академия больших данных

mail.ru group

Задача сегментации

Даниил Лысухин

Команда компьютерного зрения mail.ru

План

- Задача сегментации и ее виды
- Semantic Segmentation
- Instance Segmentation: Mask R-CNN
- BatchNorm и проблемы памяти

Задача сегментации

Локализация объектов

- На прошлых лекциях рассматривали задачу **детектирования объектов** (object detection):
  - Класс объектов
  - Положение объектов (через bounding boxes)
- Частный случай задачи **локализации**

Сегментация

- В задаче **сегментации** положение объектов задается не прямоугольниками, а **попиксельно**

(a) classification (b) detection (c) segmentation

https://research.fb.com/blog/2016/08/learning-to-segment/

Сегментация - примеры

Томографические снимки

https://soundray.org/maper/

Сегментация - примеры

Снимки с микроскопа

https://www.mathworks.com/matlabcentral/fileexchange/67313-cell-segmentation-seg-self-method

Сегментация - примеры

Спутниковые снимки / аэрофото

https://www.researchgate.net/publication/326883028_Scalable_Feature_Extraction_with_Aerial_and_Satellite_Imagery

Сегментация - примеры

Сегментация

- Сегментация позволяет более точно определять форму интересующих объектов
- Есть несколько видов сегментации с разной полнотой извлекаемой информации:
  - Semantic Segmentation
  - Instance Segmentation
  - Panoptic Segmentation

Сегментация - виды

https://arxiv.org/pdf/1801.00868.pdf

Сегментация - виды

1 пиксель = 1 класс

https://arxiv.org/pdf/1801.00868.pdf

Сегментация - виды

[Background]

[Car]

[Person]

[Road]

1 пиксель = 1 класс

[Person, #ID2]

[Person, #ID1]

1 пиксель = 1 класс
+ ID экземпляра

[Car, #ID1]

https://arxiv.org/pdf/1801.00868.pdf

Сегментация - виды

[Background]

[Car]

[Person]

[Road]

1 пиксель = 1 класс

[Person, #ID2]

[Person, #ID1]

1 пиксель = 1 класс
+ ID экземпляра

[Car, #ID1]

[Person, #ID2]

[Person, #ID1]

1 пиксель = 1 класс
(+ ID экземпляра
для “вещей”)

[Car, #ID1]

[Road]

[Sky]

https://arxiv.org/pdf/1801.00868.pdf

Сегментация

- **Семантическая** (**semantic**) - каждому пикселю (кроме фона) ставится в соответствие номер класса
- **Объектная** (**instance**) - разделяются маски разных экземпляров одного класса
- **Panoptic** - для классов типа “небо”, “дорожное полотно” и т.д. (stuff) только класс, для остальных (things) - отдельная маска

Семантическая сегментация

Semantic Segmentation

- Semantic S. = попиксельная классификация
  - На входе в модель - изображение (RGB)
  - На выходе - набор **масок** разных классов

- Рассмотрим бинарный случай - классы “фон” и “объект”

Semantic Segmentation

RGB

Ground-truth маска

Semantic Segmentation

RGB

Ground-truth маска

Semantic Segmentation

RGB

Модель

Предсказанная маска

Ground-truth маска

Semantic Segmentation

RGB

Модель

Предсказанная маска

Ground-truth маска

Функция потерь

Semantic Segmentation

- В качестве функции потерь можно использовать BCELoss
  - $BCELoss(y, \hat{y}) = -y * \log(\hat{y}) - (1 - y) * \log(1 - \hat{y})$

- Считаем в каждом пикселе, усредняем по всей площади

Semantic Segmentation

- В качестве функции потерь можно использовать BCELoss
  - $BCELoss(y, \hat{y}) = -y * \log(\hat{y}) - (1 - y) * \log(1 - \hat{y})$

- Считаем в каждом пикселе, усредняем по всей площади
  - Какие есть проблемы?

Semantic Segmentation

- В качестве функции потерь можно использовать BCELoss
  - $BCELoss(y, \hat{y}) = -y * \log(\hat{y}) - (1 - y) * \log(1 - \hat{y})$

- Считаем в каждом пикселе, усредняем по всей площади
  - Какие есть проблемы?

Semantic Segmentation

- Аналогично задаче детектирования, может быть сильный дисбаланс классов

Semantic Segmentation

- Аналогично задаче детектирования, может быть сильный дисбаланс классов
- Возможные решения:
  - Веса для балансировки вклада разных пикселей
  - Focal Loss
  - Другие функции (см. дальше)

Semantic Segmentation

- Архитектура модели должна быть такой, чтобы на выходе получались карты активаций того же размера, что и изображение на входе

RGB

→ Модель →

Предсказанная маска

FCN (2014)

- Fully convolutional networks for Semantic Segmentation (2014)
- VGG в качестве backbone
- В конце Upsampling для приведения к исходному размеру
- Какие тут проблемы?

FCN (2014)

- Fully convolutional networks for Semantic Segmentation (2014)
- VGG в качестве backbone
- В конце Upsampling для приведения к исходному размеру
- Какие тут проблемы?
  - Узкое бутылочное горлышко
  - Резкое увеличение H/W в конце

SegNet (2015)

- SegNet: A Deep Convolutional Encoder-Decoder Architecture
- Симметричная архитектура вида Encoder-Decoder
- Постепенный Upsampling

U-Net (2015)

- U-Net: CNNs for Biomedical Image Segmentation
- Добавили горизонтальные связи к Encoder-Decoder

U-Net (2015)

- Улучшилась сегментация на границах объектов
- Всего лишь 30 изображений для обучения (512x512)

Fig. 3. HeLa cells on glass recorded with DIC (differential interference contrast) microscopy. (**a**) raw image. (**b**) overlay with ground truth segmentation. Different colors indicate different instances of the HeLa cells. (**c**) generated segmentation mask (white: foreground, black: background). (**d**) map with a pixel-wise loss weight to force the network to learn the border pixels.

Feature Pyramid Networks (2017)

- Feature Pyramid Networks for Object Detection (2017)
- Та самая пирамида признаков, что использовали в RetinaNet
- Идея подойдет и для улучшения сегментации

U-Net + FPN

- Агрегируем выходы разных уровней декодера

U-Net + FPN

- Агрегируем выходы разных уровней декодера

U-Net backbone

- Аналогично Object Detection, в качестве **Encoder** можно брать любую из современных архитектур
  - ResNet
  - EfficientNet
  - ...
- **Decoder** делают “симметричным”, но с повышением размерностей вместо понижения

Further reading

- Семейство DeepLab от Google (DeepLabV3+)
- PSPNet

Альтернативные функции потерь

- Object Detection: использовали величину **IOU** (**intersection over union**) для оценки “совпадения” прямоугольников
- Обобщается и на другие “формы” множеств
  - Оно же - Jaccard index

$$Jaccard(X, Y) = \frac{|X \wedge Y|}{|X \vee Y|}$$

Jaccard index

- **Jaccard index** для сегментационных масок:

$$Jaccard(X, Y) = \frac{|X \wedge Y|}{|X \vee Y|}$$

Jaccard index

- **Jaccard index** для сегментационных масок:

$$Jaccard(X, Y) = \frac{|X \wedge Y|}{|X \vee Y|}$$

Можно дифференцировать?

Jaccard index

- **Jaccard index** не дифференцируется “в лоб”
- Можно аппроксимировать:
- $$J_{approx} = \frac{\sum_{x,y} M_{gt}(x,y) * M_{pred}(x,y)}{\sum_{x,y} M_{gt}(x,y) + M_{pred}(x,y) - M_{gt}(x,y) * M_{pred}(x,y)}$$
- Где
  - M_{gt} - маска ground truth (значения 0 и 1),
  - M_{pred} - предсказанная маска (значения от 0 до 1)

Jaccard Loss

- $$J_{approx} = \frac{\sum_{x,y} M_{gt}(x,y) * M_{pred}(x,y)}{\sum_{x,y} M_{gt}(x,y) + M_{pred}(x,y) - M_{gt}(x,y) * M_{pred}(x,y)}$$
- $JaccardLoss = -\log J_{approx}$

BCE + Jaccard

- $Loss_{total} = L_{BCE} + \alpha L_{Jaccard}$

More Losses

- Также: Dice Loss
- A survey of loss functions for semantic segmentation (2020)

Semantic Segmentation: итоги

- Используется, когда достаточно разбить пиксели по классам
- Модели получают на вход изображения и отдают маски с вероятностями для каждого класса (U-Net)
- Для обучения используют попиксельный BCE/Focal + Jaccard/Dice

Instance Segmentation

Instance Segmentation

- Если объекты не “касаются” и не перекрывают друг друга, то разделить их маски не составит труда и после semantic segmentation
- … Но это бывает редко

Mask R-CNN

- Mask R-CNN (2017)
- В основе - Faster R-CNN
- Дополнительная ветвь для предсказания бинарной маски во всех proposals
  - Маски не зависят от классов!
- Вместо RoIPool — RoIAlign

Recap: RoIPool

- Слой RoIPool в оригинальной Faster R-CNN терял пространственную информацию
- Для классификации это было не важно, но для сегментации - необходимо

Mask R-CNN: RoI Align

- В Mask R-CNN используется операция **RoIAlign**: вместо грубого округления границ и пулинга значений используется интерполяция значений по сетке

Figure 3. **RoIAlign:** The dashed grid represents a feature map, the solid lines an RoI (with 2×2 bins in this example), and the dots the 4 sampling points in each bin. RoIAlign computes the value of each sampling point by bilinear interpolation from the nearby grid points on the feature map. No quantization is performed on any coordinates involved in the RoI, its bins, or the sampling points.

Mask R-CNN

Figure 2. **Mask R-CNN** results on the COCO test set. These results are based on ResNet-101 [19], achieving a *mask* AP of 35.7 and running at 5 fps. Masks are shown in color, and bounding box, category, and confidences are also shown.

Mask R-CNN

- Есть в PyTorch

```
from torchvision.models.detection.mask_rcnn import MaskRCNNPredictor
```

Проблемы с памятью

Проблемы с памятью

- Модели для сегментации (и не только) могут быть прожорливы
  - Много карт активаций (encoder + decoder)
  - Увеличенный размер входа (часто бывает полезно)

Проблемы с памятью

- Модели для сегментации (и не только) могут быть прожорливы
  - Много карт активаций (encoder + decoder)
  - Увеличенный размер входа (часто бывает полезно)

=> **Уменьшается размер батча на 1 GPU**

Проблемы с памятью

- Модели для сегментации (и не только) могут быть прожорливы
  - Много карт активаций (encoder + decoder)
  - Увеличенный размер входа (часто бывает полезно)

=> **Уменьшается размер батча на 1 GPU**

Чем плохо?

Проблемы с памятью

- Малый размер батча = шумные статистики в BatchNorm
- Варианты решения:

Проблемы с памятью

- Малый размер батча = шумные статистики в BatchNorm
- Варианты решения:
  - Замораживать BN (если есть предобученные веса)
  - Использовать альтернативы BN (InstanceNorm, ...)
  - Учиться на нескольких GPU с синхронизацией BN (Synchronized BatchNorm)
  - Оптимизировать вычисления BN

Проблемы с памятью

- Малый размер батча = шумные статистики в BatchNorm
- Варианты решения:
  - Замораживать BN (если есть предобученные веса)
  - Использовать альтернативы BN (InstanceNorm, ...)
  - Учиться на нескольких GPU с синхронизацией BN (Synchronized BatchNorm)
  - **Оптимизировать вычисления BN**

Recap: BatchNorm

- BN: Accelerated Training by Reducing Internal Covariate Shift

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1\ldots m}\}$;
Parameters to be learned: γ, β

Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma \widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Algorithm 1: Batch Normalizing Transform, applied to activation x over a mini-batch.

Recap: BatchNorm

- BN: Accelerated Training by Reducing Internal Covariate Shift

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1\ldots m}\}$;
Parameters to be learned: γ, β

Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_\mathcal{B} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma_\mathcal{B}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu_\mathcal{B})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_\mathcal{B}}{\sqrt{\sigma_\mathcal{B}^2 + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma \widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Algorithm 1: Batch Normalizing Transform, applied to activation x over a mini-batch.

- Для обновления параметров слоя требуется хранить тензор **x** после forward pass в буфере
- При backward pass значение **x** извлекается из буфера
- Есть подходы, избавляющие от этой необходимости

BatchNorm optimisation

- In-Place Activated BN for Memory-Optimized Training of DNNs
- Рассмотрим связку **BN + Activation + Conv**

BatchNorm: vanilla

(a) Standard building block (memory-inefficient)

- Для backward pass необходимо хранить только промежуточные тензоры **x**, **z**

BatchNorm: vanilla

(a) Standard building block (memory-inefficient)

- Для backward pass необходимо хранить только промежуточные тензоры **x**, **z**
  - Для обратимых активаций вроде Sigmoid/Tanh **y** восстанавливается по **z**
  - Для ReLU по **z** нельзя восстановить **y**, но легко восстановить градиент **dz/dy**

BatchNorm: vanilla + checkpoints

(b) Checkpointing [4, 21]

- Если после forward pass буферизовать только **x**, то по сохраненным статистикам **BN** можно вычислить заново **y** и затем **z**

BatchNorm: vanilla + checkpoints

(b) Checkpointing [4, 21]

- Если после forward pass буферизовать только **x**, то по сохраненным статистикам **BN** можно вычислить заново **y** и затем **z**
- Сэкономили память, добавили вычислений
  - Вычисления **нелокальны:** layer_{i} вызывает вычисления на layer_{i-2}

BatchNorm: in-place activated

(d) In-Place Activated Batch Normalization I (proposed method)

Обратима?
Нет

Обратима?
Да

- Будем использовать только **обратимые активации**
- Тогда можно при forward pass хранить только **z**
- Вычисления теперь **локальны**

InPlace-Activated BN

- Заявленная авторами экономия VRAM: до 50%
- Можно комбинировать с SyncBN

Итоги

- Сегментация нужна для чёткого выделения границ объектов
- Разница между Semantic и Instance - в разделении объектов одного класса
- U-Net как хороший старт в Semantic Segmentation
- Для Instance Segmentation — Mask R-CNN
- Для тяжелых моделей могут требоваться хитрые оптимизации