Build a simple CNN model for image segmentation

1. **Libraries and Model Definition**
    - TensorFlow, NumPy, and Keras layers/models are used to define a custom CNN-based image segmentation model.
    - The model follows an encoder-decoder structure:
        - The **encoder** extracts features using convolution and max-pooling layers.
        - The **bottleneck** deepens the network with more filters.
        - The **decoder** upsamples using transposed convolutions to restore spatial resolution.
        - The final output layer uses a softmax activation to assign a class to each pixel.

2. Image Preprocessing

- An image is loaded, resized to 128x128, and normalized.
- It is reshaped to include the batch dimension as required by the model.

3. Model Inference

- The CNN model is created and used to predict a segmentation mask on the input image.
- The predicted output is converted into a class map (per-pixel prediction) using `argmax`.
- The original image and the segmentation mask are displayed side-by-side using Matplotlib.

4. TensorFlow Compatibility Fix

- The script modifies internal PixelLib files to switch import paths between `tensorflow.python.keras` and `tensorflow.keras` to fix potential compatibility issues.

- It uses the `fileinput` module to search and replace import lines in the source file `deeplab.py`.

5. Semantic Segmentation Using PixelLib

- PixelLib's `SemanticSegmentation` class is used to load a pre-trained DeepLabV3+ model trained on the Pascal VOC dataset.
- An image is passed through this model, and the segmentation result is saved.
- The segmented output image is loaded and displayed using Matplotlib.