Lab3 Final Report
Semantic segmentation Model 생성 및 성능 분석

정보컴퓨터공학부 202355514 강지원

1 주요 내용

본 Lab에서는 FCN(Fully Convolutional Network)을 사용하여 Semantic Segmentation을 수행하고, 이미지의 각 픽셀을 카테고리로 분류하는 예측 성능을 분석한다.

2 FCN

FCN(Fully Convolutional Network)은 계층적 특징을 생성하는 모델이다. 기존의 이미지 분류 모델과 달리 완전 연결 층(Fully Connected Layer)을 컨볼루션 층(Convolutional Layer)으로 변환하여 입력 이미지의 공간적 정보를 유지하며 예측을 수행한다. 또한 풀링 층(Pooling Layer)을 사용하여 특징을 압축한 후, 그 특징을 업샘플링하여 원본 이미지 크기로 복원하는 작업을 거친다. 이는 Transposed Convolution 등으로 구현된다.

2.1 FCN : 모델 구조

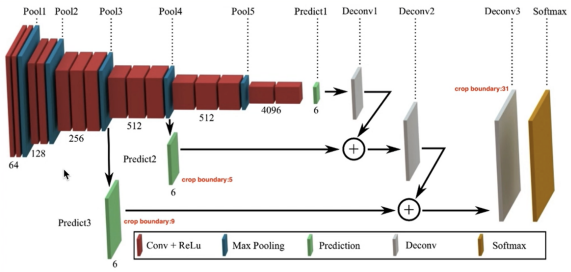

Figure 1: FCN8s 모델 구조

FCN8s 모델은 VGG16을 기반으로 한 Semantic Segmentation 네트워크이다. VGG16의 컨볼루션 레이어를 사용하여 이미지의 특징을 추출한 후, 1x1 컨볼루션을 통해 각 픽셀에 대한 클래스 확률을 예측한다. 그 후 전치 컨볼루션(Deconvolution)을 통해 다운 샘플링된 이미지를 업샘플링하여 원본 이미지 크기로 복원한다. 업샘플링 시, 풀링 층에서는 추출된 특징을 스킵 연결하여 정확한 경계 정보를 제공한다.

3 코드 구현

3.1 Data Loader 및 이미지 전처리

get_data 함수는 학습을 위한 이미지와 레이블 데이터를 전처리하는 함수이다. 이미지 크기 조정 후 각 채널(RGb)에 대해 평균값을 빼 정규화하고, 데이터 증강(Augmentation)을 위해 수평 방향 flipping을 랜덤하게 수행한다. 이후 이미지를 4D 텐서로 변환하여 이미지 전처리를 수행을 완료한다.

3.2 class: ConvolutionalVGGwithUpsample

__init__ 함수에서 VGG16의 pre-trained된 특징 추출기를 사용한 특징 추출과 Fully Convolution Layer인 fc6, fc7, fc8 레이어를 통해 이미지 크기를 유지하며 분류를 수행할 수 있게 한다. 최종 출력 크기를 원본 이미지 크기와 동일하게 만들기 위해 bilinear interpolation을 사용해 upsampling을 수행한다.

forward 함수에서는 입력 이미지를 VGG16을 통해 처리한 후, fc6, fc7, fc8을 통과시켜 특징을 추출하고 fc8(1x1 conv)을 통해 클래스 수만큼의 출력을 얻는다. 마지막에 self.upsample을 사용해 출력 이미지를 원본 해상도로 업샘플링하여 최종 예측 결과를 반환한다.

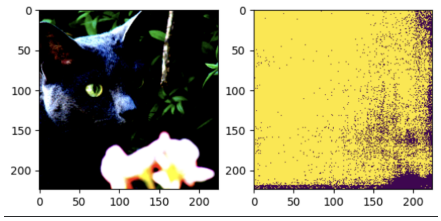

Figure 2: sliding window

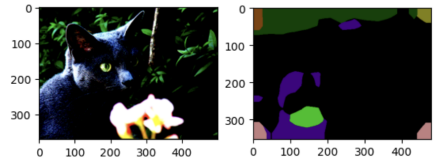

Figure 3: upsampling with bilinear

Segmentation 처리 시에 Sliding-Window 방식은 각 패치를 독립적으로 분류하기 때문에 전체적인 특징 반영이 어려워 경계가 모호하다. 본 lab에서 사용한 업샘플링 방식은 전치 컨볼루션을 통해 더 부드러운 경계를 제공하며, 이미지의 전반적인 문맥을 활용하여 더 정밀한 분할을 가능하게 한다. 다음은 아래 그림에서 확인할 수 있다.

3.3 Loss : Cross Entropy Loss

모델의 출력(out)과 실제 레이블(label)을 사용하여 Cross Entropy Loss를 계산한다. 해당 손실 함수를 통해 예측 값과 실제 값의 차이를 측정하며, 클래스 확률 분포 차이를 최소화하는데 사용한다.

4 모델학습

PASCAL VOC 데이터 셋을 이용해 전처리 및 확장 후, data_gen을 생성해 데이터를 배치단위로 나누고, 이를 모델의 입력으로 전달하였다. 이때, data_gen의 반환 값인 inputs과 label은 Numpy 배열이므로, PyTorch 모델에 입력하기 위해 tensor형식으로 변환하는 과정을 거쳤다. Adam 옵티마이저를 통해 가중치를 업데이트하였고, 학습률 1e-4 으로 20,000번의 반복 학습을 진행하였다.

5 성능 분석

Iteration	Loss
0	0.0967
100	2.1420
200	1.7307
300	1.4266
...	...
20000	0.7777

Table 1: Loss value during training

iteration 100에서 Loss는 2.14로 시작해 최종 iteration 20,000번 수행 후의 Loss는 0.78로 서서히 감소하는 추세를 확인할 수 있다.

$$\text{Mean IoU} = 0.2025969$$

모델의 성능 평가를 위해, Semantic Segmentation 모델이 얼마나 정확하게 픽셀을 분류했는지를 나타내는 Mean IoU 값을 측정해본 결과 0.2026으로

나타났다. 이는 Mean IoU 값의 범위가 0과 1 사이 임을 생각해보면 낮은 수치이다.

6 결론

본 lab에서 mIoU가 0.2로 낮게 나왔다. 이는 모델이 픽셀 단위로 구분해야 할 클래스들의 경계를 정확하게 예측하지 못했음을 의미한다. 성능 향상을 위해 데이터 전처리 과정과 학습 파라미터를 조정하는 것이 필요하다. 실제로 학습률을 5e-4, 1e-5 등으로 변경하고, 이미지 batch size를 조정하여 학습하도록 get_data를 수정했지만, 결과적으로 성능이 더 낮게 나왔다. 이는 모델 성능을 높이기 위해 추가적인 최적화와 실험이 필요함을 시사하며, 향후 더 다양한 하이퍼파라미터 튜닝과 개선된 데이터 증강 기법이 필요할 것으로 보인다.

[1] E. Shelhamer, J. Long, and T. Darrell, "Fully Convolutional Networks for Semantic Segmentation," *FCN Official Code*, . https://github.com/shelhamer/fcn.berkeleyvision.org

[2] Y. Wang, T. Zhang, L. Li, and J. Xu, "How does Sparse Convolution work?" *Upsampling Method*, . https://towardsdatascience.com/how-does-sparse-convolution-work-3257a0a8fd1