FINAL-REPORT: SEMANTIC SEGMENTATION

SongSeungWu

1 WHAT IS SEMANTIC SEGMENTATION?

Semantic Segmentation is a computer vision technique that predicts class labels for every pixel in an image. It is used to understand and distinguish objects in an image at the pixel level, with applications in autonomous driving, medical image analysis, satellite image processing, and more.

2 PIXEL WISE CLASSIFICATION USING SLIDING WINDOW (EXAMPLE WITH VGG-NET)

Objective: This code implements semantic segmentation using a sliding window approach with a pre-trained VGG16 network. Each pixel in the input image is classified based on its surrounding context, and the results are visualized to highlight the pixels classified as 'Egyptian Cat' (class 285 in ImageNet).
Approach:

- Sliding Window: A fixed-size sliding window (224x224) extracts patches from the padded input image. Each patch is classified using VGG16, and the class predictions are assigned to the center pixel of the patch.
- Pre-trained Network: A VGG16 model pre-trained on ImageNet is used for classification. It predicts the class probabilities for each patch, and the highest probability class is selected for each pixel.

Visualization: The results are visualized to show pixels classified as 'Egyptian Cat'.
Strengths:

- Pixel-wise Classification: The sliding window approach ensures that each pixel is classified individually, enabling fine-grained semantic segmentation.
- Reusability of Pre-trained Models: The implementation leverages a pre-trained VGG16 model, eliminating the need for training from scratch and allowing the network to utilize learned features.
- Visual Results: The output visualization effectively highlights the pixels classified as the target class, making it easier to interpret the results.

Weaknesses:

- Computational Efficiency: The pixel-wise classification using nested loops is computationally expensive, especially for larger images. Each iteration involves passing a patch through the VGG16 network, which is slow.
- Padding Size: Padding the image with zeros might introduce artifacts or incorrect classifications near the image boundaries.
- Limited Context: The 224x224 window captures local context but fails to account for global relationships in the image.
- Pre-trained Model Mismatch: VGG16 is trained for image-level classification, not for pixel-level tasks. The lack of adaptations for semantic segmentation.

Suggested Improvements:

- Use fully convolutional networks or segmentation-specific models.
- Optimize the sliding window process using vectorized operations or direct FCN adaptation.

3 PIXEL WISE CLASSIFICATION USING IMAGE CLASSIFICATION (CONVOLUTIONAL VGG)

Objective: The goal is to adapt a VGG16 model into a Fully Convolutional Network to perform pixel-wise classification. The implementation modifies VGG16's fully connected layers into convolutional layers, enabling dense predictions across the entire image.
Strengths:

- Conversion to Fully Convolutional Network: The modified VGG16 architecture removes the constraint of fixed input sizes by replacing fully connected layers with convolutional layers. This is essential for semantic segmentation tasks.
- Reusability of Pre-trained Weights: Parameters from the pre-trained VGG16 are transferred to the modified layers leveraging pre-trained ImageNet knowledge.
- Pixel-wise Output: The adaptation allows the network to output pixel-wise predictions instead of a single class for the entire image.
- Simple and Straightforward: The approach reuses a familiar architecture and introduces minimal changes, making it easy to understand and implement.

Weaknesses:

- Resolution Mismatch: The output resolution is smaller than the input image resolution due to downsampling in the pooling layers of VGG16.
- Misclassifications: Many pixels are incorrectly classified. This is because VGG16 was designed for image-level classification, not pixel-wise tasks.
- Lack of Upsampling: Without explicit upsampling or deconvolution layers, the model's output cannot match the original input resolution.

Suggested Improvements:

- Resolution Recovery: Use upsampling layers to match the output resolution with the input image resolution.
- Segmentation-Specific Loss: Replace standard classification loss with a segmentation loss.
- Context Enhancement: Use Dilated Convolutions to increase the receptive field without additional pooling layers. This helps the model understand larger contexts without losing resolution.

4 PIXEL WISE CLASSIFICATION USING IMAGE CLASSIFICATION (ADD SIMPLE UPSAMPLING LAYER (BILINEAR) AND RETAIN WITH PASCAL VOC 2012)

Objective: This code implements a Fully Convolutional Network by modifying the VGG16 network for semantic segmentation. The output resolution is adjusted to match the input resolution using Bilinear Upsampling. The model is trained on the PASCAL VOC 2012 dataset, which contains 21 classes.
Code Review:

- Model Transformation: Fully Connected Layers (fc6, fc7, fc8) of VGG16 are replaced with Convolutional Layers.

- Bilinear Upsampling: nn.functional.interpolate is used to upsample the output to match the input image size.
- Pre-trained Weights: Weights from the pre-trained VGG16 are transferred to fc6 and fc7 for initialization.
- Visualization: The decode_labels function maps pixel values to RGB colors for PASCAL VOC classes.

Strengths:

- Output Matches Input Resolution: Bilinear Upsampling ensures the output matches the input size, resolving resolution mismatches common in CNN architectures.
- Simple Architecture: The code efficiently modifies VGG16 to produce dense predictions, retaining its straightforward structure.
- Pre-trained Weights: Leveraging pre-trained VGG16 weights significantly reduces training time and improves initial performance.

Weaknesses:

- Misclassification: Many pixels are misclassified, especially at boundaries or for small objects.
- Lack of Global Context: Pooling layers in VGG16 reduce spatial resolution and discard contextual information, which is critical for semantic segmentation.

Suggested Improvements:

- Use Dilated Convolutions: Replace standard convolutions in deeper layers with dilated convolutions to expand the receptive field without increasing the number of parameters.
- Add Skip Connections: Incorporate skip connections from lower layers to the final output to recover fine-grained spatial details.

5 FCN8s MODEL

Objective: This implementation uses Transposed Convolutions to perform semantic segmentation. The network, based on a modified VGG16 architecture, uses transposed convolutions to upsample the low-resolution feature maps back to the input resolution for dense pixel-wise predictions. The model is trained on augmented PASCAL VOC 2012 data.
Transposed Convolutions:

- What Are Transposed Convolutions?: Transposed convolutions are used to upsample feature maps. Unlike standard convolutions, which aggregate spatial information, transposed convolutions distribute a single point of input information to multiple output locations.
- Advantages: 1. Unlike bilinear interpolation, transposed convolutions learn filters, enabling adaptive upsampling that can preserve finer details.
  2. By adjusting kernel size, stride, and padding, the model can control the scale and quality of upsampling.

Architecture:

- Base Model: VGG16 (pre-trained on ImageNet).
- Fully Convolutional Layers: Converts fully connected layers into convolutional layers.
- Adds transposed convolution layers for progressive upsampling.
- Combines outputs from intermediate layers with upsampled predictions for finer details.

Training:

- Dataset: PASCAL VOC 2012 with augmented training data.
- Loss: Pixel-wise Cross-Entropy Loss.
- Optimizer: Adam with a learning rate of 0.00001
- Data Augmentation: Random scaling and flipping to improve generalization.

Evaluation:

- Metric: Mean Intersection over Union (mIoU).

Strengths:

- Transposed convolutions allow learnable upsampling.
- Skip connections improve boundary predictions.
- Data augmentation enhances generalization.

Weaknesses:

- Limited receptive field due to pooling layers in VGG16.

6 FCN8s MODEL RESULT

Training Results: The training process was conducted over 20,000 iterations with a learning rate of 0.00001. The loss values showed a steady decrease in the early stages and later stabilized as the model converged. Key training loss observations include:

- At the 100th iteration, the average loss was 2.9209, indicating a relatively high loss during the initial learning phase.
- By the 1,000th iteration, the loss had significantly decreased to 1.2218, demonstrating effective learning.
- In the later stages of training (15,000th iteration), the loss stabilized at 0.4530, with minimal fluctuations.
- At the end of training (20,000th iteration), the loss settled at 0.5420, marking the conclusion of the training process.

These results suggest that the model successfully learned patterns in the data, achieving a substantial reduction in loss over the course of training.

Validation Results: The model was evaluated on the PASCAL VOC 2012 validation set using Mean Intersection over Union (mIoU) as the performance metric. A total of 1,449 validation images were used, and the model achieved a final mIoU score of 0.4609, which measures the overlap between the predicted classes and the ground truth at the pixel level. Key observations from the validation process:

- The model performed well in segmenting large objects and simple shapes.
- However, it struggled with small objects and thin boundaries, often misclassifying such regions.

7 CONCLUSION

The implemented FCN8s model demonstrated a reasonable capability for semantic segmentation on the PASCAL VOC 2012 dataset. Over 20,000 iterations, the model successfully reduced the loss, indicating effective learning of data patterns. The final validation mIoU score of 0.4609 showcases the model's ability to perform pixel-wise classification, particularly for larger and simpler objects.

However, the performance fell slightly short of the desired benchmark of 0.5 mIoU, with notable challenges in segmenting small objects and thin boundaries. Additionally, checkerboard artifacts were observed in some predictions, likely stemming from the use of transposed convolutions.