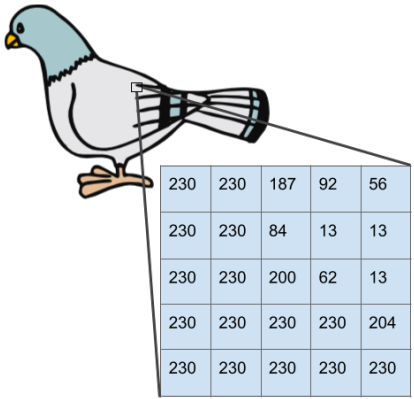
Поиск похожих изображений

Андрей Шадриков

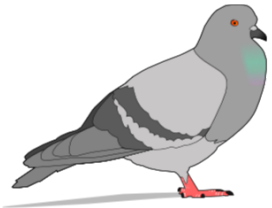
Март 2021

Сравнивать картинки сложно!

- Компьютер не понимает смысла картинок.
- Две «похожие» картинки могут иметь сильные отличия.



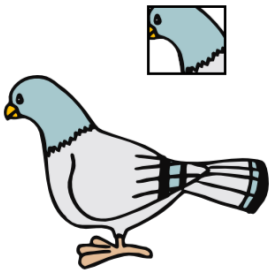
230	230	187	92	56
230	230	84	13	13
230	230	200	62	13
230	230	230	230	204
230	230	230	230	230



Сопоставление шаблона

Можно искать определённый шаблон на картинке скользящим окном:

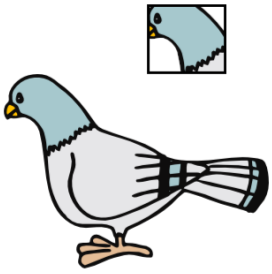
- Через корреляцию шаблона и текущего положения окна.
- Или через квадрат разницы.



https://docs.opencv.org/master/d4/dc6/tutorial_py_template_matching.html

Сопоставление шаблона

- Подправим способ сравнения, учтя общую освещённость.
- Для корреляции это помогло найти правильное местоположение.



https://docs.opencv.org/master/d4/dc6/tutorial_py_template_matching.html

Глобальные статистики

Какая глобальная информация о картинке может быть полезна для её поиска?

- Гистограмма всех цветов (яркости).
- Переход в другое цветовое представление и гистограмма в нём.
- Применить преобразование Фурье, и смотреть статистики частот.
- Выбрать набор текстур, и описывать картинки через наличие в них таких текстур.

А ещё...

Глобальные статистики

Какая глобальная информация о картинке может быть полезна для её поиска?

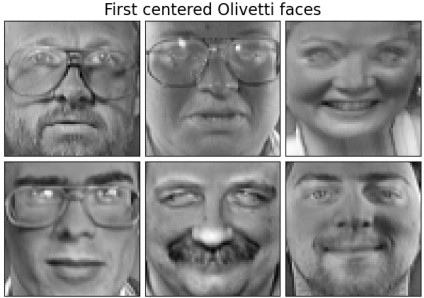
- Гистограмма всех цветов (яркости).
- Переход в другое цветовое представление и гистограмма в нём.
- Применить преобразование Фурье, и смотреть статистики частот.
- Выбрать набор текстур, и описывать картинки через наличие в них таких текстур.

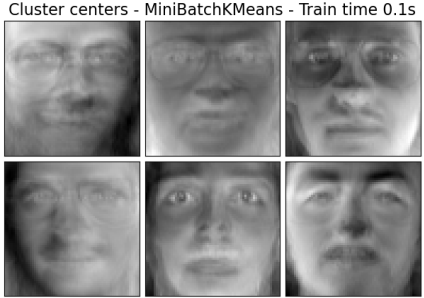
А ещё...

- Представить картинку как взвешенную сумму других картинок.

Разложение на базис

Можно использовать кластеризацию для нахождения «средних» изображений, из которых будут составляться остальные.

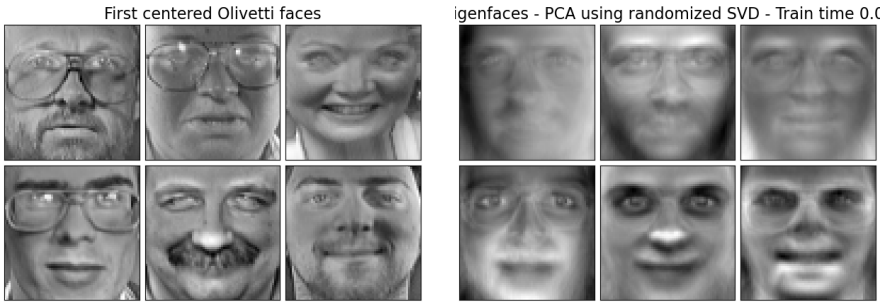




https://scikit-learn.org/stable/auto_examples/decomposition/plot_faces_decomposition.html

Разложение на базис

Можно вспомнить метод главных компонент, чтобы найти нужный базис.



https://scikit-learn.org/stable/auto_examples/decomposition/plot_faces_decomposition.html

Разложение на базис

Или даже добавить ограничение неотрицательности, чтобы полученные матрицы можно было хоть как-то считать картинками.

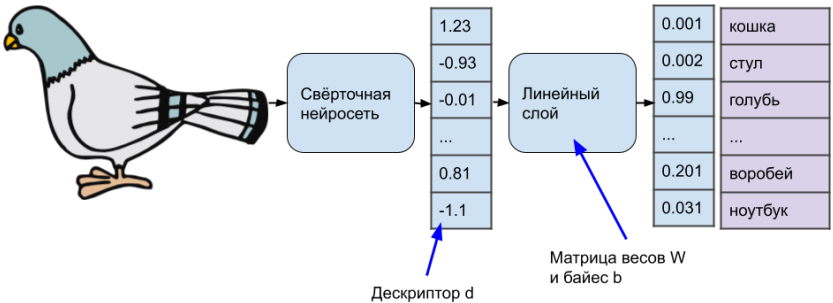
First centered Olivetti faces

Non-negative components - NMF - Train time 0.2s

https://scikit-learn.org/stable/auto_examples/decomposition/plot_faces_decomposition.html

Дескрипторы классификационной CNN

- Свёрточные нейросети умеют правильно классифицировать картинки.
- Внутреннюю информацию можно извлечь и использовать в качестве описания картинки!



Итоговые предсказания класса i: $\sigma_i(W_i^\mathsf{T} d + b_i) = \frac{\exp(W_i^\mathsf{T} d + b_i)}{\sum_j \exp(W_j^\mathsf{T} d + b_j)}$

Вспоминаем задачу классификации

Мы пытаемся минимизировать кроссэнтропию:

$$CE(X, Y) = -\frac{1}{B}\sum_{k=1}^{B} \ln \frac{e^{W_{y_k}^\mathsf{T} d_k + b_{y_k}}}{\sum_{j=1}^{N} e^{W_j^\mathsf{T} d_k + b_j}} \to \min_{W,\, b,\, d}$$

Сконцентрируемся только на одном объекте класса i:

$$-\ln \frac{e^{W_i^\mathsf{T} d + b_i}}{\sum_{j=1}^{N} e^{W_j^\mathsf{T} d + b_j}} = \ln \left(\sum_{j=1}^{N} e^{W_j^\mathsf{T} d + b_j} \right) - \ln e^{W_i^\mathsf{T} d + b_i} = \ln \left(\sum_{j=1}^{N} e^{W_j^\mathsf{T} d + b_j} \right) - \left(W_i^\mathsf{T} d + b_i \right)$$

Мы хотим, чтобы дескриптор d был похож на столбец W_i больше, чем на другие столбцы!

Минимизация угла

Ещё раз про классификацию из дескриптора:

$$\sigma_i(W_i^\mathsf{T} d + b_i) = \frac{\exp(W_i^\mathsf{T} d + b_i)}{\sum_j \exp(W_j^\mathsf{T} d + b_j)}$$

Если мы отнормируем дескиптор d и столбцы W, то скалярное произведение внутри экспоненты можно переписать:

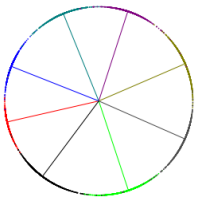
$$\frac{\exp(\|W_i\|_2 \|d\|_2 \cos\theta_i + b_i)}{\sum_j \exp(\|W_j\|_2 \|d\|_2 \cos\theta_j + b_j)} = \{\|d\|_2 = 1,\ \|W_i\|_2 = 1\} = \frac{\exp(\cos\theta_i + b_i)}{\sum_j \exp(\cos\theta_j + b_j)}$$

В этом случае мы уменьшаем непосредственно угол между дескриптором и нужным столбцом матрицы W.

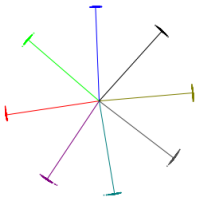
Angular Loss

А что если нам идею с зазором добавить в оптимизируемый нами угол?

$$AngularLoss : \frac{e^{\cos(m_1\theta_i+m_2)+m_3}}{e^{\cos(m_1\theta_i+m_2)+m_3} + \sum_{j\neq i} e^{\cos\theta_j}}$$

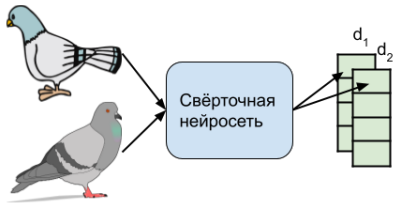


(a) Softmax



(b) ArcFace

Сиамские сети, обучения по парам



$$\min_{d_1,\,d_2}\left(1-\frac{d_1^{\mathsf{T}}d_2}{\|d_1\|_2\|d_2\|_2}\right)$$

$$\max_{d_1,\,d_3}\left(1-\frac{d_1^{\mathsf{T}}d_3}{\|d_1\|_2\|d_3\|_2}\right)$$

Если передавать метку 1 для положительных пар и -1 для отрицательных, функцию ошибки можно записать в общем виде: $y_{ij}\left(1-\frac{d_i^{\mathsf{T}}d_j}{\|d_i\|_2\|d_j\|_2}\right)\to\min_{d_1,\,d_2}$

Триплеты, схемы семплирования

— случайное семплирование

— semi-hard negative sampling

— hard negative sampling, hard positive sampling

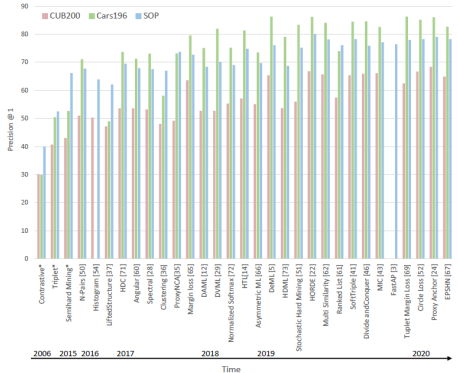
Мы хотим, чтобы расстояние от якорного примера до положительного было меньше, чем до отрицательного.
Hinge loss: $[m + dist(d_A, d_P) - dist(d_A, d_N)]_+$

Ещё обучения метрики

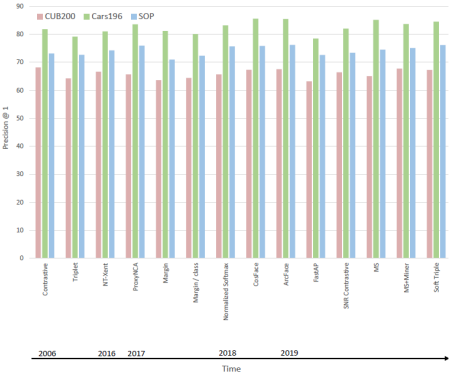
- Можно добавить пару к негативному примеру, получим квадруплеты!
- Можно пользоваться особенностью обучения по батчам, и семплировать примеры только изнутри батча.
- Используя матричные формы функции ошибки можно без семплирования оптимизировать сразу по всем возможным парам из батча.
- Можно формировывать псевдо-классы. И вместо отсемплированных примеров, брать эти псевдоклассы (Proxy-NCA).
- И ещё много идей...

Но в реальности...

Так всё это обучение метрик вообще помогает?



(a) The trend according to papers

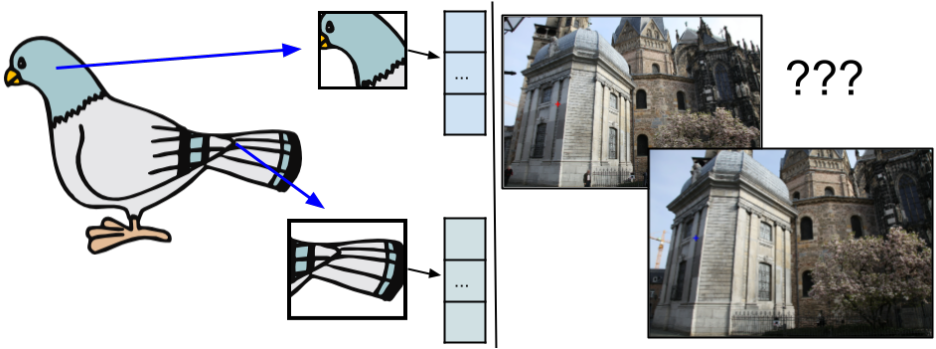


(b) The trend according to reality

Kevin Musgrave, Serge Belongie и Ser-Nam Lim. *A Metric Learning Reality Check*. 2020. arXiv: 2003.08505 [cs.CV]

Локальность признаков

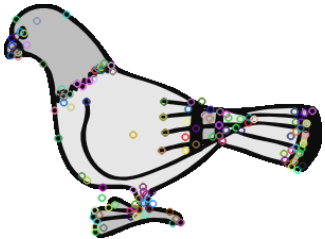
А что если мы векторизовывать будем не всё изображения, а его части?



Как понять, какие части изображения нам интересны?

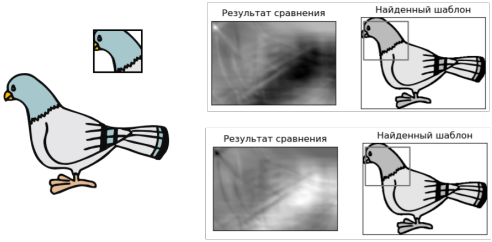
Особые точки

- Опишем набор особых точек: углы, локальные максимумы яркости, границы, и т.д.
- Теперь используя это описание будем искать такие наборы точек на нашем изображении.



SIFT-подобные дескрипторы

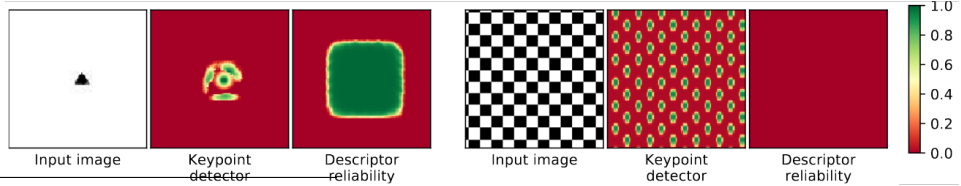
Нам не достаточно находить только координаты.



- Можно использовать «похожесть» шаблона в качестве дескриптора точки.
- Ещё можно считать «уникальность» пикселя по сравнению с соседями.

Нейросети

- Если у нас есть фиксированный набор «особенностей», можно обучить нейросетевой детектор.
- А затем дообучить его, чтобы на похожих точках он выдавал лизкие дескрипторы (SuperPoint[1]).
- У нас могут получиться плохие дескрипторы, давайте дополнительно оценивать «надёжность» (R2D2[2])



[1] Daniel DeTone, Tomasz Malisiewicz и Andrew Rabinovich. “SuperPoint: Self-Supervised Interest Point Detection and Description”. в: *CVPR Deep Learning for Visual SLAM Workshop*. 2018. URL: http://arxiv.org/abs/1712.07629.

[2] Jerome Revaud и др. “R2D2: Repeatable and Reliable Detector and Descriptor”. в: *NeurIPS*. 2019.

Матчинг дескрипторов

- Мы можем сравнивать дескрипторы как обычные вектора.
- Но чтобы сравнивать изображения надо сравнивать наборы дескрипторов друг с другом.
- При сравнении мы можем построить граф расстояний, и использовать его для дополнительной информации.
- Например обучить графовую нейросеть! (SuperGlue[3])

[3] Paul-Edouard Sarlin и др. “SuperGlue: Learning Feature Matching with Graph Neural Networks”. в: *CVPR*. 2020. URL: https://arxiv.org/abs/1911.11763.

Прочие способы поиска

Можно улучшать поиск картинок, добавляя:

- картинки того же объекта в разных положениях (аггрегация дескрипторов),
- мета-информацию об объекте (тэги, размеры),
- текстовое описание картинки или объекта,
- смешивая всё выше с разными весами (ансамблирование).

Что осталось за рамками

- Построение базы векторов.
- Инвертированный индекс (и мульти-индекс).
- Ускорение поиска ближайшего соседа внутри базы.
- Смешанные схемы по ускорению поиска ближайшего соседа.
- Нейросетевые подходы для поиска приближённого ближайшего соседа.

Спасибо за внимание!


@vuvko


@vuvko


vuvko

vuvko@fmap.me


vuvko@fmap.me