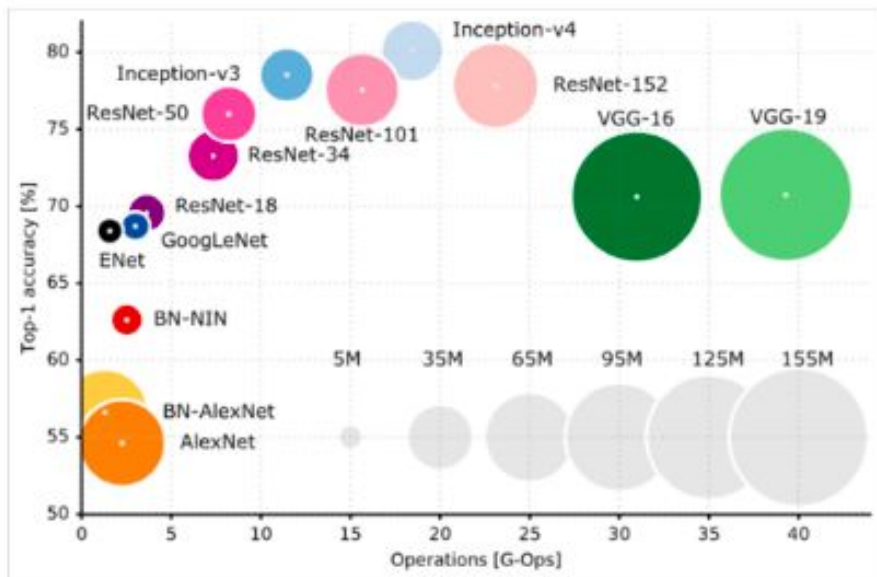
Задачи DL

Классификация (ImageNet - 1000 категорий)



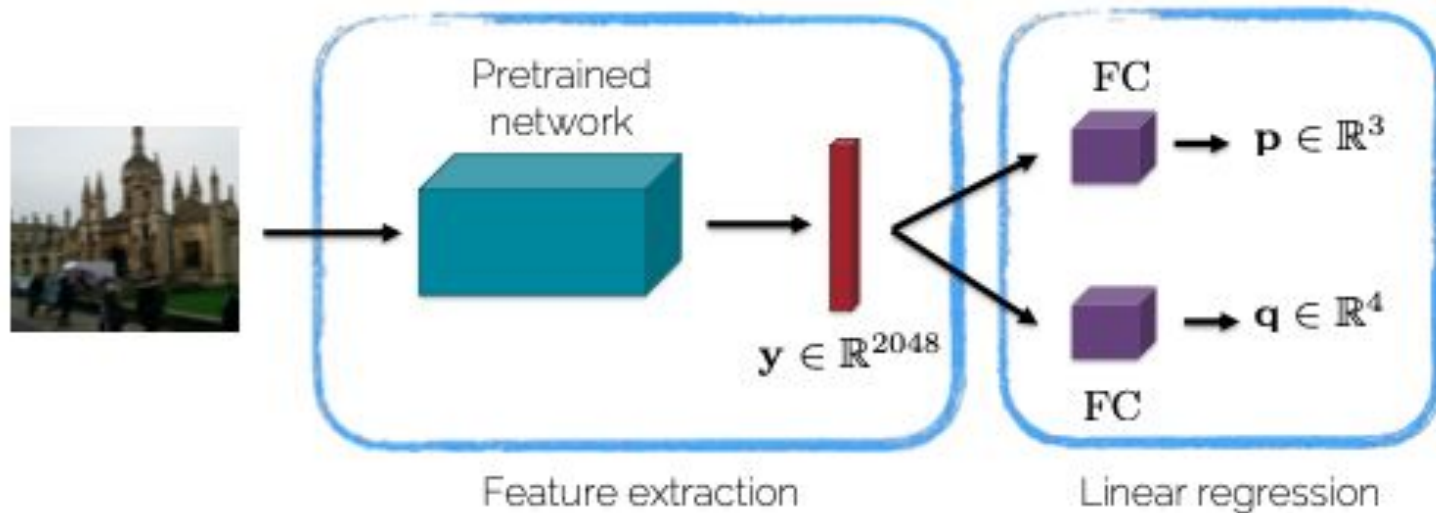
Задачи DL

Performance on ImageNet. Размер круга - количество параметров



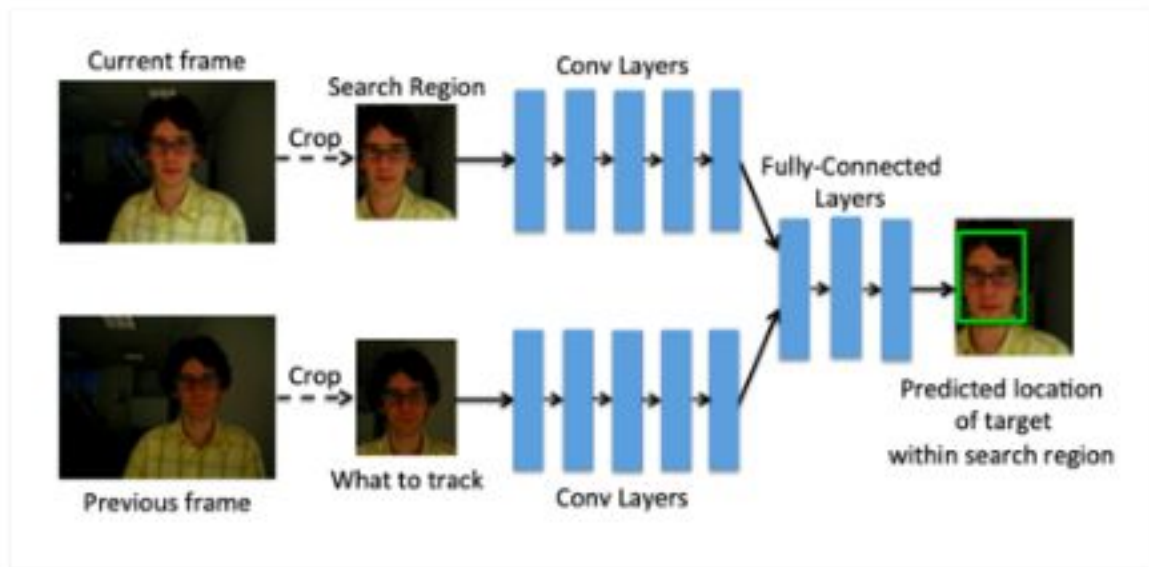
Задачи DL

Регрессия: pose regression



Задачи DL

Регрессия: bounding box regression



D. Held et al. „Learning to Track at 100 FPS with Deep Regression Networks“. ECCV 2016

Задачи DL

Задачи с лицами

A



Classification: person, face, female

B



Classification: person, face, male

Задачи DL

Задачи с лицами

A



B



Is it the same person?

Задачи DL

Задачи с лицами: similarity learning



- Comparison
- Ranking

Задачи DL

Приложение: разблокировка айфона с FaceID

Training



Задачи DL

Приложение: разблокировка айфона с FaceID

A



YES

B



NO

Testing

Can be solved as a
classification problem



Задачи DL

Приложение: распознавание лиц студентов на экзамене

Training

Person 1



Person 2



Person 3



Задачи DL

Приложение: распознавание лиц студентов на экзамене

В чем проблема?

Задачи DL

Приложение: распознавание лиц студентов на экзамене

В чем проблема?

Немасштабируема! Нужно каждый раз обучать модель заново, если появляется новый студент

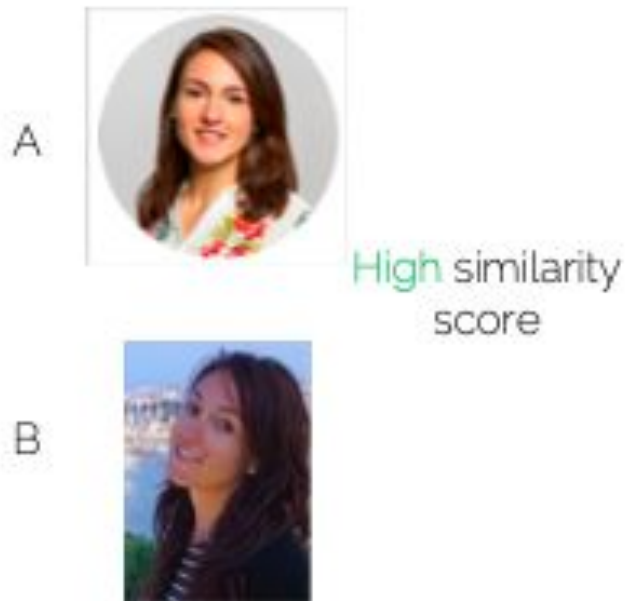
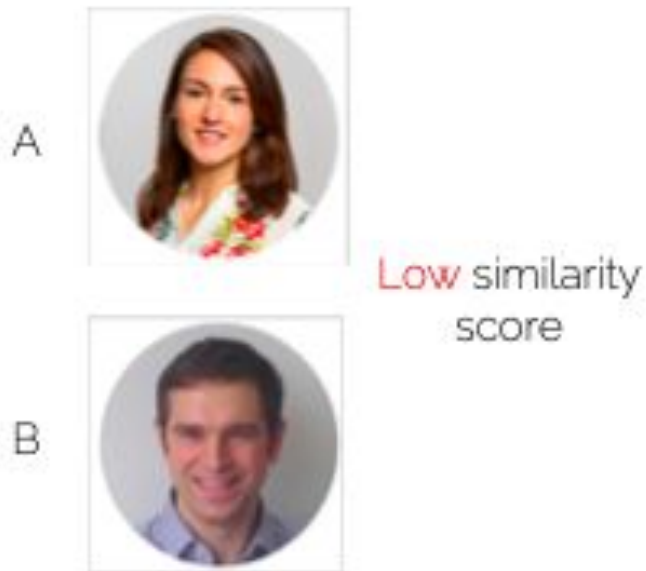
Задачи DL

Приложение: распознавание лиц студентов на экзамене

Можем ли мы обучить один раз модель и использовать ее каждый раз?

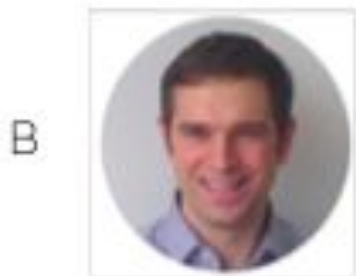
Задачи DL

Обучаем функции похожести (similarity function)



Задачи DL

Обучаем функции похожести: test time



$$d(A, B) > \tau$$

Not the same
person

Задачи DL

Обучаем функции похожести: test time

Same person

$$d(A, B) < \tau$$

A



B



Similarity learning

Как обучить сетку определять похожесть?

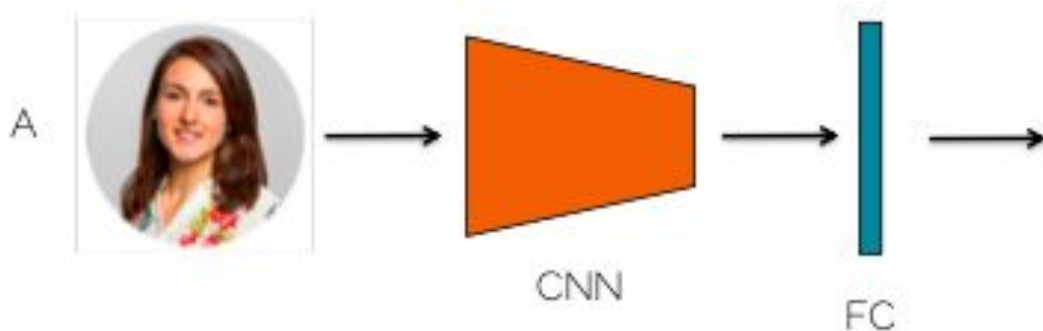


Siamese Neural Networks



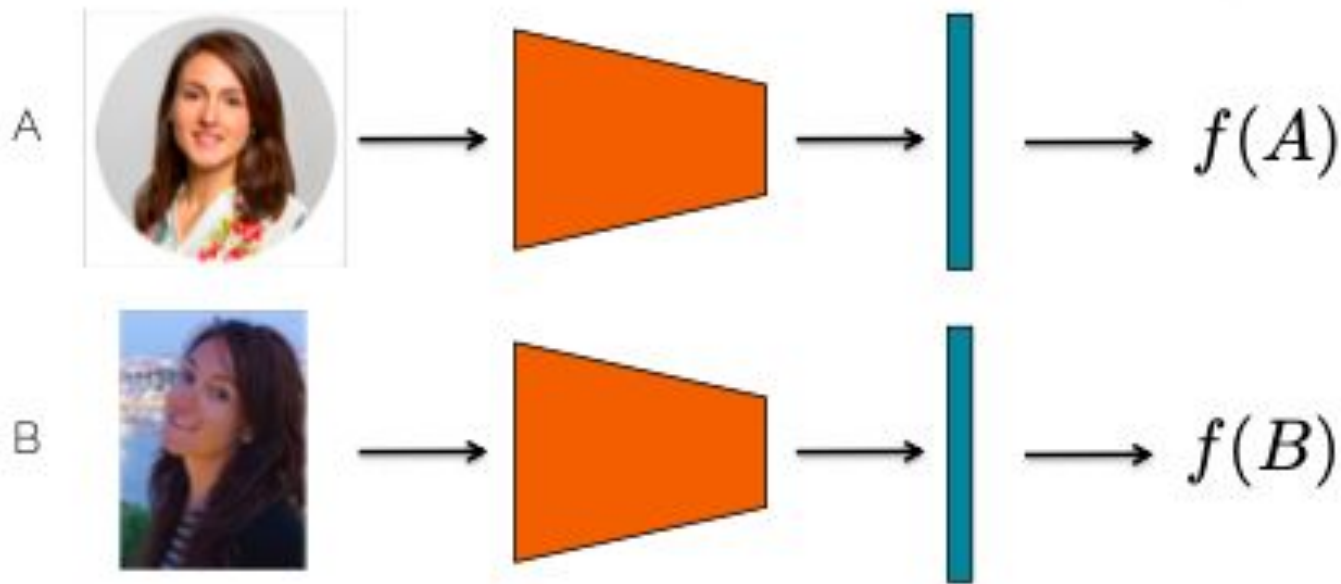
Similarity learning

Как обучить сетку определять похожесть?



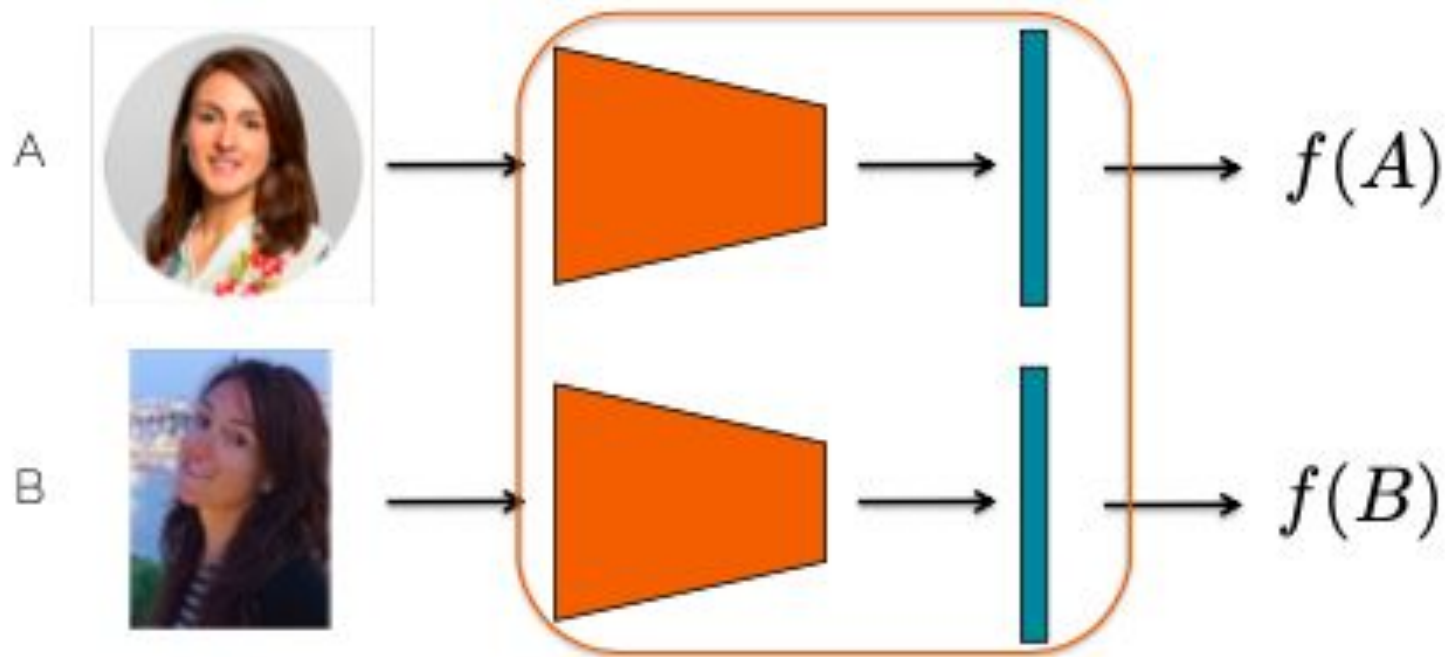
Similarity learning

Как обучить сетку определять похожесть?



Similarity learning

Siamese network - общие веса



Similarity learning

Siamese network - общие веса

Мы используем сетку, чтобы получить эмбединг
изображение $f(A)$

Осталось сравнить эмбединги!

Similarity learning

Функция расстояния

$$d(A, B) = ||f(A) - f(B)||^2$$

Во время обучения подбираем такие параметры чтобы:

- Если A и B - изображения одного и того же человека, $d(A, B)$ - маленькое
- Если A и B - изображения разных людей, $d(A, B)$ - большое

Similarity learning

Функция потерь для положительной пары:

- Если A и B - изображения одного и того же человека, $d(A, B)$ - маленькое

$$\mathcal{L}(A, B) = \|f(A) - f(B)\|^2$$

Similarity learning

Функция потерь для отрицательной пары:


- Если A и B - изображения разных людей, $d(A, B)$ - большое

$$\mathcal{L}(A, B) = \max(0, m^2 - \|f(A) - f(B)\|^2)$$


Similarity learning

Contrastive loss

$$\mathcal{L}(A, B) = y^* \|f(A) - f(B)\|^2 + (1 - y^*) \max(0, m^2 - \|f(A) - f(B)\|^2)$$



Positive pair,
reduce the distance
between the
elements



Negative pair,
brings the elements
further apart up to a
margin

Similarity learning

Обучение siamese networks:

- Обновляем веса для каждой сетки-близнеца независимо, потом усредняем параметры

Эта функция потерь позволяет сблизить положительные пары, и раздвинуть отрицательные

Triplet Loss

Triplet loss - появляется ранжирование

Мы хотим: $\|f(A) - f(P)\|^2 < \|f(A) - f(N)\|^2$



Anchor (A)



Positive (P)

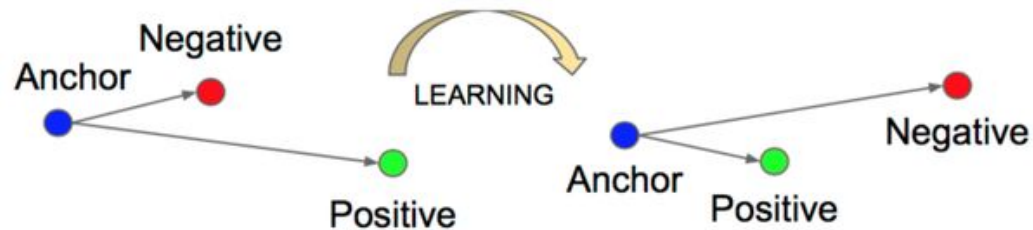


Negative (N)

Triplet Loss

Triplet loss - появляется

Мы хотим:



Anchor (A)



Positive (P)



Negative (N)

Triplet Loss

Triplet loss - появляется ранжирование

$$||f(A) - f(P)||^2 < ||f(A) - f(N)||^2$$

$$||f(A) - f(P)||^2 - ||f(A) - f(N)||^2 < 0$$

$$||f(A) - f(P)||^2 - ||f(A) - f(N)||^2 + m < 0$$



margin

Triplet Loss

Triplet loss - появляется ранжирование

$$\|f(A) - f(P)\|^2 < \|f(A) - f(N)\|^2$$

$$\|f(A) - f(P)\|^2 - \|f(A) - f(N)\|^2 < 0$$

$$\|f(A) - f(P)\|^2 - \|f(A) - f(N)\|^2 + m < 0$$

$$\mathcal{L}(A, P, N) = \max(0, \|f(A) - f(P)\|^2 - \|f(A) - f(N)\|^2 + m)$$

Triplet loss

Обучение сложных примеров:

$$\mathcal{L}(A, P, N) = \max(0, \|f(A) - f(P)\|^2 - \|f(A) - f(N)\|^2 + m)$$

Обучаемся несколько эпох

Выбираем сложные примеры, где $d(A, P) \approx d(A, N)$

Дообучаем на сложных примерах

Triplet loss

Обучение сложных примеров:

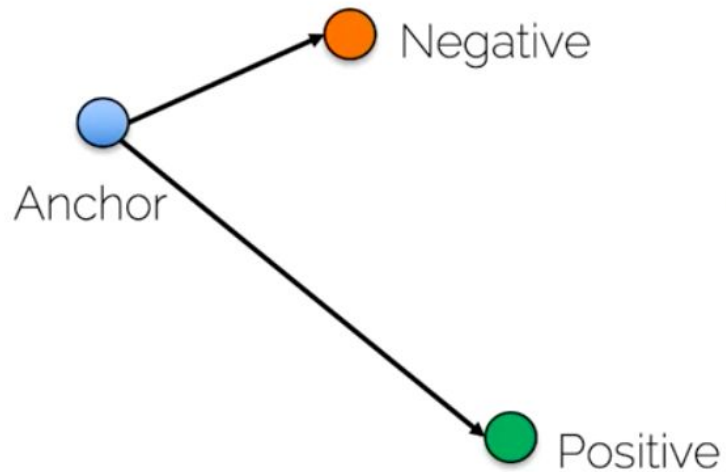
$$\mathcal{L}(A, P, N) = \max(0, \|f(A) - f(P)\|^2 - \|f(A) - f(N)\|^2 + m)$$

Обучаемся несколько эпох

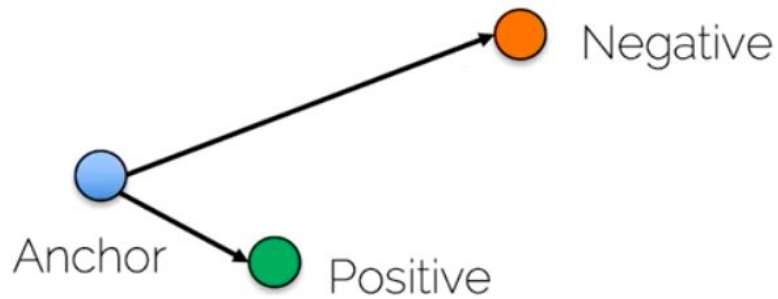
Выбираем сложные примеры, где $d(A, P) \approx d(A, N)$

Дообучаем на сложных примерах

Triplet loss



Training
→



Triplet loss: test time

Просто nearest neighbor search



Triplet loss: sampling

Random sampling не работает - количество возможных триплетов $O(n^3)$, поэтому нужно будет слишком долго обучать

Даже с hard negative mining, есть вероятность застрять в локальном минимуму

Improving similarity learning

Loss:

-Contrastive vs triplet loss

Sampling:

- Выбор наилучших троек для обучения, умное сэмплирование - разнородность классов + сложные примеры

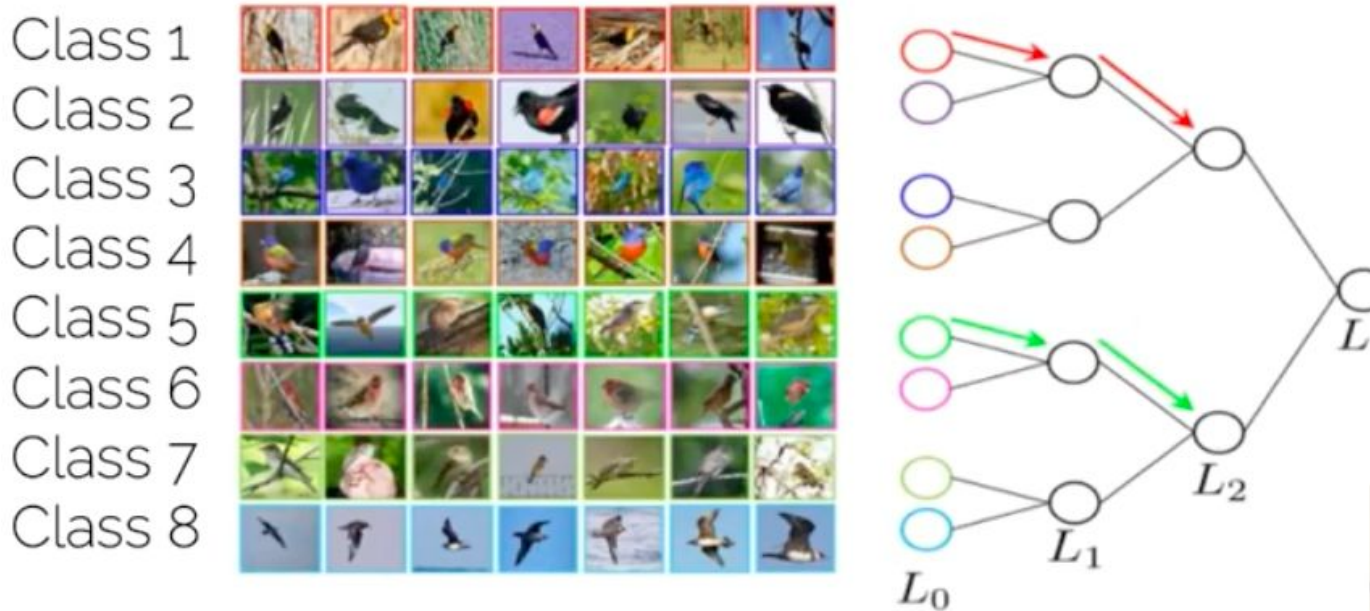
Ансамблирование

- Почему бы не использовать несколько сеток, каждая из которых обучена на подмножестве троек

Использование классификационного лосса для similarity learning

Sampling: Hierarchical Triplet loss

Строим дерево классов, где каждый лист дерева представляет класс изображения. Рекурсивно мерджим их, пока не дойдем до корневого узла



Sampling: Hierarchical Triplet loss

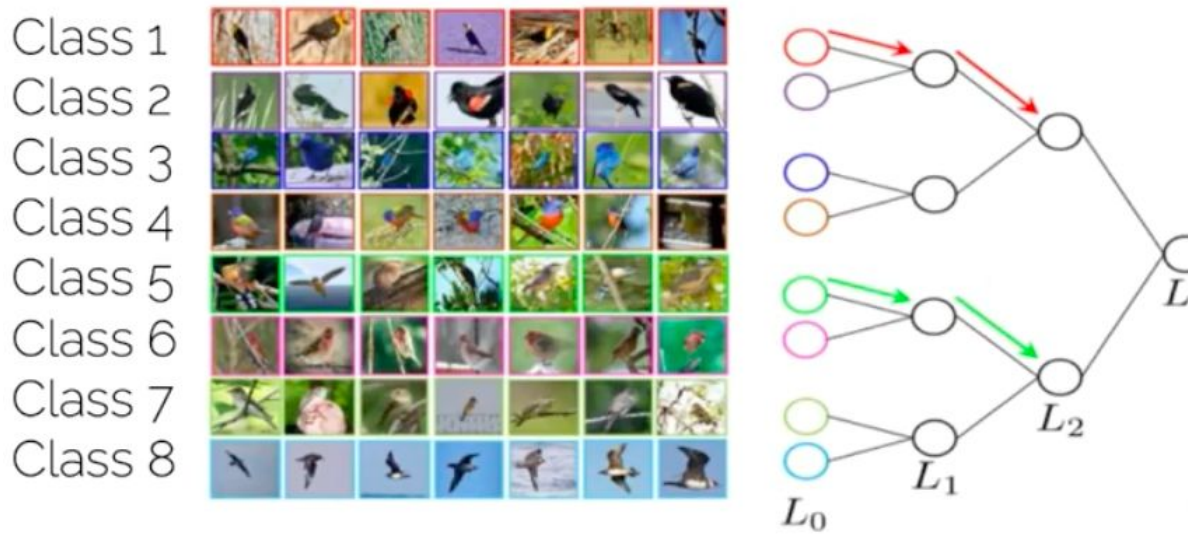
Для того чтобы создать дерево, мы вначале определяем расстояние между классами. Интуиция: если расстояние маленькое - они смерджатся на следующем уровне дерева

$$d(p, q) = \frac{1}{\underbrace{n_p n_q}_{\text{The cardinality of classes p and q (how many}} \sum_{i \in p, j \in q} \underbrace{\|\mathbf{r}_i - \mathbf{r}_j\|^2}_{\text{Deep features of images i and j}}$$

Sampling: Hierarchical Triplet loss

Рандомно выбираем l' узлов на нулевом уровне

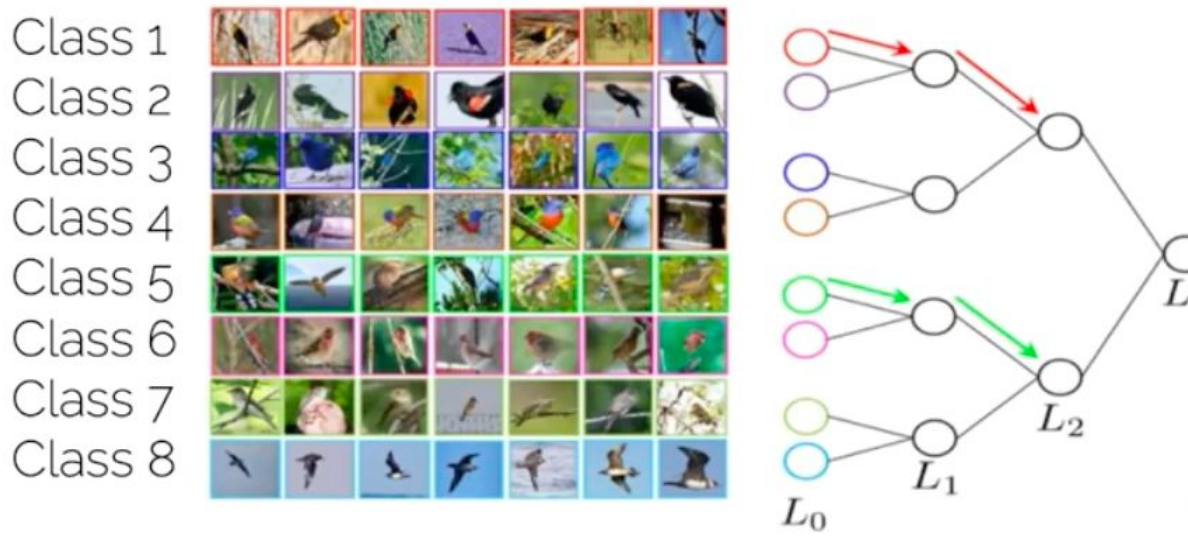
- Для того чтобы сохранить разнообразие классов в мини-батчах



Sampling: Hierarchical Triplet loss

Рандомно выбираем l' узлов на нулевом уровне

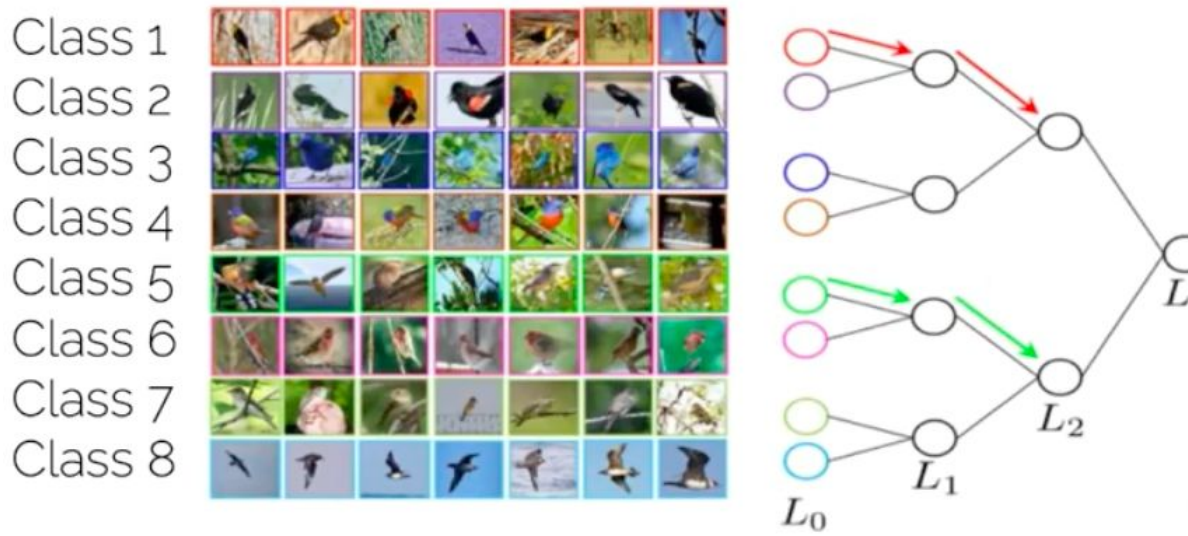
- Для того чтобы сохранить разнообразие классов в мини-батчах $m-1$ ближайших классов на нулевом уровне выбираются для каждого из l' узлов по расстоянию в пространстве фичей
- Мы хотим научить модель распознавать визуально похожие классы



Sampling: Hierarchical Triplet loss

Рандомно выбираем l' узлов на нулевом уровне

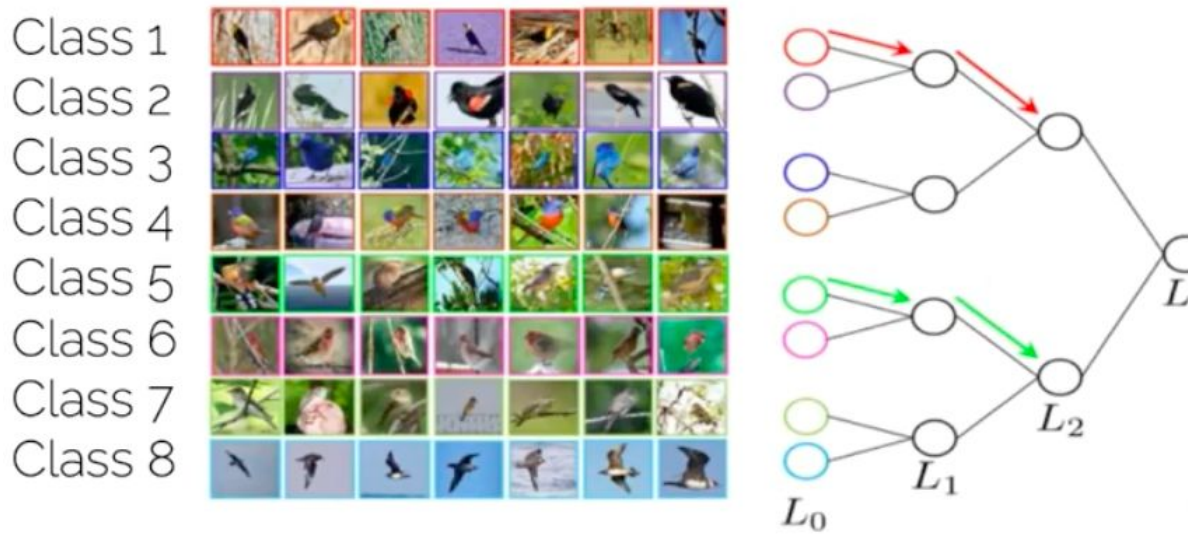
- Для того чтобы сохранить разнообразие классов в мини-батчах $m-1$ ближайших классов на нулевом уровне выбираются для каждого из l' узлов по расстоянию в пространстве фичей
- Мы хотим научить модель распознавать визуально похожие классы t изображений на каждый класс выбираются случайно



Sampling: Hierarchical Triplet loss

Рандомно выбираем l' узлов на нулевом уровне

- Для того чтобы сохранить разнообразие классов в мини-батчах $m-1$ ближайших классов на нулевом уровне выбираются для каждого из l' узлов по расстоянию в пространстве фичей
- Мы хотим научить модель распознавать визуально похожие классы t изображений на каждый класс выбираются случайно $t*m*l'$ изображений в минибатче



HTL: Loss

$$\mathcal{L}_{\mathcal{M}} = \frac{1}{2Z_{\mathcal{M}}} \sum_{\mathcal{T}^z \in \mathcal{T}^{\mathcal{M}}} [\|\mathbf{x}_a^z - \mathbf{x}_p^z\| - \|\mathbf{x}_a^z - \mathbf{x}_n^z\| + \alpha_z]_+$$

all the triplets

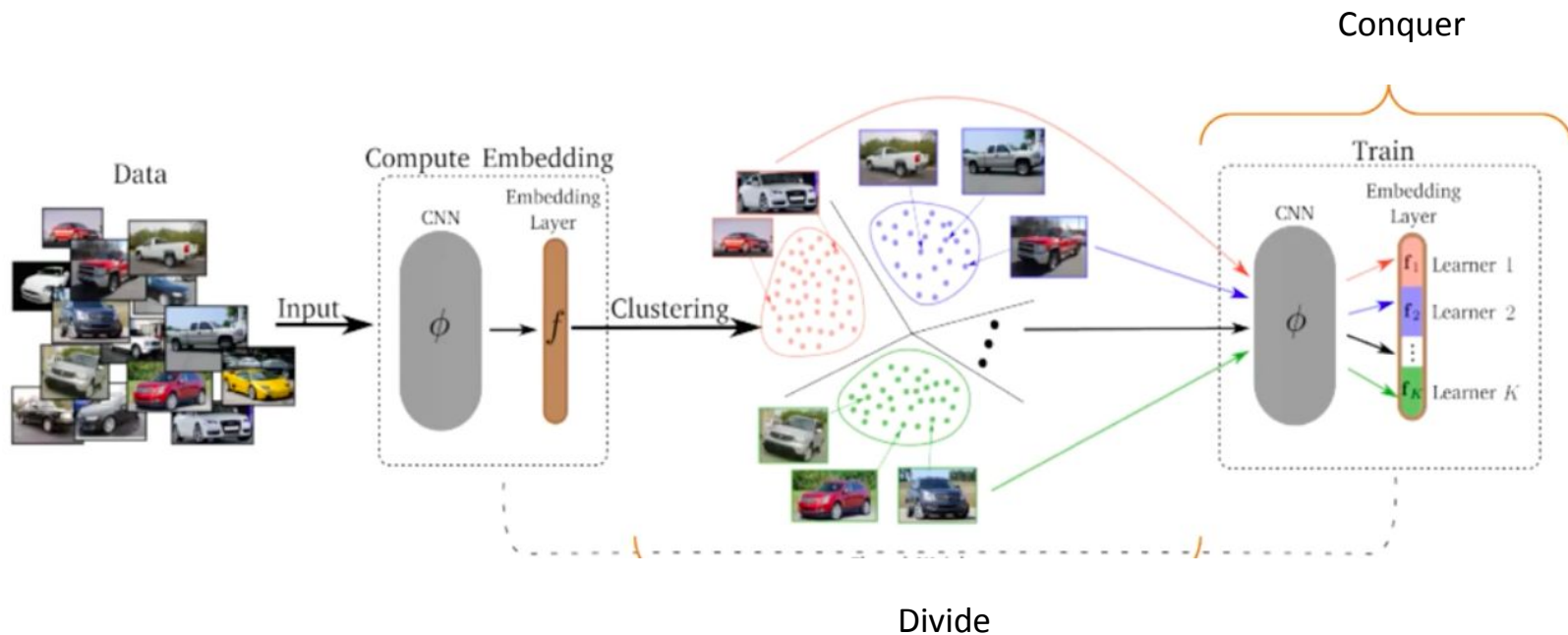
The margin actually depends on the distances computed on the hierarchical tree. The idea is that it can adapt to class distributions and differences of the samples within the classes.

Sampling: interesting works

- Manmatha et al., Sampling matter for deep metric learning, (ICCV 2017) - original sampling method
- Xu et al., Deep asymmetric metric learning via rich relationship mining, (CVPR 2019)
- Duan et al., Deep embedding learning with discriminative sampling policy, (CVPR 2019)
- Wang et al., Ranked list loss for deep metric learning (CVPR 2019)

Ensembles

Идея: разделим пространство на K кластеров, и для каждого кластера будем иметь свой классификатор

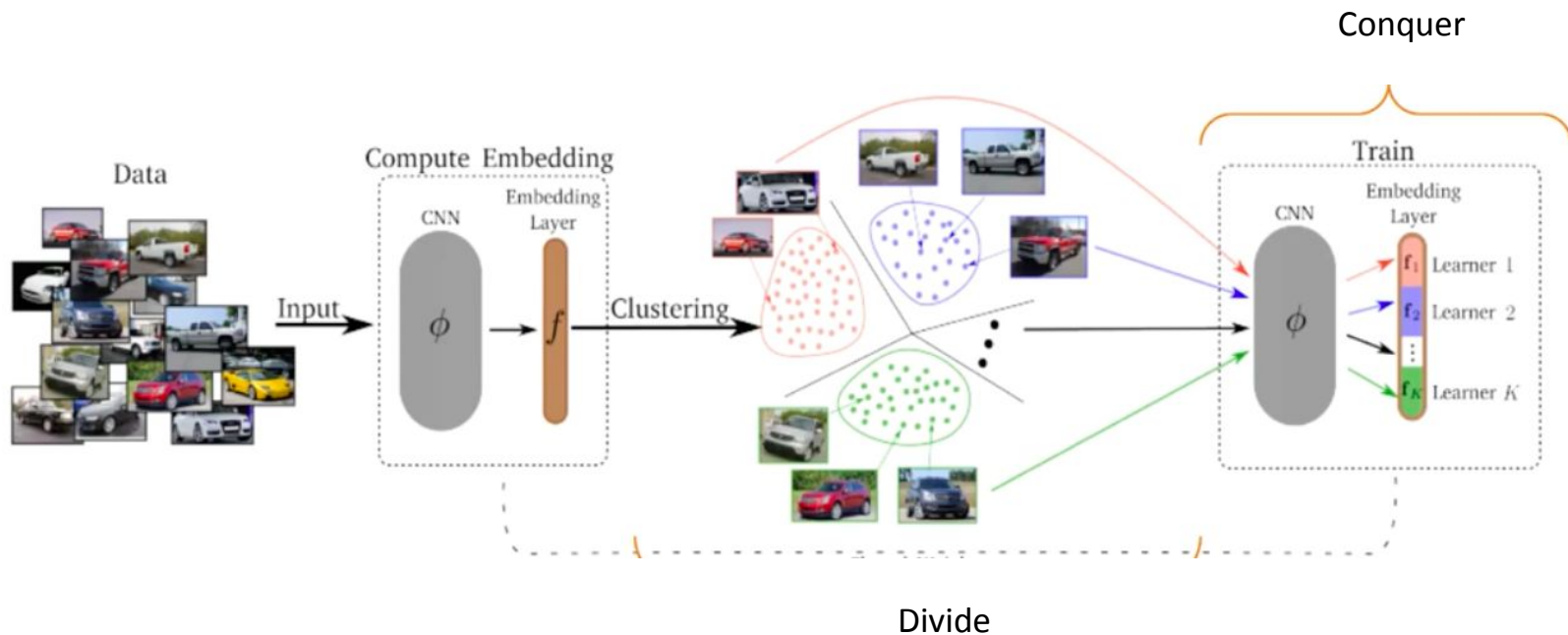


Ensembles: Divide and conquer

- 1) Кластеризуем пространство эмбеддингов в K кластеров используя K-means
- 2) Построить K независимых моделей (полносвязные слои) после CNN, где каждая модель отвечает за один кластер **DIVIDE**
- 3) После того как модель обучилась, сэмплировать один мини батч из рандомного кластера, и обновлять только соответствующую ему модель
- 4) Теперь файнтюним все модели в одно и тоже время **CONQUER**
- 5) Возвращаем на шаг (1) и повторяем несколько раз

Ensembles

Идея: разделим пространство на K кластеров, и для каждого кластера будем иметь свой классификатор



Ensembles: interesting works

- Opitz et al., BIER - Boosting Independent Embeddings Robustly, ICCV 2017 - train K independent networks.
- Elezi et al., The Group Loss for Metric Learning, arXiv 2020 - train K independent networks and concatenate their features.
- Yuan et al., Hard-Aware Deeply Cascaded Embedding, CVPR 2017 - concatenate features from different levels of the network.
- Wang et al., Ranked list loss for deep metric learning, CVPR 2019 - concatenate features from different levels of the network.
- Kim et al., Attention-based Ensemble for Deep Metric Learning, ECCV 2018 - use an attention mechanism such that each learner

Divide

Improving similarity learning

Loss:

-Contrastive vs triplet loss

Sampling:

- Выбор наилучших троек для обучения, умное сэмплирование - разнородность классов + сложные примеры

Ансамблирование

- Почему бы не использовать несколько сеток, каждая из которых обучена на подмножестве троек

Использование классификационного лосса для similarity learning

Classification loss: interesting works

- Movshovitz-Attias et al., No Fuss Distance Metric Learning using Proxies, ICCV 2017 - learn "proxy" samples to keep as positives and negatives in the mini-batch).
- Teh et al., ProxyNCA++: Revisiting and Revitalizing Proxy Neighborhood Component Analysis, arXiv 2020 - a better way of using proxies, some of the best results in the field.
- Qian et al., SoftTriple Loss: Deep Metric Learning Without Triplet Sampling, ICCV 2019 - using multiple centers for class
- Elezi et al., The Group Loss for Deep Metric Learning, arXiv 2020 - refine the softmax probabilities via a dynamical system for better

Some results

	CUB-200-2011					CARS 196					Stanford Online Products			
Loss	R@1	R@2	R@4	R@8	NMI	R@1	R@2	R@4	R@8	NMI	R@1	R@10	R@100	NMI
Triplet	42.5	55	66.4	77.2	55.3	51.5	63.8	73.5	82.4	53.4	66.7	82.4	91.9	89.5
Lifted Structure	43.5	56.5	68.5	79.6	56.5	53.0	65.7	76.0	84.3	56.9	62.5	80.8	91.9	88.7
Npairs	51.9	64.3	74.9	83.2	60.2	68.9	78.9	85.8	90.9	62.7	66.4	82.9	92.1	87.9
Facility Location	48.1	61.4	71.8	81.9	59.2	58.1	70.6	80.3	87.8	59.0	67.0	83.7	93.2	89.5
Angular Loss	54.7	66.3	76	83.9	61.1	71.4	81.4	87.5	92.1	63.2	70.9	85.0	93.5	88.6
Proxy-NCA	49.2	61.9	67.9	72.4	59.5	73.2	82.4	86.4	88.7	64.9	73.7	-	-	90.6
Deep Spectral	53.2	66.1	76.7	85.2	59.2	73.1	82.2	89.0	93.0	64.3	67.6	83.7	93.3	89.4
Classification	59.6	72	81.2	88.4	66.2	81.7	88.9	93.4	96	70.5	73.8	88.1	95	89.8
Bias Triplet	46.6	58.6	70.0	-	-	79.2	86.7	91.4	-	-	63.0	79.8	90.7	-
Group Loss	64.3	75.8	84.1	90.5	67.9	83.7	89.9	93.7	96.3	70.7	75.1	87.5	94.2	90.8
SoftTriple	65.4	76.4	84.5	90.4	69.3	84.5	90.7	94.5	96.9	70.1	78.3	90.3	95.9	92
HORDE	66.8	77.4	85.1	91	-	86.2	91.9	95.1	97.2	-	80.1	91.3	96.2	-

Some results

Loss+Sampling	CUB-200-2011					CARS 196					Stanford Online Products			
	R@1	R@2	R@4	R@8	NMI	R@1	R@2	R@4	R@8	NMI	R@1	R@10	R@100	NMI
Samp. Matt.	63.6	74.4	83.1	90.0	69.0	79.6	86.5	91.9	95.1	69.1	72.7	86.2	93.8	90.7
Hier. triplet	57.1	68.8	78.7	86.5	-	81.4	88.0	92.7	95.7	-	74.8	88.3	94.8	-
DAMLRMM	55.1	66.5	76.8	85.3	61.7	73.5	82.6	89.1	93.5	64.2	69.7	85.2	93.2	88.2
DE-DSP	53.6	65.5	76.9	61.7	-	72.9	81.6	88.8	-	64.4	68.9	84.0	92.6	89.2
RLL 1	57.4	69.7	79.2	86.9	63.6	74	83.6	90.1	94.1	65.4	76.1	89.1	95.4	89.7
GPW	65.7	77.0	86.3	91.2	-	84.1	90.4	94.0	96.5	-	78.2	90.5	96.0	-
Teacher-Student														
RKD	61.4	73.0	81.9	89.0	-	82.3	89.8	94.2	96.6	-	75.1	88.3	95.2	-
Loss+Ensembles														
BIER 6	55.3	67.2	76.9	85.1	-	75.0	83.9	90.3	94.3	-	72.7	86.5	94.0	-
HDC 3	54.6	66.8	77.6	85.9	-	78.0	85.8	91.1	95.1	-	70.1	84.9	93.2	-
ABE 2	55.7	67.9	78.3	85.5	-	76.8	84.9	90.2	94.0	-	75.4	88.0	94.7	-
ABE 8	60.6	71.5	79.8	87.4	-	85.2	90.5	94.0	96.1	-	76.3	88.4	94.8	-
A-BIER 6	57.5	68.7	78.3	86.2	-	82.0	89.0	93.2	96.1	-	74.2	86.9	94.0	-
D and C 8	65.9	76.6	84.4	90.6	69.6	84.6	90.7	94.1	96.5	70.3	75.9	88.4	94.9	90.2
RLL 3 [45]	61.3	72.7	82.7	89.4	66.1	82.1	89.3	93.7	96.7	71.8	79.8	91.3	96.3	90.4

Какой метод использовать?



CUB	Concatenated (512-dim)		
	P@1	RP	MAP@R
Pretrained	51.05	24.85	14.21
Contrastive	67.21 \pm 0.49	36.92 \pm 0.28	26.19 \pm 0.28
Triplet	64.40 \pm 0.38	34.63 \pm 0.36	23.79 \pm 0.36
ProxyNCA	66.14 \pm 0.32	35.48 \pm 0.18	24.56 \pm 0.18
Margin	65.48 \pm 0.50	35.04 \pm 0.24	24.10 \pm 0.26
N. Softmax	65.43 \pm 0.23	35.98 \pm 0.22	25.20 \pm 0.21
CosFace	67.19 \pm 0.37	37.36 \pm 0.23	26.53 \pm 0.23
ArcFace	67.06 \pm 0.31	37.23 \pm 0.17	26.35 \pm 0.17
FastAP	63.64 \pm 0.24	34.45 \pm 0.21	23.71 \pm 0.20
SNR	67.26 \pm 0.46	36.86 \pm 0.20	26.10 \pm 0.22
MS	65.93 \pm 0.16	35.91 \pm 0.11	25.16 \pm 0.10
MS+Miner	65.75 \pm 0.34	35.95 \pm 0.21	25.21 \pm 0.22
SoftTriple	66.20 \pm 0.37	36.46 \pm 0.20	25.64 \pm 0.21

CARS	Concatenated (512-dim)		
	P@1	RP	MAP@R
Pretrained	46.89	13.77	5.91
Contrastive	81.57 \pm 0.36	35.72 \pm 0.35	25.49 \pm 0.41
Triplet	77.48 \pm 0.57	32.85 \pm 0.45	22.13 \pm 0.45
ProxyNCA	83.25 \pm 0.37	36.63 \pm 0.34	26.39 \pm 0.41
Margin	82.08 \pm 2.41	34.71 \pm 2.17	24.14 \pm 2.25
N. Softmax	83.58 \pm 0.29	36.56 \pm 0.19	26.36 \pm 0.21
CosFace	85.27 \pm 0.23	36.72 \pm 0.20	26.86 \pm 0.22
ArcFace	83.95 \pm 0.23	35.44 \pm 0.26	25.24 \pm 0.27
FastAP	78.20 \pm 0.74	33.39 \pm 0.67	22.90 \pm 0.69
SNR	81.87 \pm 0.35	35.40 \pm 0.44	25.14 \pm 0.49
MS	85.29 \pm 0.31	37.96 \pm 0.63	27.84 \pm 0.77
MS+Miner	84.59 \pm 0.29	37.70 \pm 0.37	27.59 \pm 0.43
SoftTriple	83.66 \pm 0.22	36.31 \pm 0.16	26.06 \pm 0.19

Когда обучение корректно (и используется везде один и тот же backbone, одно и то же пространство эмбеддингов и никаких дополнительных трюков) разность в точность между различными моделями не велика

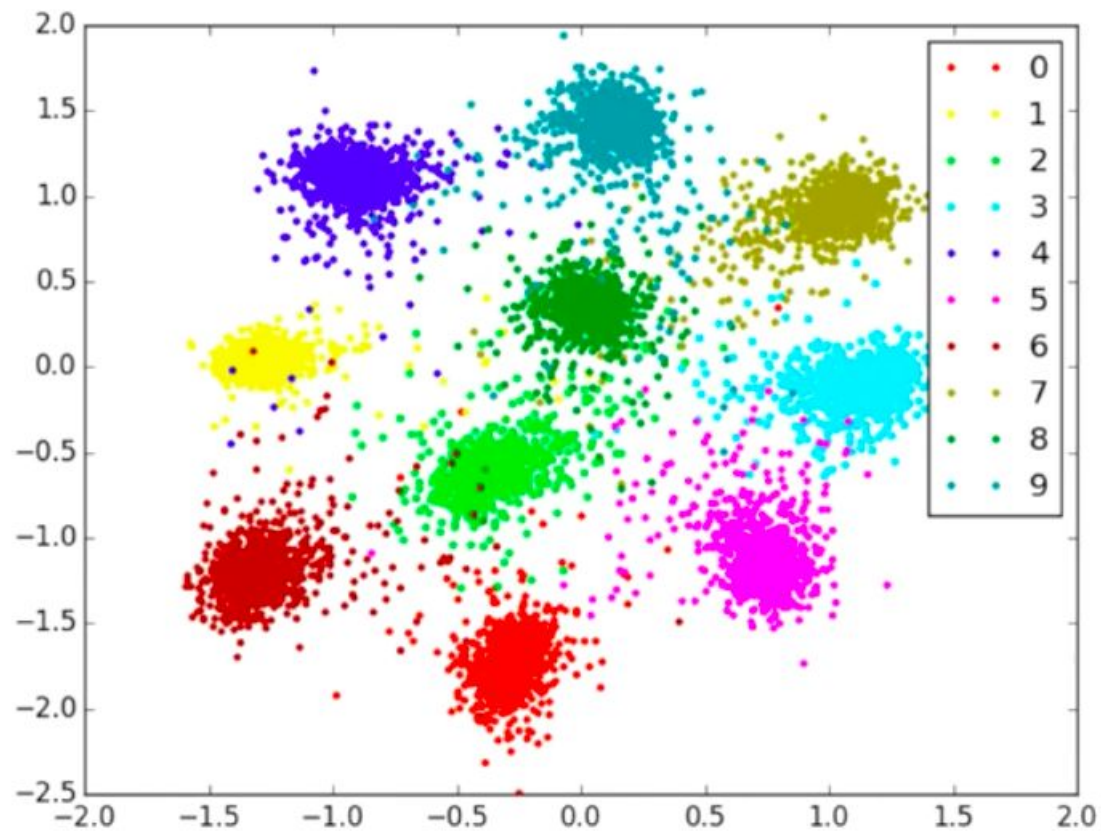
Советы:

1. Бейзлановые лоссы (contrastive loss, triplet loss) хорошо обучаются, когда обучение корректно
2. Сэмплирование - очень важно. Каждый метод может быть улучшен, если с умом подойти к выбору стратегии сэмплирования
3. Еще несколько трюков для улучшения перфоманса (температура для софтмакса, замораживание батч-норм слоев, использование множественных центров на один класс, и т.д.)
4. Даже наивные ансамбли значительно улучшают качество
5. Хороший out-of-box выбор: Proxy-NCA и SoftTriple Loss -> они хорошо работают, и не нуждаются в сложном поиске гиперпараметров(и есть код!)
6. Чем лучше backbone (напр. densenet), тем лучше результат

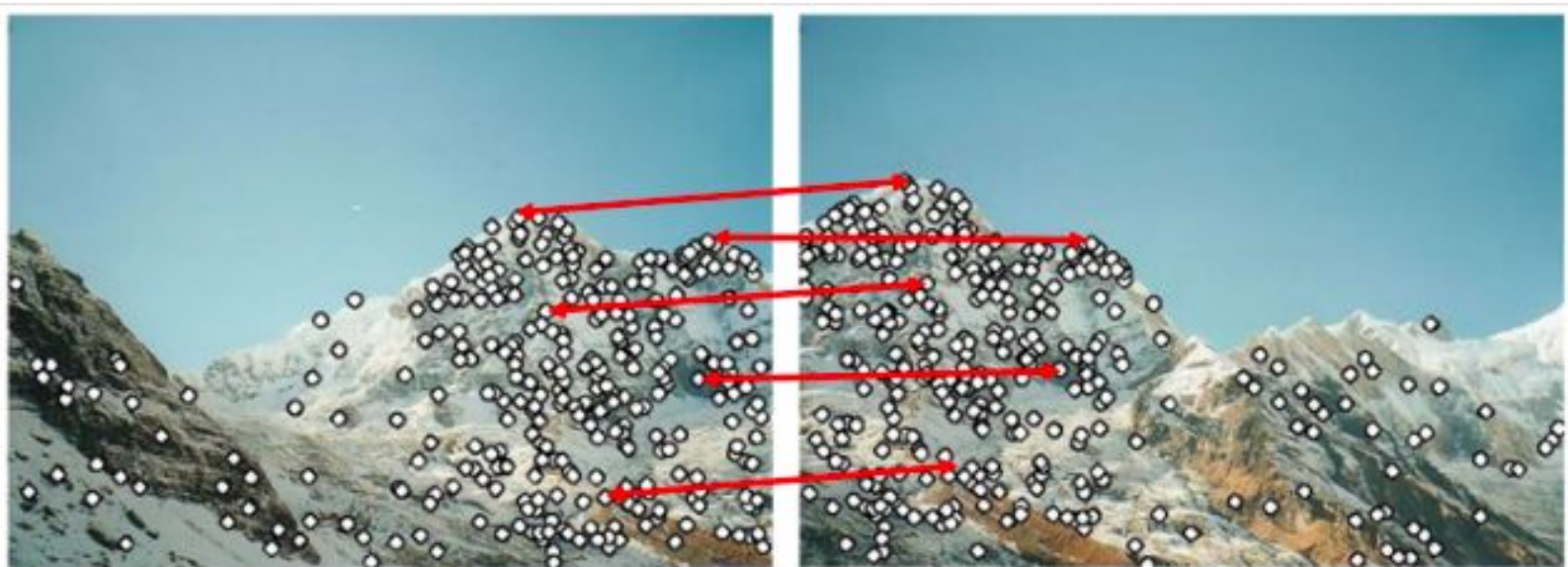


Приложения в компьютерном зрении

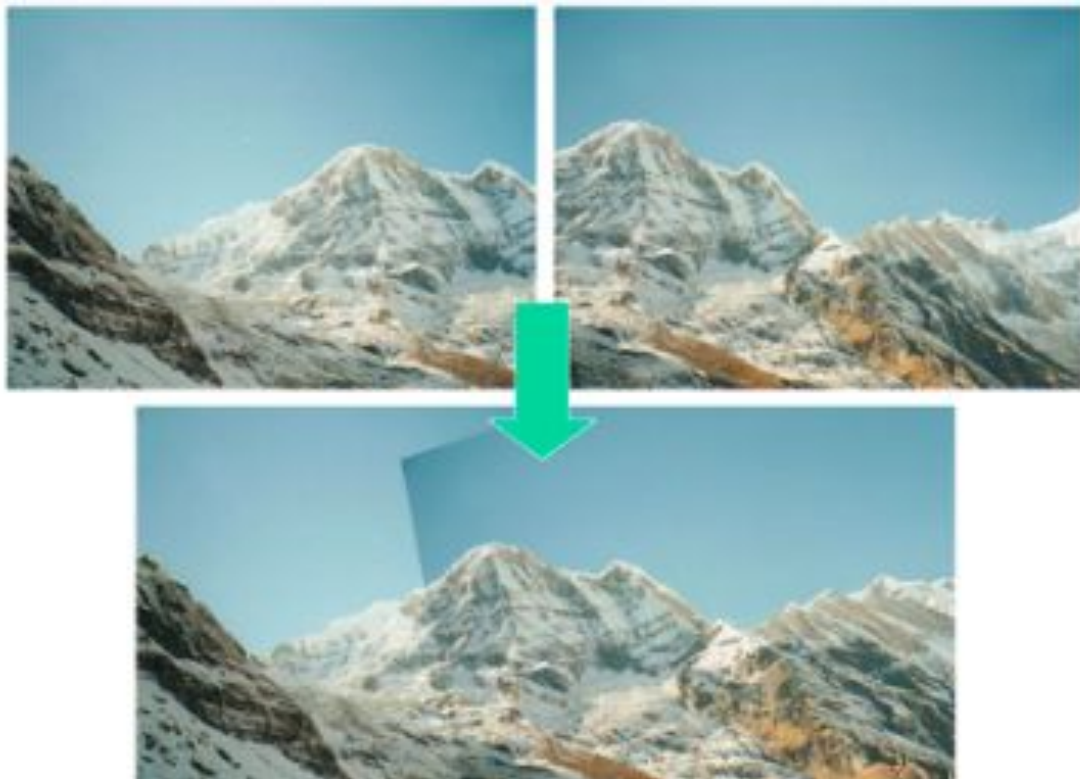
Siamese network on MNIST



Поиск мэтчей



Поиск мэтчей



Поиск мэтчей

Используется во многих областях компьютерного зрения:

- Image stitching
- Object recognition
- 3D reconstruction
- Object tracking
- Image retrieval

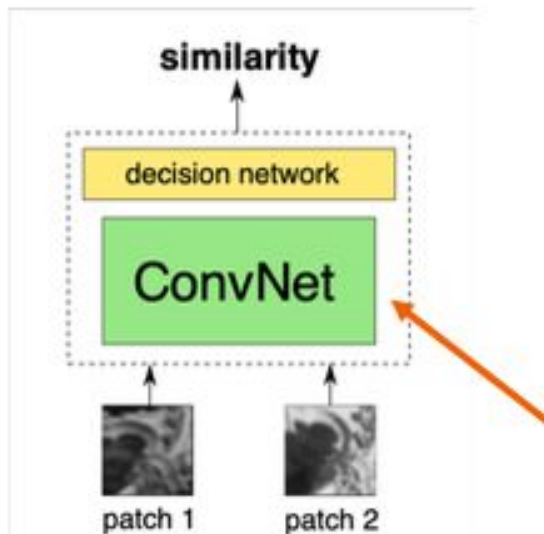
Все это можно делать с нейросетками!

Классический пайплайн:

- Извлечь дескрипторы
 - Harris, SIFT, SURF: в основе градиенты патча
 - Они сильно зависят от изменения освещенности или точки зрения
 - Не умею извлекать dense features
- Сматчить дескрипторы из двух изображений
 - Многие дескрипторы похожие, нужно избавиться от плохих матчей, и оставить только хорошие

Поиск мэтчей

- End-to-end learning for patch similarity



Быстрая, позволяет извлекать
dense features

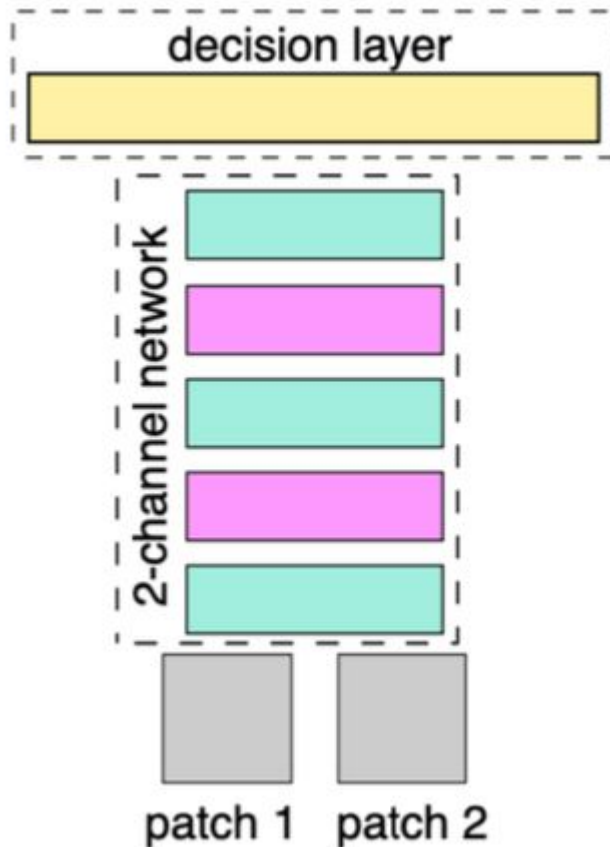
Инвариантна ко многим
преобразованиям

Siamese network

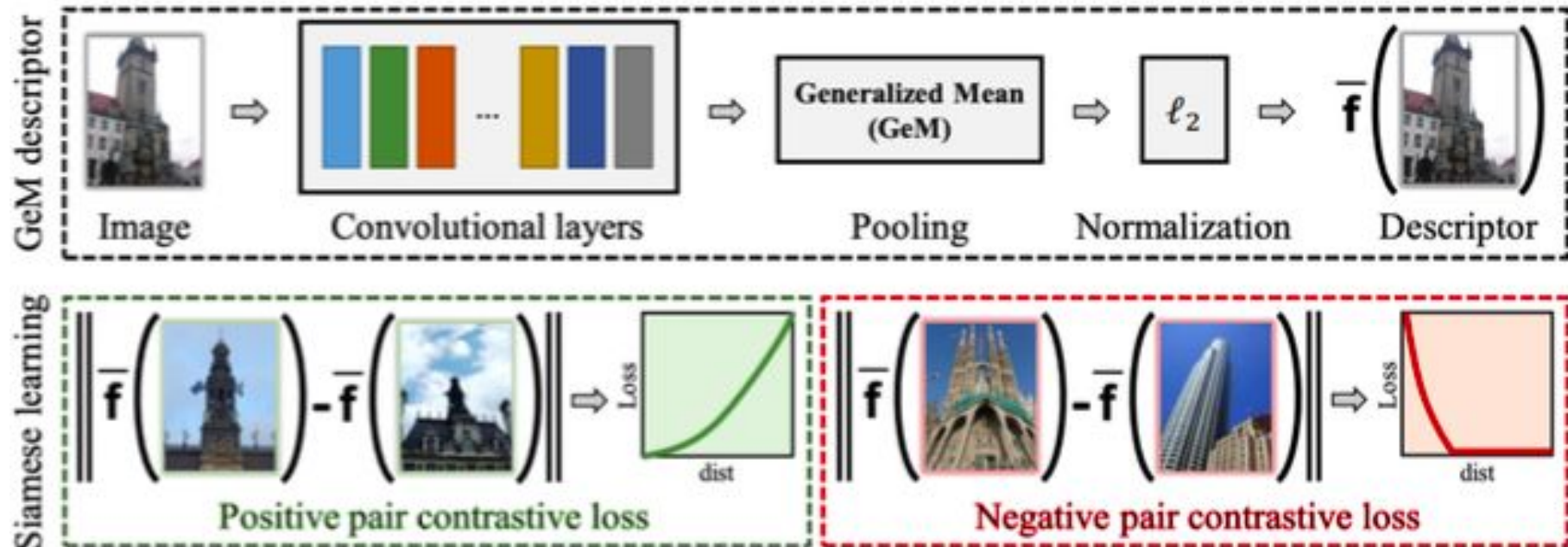
Поиск мэтчей

Классическая Siamese architecture

- Общие слои
 - Извлечение фичей
- Один слой для принятия решения
 - да/нет



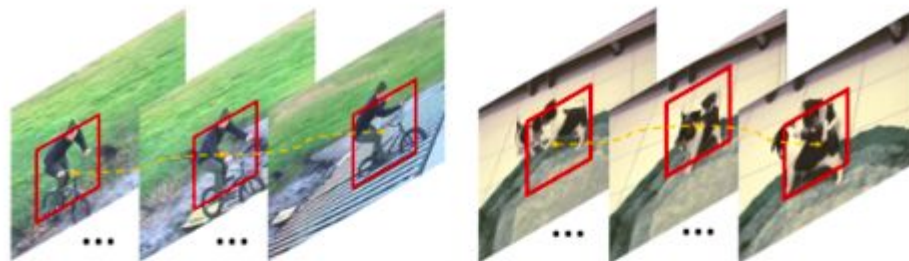
Поиск изображений



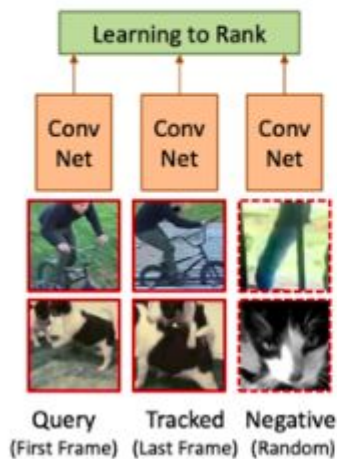
Unsupervised learning

Обучение на видео

- Трекинг - своего рода супервайзинг
- Используем их как положительные примеры
- Извлекаем случайные патчи как негативные примеры



(a) Unsupervised Tracking in Videos



(b) Siamese-triplet Network

$$\begin{aligned} D \left(\begin{bmatrix} \text{Query} \\ \text{Tracked} \end{bmatrix} \right) &< D \left(\begin{bmatrix} \text{Query} \\ \text{Negative} \end{bmatrix} \right) \\ D \left(\begin{bmatrix} \text{Query} \\ \text{Tracked} \end{bmatrix} \right) &< D \left(\begin{bmatrix} \text{Query} \\ \text{Negative} \end{bmatrix} \right) \end{aligned}$$

D : Distance in deep feature space

(c) Ranking Objective

Optical flow

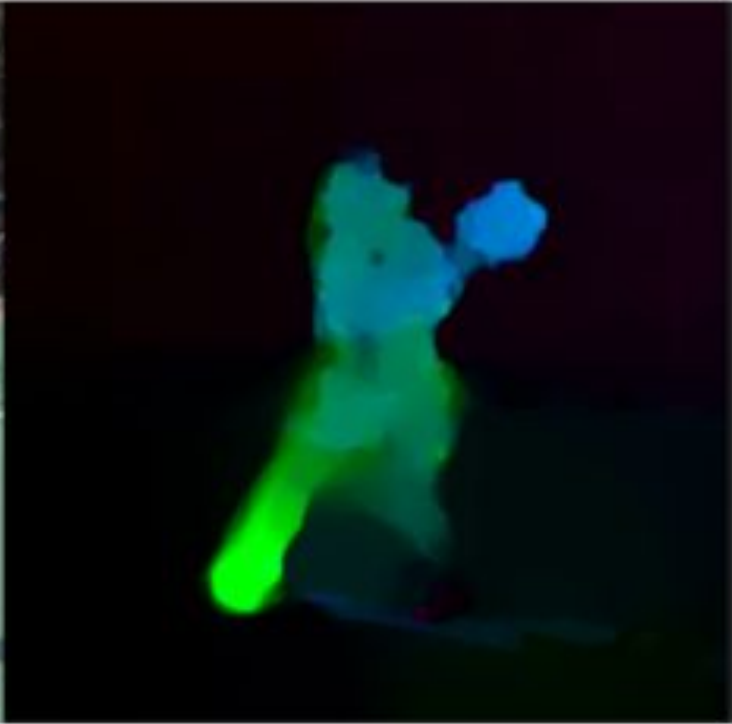
- Инпут:
2 последовательных изображения (напр. из видео)
- Аутпут:
смещение каждого пикселя из изображения А в изображение Б

Получается 2Д смещение объектов (не совсем реальное движение объектов)

Optical flow

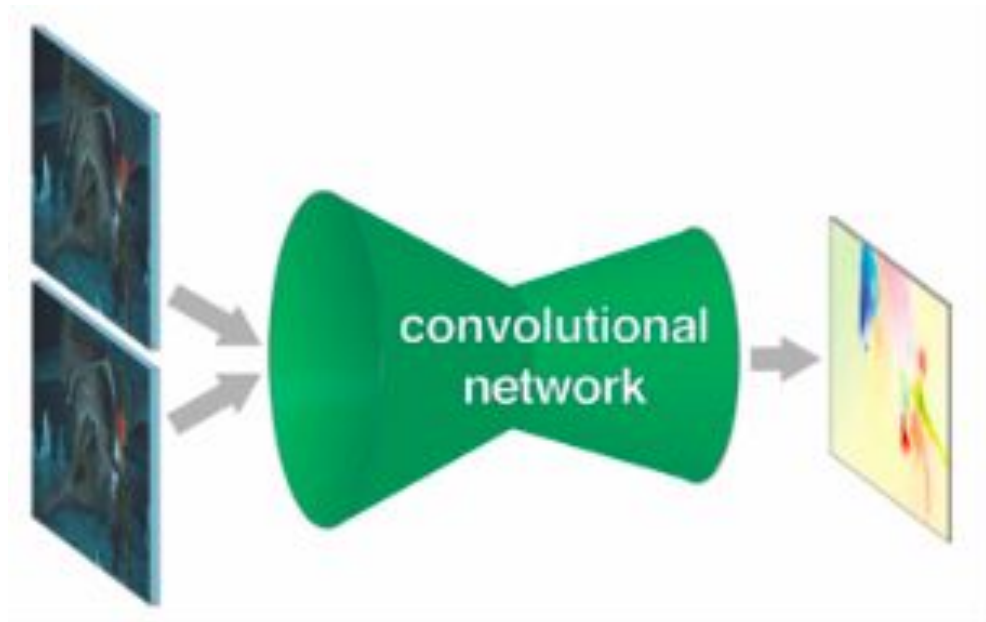


Optical flow



Optical flow

End-to-end supervised learning of optical flow



Optical flow

FlowNet: Learning Optical Flow with Convolutional Networks

Img1

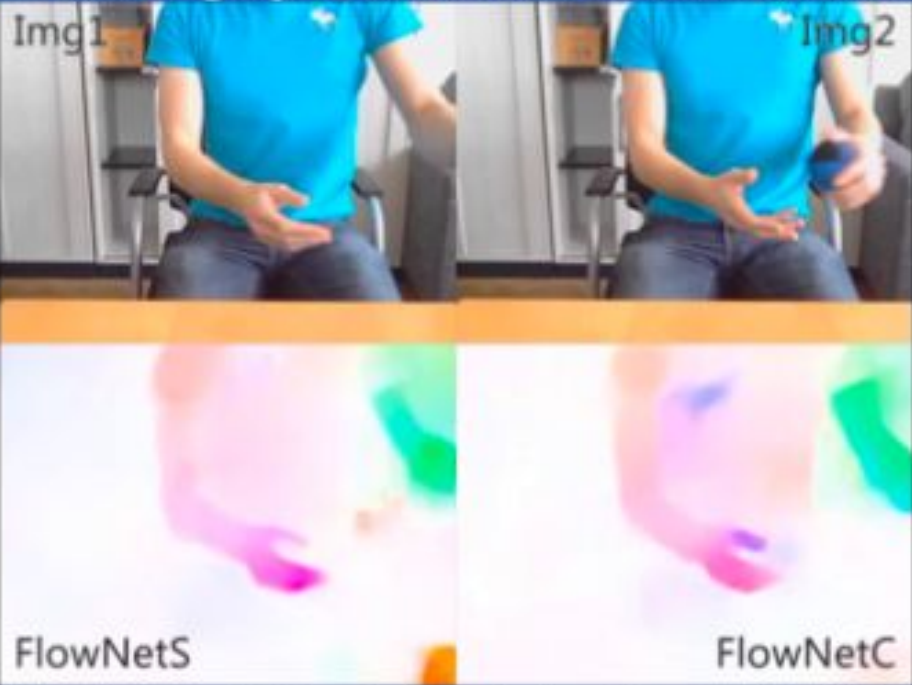
Img2

FlowNet
P. Fischer,
A. Dosovitskiy,
E. Ilg,
P. Häusser,
C. Hazirbas,
V. Golkov,
P. v.d. Smagt,
D. Cremers,
T. Brox

FlowNetS

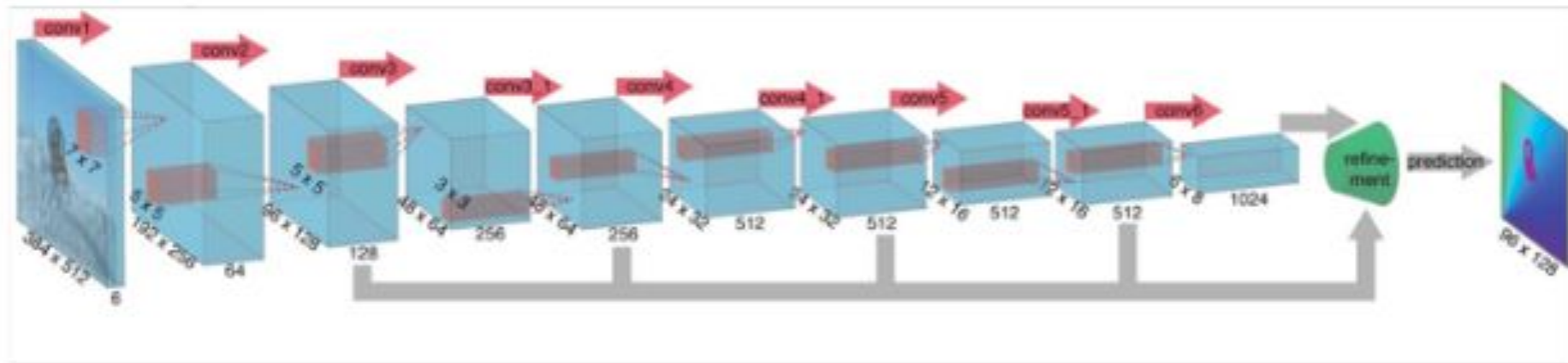
FlowNetC

We train convolutional networks to estimate optical flow.



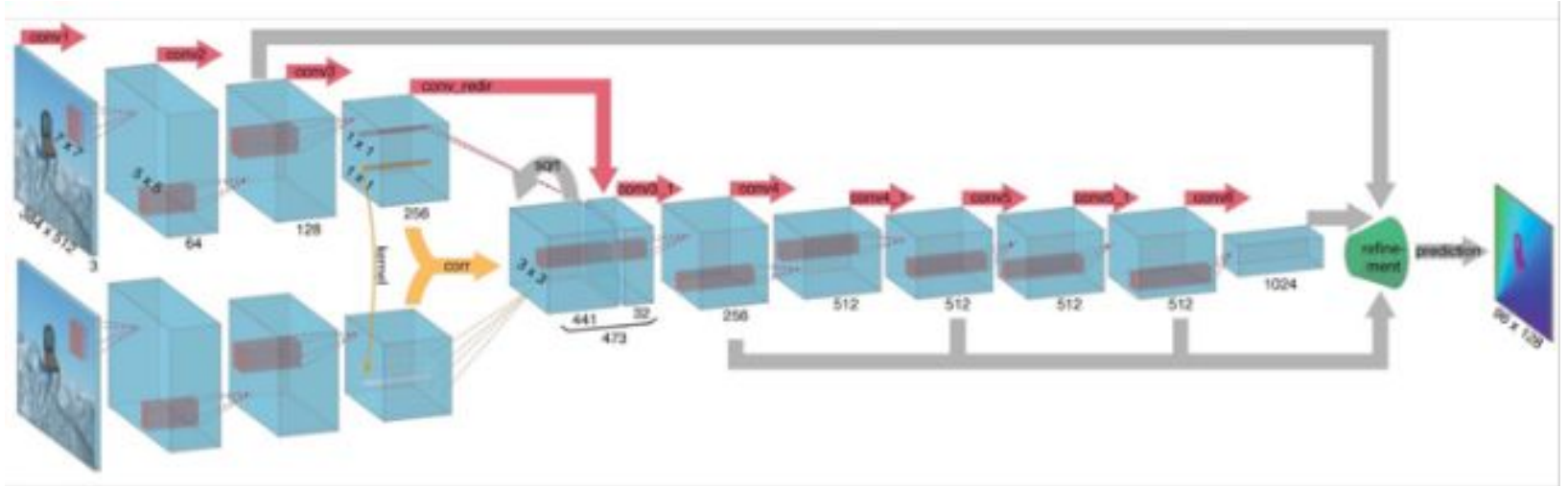
FlowNet1

Стэкаем оба изображения: инпут теперь 2xRGB= 6 каналов



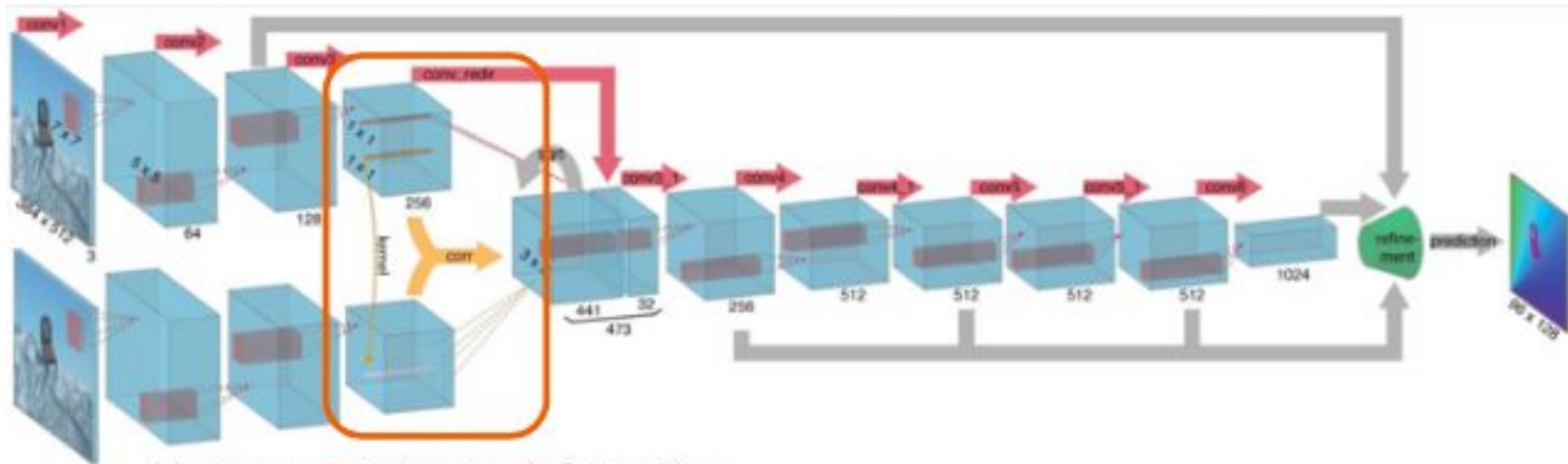
FlowNet 2

Siamese architecture



FlowNet 2

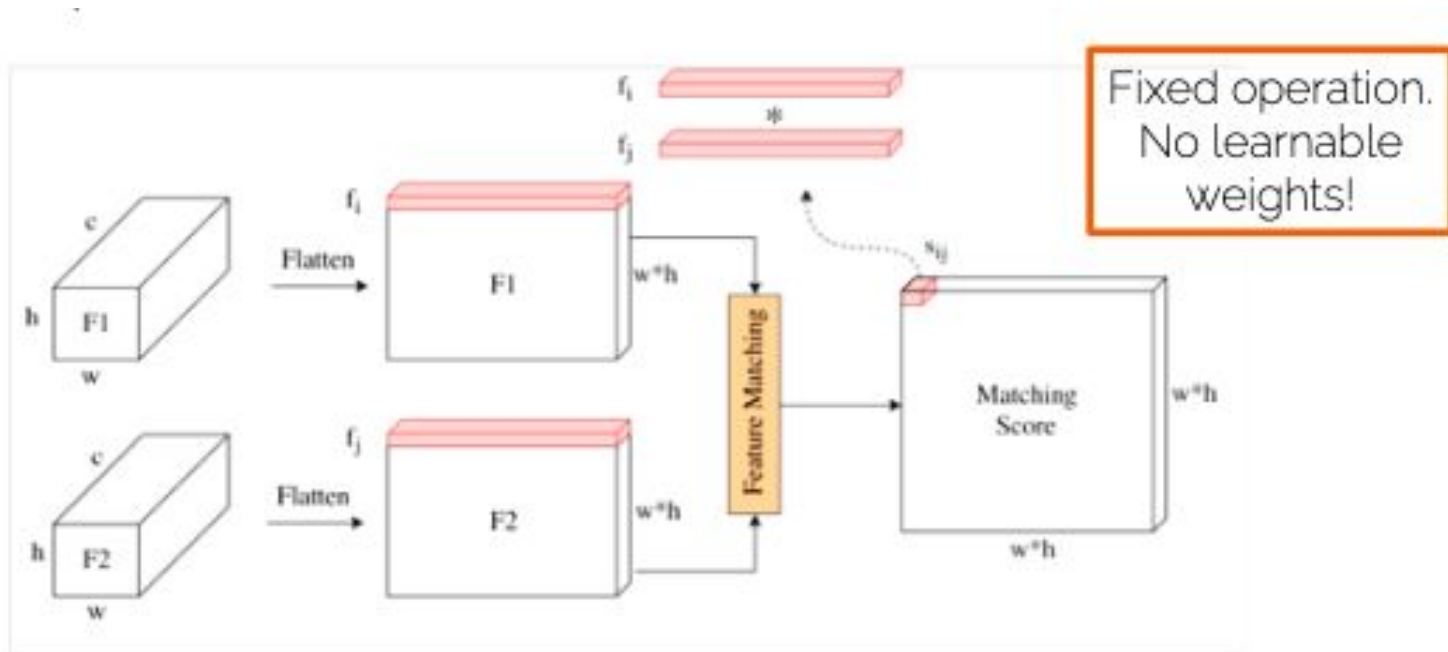
2 важные особенности



How to combine the information
from both images?

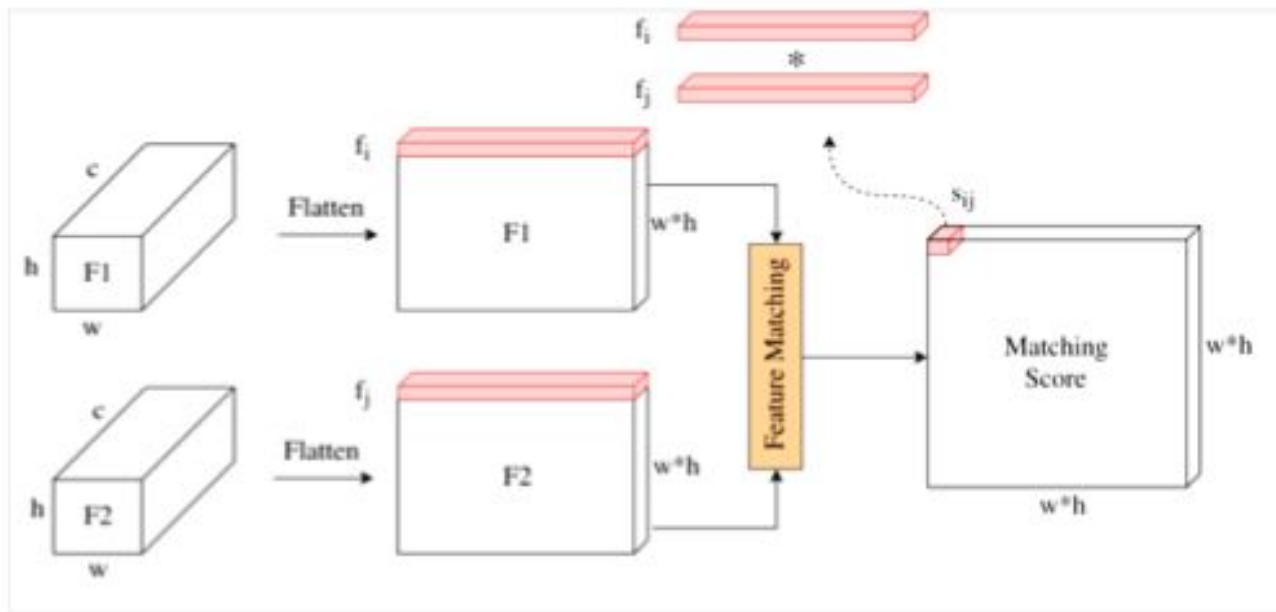
Correlation layer

Умножаем вектор фичей на другой вектор фичей



Correlation layer

Matching score определяет насколько скоррелированы эти два вектора фичей

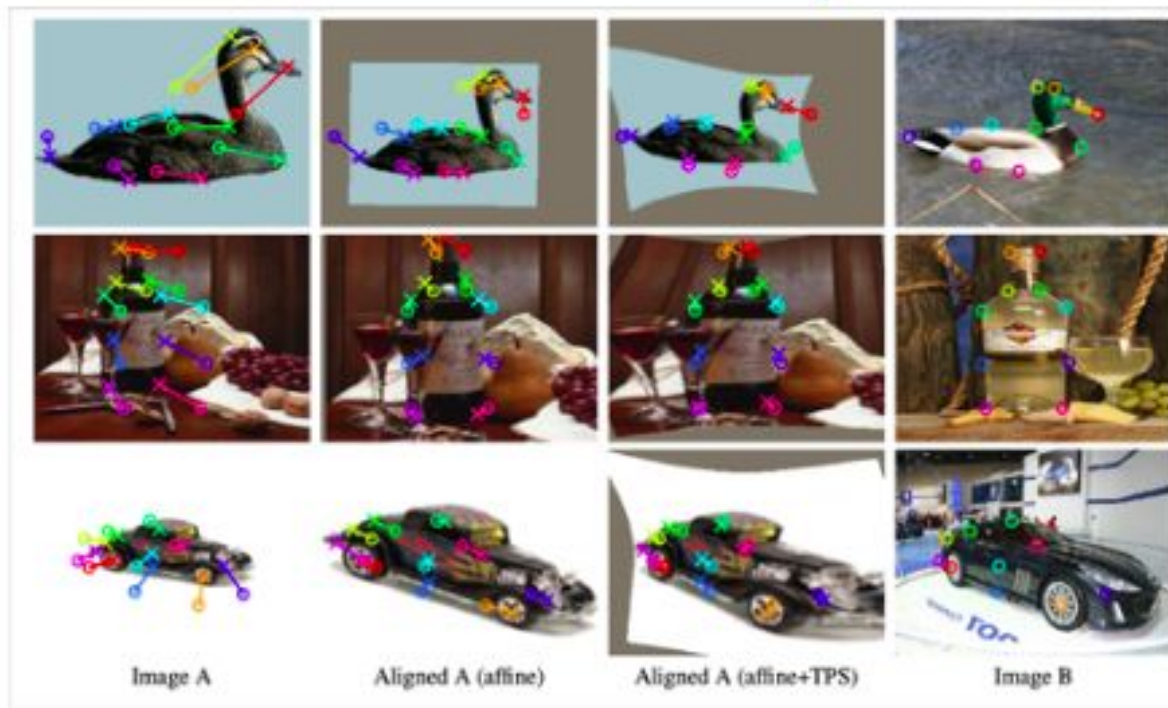


Correlation layer

Полезно для нахождения преобразований между изображениями

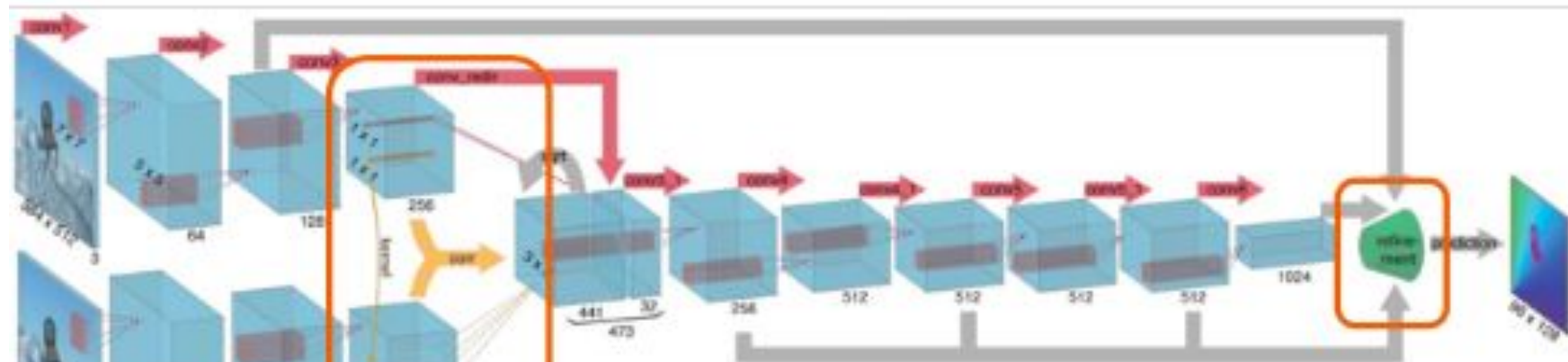


Correlation layer



I. Rocco et al. "Convolutional neural network architecture for geometric matching. CVPR 2017.

FlowNet 2



How to combine the information from both images?

How to obtain high-quality results?

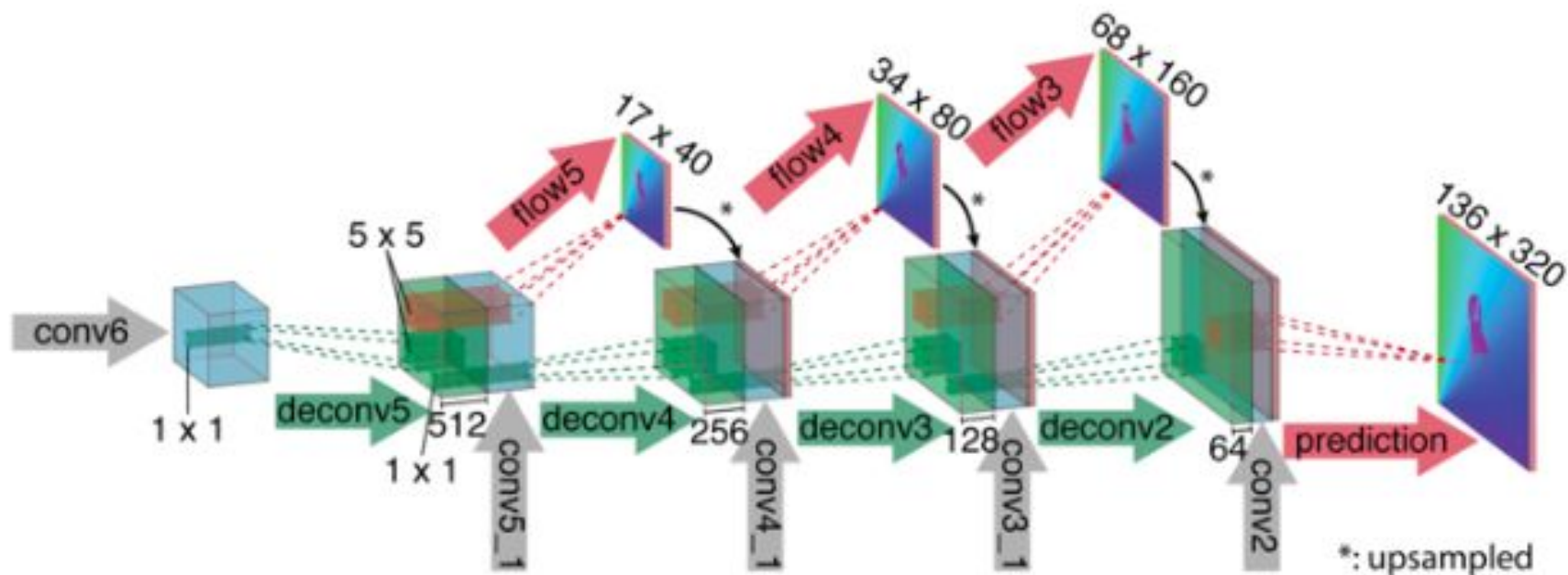
FlowNet 2

Свертки + пулинг отлично агрегируют информацию из различных частей изображения

Это также уменьшает количество вычислений

Проблема: значительно уменьшается размер инпута, а мы хотим, аутпут с большим количеством деталей

Refinement architecture

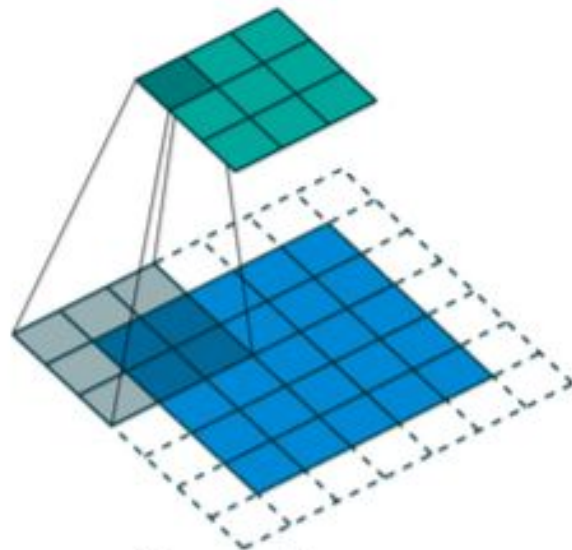


Transpose convolution

Вспомним:



Convolution
no padding, no stride



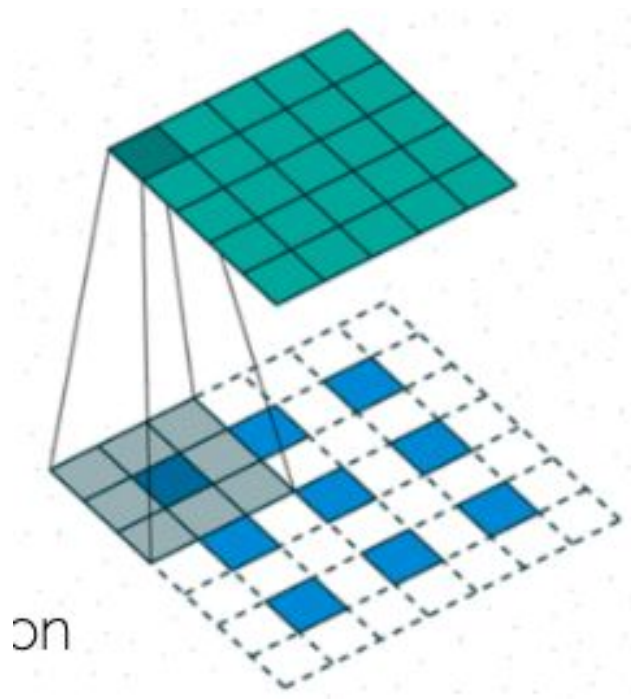
Convolution
padding, stride

Transpose convolution

Мы хотим преобразовать 3x3 инпут в 5x5 аутпут

“Умный” паддинг + нормальная конволюция

Unpooling + conv = upconvolution





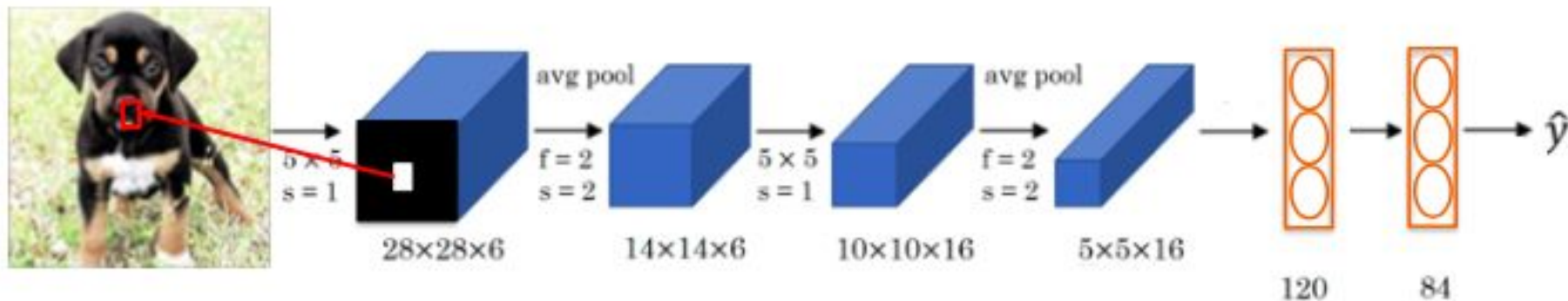
Visualization

Визуализация CNN

- Визуализация фичей
- Визуализация активация
- Визуализация градиентов
- T-SNE Visualization
- DeepDream
- ...

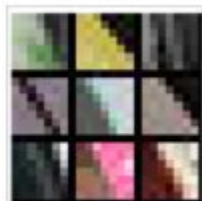
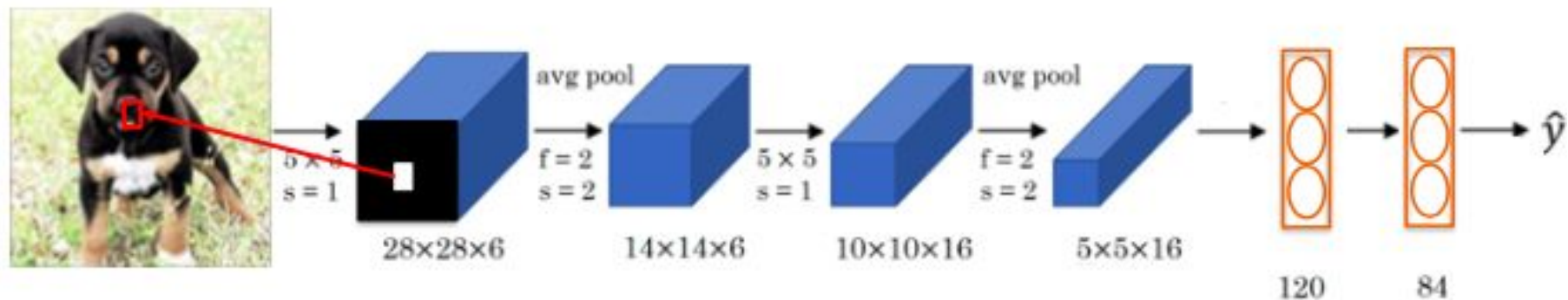
Визуализация - отличный инструмент для дебаггинга!

Визуализация



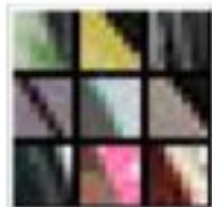
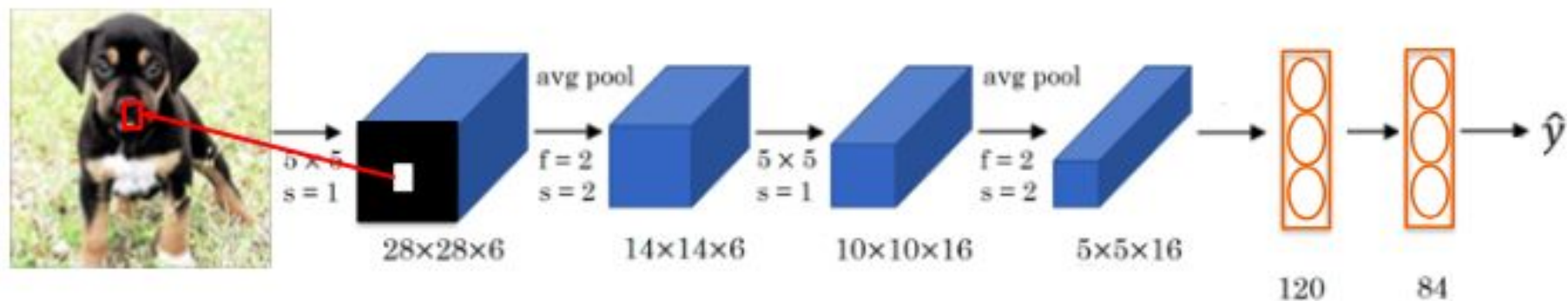
- Выбрать юнит в первом слое
- Найти 9 патчей изображения в датасет которые максимизируют активацию в этом слое

Визуализация



Feature map 1, layer 1, 9 image patches that provided the highest activation

Визуализация



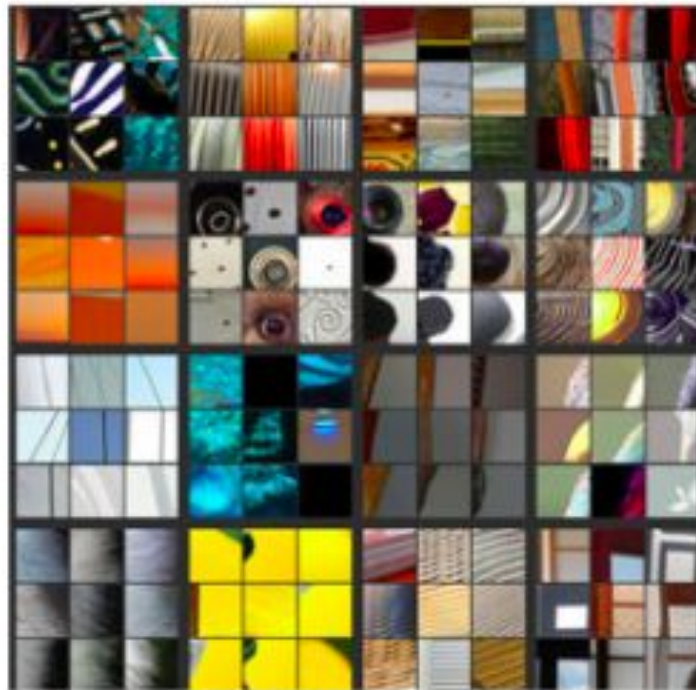
Feature map 2, layer 1, 9 image patches that provided the highest activation

Визуализация

Layer 1

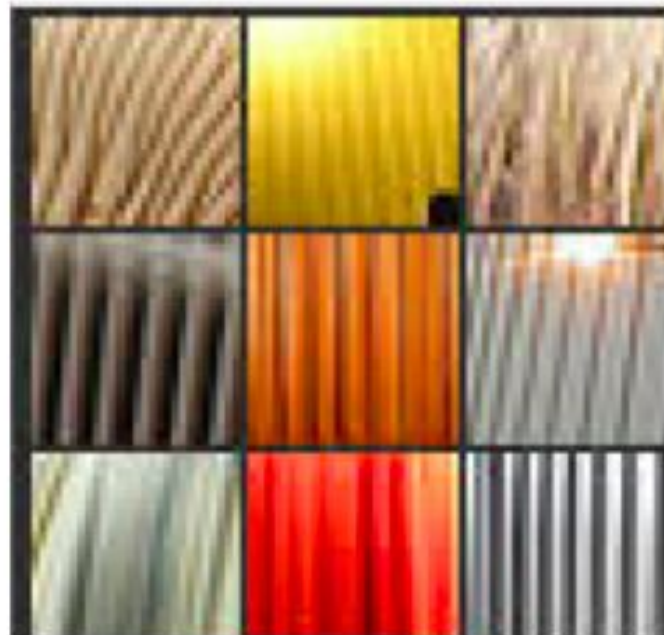
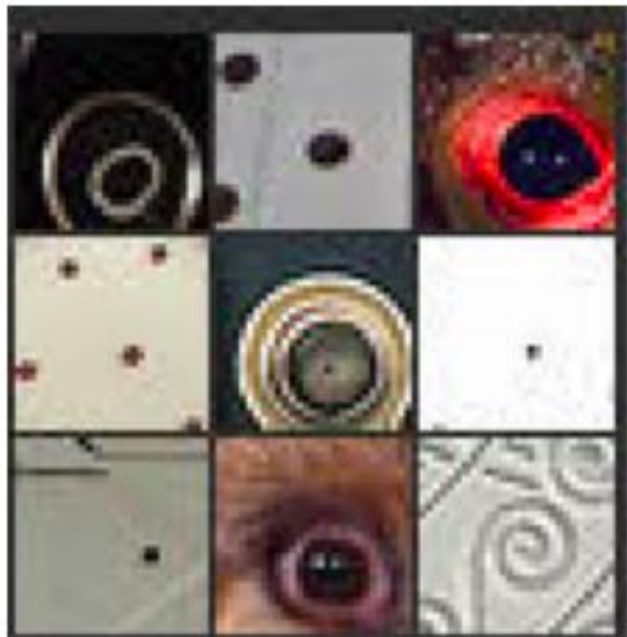


Layer 2



Визуализация

Zoom in, examples of Layer 2



Визуализация

Zoom in, examples of Layer 5



Визуализация

Zoom in, examples of Layer 5



Occlusion experiment

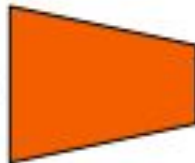
Закрываем различные части изображения и смотрим как
меняются предсказания модели



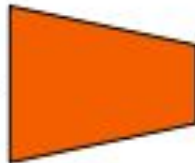
DOG 0.96

Occlusion experiment

Закрываем различные части изображения и смотрим как
меняются предсказания модели



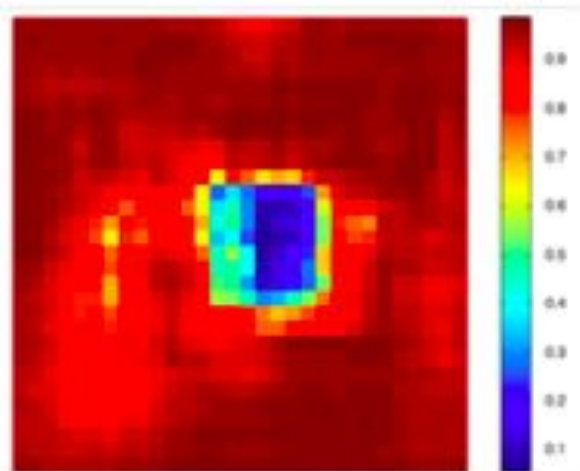
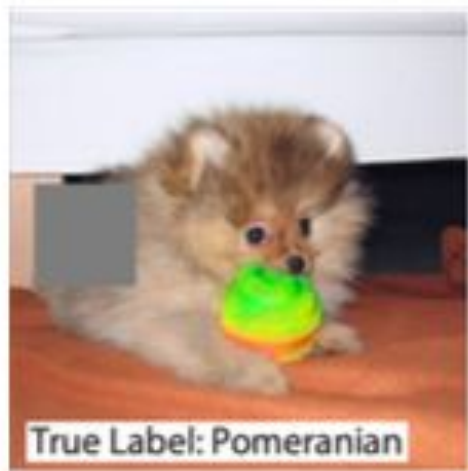
DOG 0.95



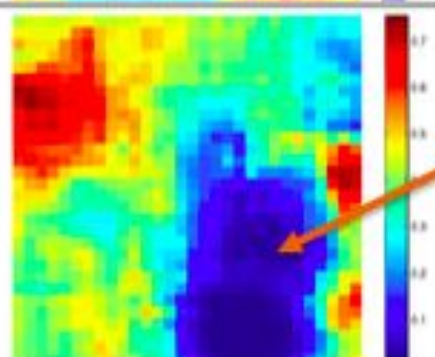
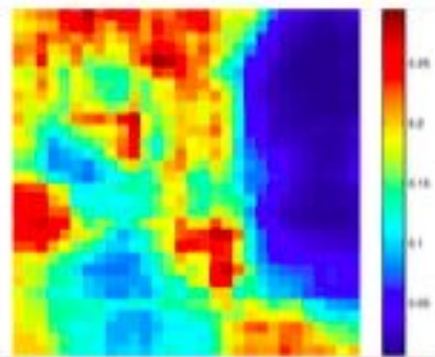
DOG 0.35

Occlusion experiment

Создаем хитмэп где каждый пиксель отражает вероятность “собаки”, если серый квадрат поставить на этот пиксель

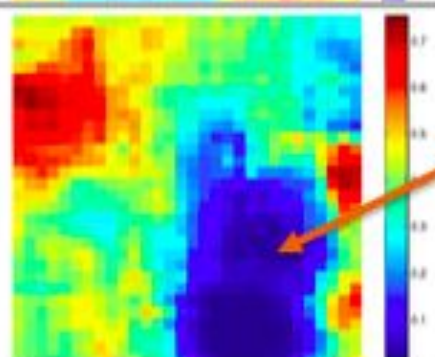
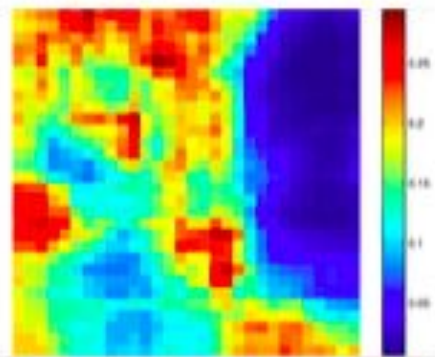


Occlusion experiment

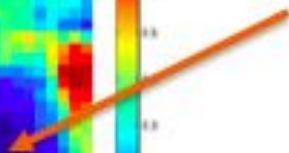


Most important
pixels for
classification

Occlusion experiment

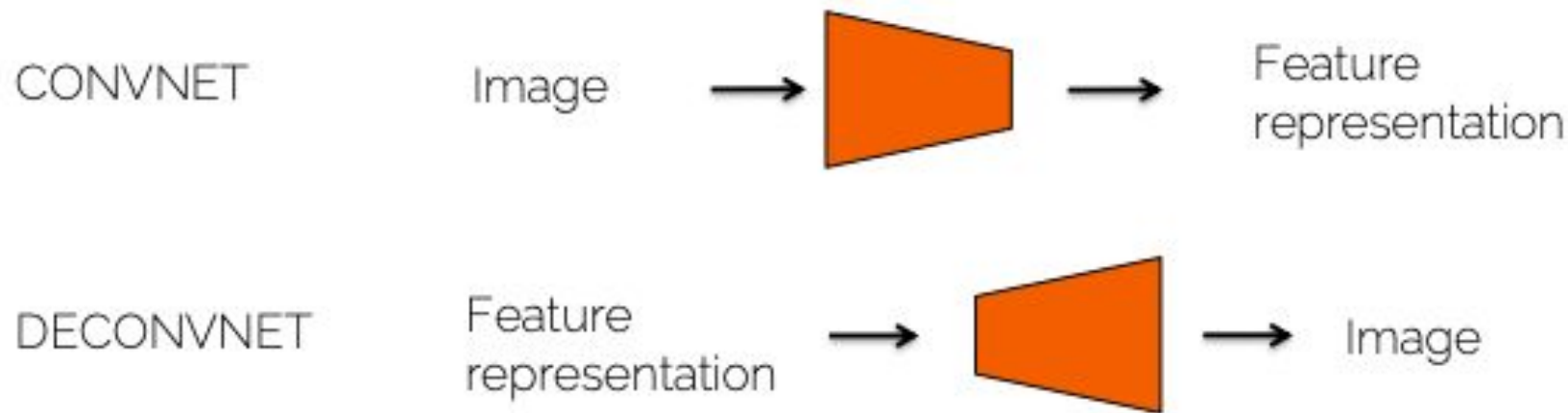


Most important
pixels for
classification



DeconvNet

Отображаем фичи обратно в пространство изображений

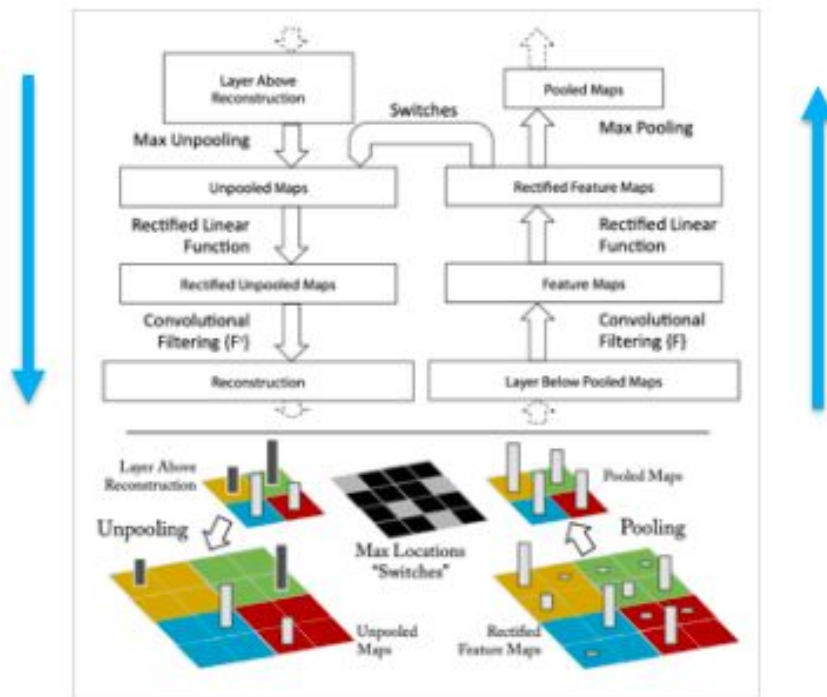


DeconvNet: визуализация

- Выбираем входное изображение
- Делаем форвард пасс
- Наблюдение: в 15 фильтре третьего слоя большое значение активаций с этого изображения
- Задача: визуализировать 15 фильтр 3 слоя
- Зануляем все остальные фильтры
- Прогоняем обратно через DeconvNet

DeconvNet: визуализация

DECONVNET



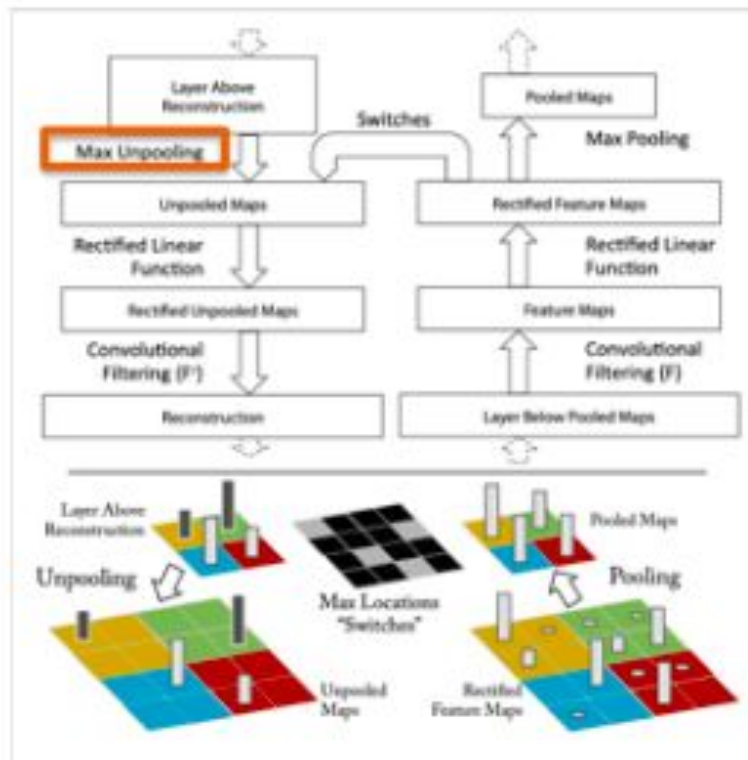
CONVNET

Zeiler and Fergus. „Visualizing and understanding convolutional neural networks“. ECCV 2014

DeconvNet: визуализация

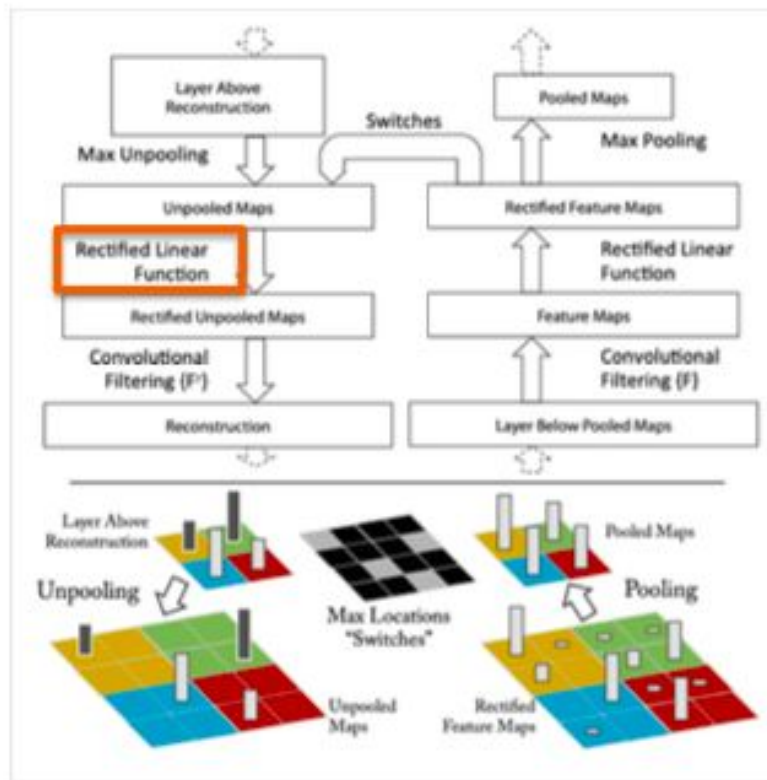
- Unpooling

Keep the
locations
where the max
came from



DeconvNet: визуализация

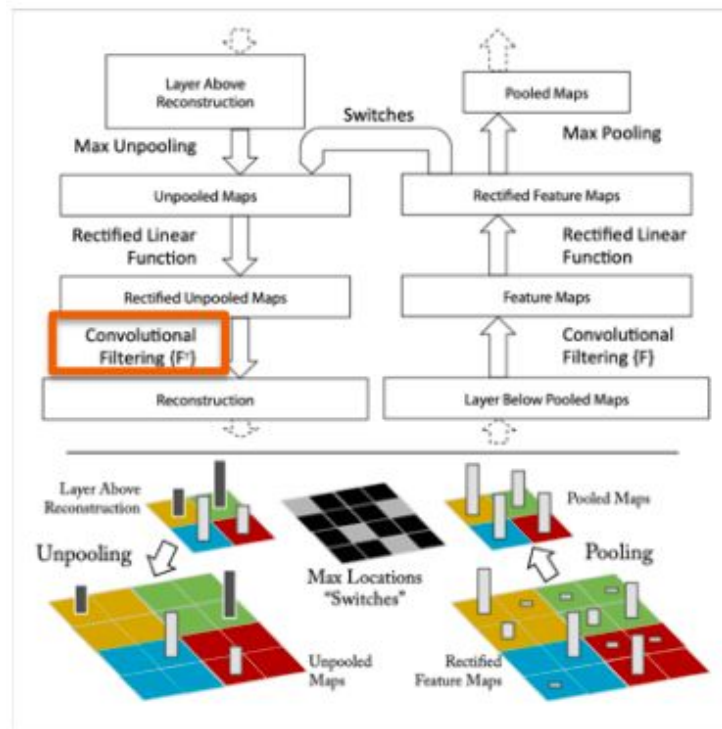
ReLU: нам нужны
положительные
фичи для
визуализации



DeconvNet: визуализация

Deconvolution

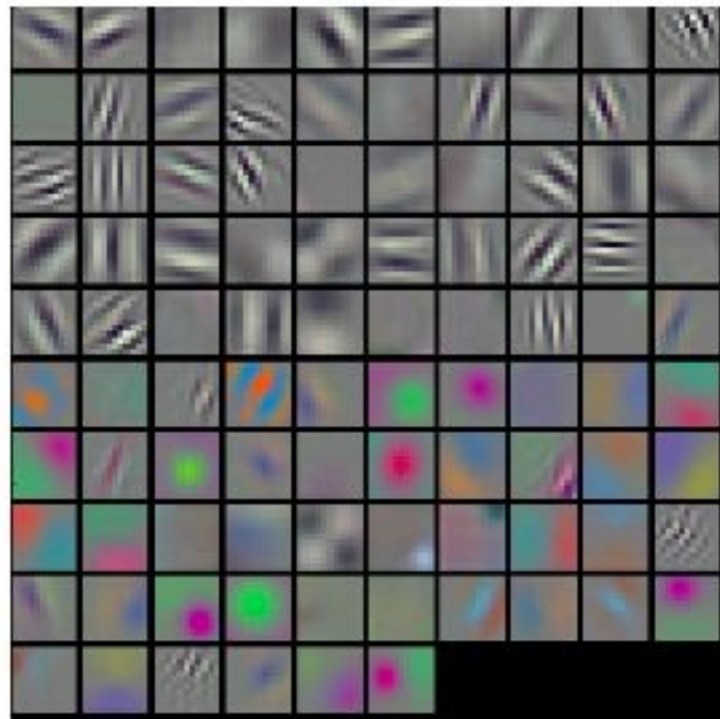
На практике
делаем
операцию
свертку с
transposed
выученного
фильтра



DeconvNet: визуализация



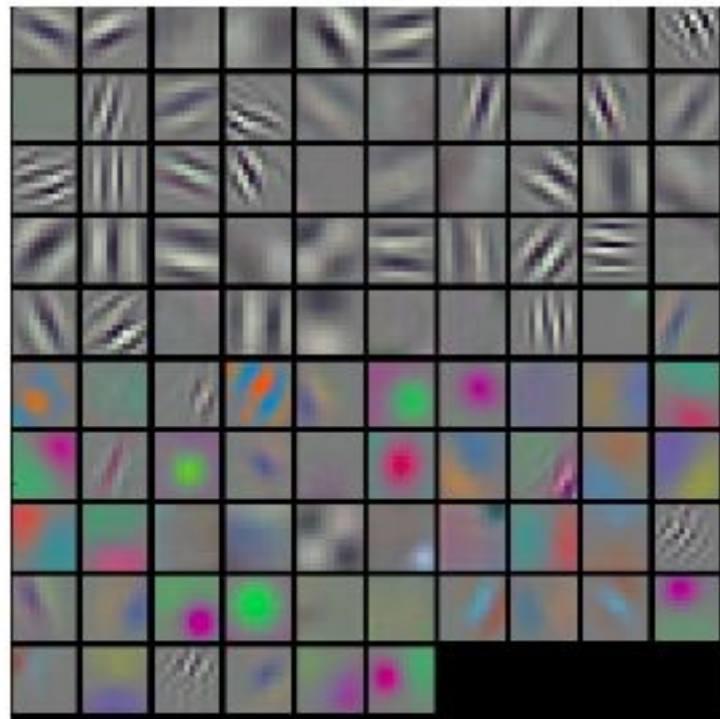
Layer 1



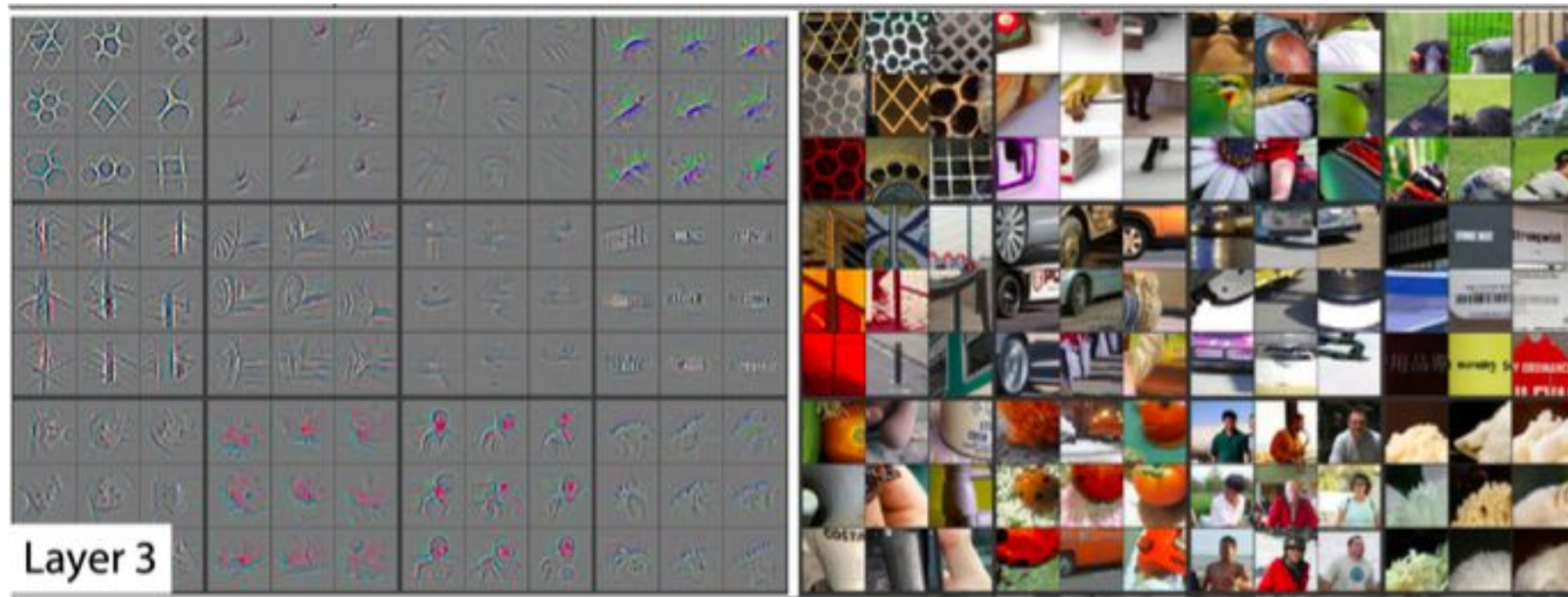
DeconvNet: визуализация



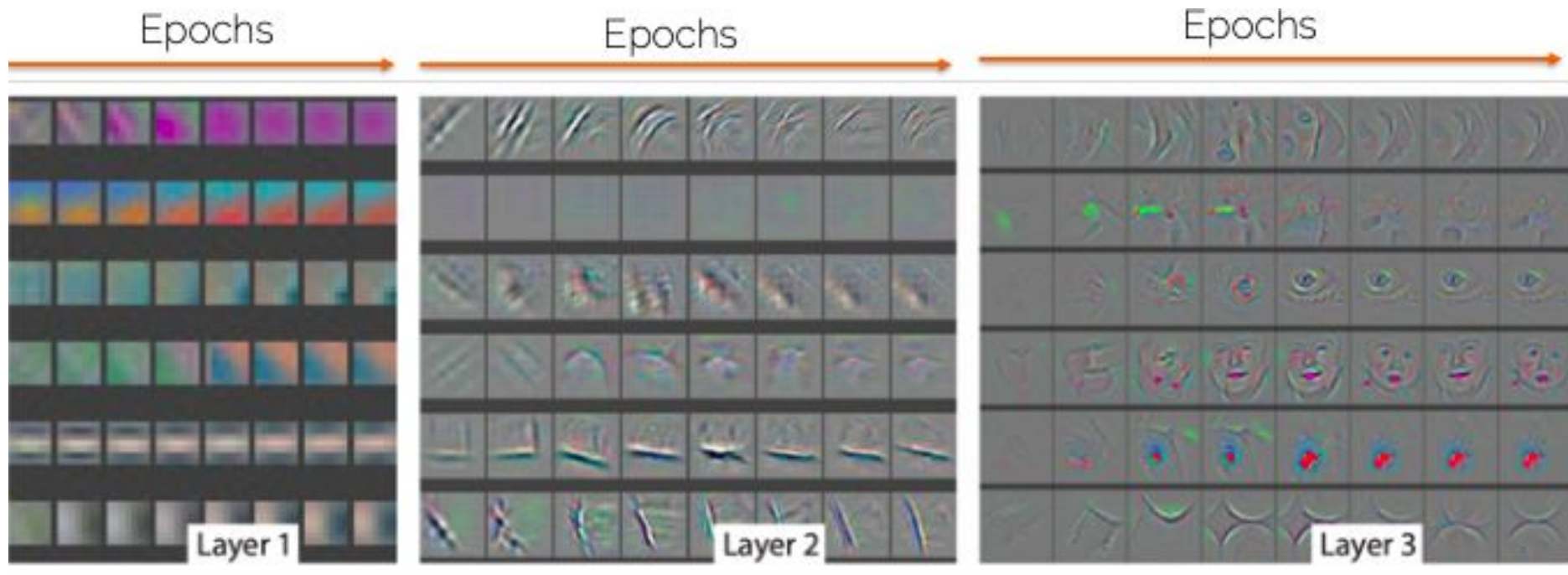
Layer 1



DeconvNet: визуализация



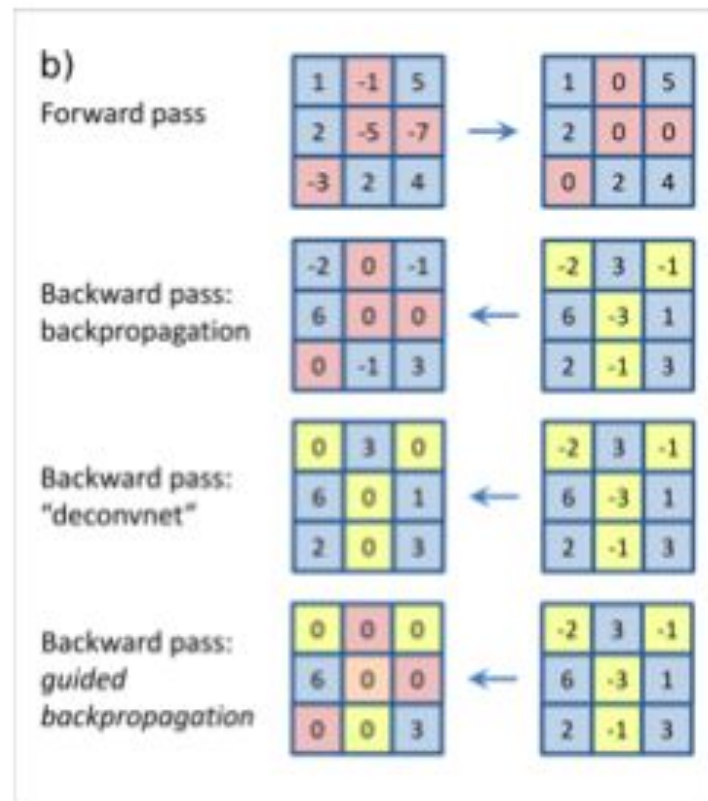
Визуализация: эволюция фичей



Другие способы обратного ReLU

Springenberg et al. "Striving for simplicity: the all convolutional net". ICLR Workshop 2015

<https://arxiv.org/pdf/1412.6806.pdf>



Визуализация спешит на помощь

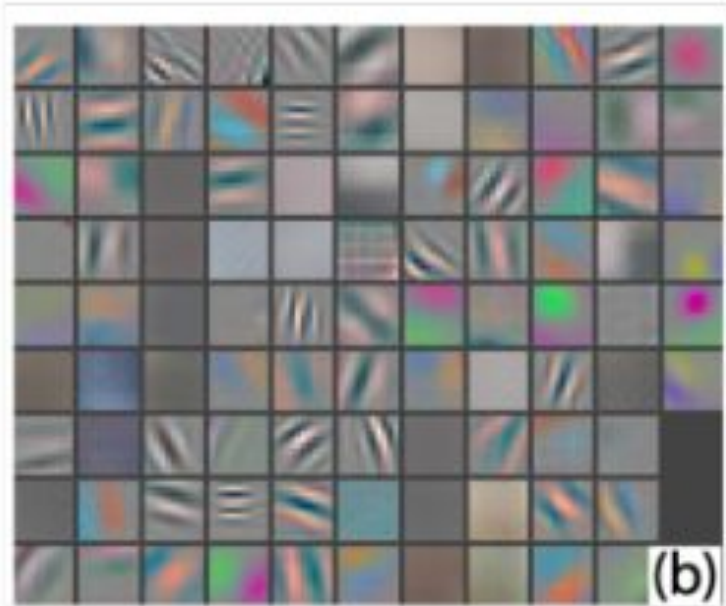
Наблюдения за Alexnet

1. Первый слой - смесь
низкочастотной и высокочастотной
информации, нет среднечастотной

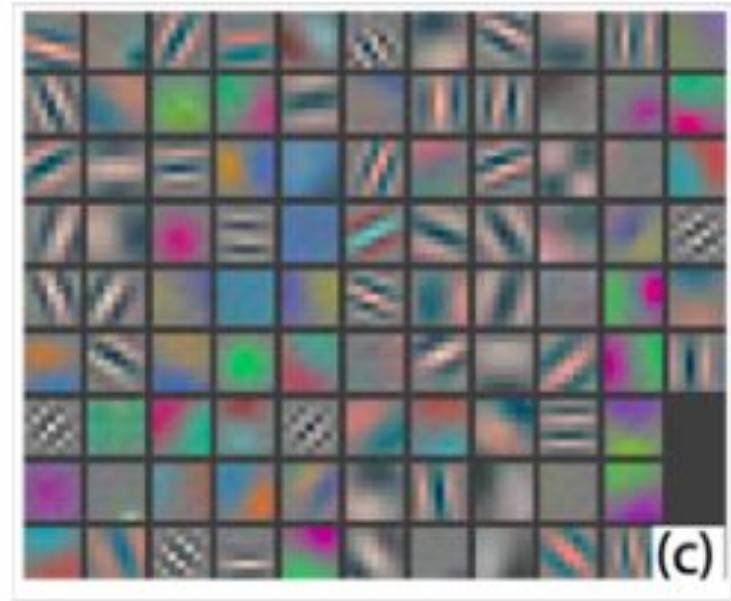
Решение: изменить с 11x11 до 7x7

Другие способы обратного ReLU

11x11



7x7



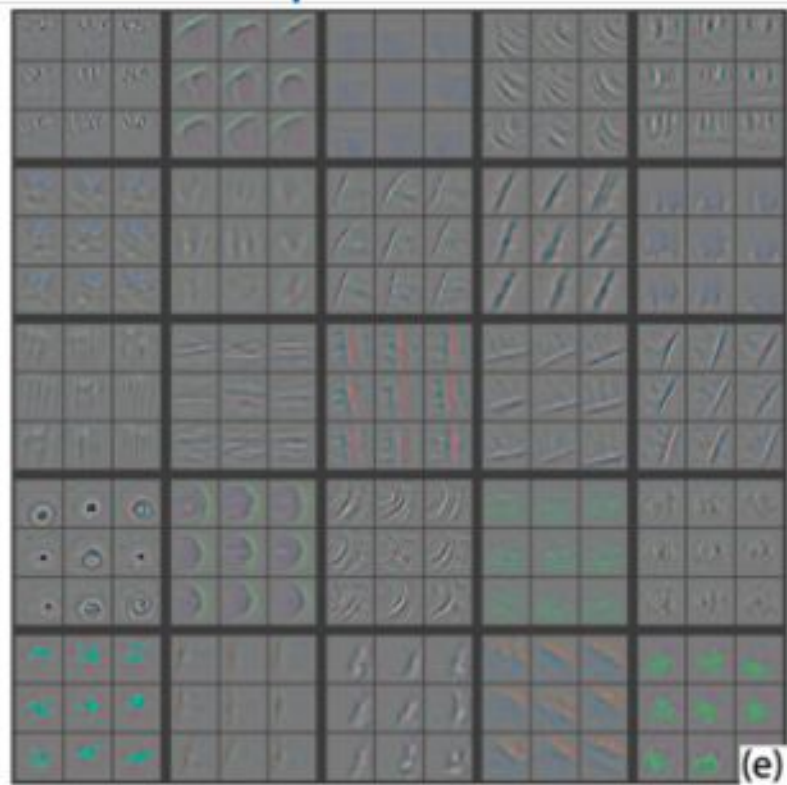
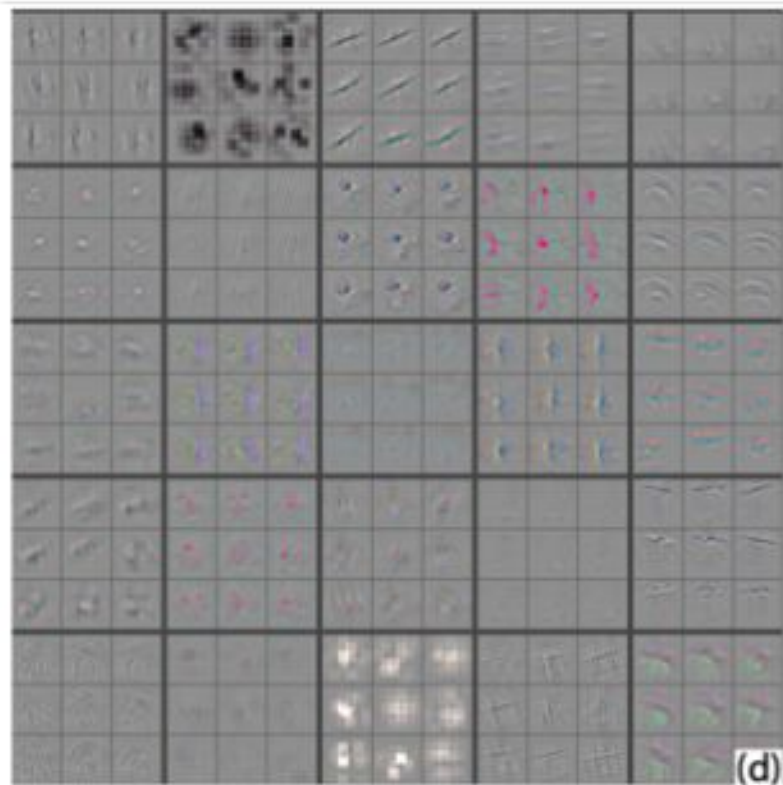
Визуализация спешит на помощь

Наблюдения за Alexnet

2. Плохие артефакты во втором слое, вызванные большим страйдом

Решение: изменить stride 4 до stride 2

Визуализация спешит на помощь



Визуализация спешит на помощь

- Точность увеличилась на 2%
- Активно используйте визуализацию для дебаггинга CNN

Визуализация

1. DeconvNet: нахождение частей изображения на которые больше всего реагирует фильтр
2. Градиентный подъем (gradient ascent): сгенерировать синтетическое изображение, которое максимально активирует фильтр

Визуализация 2

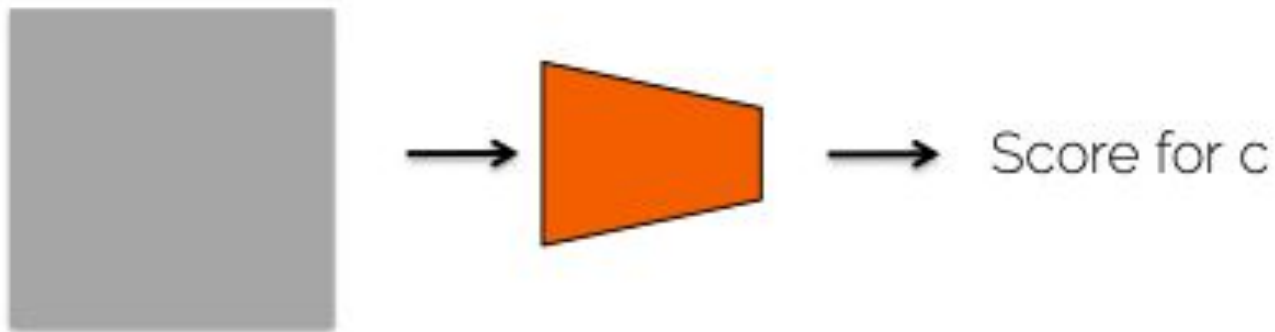
- Хотим найти изображение которое максимальное увеличивает вероятность выбранного класса

$$\arg \max_I S_c(I) + \lambda ||I||_2^2$$

- Скор высчитывается до софтмакса. Аутпут полносвязного слоя
- L2 норма чтобы избежать чересчур больших значений пикселя

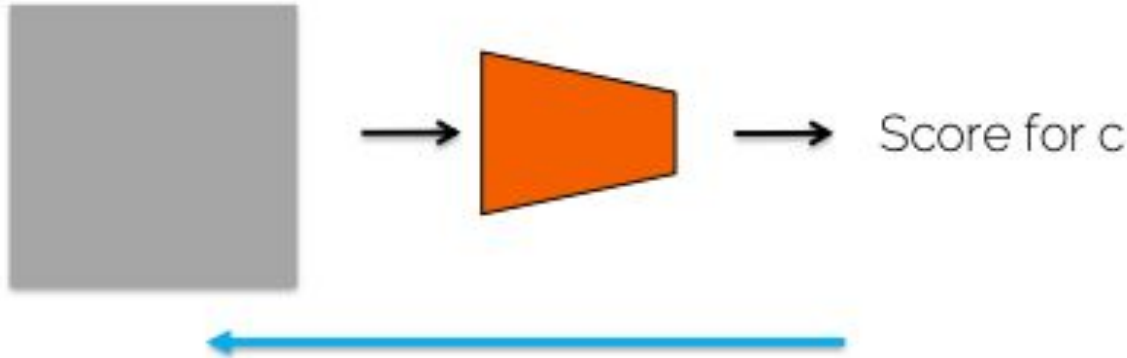
Визуализация 2

1. Берем предобученную CNN (вначале идет нормализация изображения)
2. Форвардпассим нулевое изображения для CNN



Визуализация 2

3. Максимизируем скор -> используем градиентный подъем и бэкпропаем



4. Делаем небольшой апдейт на изображении

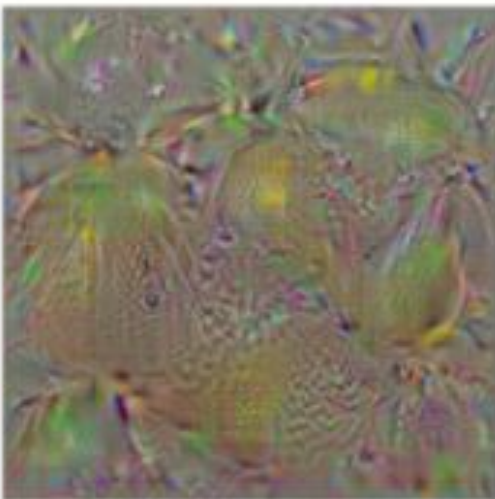
Визуализация 2

5. Повторяем

6. Визуализируем добавлением training mean image



bell pepper



lemon



husky

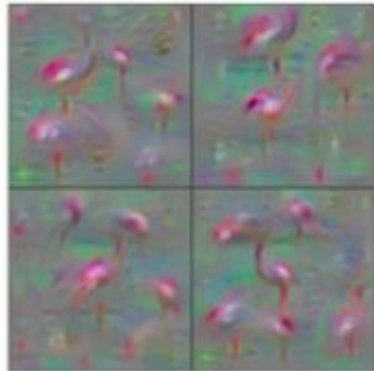
Визуализация 2.1

Улучшаем визуализацию с другой регуляризацией

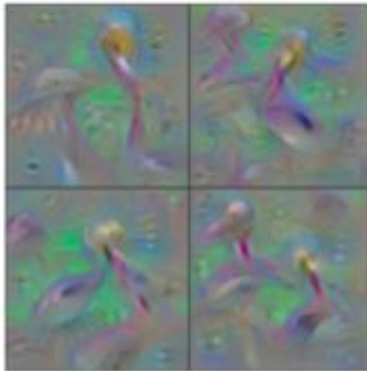
$$\arg \max_I S_c(I) + \lambda ||I||_2^2$$

Другие регуляризации: использование gaussian blur на изображении, обрезание пикселей с маленьким значением к нулю, обрезание градиентов с маленьким значением к нулю

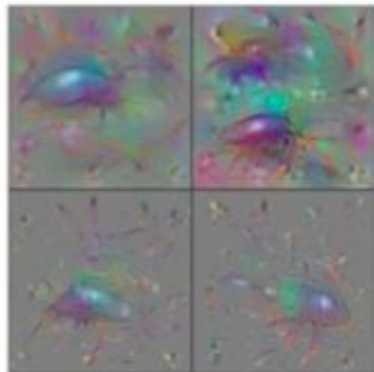
Визуализация 2.1



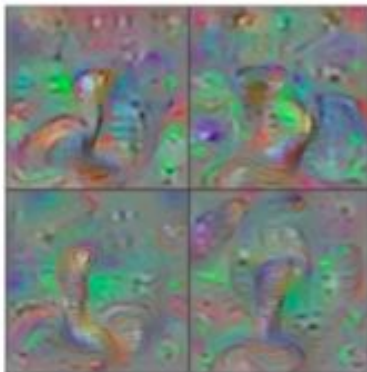
Flamingo



Pelican

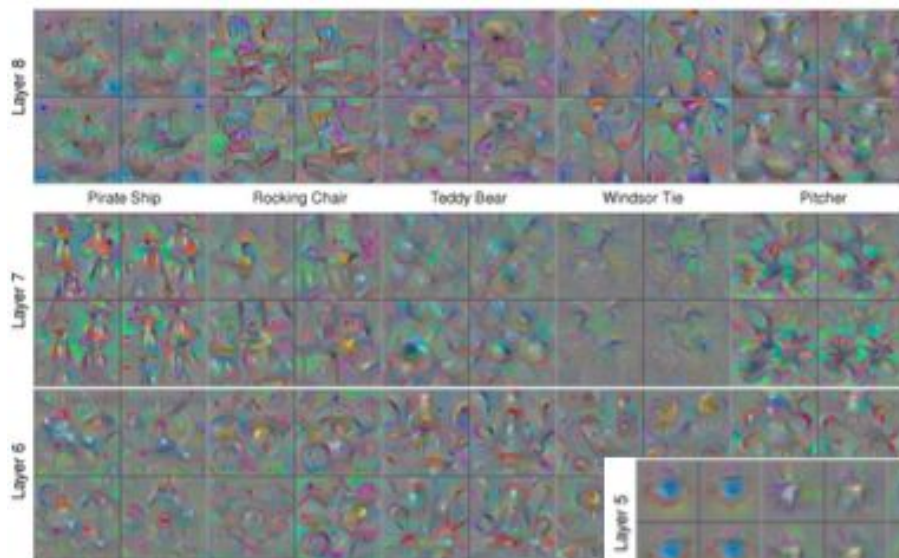


Ground Beetle

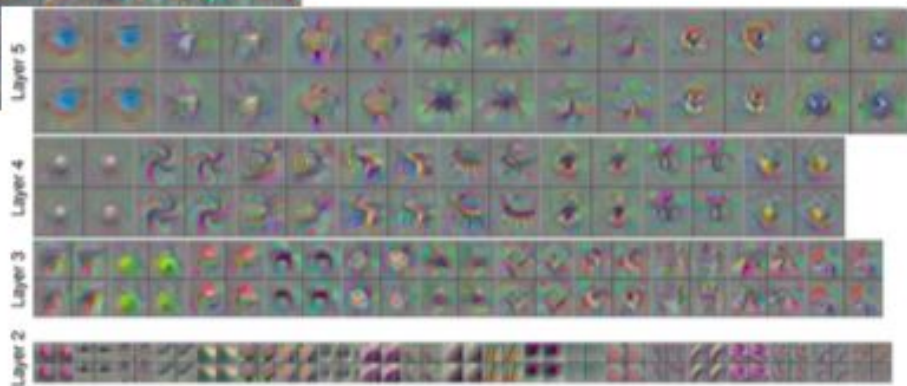


Indian Cobra

Визуализация 2.1



You can also visualize the features
DeepVis [Yosinski et al. 15]
<http://yosinski.com/deepvis>



DeepDream

Делали раньше: синтезировали изображение чтобы максимизировать определенную фиче

Теперь: усиливаем активации фичей в конкретном слое сети

DeepDream

1. Скармливаем изображение сетке
2. Выбираем слои и говорим сетки: усиль все, что смогла найти -> если видишь собак, покажи мне больше собак

DeepDream

1. Форвардпассим изображения до слоя L
2. Приравниваем градиенты слоя к полученным активациям
 - а. Большие активации для фильтра “собаки” сделают большие градиенты
 - б. Изображение будет “показывать больше собак”
3. Бэкпропаем
4. Апдейтим изображение

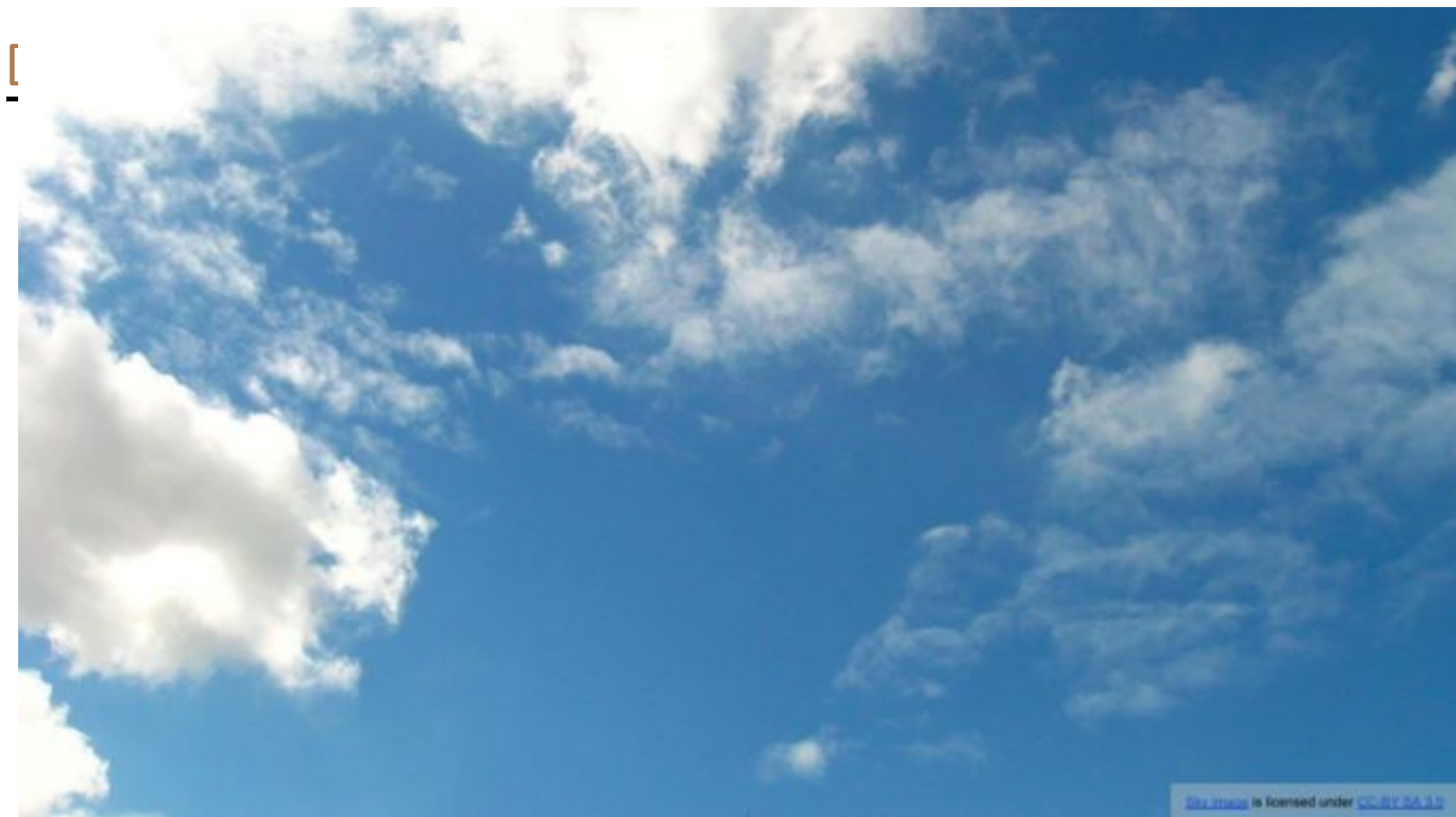
DeepDream

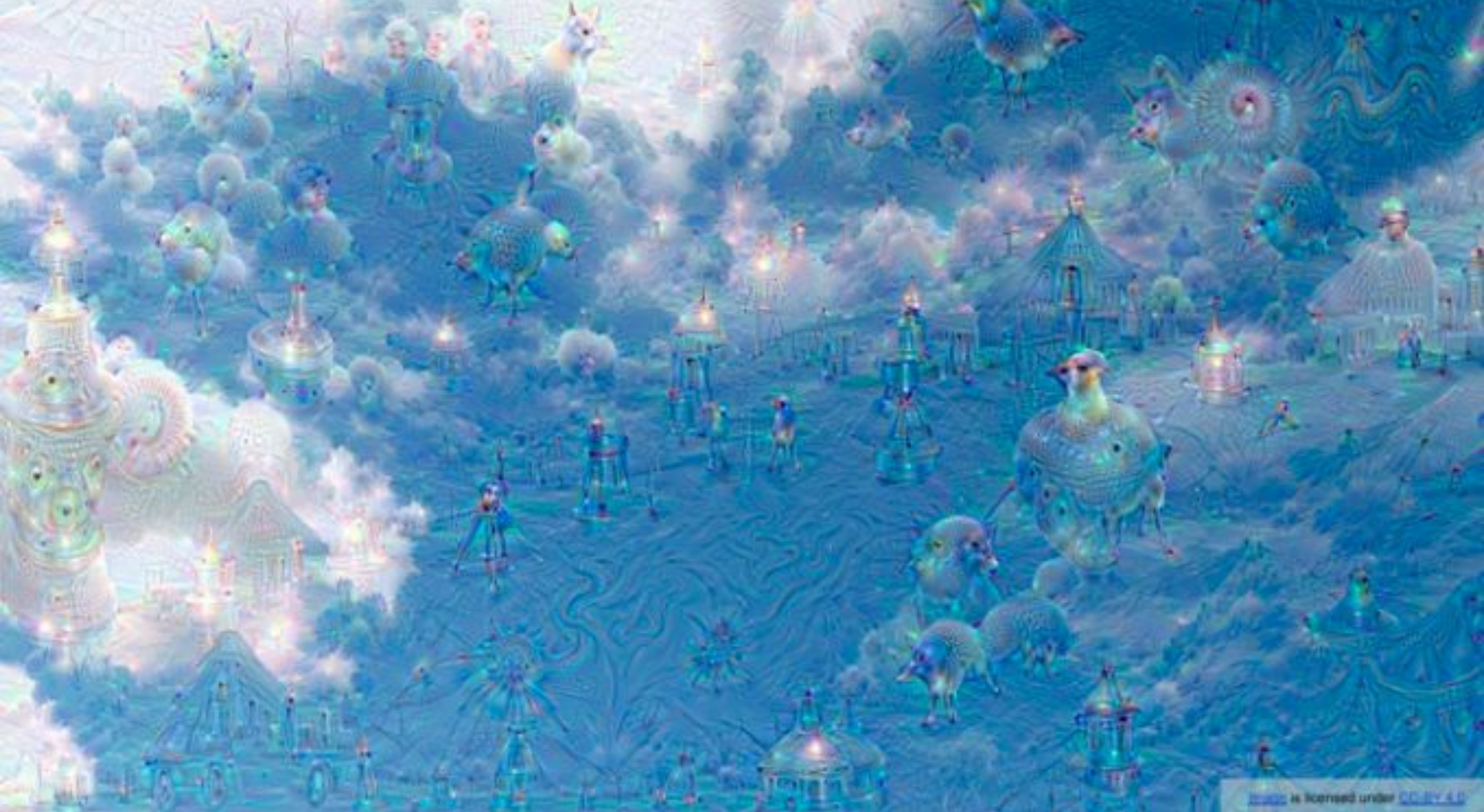
Низкоуровневые признаки



DeepDream

Глубокие слои: мы начинаем видеть целые объекты





De



Prof. Levent Ural and Prof. N. Nesim

IMoradotsev et al. '15 DeepDream

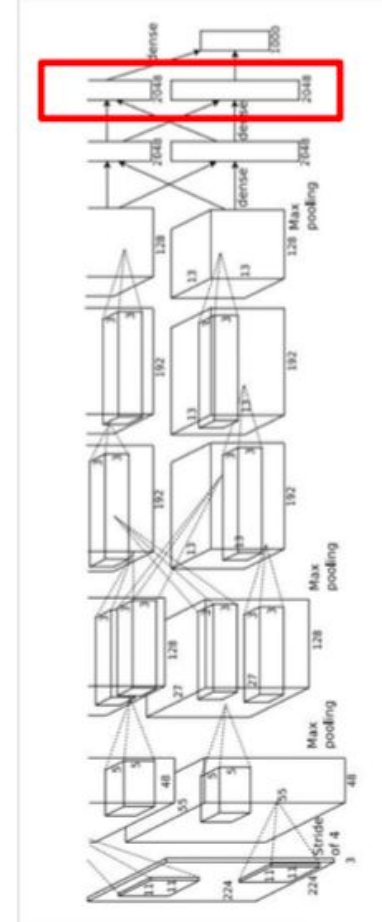


t-SNE

AlexNet

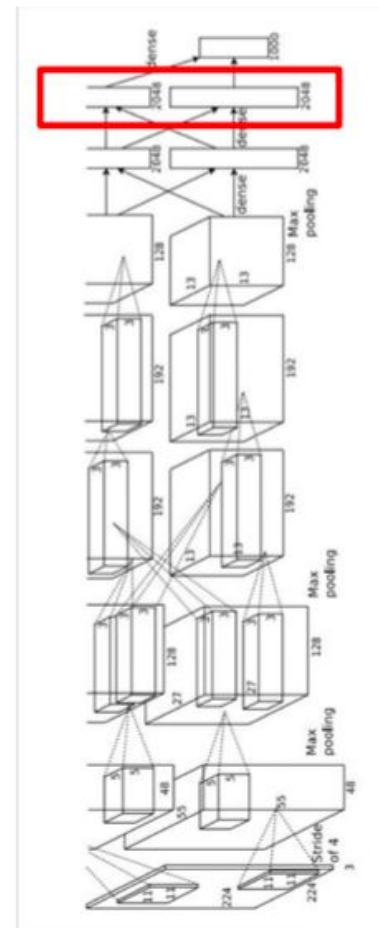
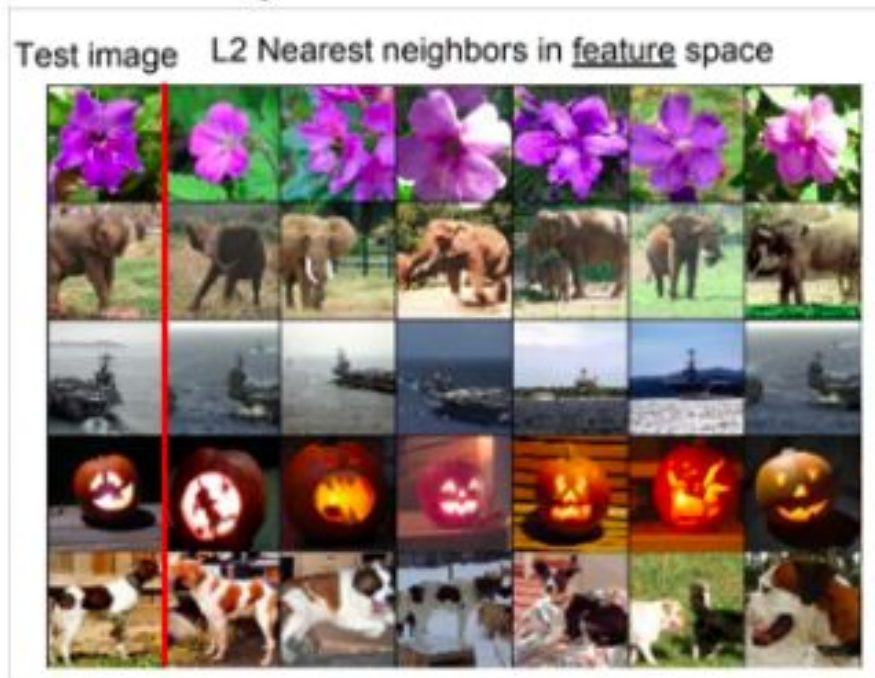
Мы хотим визуализировать
последний полносвязный слой
AlexNet (размерность 4096)

Мы делаем форвард пасс всех
изображений и получаем их
представления



AlexNet

Визуализация Nearest neighbor

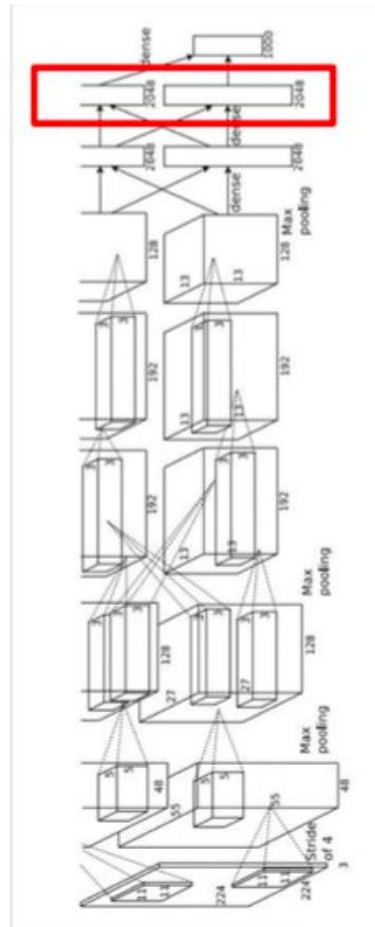


AlexNet

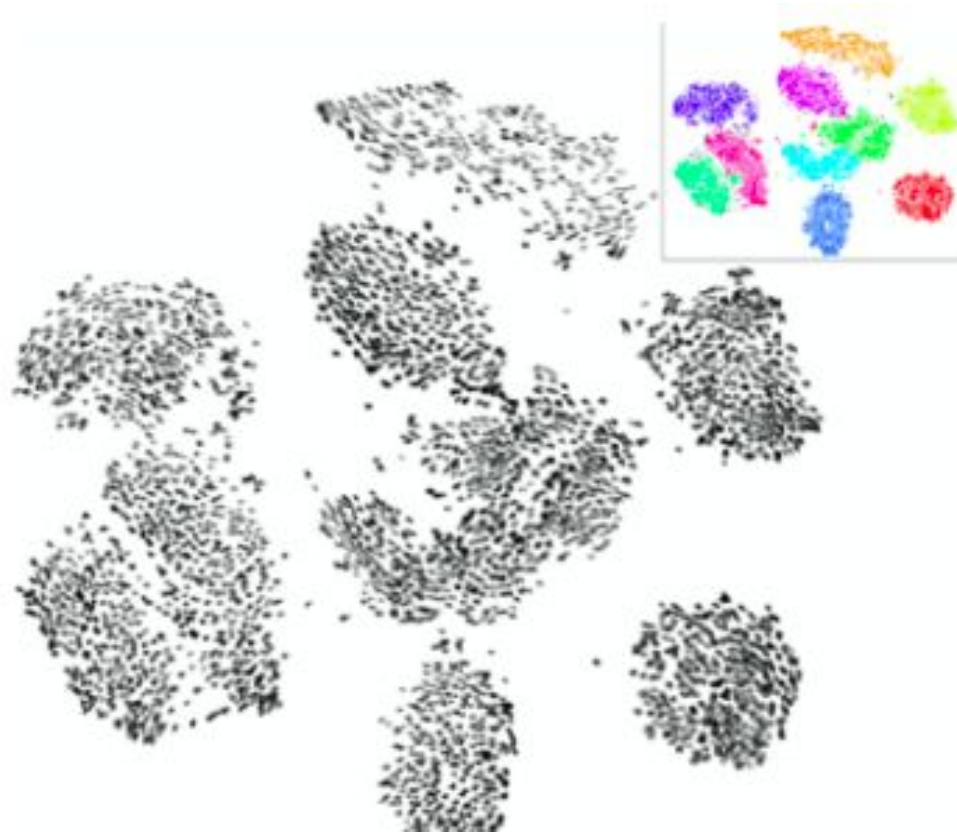
Как визуализировать эти кластеры в пространстве признаков?

Отобразить высокостранственный эмбединг в 2Д так, чтобы сохранить попарное расстояние между точками

Этот мэппинг делается с помощью t-SNE



t-SNE visualization: MNIST



t-SNE visualization: ImageNet



t-SNE visualization: ShapeNet



Когда использовать t-SNE

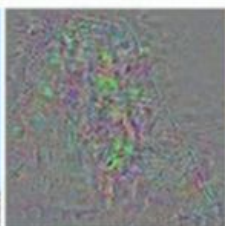
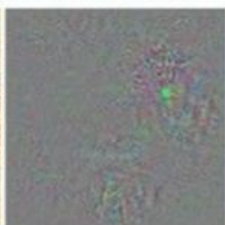
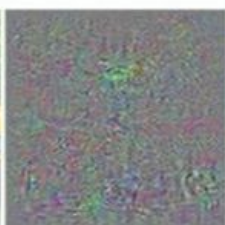
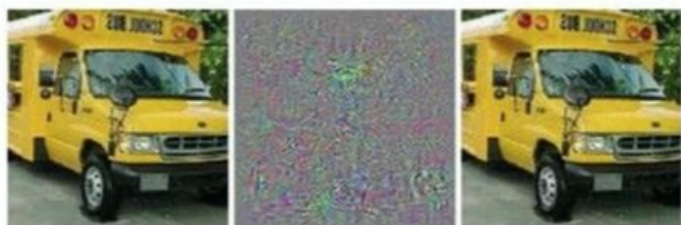
- Используйте для дебаггинга сетки
- Хорошо для визуализации кластеров полученных из Siamese network

Больше визуализаций!

- Saliency visualization: Simonyan et al. „Deep inside convolutional networks: visualizing image classification models and saliency maps“. ICLR Workshop 2014
- [Grad-CAM: Why did you say that? Visual Explanations from Deep Networks via Gradient-based Localization](#)
Ramprasaath R. Selvaraju, Abhishek Das, Ramakrishna Vedantam, Michael Cogswell, Devi Parikh, Dhruv Batra



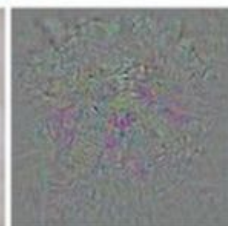
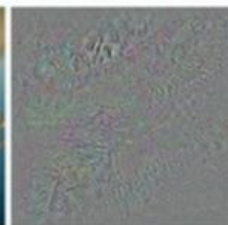
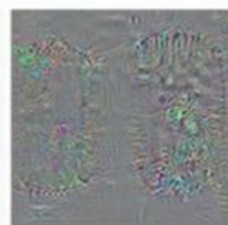
Adversarial attacks



correct

+distort

ostrich



correct

+distort

ostrich

Adversarial construction

$$I_{k+1} = I_k + \lambda \nabla \mathcal{L}(f(I_k), y_{\text{target}})$$

Adversarial construction

$$I_{k+1} = I_k + \lambda \nabla \mathcal{L}(f(I_k), y_{\text{target}})$$

Инициализируем изображение и максимизируем лосс

```
model = torchvision.models.alexnet(constr  
    pretrained = True)  
target = model(input).max(1)[1].view(-1)  
  
cross_entropy = nn.CrossEntropyLoss()  
optimizer = optim.SGD([input], lr = -1e-1)  
  
for in range(15):  
    output = model(input)  
    loss = cross_entropy(output, target)  
    optimizer.zero_grad()  
    loss.backward()  
    optimizer.step()
```




Style Transfer



Style Transfer



content



style



style



style transfer

Image Style Transfer Using Convolutional Neural Networks

Leon A. Gatys

Centre for Integrative Neuroscience, University of Tübingen, Germany
Bernstein Center for Computational Neuroscience, Tübingen, Germany
Graduate School of Neural Information Processing, University of Tübingen, Germany
leon.gatys@bethgelab.org

Alexander S. Ecker

Centre for Integrative Neuroscience, University of Tübingen, Germany
Bernstein Center for Computational Neuroscience, Tübingen, Germany
Max Planck Institute for Biological Cybernetics, Tübingen, Germany
Baylor College of Medicine, Houston, TX, USA

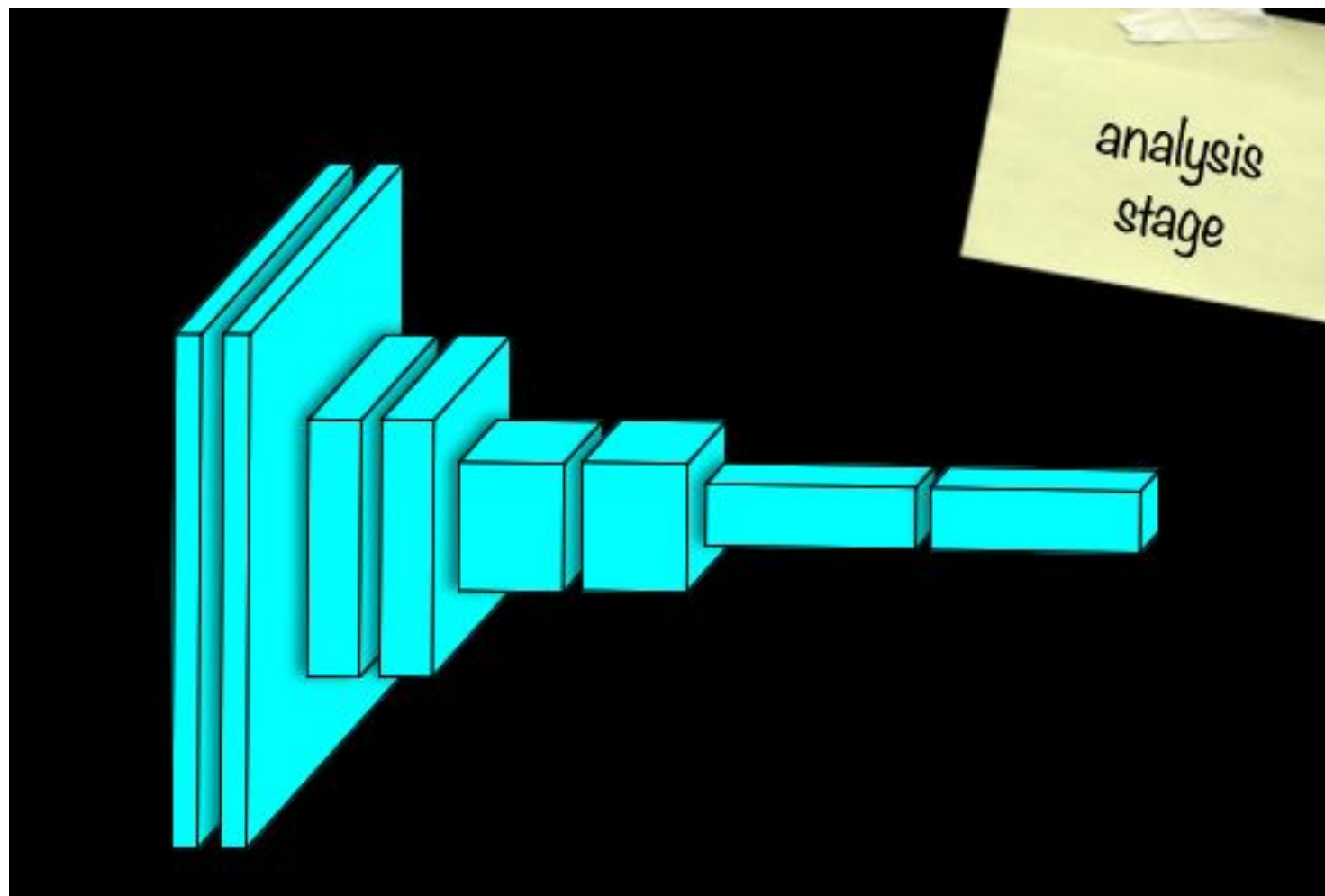
Matthias Bethge

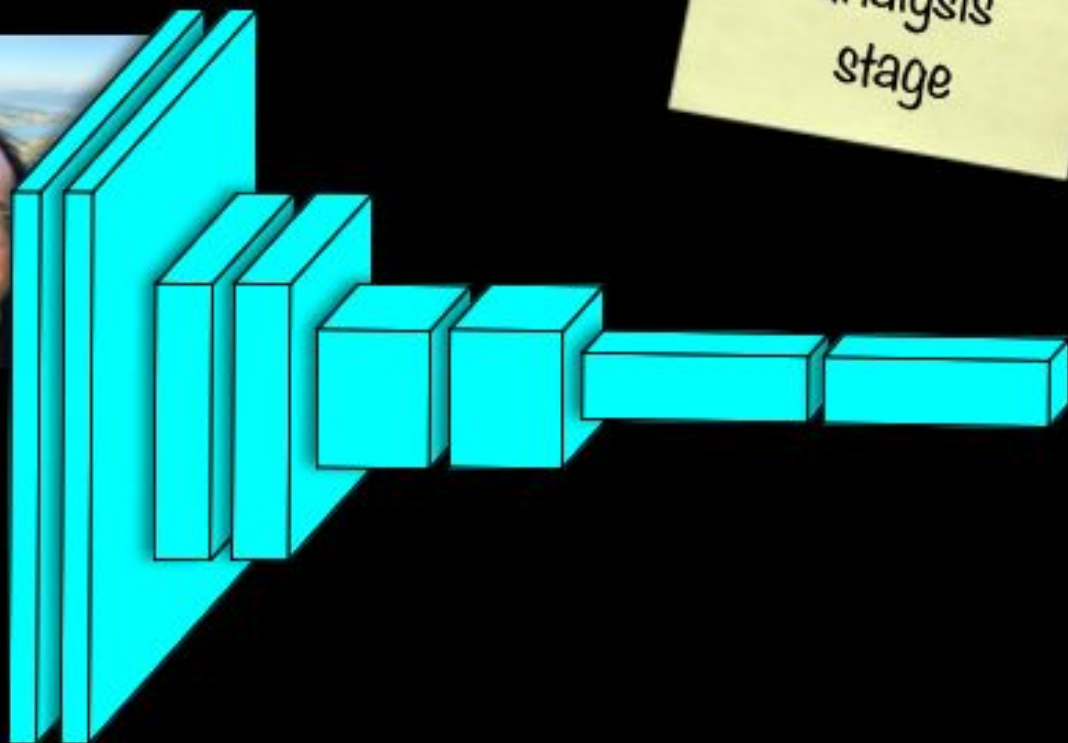
Centre for Integrative Neuroscience, University of Tübingen, Germany
Bernstein Center for Computational Neuroscience, Tübingen, Germany
Max Planck Institute for Biological Cybernetics, Tübingen, Germany

Abstract

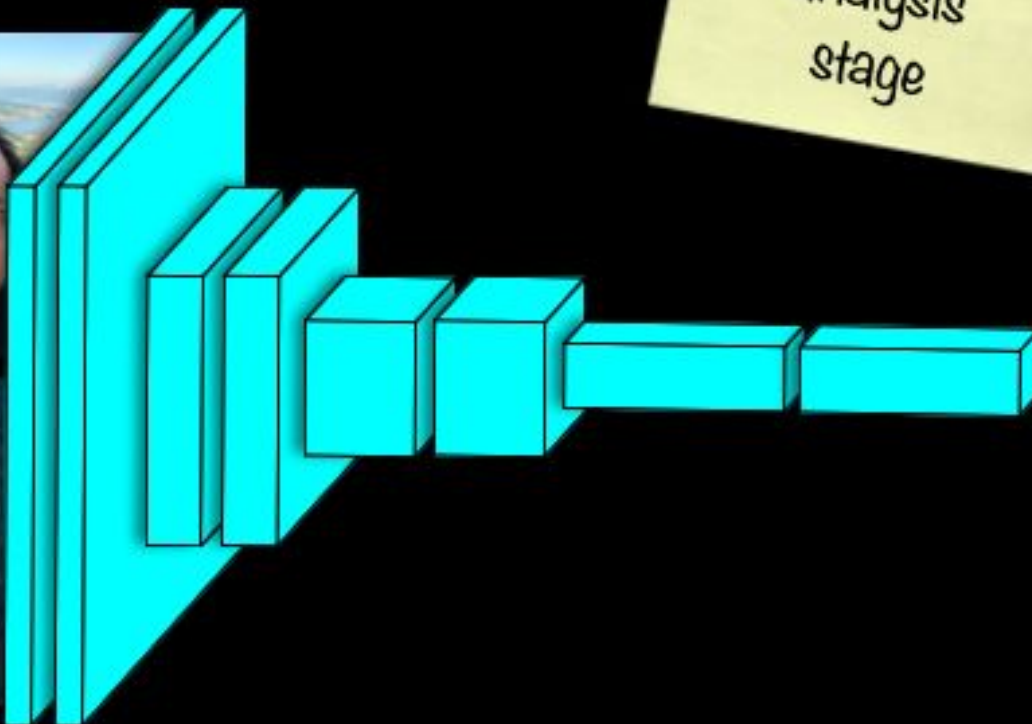
Rendering the semantic content of an image in different styles is a difficult image processing task. Arguably, a major

there exist a large range of powerful non-parametric algorithms that can synthesise photorealistic natural textures by resampling the pixels of a given source texture [7, 30, 8, 30]. Most previous texture transfer algorithms rely on these non-

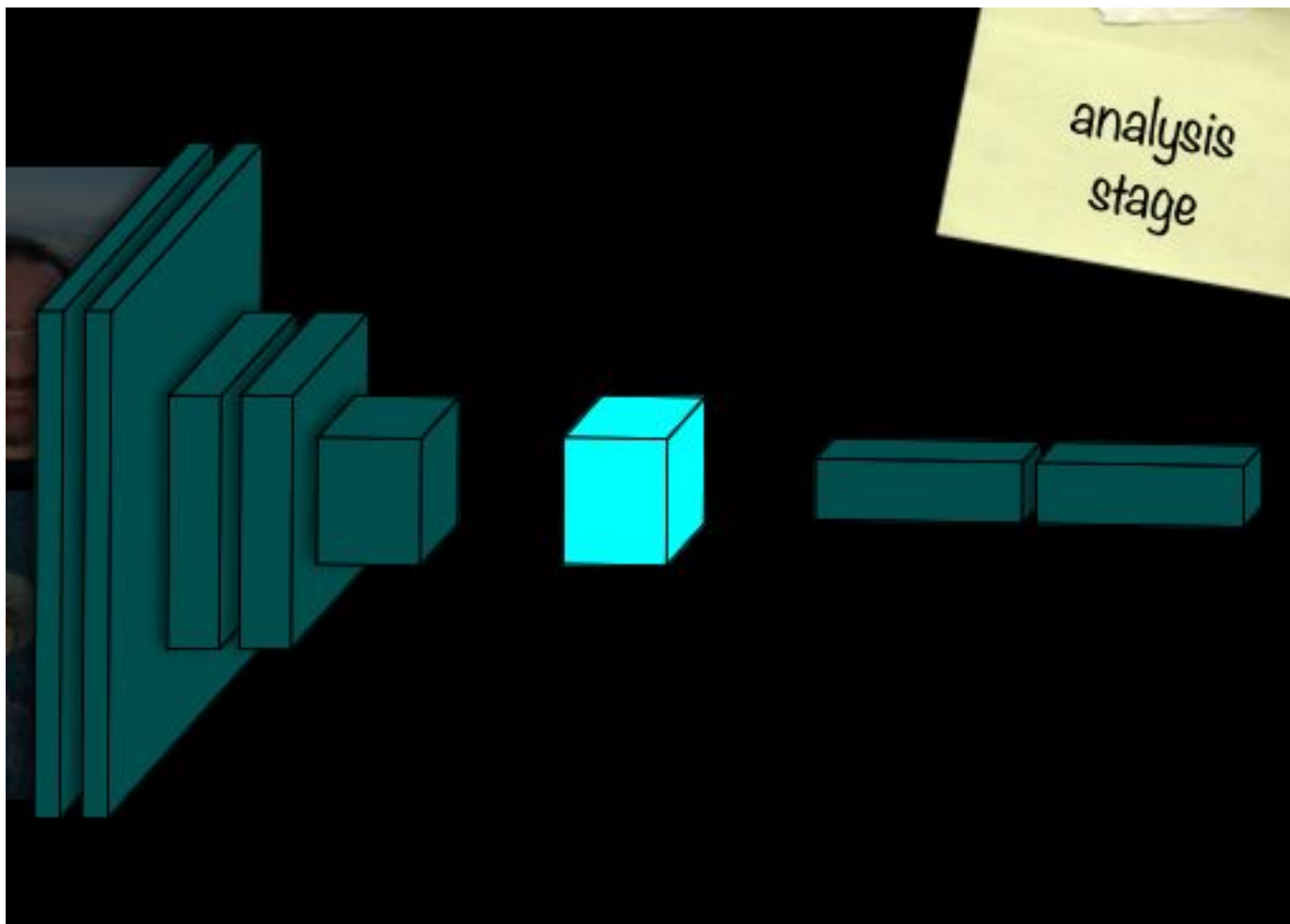




analysis
stage



analysis
stage





content + style

content + **style**

content

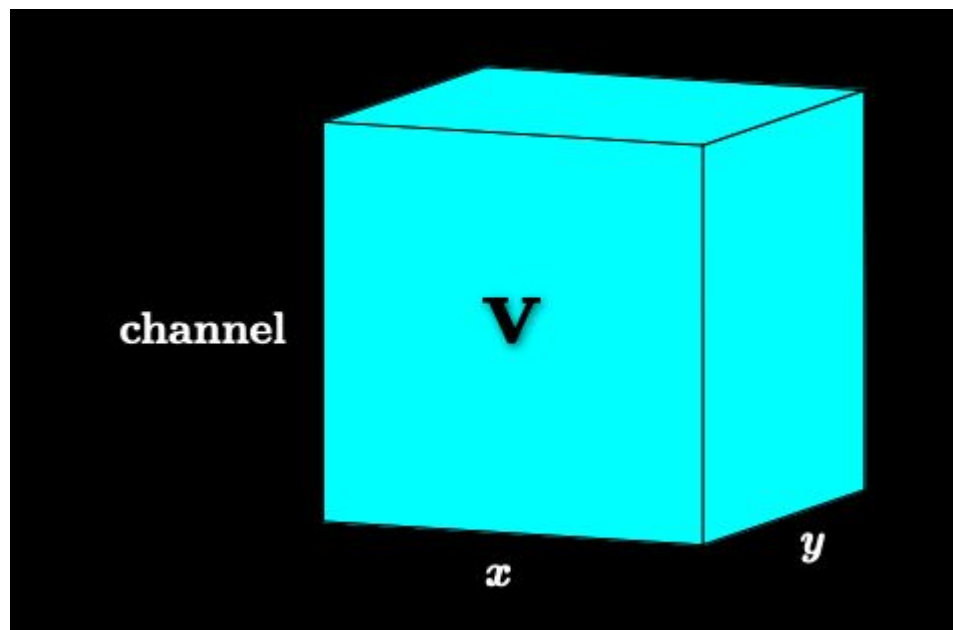
+

style

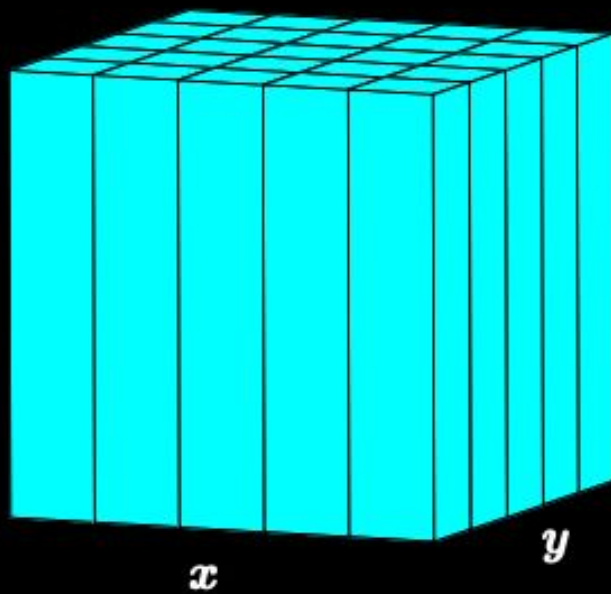
convolutional layer activations from content image

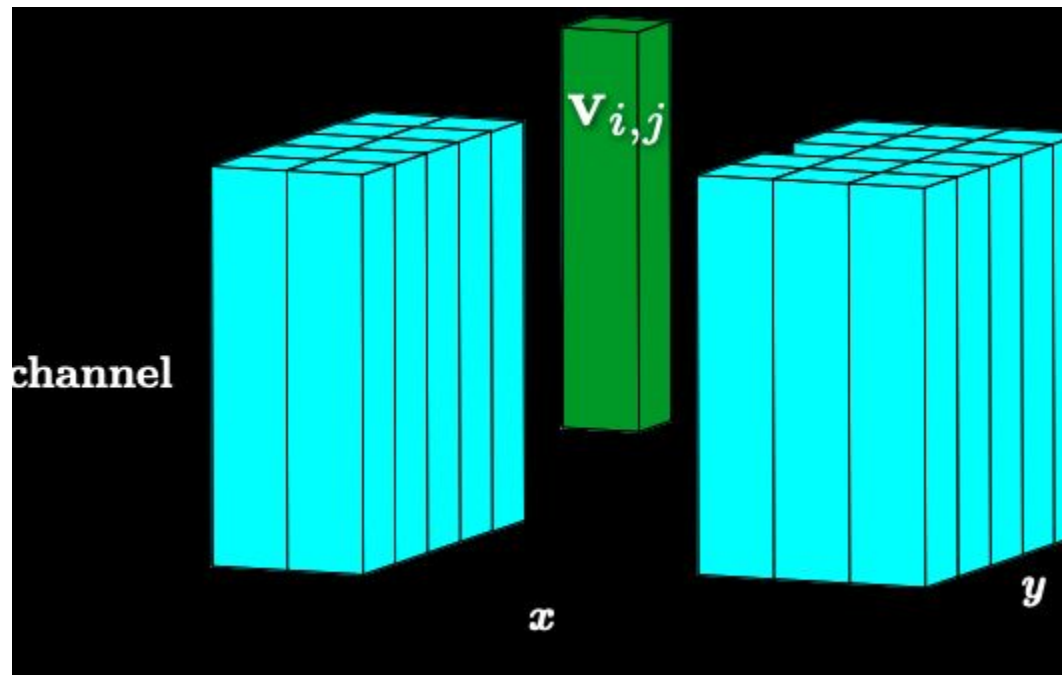
content + **style**

content + style
Gram matrix of layer activations from style image

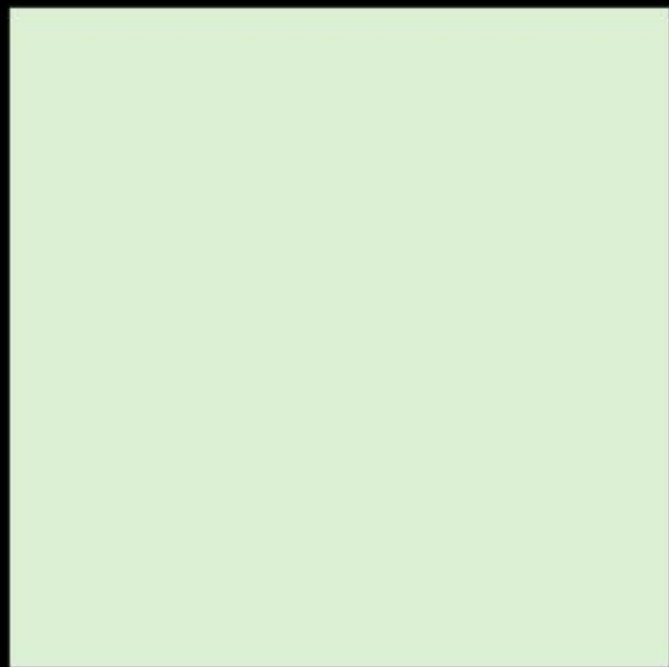


channel






$G =$



channels \times channels


G =

$$\sum_{i=1}^{\text{rows}} \sum_{j=1}^{\text{cols}} \mathbf{v}_{i,j,k} \mathbf{v}_{i,j,k'}$$


$[k, k']$

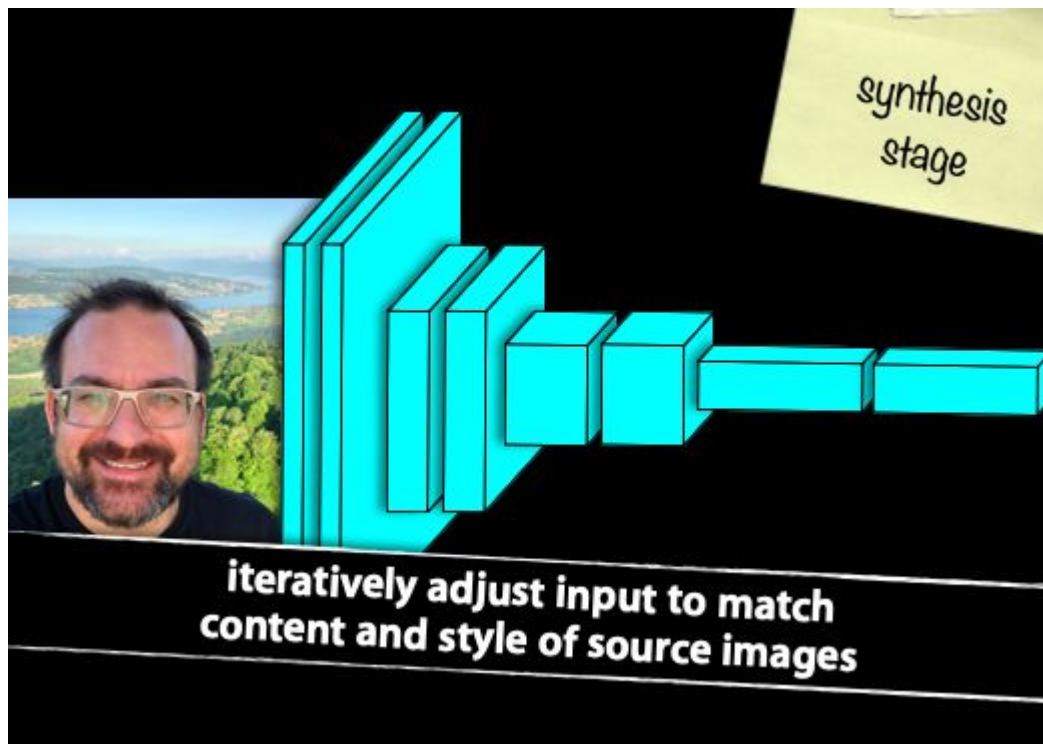
channels \times channels

G =

$$\sum_{i=1}^{\text{rows}} \sum_{j=1}^{\text{cols}} \mathbf{v}_{i,j,k} \mathbf{v}_{i,j,k'}$$


$[k, k']$

channels \times channels



style transfer
loss

$$\begin{aligned}\mathcal{L}(I_{\text{generated}}) = & \\ & \lambda_1 \|\mathbf{A}^{[\ell]}(I_{\text{content}}) - \mathbf{A}^{[\ell]}(I_{\text{generated}})\|_F^2 \\ & + \\ & \lambda_2 \|\mathbf{G}^{[\ell]}(I_{\text{style}}) - \mathbf{G}^{[\ell]}(I_{\text{generated}})\|_F^2\end{aligned}$$

Iterative style transfer **очень** медленный, нужно много
форвард и бэквард пассов

Нужна feed-forward сетка!

Iterative style transfer **очень** медленный, нужно много
форвард и бэквард пассов

Нужна feed-forward сетка!

Johnson et al., Perceptual Losses for Real-Time Style Transfer and Super-Resolution,
<https://arxiv.org/abs/1603.08155>

Ulyanov et al., Texture Networks: Feed-forward synthesis of textures and Stylized images
<https://arxiv.org/abs/1603.08155>



It's coding time!