TensorFlow in Action (Part 2)

Chapter 8: Telling Things Apart – Image Segmentation

Introduction

Chapter 8 marks a significant shift in computer vision tasks—from image classification (assigning a single label to the entire image) to image segmentation (assigning a label to each pixel in the image). This shift from image-level to pixel-level predictions substantially increases the granularity and complexity of the task. While classification networks output a single probability distribution over classes, segmentation networks must output a spatial map with the same resolution as the input image (or a scaled-down version), where each pixel has a class assignment. This shift opens up possibilities for applications that require detailed spatial understanding—from medical imaging requiring precise tumor boundaries to autonomous vehicles needing to recognize exact road regions.

Semantic Segmentation: Per-Pixel Classification

Semantic segmentation is the most fundamental formulation of the segmentation task. The objective is simple in concept but challenging in execution—for each pixel in the image, predict its class. If the image is a medical scan with a tumor, each pixel is either background, healthy tissue, or tumor. If the image is a scene from an autonomous vehicle, each pixel is either road, sidewalk, car, pedestrian, sky, building, etc.

Semantic segmentation crucially does not distinguish between different instances of the same class. If an image contains five cars, semantic segmentation only shows pixels that belong to the car class—it does not identify five separate car instances. This is a fundamental limitation of semantic segmentation, addressed by instance segmentation approaches discussed later in the chapter.

A fundamental challenge in semantic segmentation is maintaining spatial resolution. Classification networks progressively reduce spatial dimensions through pooling and strided convolutions, culminating in a single classified vector. Segmentation networks require detailed spatial information and therefore cannot aggressively reduce resolution. The solution is encoder-decoder architectures that explicitly restore spatial information.

Encoder-Decoder Architecture Paradigm

The encoder-decoder architecture consists of two complementary components. The encoder works similarly to a classification network—applying a sequence of convolutional layers with pooling to progressively reduce spatial dimensions while extracting and strengthening semantic features. The encoder output is a compressed, highly semantic representation of the input image. Within this reduced dimensionality, features are highly discriminative—the network has learned to preserve the most important information while discarding less important details.

The decoder then takes the compressed representation and progressively upsamples the spatial dimensions, culminating in an output resolution matching the input. Decoders use transpose convolutions (also called deconvolutions) for spatial upsampling. Transpose convolution is conceptually the reverse of regular convolution—it takes a small input map and upsamples it into a larger output map. This process can be thought of as "unpooling" or "deconvolving" information.

The key insight is the use of skip connections between the encoder and decoder. The encoder, in the early stages, processes the full-resolution input and extracts detailed spatial features such as edges and textures. As encoding progresses, the spatial resolution decreases but the semantic information strengthens. Skip connections allow the decoder to access fine-grained spatial details from the encoder, combining them with high-level semantic features. Concatenating encoder features with decoder features at each upsampling stage enriches the decoder input with spatial information, significantly improving segmentation quality.

U-Net: Elegant Architecture for Medical Image Segmentation

U-Net is a seminal architecture for image segmentation, particularly powerful for biomedical image segmentation. Named for its characteristic U-shaped architecture (an encoder path that gradually downsamples, followed by a decoder path that upsamples), U-Net uses a symmetric encoder-decoder structure with extensive skip connections.

The U-Net architecture begins with a contraction path (encoder) consisting of the repeated application of two 3x3 convolutions (each followed by ReLU), followed by 2x2 max pooling. A typical contraction path consists of four downsampling steps. After reaching the bottleneck (lowest resolution, highest feature depth), the expansion path (decoder) begins. The expansion path repeatedly applies a 2x2 transpose convolution for upsampling, followed by two 3x3 convolutions. At each expansion step, features from the corresponding contraction level are concatenized with the upsampled features, providing rich spatial context.

The advantages of U-Net are manifold. First, by combining skip connections from the encoder with learned features in the decoder, the network can effectively utilize multi-scale information. First, U-Net uses parameter sharing extensively and works well with relatively small training datasets—critical for medical imaging, where annotated data is often scarce. Third, its simple and elegant architecture is easily modified to meet specific requirements. Fourth, U-Net achieves state-of-the-art results on various medical imaging benchmarks.

Variations of U-Net include recursive U-Nets (applying U-Net recursively), wide U-Nets (increasing the number of channels), and deep U-Nets (increasing the number of layers). Hyperparameter choices such as depth and feature channel counts can be adjusted to balance model capacity, computational requirements, and dataset size.

Instance Segmentation: Telling Apart Individuals

Semantic segmentation recognizes the class of each pixel but does not distinguish between instances. Instance segmentation extends semantic segmentation by identifying and separating individual object instances of the same class. If an image contains three dogs, instance segmentation identifies not only which pixels are dogs (semantic), but also which pixels belong to the first dog, which to the second, and which to the third.

Instance segmentation is significantly more challenging than semantic segmentation—the network must simultaneously solve classification, localization (bounding box detection), and pixel-level segmentation. The dominant approach is Mask R-CNN, an elegant extension of the Faster R-CNN object detection framework.

Mask R-CNN works in several stages. First, a feature extraction backbone (typically ResNet) processes the input image, producing feature maps. Second, a Region Proposal Network (RPN) generates candidate bounding boxes (region proposals) that are likely to contain objects. Third, for each region proposal, classification (determining the class of the object within the region), bounding box regression (refining box coordinates), and mask prediction (segmentation mask for pixels belonging to that object) are performed.

The mask branch in Mask R-CNN consists of a small FCN (fully convolutional network) that predicts a binary mask for each region proposal. Masks are predicted at low resolution and upsampled to the original resolution. The loss function combines classification loss, bounding box regression loss, and mask prediction loss. This multi-task learning forces the network to learn representations that are useful for all three tasks.

Loss Functions for Segmentation Tasks

Segmentation networks require appropriate loss functions for pixel-level predictions. The cross-entropy loss is standard—for each pixel, the standard cross-entropy loss between predicted class probabilities and the ground truth class is computed. For multi-class problems, it is summed or averaged across all pixels. Cross-entropy loss works well but can be heavily weighted toward pixel-rich classes—if 95% of pixels are background, the network can achieve 95% accuracy by simply predicting background everywhere.

To address class imbalance, weighted cross-entropy loss assigns higher weights to pixels from minority classes. The weighted loss encourages the network to pay attention to rarer classes. The choice of weights can be made based on inverse class frequencies or manual specification.

Dice loss is a popular alternative in medical imaging, inspired by the Dice coefficient similarity metric. Dice loss directly optimizes the overlap between predicted and ground truth masks, making it particularly suitable for segmentation. The Dice score ranges between 0 (no overlap) and 1 (perfect overlap). Dice loss is defined as 1 - Dice, subtracted for optimization.

Intersection over Union (IoU) loss, also called Jaccard loss, is similar to Dice loss but with a different mathematical formulation. IoU measures the ratio of the intersection area between the prediction and ground truth to the union area. IoU is a stricter measure than Dice—it rewards high overlap but penalizes misses more severely.

A combination of multiple loss functions (multi-loss objective) is sometimes used, with a weighted combination of cross-entropy, Dice, and IoU losses. Weighting can be statically fixed or dynamically adjusted during training (uncertainty weighting, self-adaptive weighting).

Post-Processing and Refinement

Raw output from segmentation networks is often noisy and has disconnected regions or small isolated pixels. Post-processing steps can significantly improve the quality of segmentation outputs.

Morphological operations (erosion and dilation) of classical image processing can provide clean predictions. Erosion removes small objects and noise, shrinking region boundaries. Dilation expands regions, potentially filling small holes. The combination of erosion and dilation (morphological opening and closing) effectively removes noise while preserving larger structures.

Connected components analysis identifies connected regions in predictions, labeling each connected region with a unique identity Individual instances of the same class are mistakenly segmented as a single region.

Conditional Random Fields (CRFs) are probabilistic graphical models that can refine segmentation by considering spatial correlations. CRFs model relationships between neighboring pixels—neighboring pixels are likely to have the same label. Posterior inference in CRFs can be done efficiently using mean field approximation, refining segmentation by encouraging spatial consistency.

Implementation with TensorFlow

Chapter 8 covers the practical implementation of segmentation networks using the TensorFlow Keras API. Custom layers can be defined to handle specific operations. Pre-trained backbones from TensorFlow Hub or Keras Applications (VGG, ResNet, EfficientNet) can serve as efficient feature extractors, with custom decoder heads added for segmentation.

Data loading and preprocessing are critical for segmentation—images must be resized, pixels normalized, and masks must be in the proper format. TensorFlow's tf.data API provides efficient data pipelines for loading, preprocessing, and batching data. Augmentation strategies are similar to classification but must consistently apply to both images and masks.

Training loops can use the high-level Keras API (model.fit) or custom training loops for greater control. Callbacks for early stopping, learning rate scheduling, and checkpoint saving standard practice. Evaluation metrics must be appropriate for segmentation—mean Intersection over Union (mIoU), Dice coefficient, boundary metrics.

Real-World Applications

Segmentation applications are vast and impactful. In medical imaging, tumor segmentation from CT/MRI scans enables precise radiotherapy planning and surgical guidance. Organ segmentation assists in surgical planning and volume measurements. Retinal vessel segmentation helps diagnose diabetic retinopathy. In autonomous vehicles, segmentation of roads, sidewalks, and obstacles provides critical environmental understanding beyond simple detection. In satellite imagery, land cover segmentation (water, vegetation, urban areas) aids environmental monitoring and urban planning. Industrial quality control uses segmentation to detect defects in manufactured products. In robotics, segmentation helps grippers identify grasping regions on objects.

Conclusion and Significance

Chapter 8 shows that with proper architectural design (encoder-decoder, skip connections), appropriate loss functions, and post-processing, networks can achieve remarkable pixel-level understanding of images. The transition from classification to segmentation requires new thinking about network design and loss functions, but the fundamental principles remain. The powerful segmentation capabilities discussed in this chapter enable sophisticated computer vision applications that are not possible with classification or detection alone.