B.Tech. Project: Spring 2022
Open Set Segmentation for Remote Sensing

Lalit Saini
Roll no. 180070030

Guide : Prof. Biplab Banerjee
Co Guide: Prof. V. Rajbabu

Department of Electrical Engineering
IIT Bombay

May 2022

Outline

Open Set Segmentation

- Addressing problem of identifying, during the testing phase, instances of classes not seen during training phase.

Open Set Segmentation

- Addressing problem of identifying, during the testing phase, instances of classes not seen during training phase.
- System Should be able to correctly classify pixels of classes seen during training, whereas recognizing pixels of classes not seen during training.

Open Set Segmentation

- Addressing problem of identifying, during the testing phase, instances of classes not seen during training phase.
- System Should be able to correctly classify pixels of classes seen during training, whereas recognizing pixels of classes not seen during training.
- Open set Approach, assume that all the possible classes are not known during training.

OpenMAX

- Re-estimates soft max probabilities and rejects unknown and uncertain inputs.

[1] Abhijit Bendale and Terrance Boult. *Towards Open Set Deep Networks*. 2015. DOI: 10.48550/ARXIV.1511.06233. URL: https://arxiv.org/abs/1511.06233

OpenMAX

- Re-estimates soft max probabilities and rejects unknown and uncertain inputs.
- Probabilities of top α classes are modified. The input and the class parameters are used to fit a Weibull distribution to get weights for each of the top α classes.

[1] Abhijit Bendale and Terrance Boult. *Towards Open Set Deep Networks*. 2015. DOI: 10.48550/ARXIV.1511.06233. URL: https://arxiv.org/abs/1511.06233

OpenMAX

- Re-estimates soft max probabilities and rejects unknown and uncertain inputs.
- Probabilities of top α classes are modified. The input and the class parameters are used to fit a Weibull distribution to get weights for each of the top α classes.
- Using the weights we define activation vector for unknown class as $\hat{v}_0 = \Sigma_i v_i(x)(1 - w_i(x))$. Now we have activation for C+1 classes.

[1] Abhijit Bendale and Terrance Boult. *Towards Open Set Deep Networks*. 2015. DOI: 10.48550/ARXIV.1511.06233. URL: https://arxiv.org/abs/1511.06233

OpenMAX

- Re-estimates soft max probabilities and rejects unknown and uncertain inputs.
- Probabilities of top α classes are modified. The input and the class parameters are used to fit a Weibull distribution to get weights for each of the top α classes.
- Using the weights we define activation vector for unknown class as $\hat{v}_0 = \Sigma_i v_i(x)(1 - w_i(x))$. Now we have activation for C+1 classes.
- All the activation vectors are now used to calculate revised scores.

[1] Abhijit Bendale and Terrance Boult. *Towards Open Set Deep Networks*. 2015. DOI: 10.48550/ARXIV.1511.06233. URL: https://arxiv.org/abs/1511.06233

OpenMAX

- Re-estimates soft max probabilities and rejects unknown and uncertain inputs.
- Probabilities of top α classes are modified. The input and the class parameters are used to fit a Weibull distribution to get weights for each of the top α classes.
- Using the weights we define activation vector for unknown class as $\hat{v_0} = \Sigma_i v_i(x)(1 - w_i(x))$. Now we have activation for C+1 classes.
- All the activation vectors are now used to calculate revised scores.
- If 0^{th} class (i.e. unknown class) has highest score or all other classes have score less than a threshold the input is considered unknown.

[1] Abhijit Bendale and Terrance Boult. *Towards Open Set Deep Networks*. 2015. DOI: 10.48550/ARXIV.1511.06233. URL: https://arxiv.org/abs/1511.06233

OpenMAX

- Re-estimates soft max probabilities and rejects unknown and uncertain inputs.
- Probabilities of top α classes are modified. The input and the class parameters are used to fit a Weibull distribution to get weights for each of the top α classes.
- Using the weights we define activation vector for unknown class as $\hat{v}_0 = \Sigma_i v_i(x)(1 - w_i(x))$. Now we have activation for C+1 classes.
- All the activation vectors are now used to calculate revised scores.
- If 0^{th} class (i.e. unknown class) has highest score or all other classes have score less than a threshold the input is considered unknown.
- Algorithm requires Weibull shape parameters and shifting paramter fit using EVT from correctly classified training examples.

[1] Abhijit Bendale and Terrance Boult. *Towards Open Set Deep Networks*. 2015. DOI: 10.48550/ARXIV.1511.06233. URL: https://arxiv.org/abs/1511.06233

OpenFCN

- Based on traditional FCN-based architecture. A meta recognition module is added to the prediction procedure of traditional FCN.

[1] Hugo Oliveira, Caio Silva, Gabriel L. S. Machado, et al. *Fully Convolutional Open Set Segmentation*. 2020. arXiv: 2006.14673 [cs.CV]

OpenFCN

- Based on traditional FCN-based architecture. A meta recognition module is added to the prediction procedure of traditional FCN.
- In training, the FCN backbone is trained on C known classes using Cross Entropy loss. This yields an activation for each pixel after the last layer of FCN network.

[1] Hugo Oliveira, Caio Silva, Gabriel L. S. Machado, et al. *Fully Convolutional Open Set Segmentation*. 2020. arXiv: 2006.14673 [cs.CV]

OpenFCN

- Based on traditional FCN-based architecture. A meta recognition module is added to the prediction procedure of traditional FCN.
- In training, the FCN backbone is trained on C known classes using Cross Entropy loss. This yields an activation for each pixel after the last layer of FCN network.
- Using the activations OpenMax is computed and the probabilities are threshold in order to predict Unknown classes.

[1] Hugo Oliveira, Caio Silva, Gabriel L. S. Machado, et al. *Fully Convolutional Open Set Segmentation*. 2020. arXiv: 2006.14673 [cs.CV]

OpenFCN

- Based on traditional FCN-based architecture. A meta recognition module is added to the prediction procedure of traditional FCN.
- In training, the FCN backbone is trained on C known classes using Cross Entropy loss. This yields an activation for each pixel after the last layer of FCN network.
- Using the activations OpenMax is computed and the probabilities are threshold in order to predict Unknown classes.
- Depending on threshold value on recalculated probabilities through OpenMax, pixels are categorized as unknown and if it is known correct class label assigned.

[1] Hugo Oliveira, Caio Silva, Gabriel L. S. Machado, et al. *Fully Convolutional Open Set Segmentation*. 2020. arXiv: 2006.14673 [cs.CV]

OpenFCN

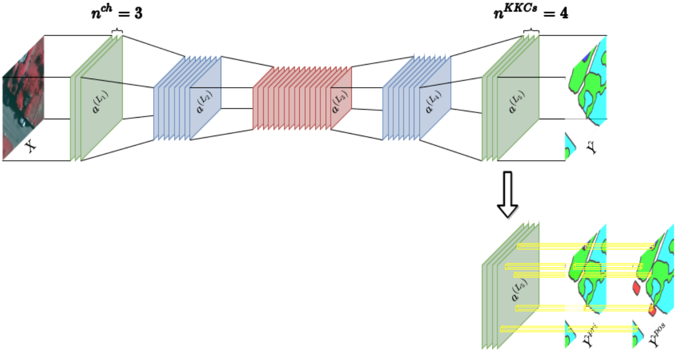

Figure 1: Architecture of OpenFCN

[1] Hugo Oliveira, Caio Silva, Gabriel L. S. Machado, et al. *Fully Convolutional Open Set Segmentation*. 2020. arXiv: 2006.14673 [cs.CV]

OpenPCS

- Using FCN backbone like OpenFCN, instead of just last layer activation features, it uses intermediate layer activation also.

[1] Hugo Oliveira, Caio Silva, Gabriel L. S. Machado, et al. *Fully Convolutional Open Set Segmentation*. 2020. arXiv: 2006.14673 [cs.CV]

OpenPCS

- Using FCN backbone like OpenFCN, instead of just last layer activation features, it uses intermediate layer activation also.
- The intermediate activations are scaled properly and concatenated to get a bigger activation vector a^*.

[1] Hugo Oliveira, Caio Silva, Gabriel L. S. Machado, et al. *Fully Convolutional Open Set Segmentation*. 2020. arXiv: 2006.14673 [cs.CV]

OpenPCS

- Using FCN backbone like OpenFCN, instead of just last layer activation features, it uses intermediate layer activation also.
- The intermediate activations are scaled properly and concatenated to get a bigger activation vector a^*.
- Convolutional layers's activations have large redundancy. OpenPCS mitigate this by performing PCA on a^* and obtaining a lower dimension representation a^{low}.

[1] Hugo Oliveira, Caio Silva, Gabriel L. S. Machado, et al. *Fully Convolutional Open Set Segmentation*. 2020. arXiv: 2006.14673 [cs.CV]

OpenPCS

- Using FCN backbone like OpenFCN, instead of just last layer activation features, it uses intermediate layer activation also.
- The intermediate activations are scaled properly and concatenated to get a bigger activation vector a^*.
- Convolutional layers's activations have large redundancy. OpenPCS mitigate this by performing PCA on a^* and obtaining a lower dimension representation a^{low}.
- One multivariate Gaussian is fit for each class using a^{low}. Thresholds defined according to likelihood from these models is used to identify unknown classes.

[1] Hugo Oliveira, Caio Silva, Gabriel L. S. Machado, et al. *Fully Convolutional Open Set Segmentation*. 2020. arXiv: 2006.14673 [cs.CV]

OpenPCS

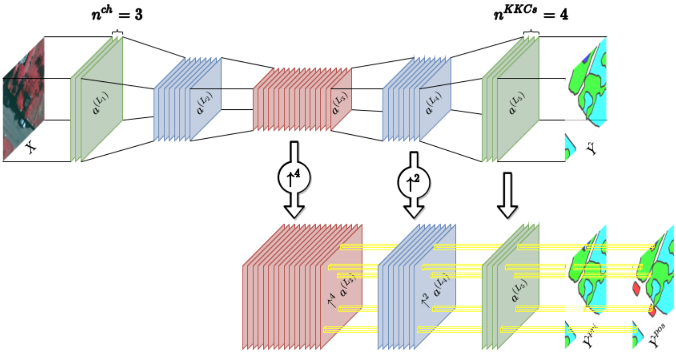

Figure 2: Architecture of OpenPCS

[1] Hugo Oliveira, Caio Silva, Gabriel L. S. Machado, et al. *Fully Convolutional Open Set Segmentation*. 2020. arXiv: 2006.14673 [cs.CV]

OpenPCS

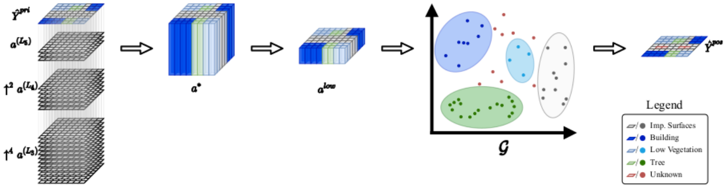

Figure 3: Fitting of multivariate Gaussians

[1] Hugo Oliveira, Caio Silva, Gabriel L. S. Machado, et al. *Fully Convolutional Open Set Segmentation*. 2020. arXiv: 2006.14673 [cs.CV]

Model 1 : Deep OpenPCS

- Created activation vector a^* using intermediate activation layers like OpenPCS.

Model 1 : Deep OpenPCS

- Created activation vector a^* using intermediate activation layers like OpenPCS.
- The activation vector a^* passed through another model called classifier to predict pixel from unknown class.

Model 1 : Deep OpenPCS

- Created activation vector a^* using intermediate activation layers like OpenPCS.
- The activation vector a^* passed through another model called classifier to predict pixel from unknown class.
- The FCN backbone is trained using the training set whereas the classifier is trained using the validation set.

Model 1 : Deep OpenPCS

- Created activation vector a^* using intermediate activation layers like OpenPCS.
- The activation vector a^* passed through another model called classifier to predict pixel from unknown class.
- The FCN backbone is trained using the training set whereas the classifier is trained using the validation set.
- During testing if pixel classified as known then appropriate class is assigned by prediction of backbone else it is classified as pixel from unknown class.

Model 1 : Deep OpenPCS

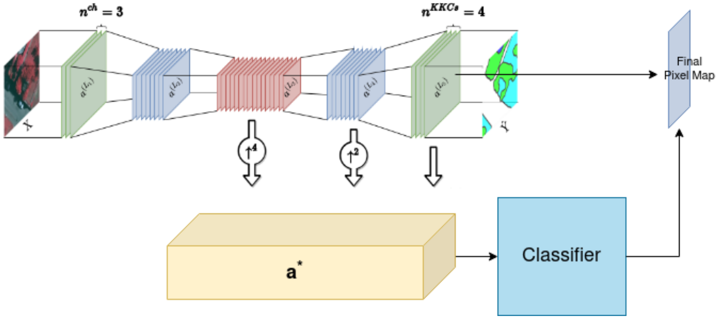

Figure 4: Architecture of Model 1

Model 1 : Deep OpenPCS

- Every image was of 4 channels namely R, G, B and last channel of normalised heights. Images were taken from Potsdam dataset.

Model 1 : Deep OpenPCS

- Every image was of 4 channels namely R, G, B and last channel of normalised heights. Images were taken from Potsdam dataset.
- FCN backbone trained for 200 epochs.

Model 1 : Deep OpenPCS

- Every image was of 4 channels namely R, G, B and last channel of normalised heights. Images were taken from Potsdam dataset.
- FCN backbone trained for 200 epochs.
- Classifier is convolutional layer, trained for 400 epochs using FCN backbone trained.

Model 1 : Deep OpenPCS

- Every image was of 4 channels namely R, G, B and last channel of normalised heights. Images were taken from Potsdam dataset.
- FCN backbone trained for 200 epochs.
- Classifier is convolutional layer, trained for 400 epochs using FCN backbone trained.
- Obtained a Dice Score of $\mathbf{0.166}$.

Model 1 : Deep OpenPCS

- Every image was of 4 channels namely R, G, B and last channel of normalised heights. Images were taken from Potsdam dataset.
- FCN backbone trained for 200 epochs.
- Classifier is convolutional layer, trained for 400 epochs using FCN backbone trained.
- Obtained a Dice Score of $\mathbf{0.166}$.
- Conclusion : For this kind of approach re-weighting works better rather than just depending on prediction of classifier.

Model 1 Results

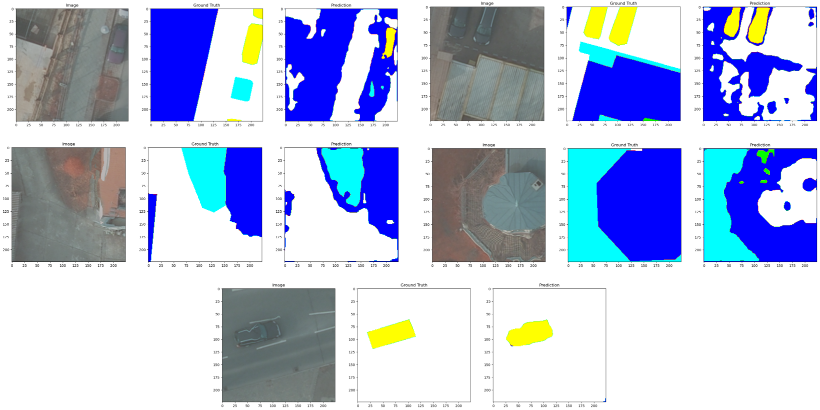

Figure 5: Model 1 Results

Model 2 : Using GAN and ResUNet

- Using the available training set DCGan was trained to half way to completion.

Model 2 : Using GAN and ResUNet

- Using the available training set DCGan was trained to half way to completion.
- Using the DCGAN unknown class examples generated, with distribution near to known classes but not exactly from a known class.

Model 2 : Using GAN and ResUNet

- Using the available training set DCGan was trained to half way to completion.
- Using the DCGAN unknown class examples generated, with distribution near to known classes but not exactly from a known class.
- A ResUnet is trained on the real and fake samples to predict C+1 classes.

Model 2 : Using GAN and ResUNet

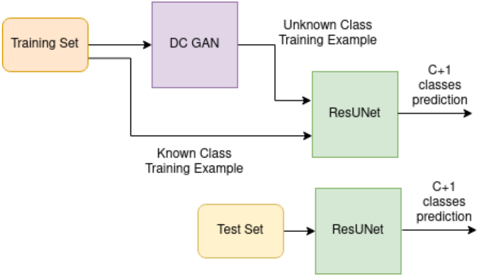

Figure 6: Architecture of Model 2

Model 2 : Using GAN and ResUNet

- Every image was of 4 channels namely R, G, B and last channel of normalised heights. Images were taken from Potsdam dataset.

Model 2 : Using GAN and ResUNet

- Every image was of 4 channels namely R, G, B and last channel of normalised heights. Images were taken from Potsdam dataset.
- DCGAN was trained for 600 epochs with noise vector dimension 100.

Model 2 : Using GAN and ResUNet

- Every image was of 4 channels namely R, G, B and last channel of normalised heights. Images were taken from Potsdam dataset.
- DCGAN was trained for 600 epochs with noise vector dimension 100.
- ResUNet was trained for 40 epochs using the generator of above GAN for unknown class samples with batch size of 8.

Model 2 : Using GAN and ResUNet

- Every image was of 4 channels namely R, G, B and last channel of normalised heights. Images were taken from Potsdam dataset.
- DCGAN was trained for 600 epochs with noise vector dimension 100.
- ResUNet was trained for 40 epochs using the generator of above GAN for unknown class samples with batch size of 8.
- Obtained a Dice Score of **0.217**.

Model 2 Results

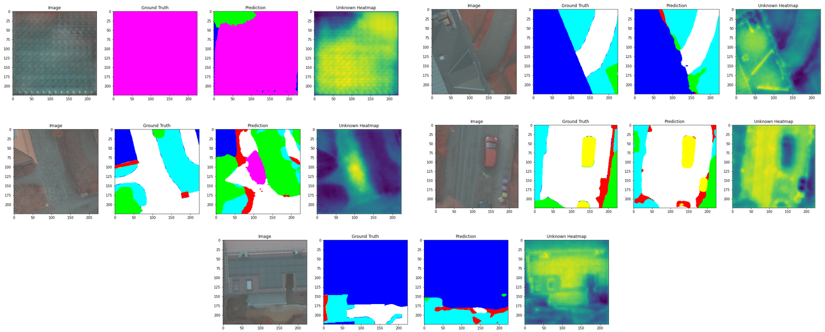

Figure 7: Model 2 Results

Thank you