```python
import numpy as np
import pydensecrf.densecrf as dcrf
from pydensecrf.utils import unary_from_softmax, create_pairwise_bilateral


def crf_proc(imgs, probs):
    _, _, H, W = imgs.shape
    imgs = imgs.transpose((0, 2, 3, 1))
    lbls = []
    for img, prob in zip(imgs, probs):
        prob = np.concatenate((1-prob[None, ...], prob[None, ...]), 0)
        # Example using the DenseCRF class and the util functions
        d = dcrf.DenseCRF2D(img.shape[1], img.shape[0], 2)

        # get unary potentials (neg log probability)
        U = unary_from_softmax(prob)
        d.setUnaryEnergy(U)

        # This creates the color-dependent features and then add them to the CRF
        feats = create_pairwise_bilateral(sdims=(80, 80), schan=(13, 13, 13),
                                          img=img, chdim=2)
        d.addPairwiseGaussian(sxy=(3, 3), compat=3, kernel=dcrf.DIAG_KERNEL, normalization=dcrf.NORMALIZE_SYMMETRIC)
        d.addPairwiseEnergy(feats, compat=10,
                            kernel=dcrf.DIAG_KERNEL,
                            normalization=dcrf.NORMALIZE_SYMMETRIC)

        # Run five inference steps.
        Q = d.inference(5)

        # Find out the most probable class for each pixel.
        MAP = np.argmax(Q, axis=0).reshape((H, W))
        lbls.append(MAP)
    lbls = np.stack(lbls)
    return lbls

```