THE EYE IN THE SKY

INDIAN INSTITUTE OF TECHNOLOGY, KANPUR

Kushagra Gupta
Department of Mathematics and Statistics
IIT Kanpur
kushgpt@iitk.ac.in

Harish Rajagopal
Department of Computer Science
IIT Kanpur
rharish@iitk.ac.in

Harsh Sinha
Department of Aerospace Engineering
IIT Kanpur
harshsin@iitk.ac.in

Apurv Gupta
Department of Mathematics
IIT Kanpur
apurv@iitk.ac.in

January 23, 2019

Contents

1 Introduction

Advances in image capturing technology has enabled us to capture detailed pictures of objects from very large distances. An immediate application is Satellite Imagery, which is being used extensively by governments, researchers and businesses across the globe for surveillance and analysis. Hundreds of remote sensing satellites suspended over the Earth provide highly detailed images which are rich in information. Developments in satellite imagery presents a challenging task of making sense of this surplus data and draw meaningful, useful inferences. One of these tasks is to classify objects present in the satellite images. This information could either be used by cartographers to create custom maps, or by a country's defence forces for better surveillance.

Classification, by design, is a problem in Learning theory. Deep Learning architectures have proven to beat classical classification algorithms, and humans too, at supervised classification tasks. The problem at hand naturally fits into a multi-label classification problem, with each of the given 9 classes of objects representing a label. We will attempt to solve it by using deep neural networks that systematically extract simple and complex features from the provided images in order to perform classification.

Problem Statement: The task is to classify all pixels in a given satellite image, as belonging to one of given 9 classes. The classes include: Roads, Buildings, Trees, Grass, Bare Soil, Water, Railways, Swimming pools and Unidentified class. Each class is represented by a colour, to make each class visually distinguishable from the other. Some pixels in the ground truth are assigned the colour white, which means that pixel belongs to unidentified class.
NOTE:- We are treating the given problem as 9 class problem treating white as one of the classes.

2 Classification Approach

2.1 Motivation

2.1.1 Segmentation

Progress in the field of satellite imagery has played an important role in improving our understanding of the planet over the years. It has enabled researchers to make great strides in various fields of application, ranging everything from monitoring effects of global warming to mobilising resources during disasters. Satellite image classification is a subject of considerable interest in the scientific community owing to the recent developments in computer vision and deep learning, along with the procurement of vast amounts of satellite imagery. The possibility of accurately distinguishing different types of objects in aerial images, such as buildings, roads, vegetation and other categories, finds applications in various fields, such as environment monitoring, disaster relief, keeping up-to-date maps, and improving urban planning. Besides the practical need for accurate aerial image interpretation systems, this domain also poses a scientific challenge to computer vision.

Satellite image classification is a two-step approach which involves Semantic Segmentation of images followed by a pixel-level Classification into predefined classes. The aim of segmentation is to divide the image into homogeneous regions, which are analysed to locate objects and boundaries (lines, curves, etc) in images. More precisely, image segmentation is the process of assigning a label to every pixel in an image such that pixels with the same label share certain visual characteristics. The Convolutional Neural Networks (CNN) is one of the main supervised approaches that is successfully used for this task. The most significant advancement of these networks is its ability to learn appropriate feature representation in an end to end manner while avoiding creation of feature that are hand-crafted and which require too much tuning to make them work for a particular case.

2.2 Literature Survey and Baseline Models Implementation

2.2.1 One for all U-Net model

Approach Considering the lack of training data, our first approach for image segmentation was a U-Net architecture. U-Nets have an ability to learn in environments of low to medium quantities of training data, and this matched with the amount of training data available for the competition. The architecture of a U-Net is similar to that of a convolutional auto-encoder, but with skip-like connections with the feature maps located before the bottleneck (compressed embedding) layer, in such a way that in the decoder part some information comes from previous layers, bypassing the compressive bottleneck. This results in the data being recovered from a compression along with being concatenated with the information's state before it was passed into the compression bottleneck. The concatenation allows us to augment context for the next decoding layer to come. This allows the U-Net to not only learn to generalise in the compressed latent representation, but also to recover its latent generalisations to a spatial representation with the proper per-pixel semantic alignment in the expansive path of the U-Net.

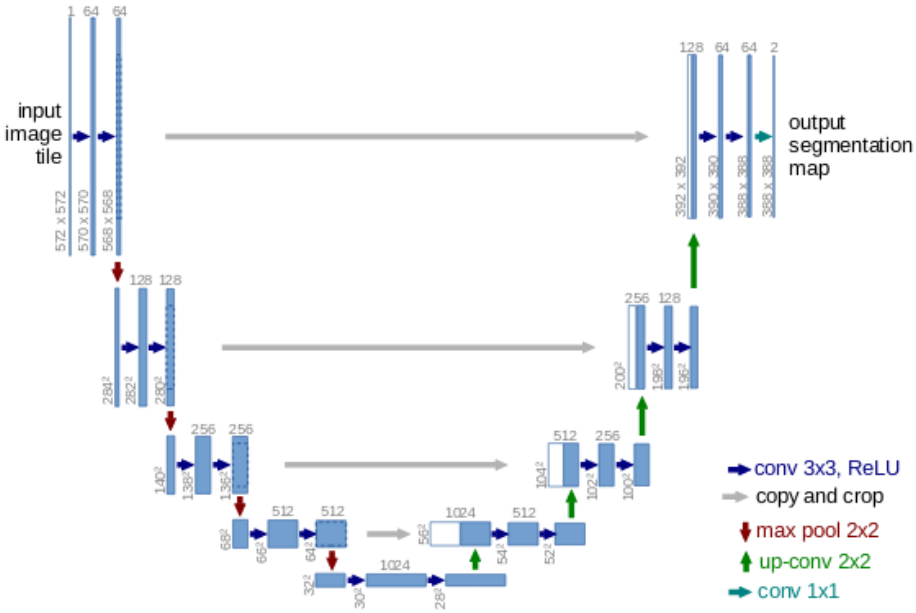

Figure 1: U-net architecture (example for 112x112 pixels in the lowest resolution). Each blue box corresponds to a multi-channel feature map. The number of channels is denoted on top of the box. The arrows denote the different operations.

Architecture As represented in the figure, the network consists of a contracting path (left side) and an expansive path (right side). The contracting path follows the typical architecture of a convolutional network. It consists of the repeated application of two 3x3 convolutions (unpadded convolutions), each followed by a rectified linear unit (Re-LU) and a 2x2 max pooling operation with stride 2 for down sampling. At each down-sampling step we double the number of feature channels. Every step in the expansive path consists of an up-sampling of the feature map followed by a 2x2 convolution ("up-convolution") that halves the number of feature channels, a concatenation with the correspondingly cropped feature map from the contracting path, and two 3x3 convolutions, each followed by a Re-LU. The cropping is necessary due to the loss of border pixels in every convolution. At the final layer a 1x1 convolution is used to map each 64-component feature vector to the desired number of classes. The reference for the network architecture can be seen in the official paper of the U-Net [1]. We used Nadam Optimiser (Adam with Nesterov momentum) with a learning of 5e-4 during training. The model was trained on 112x112 patches of the given data-set for 50 epochs. We employed multiple metrics during training, including jaccard similarity index, precision, recall and categorical accuracy on binary cross entropy loss function.

Corresponding to this model we obtained a validation accuracy of **76**%. In our submission directory, code for this model can be found at code/UnetModel/.

2.2.2 P-Net model

Approach This method uses a Fully Convolutional Network trained on a modified dataset as given in section 2.4.1. The output of the network is trained on a loss incorporating image-specific fine-tuning as done by Wang et al. [2]. The difference between our approach and theirs is that they incorporate interactive segmentation using user input scribbles, whereas the fine-tuning component of our loss is unsupervised.

Let X be the input and let Y be the target output. Let the set of parameters of the network be denoted by θ. Our objective function is:

$$\underset{Y,\theta}{\operatorname{argmin}} \left\{ E(Y,\theta) = \sum_i \phi(y_i|X,\theta) + \lambda \sum_{i,j} \psi(y_i, y_j|X,\theta) \right\}$$

ϕ and ψ are unary and pairwise energy terms respectively. λ is the weight of ψ.

ϕ is given by:

$$\phi(y_i|X,\theta) = -w(i) \log P(y_i|X,\theta)$$

where $P(y_i|X,\theta)$ is the softmax probability as given by the network's output for the pixel i corresponding to the class y_i, and

$$w(i) = \begin{cases} 0, \text{ if } P(y_i|X,\theta) < t_0 \\ 1, \text{ otherwise} \end{cases}$$

where t_0 is a lower threshold for the softmax probability of class y_i.

The energy term ψ that represents the image-specific fine-tuning is given by:

$$\psi(y_i, y_j|X,\theta) = \mathbb{I}[y_i \neq y_j] \exp\left(-\frac{(X(i) - X(j))^2}{2\sigma^2}\right) \cdot \frac{1}{d_{ij}}$$

where $\mathbb{I}[y_i \neq y_j]$ is 1 if $y_i \neq y_j$ and 0 otherwise. d_{ij} is the Euclidean distance between pixels i and j in the 2D image. σ is a hyperparameter that controls the effect of intensity difference.

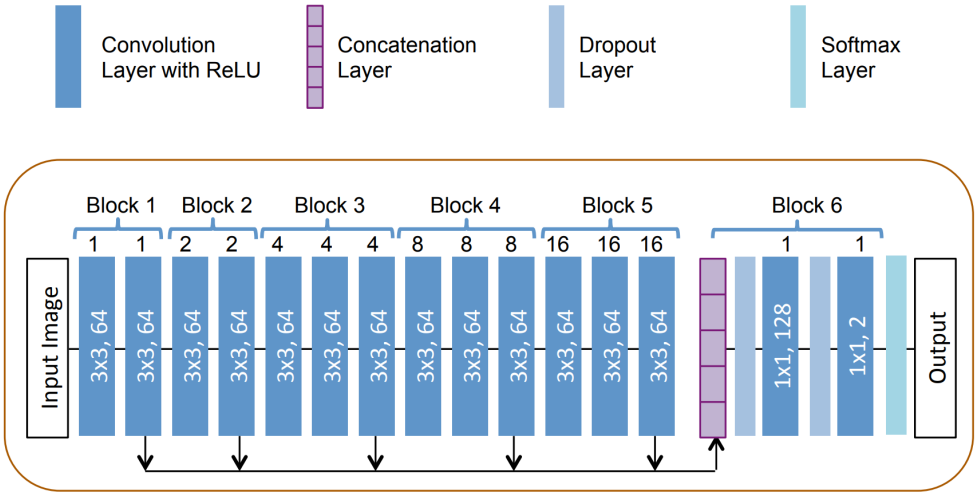

Figure 2: P-Net for segmentation. The numbers in dark blue boxes denote convolution kernel sizes and numbers of output channels, and the numbers on the top of these boxes denote dilation parameters.

Network Architecture We use a P-Net [3] architecture similar to Wang et al. [2] as shown in Fig.4. The network preserves the image resolution using padding. along with . It consists of six blocks with a kernel size of 3×3 for all the convolutional layers. Each 2D convolutional layer uses dilated convolutions [4] along with batch normalization [5]. The first five blocks have dilation parameters of 1, 2, 4, 8, and 16, respectively, so that they capture features at different scales. Features from these five blocks are concatenated and fed into block 6 that serves as a classifier. Dropout [6] is used during training time before each convolutional layer in this block. A softmax layer is used at the end to obtain probability-like outputs.

Hyperparameter Optimisation Hyperparameter Optimization can be described by the following expression:

$$x^* = \underset{x \in X}{\operatorname{argmin}}\, f(x)$$

where x^* is the set of optimal hyperparameters, X is the set of all possible hyperparameters, and $f(x)$ is the valuation of the current model given x set of hyperparameters. The valuation of the model corresponds to the validation accuracy of the model. We select a set of hyperparameters from X and fit the model on the training data. Once the model converges, we compute the validation accuracy which forms the objective function for this optimization problem.

In order to achieve the best possible fit for the training data, we perform hyperparameter search using Tree-Structured Parzen Estimators method [7]. We use the open source library hyperopt [8] for efficiently searching the optimal hyperparameter. The process of hyperparamter optimization using TPE is described below.

1. We build a surrogate probability model of the objective function.

$$\mathbb{P}(score|hyperparameters)$$

2. We choose the next set of hyperparameters based on a parameter known as Expected Improvement.

$$EI_{y^*}(x) = \int_{-\infty}^{y^*} (y^* - y)\mathbb{P}(y|x)dy$$

where y^* is the Threshold value of Objective Function and y is the Actual value of the Objective Function.

$$\mathbb{P}(x|y) = \begin{cases} \text{l(x)} & \text{if y} < \text{y}^* \\ \text{g(x)} & \text{if y} \geq y^* \end{cases}$$

By applying Bayes' Rule and some simplification we get

$$EI_{y^*}(x) \approx (\gamma + \frac{g(x)}{l(x)}(1-\gamma))^{-1}$$

Thus to maximise the EI we maximise ratio $\frac{l(x)}{g(x)}$.

3. Apply the set of new hyperparameters chosen to the model and train it.
4. Update the surrogate model incorporating the results corresponding to the fitted model.

Next we present our hyperparameter search space.

Hyperparameter	Search Space	Range/Choices
Learning Rate	Continuous Uniform Distribution	$(10^{-5}, 10^{-3})$
Weight Decay	Continuous Uniform Distribution	$(5 \times 10^{-5}, 5 \times 10^{-3})$
Batch Size	Discrete Choices	$[16, 32, 64]$
Optimizer	Discrete Choices	[Adam, Adagrad, RMSProp]
Dropout Rate	Continuous Uniform Distribution	$(0.0, 0.7)$
Activation	Discrete Choices	[relu, elu]
t_0	Continuous Uniform Distribution	$(0.0, 1.0)$
σ	Continuous Uniform Distribution	$(0.0, 100.0)$
λ	Continuous Uniform Distribution	$(0.0, 100.0)$

After performing a search for 100 iterations we obtain the following set of hyperparameters:

Hyperparameter	Optimal Value	Hyperparameter	Optimal Value
Learning Rate	0.0008408132388618728	Weight Decay	0.003683848079337278
Batch Size	32	Optimizer	Adam
Dropout Rate	0.6275728419832726	Activation	elu
t_0	0.10805612575300722	σ	56.89889776692406
λ	45.40359136227982		

Corresponding to this model we obtained a validation accuracy of **78**%. In our submission directory, code for this model can be found at code/UnetModel/.

2.3 Methodology

This is our final model which we have used for producing final output.

2.3.1 One vs ALL U-Net Model

Several multi-label classification approaches draw from their binary counterparts to do classification. In such approaches, a separate, albeit identical network is used to learn a binary classifier for every class. The final label is predicted based on a probability argmax over all the binary classifiers. Such an approach, also termed the "One vs All" classification paradigm, has been shown to beat several simple neural network architectures that do multi-label classification.

We employ such a One vs All U-Net, drawn from the basic U-Net model which is shown the figure 3. Similar network is widely used for binary segmentation problems. A stack of 9 U-Nets, trained parallelly on the dataset, each learn to do binary segmentation of an image by assigning a 1 or 0 label to each pixel in the image. This binary classification naturally stems from a softmax, which implies each pixel has an associated probability, when being predicted for, by one of the U-Nets. Taking the argmax of these probabilities over all U-Nets, we can determine which class the given pixel is most likely to belong to. This will then give us a multi class segmentation of the query image.

2.4 Implementation

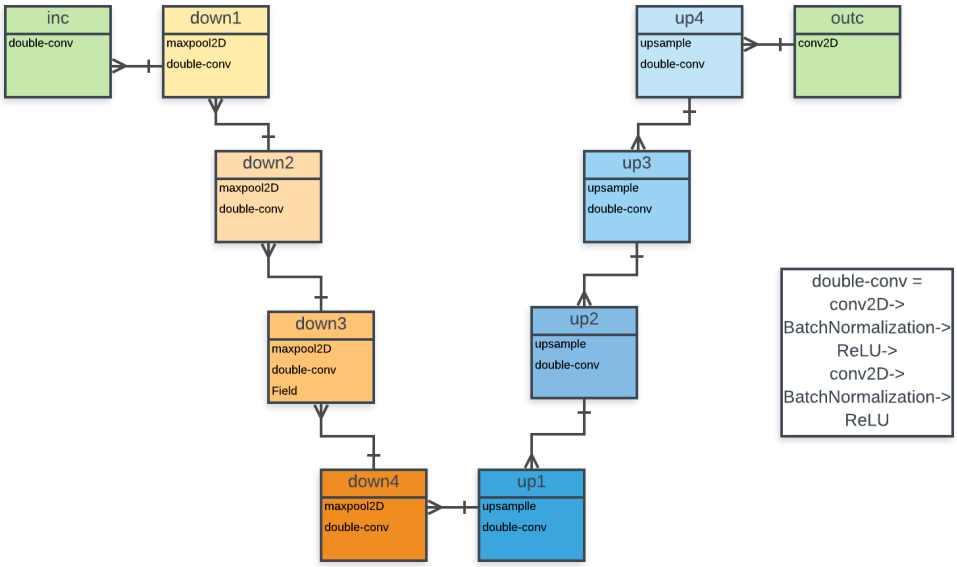

Figure 3: Architecture of one of 9 U-Nets trained for each class.

2.4.1 Dataset Generation

The original dataset given for the problem contained just 14 images. We randomly took 11 of these images for training. We decided to generate a modified dataset out of the given images by rotating the training images by 90 degrees clockwise and anticlockwise followed by splitting the original and rotated images into tiles. The angle of rotation was set as 90 due to the possible modification in the values of pixels in tiles generated in the rotated images with other angles of rotation. This modification can occur in the case of rotation angle not being 90 if the boundary of a tile passes through the middle of a pixel in the rotated image. The value of the bifurcated pixel becomes ambiguous which causes problems during training. Rotation of the image tripled the size of the dataset. Splitting the images into smaller sized tile had a twofold advantage. Firstly, the images provided in the training dataset were not of the same dimensions. Therefore, it was necessary to generate images of uniform size which can be passed as input to the network of different types. Secondly and more importantly, having smaller sized patches to train on allows the network to optimise its parameters with a higher frequency. Additionally, breaking the parent image into smaller tiles allows localised updating of parameters, rather than a wholesale update of all the parameters after a single pass. The modified dataset was used to train all the networks in all the approaches.

2.4.2 Training-Validation Procedure

We randomly selected 3 images **(8, 10, 12).tif** for the validation set. For training set, we took the other 11 images and then used the dataset generation method above to generate the final training dataset. We trained our final model on this set for 100 epochs. For the validation purpose, we directly passed the above mentioned three images without breaking it in patches since we have a CNN model and that won't affect the output. The accuracy metrics corresponding to these images is reported in section 4.

3 Results

The following are results corresponding to couple of training image and one of the validation image. Left is the satellite image and the right one is our prediction.

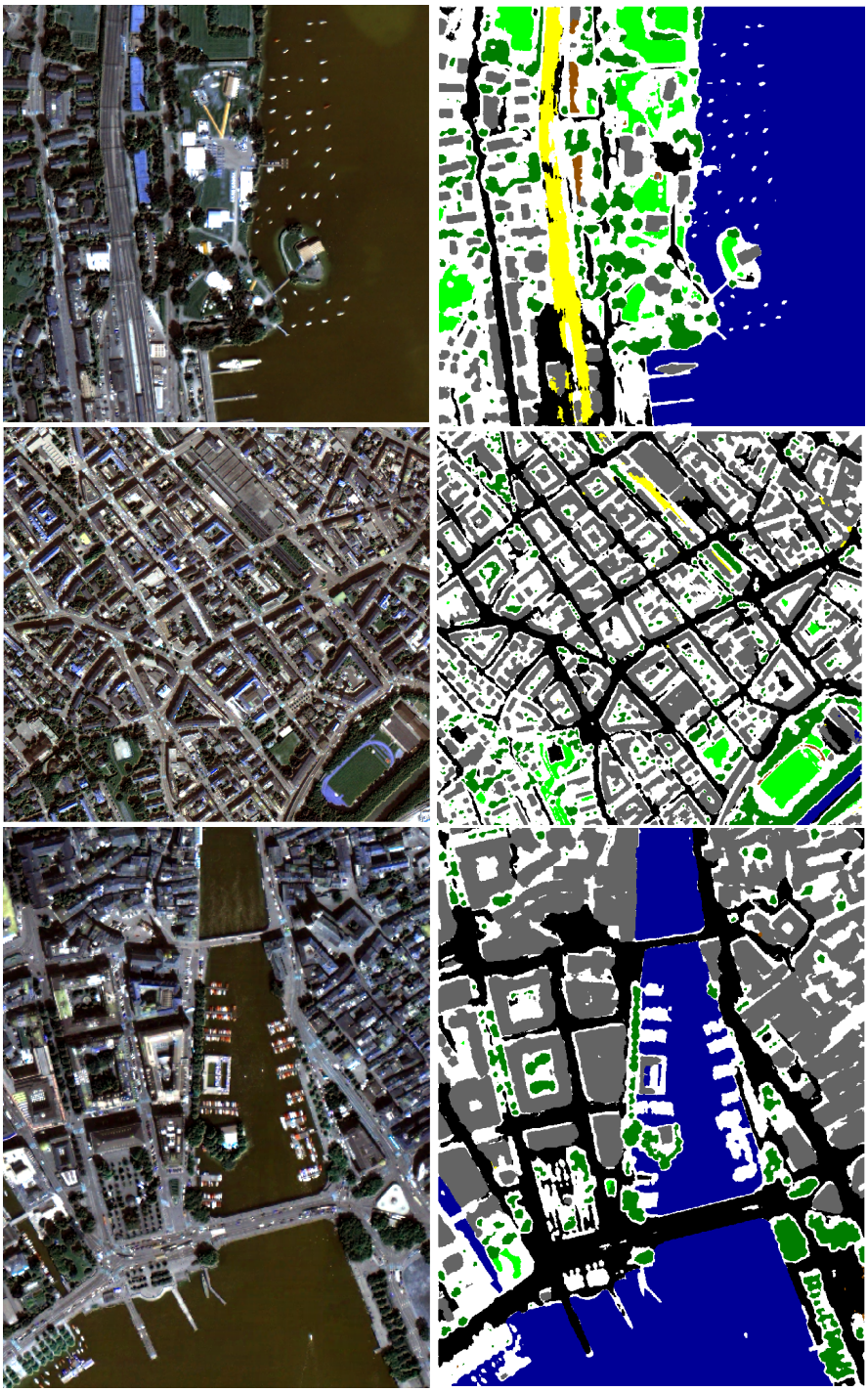

Figure 4: Prediction for training set images 2.tif and 5.tif and validation image 8.tif

4 Accuracy

Corresponding to our One vs All U-NET model, we obtained the following accuracy metrics:
As explained earlier we excluded three images at random from the training set and used them to obtain the following numbers.

4.1 Confusion Matrix:

The normalized confusion matrix is shown in the fig. below

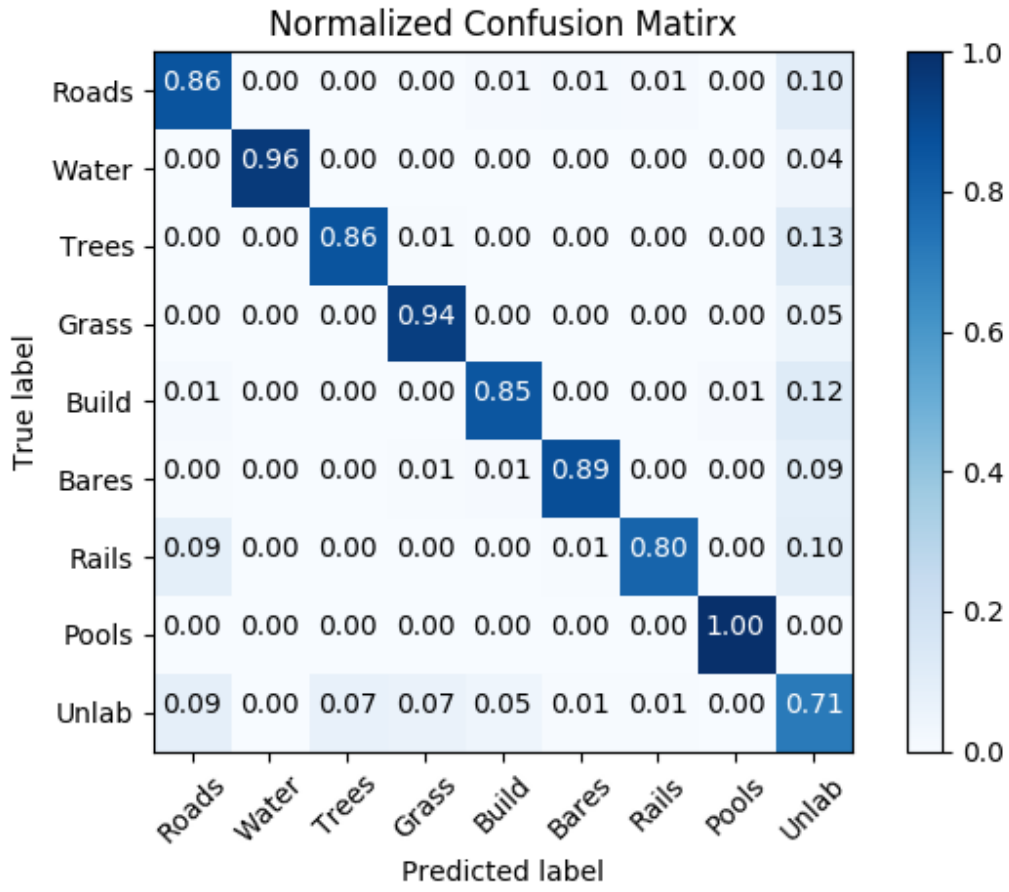

Figure 5: Normalized Confusion Matrix

4.2 Kappa coefficent

The kappa score for our model is **0.7577**.

4.3 Overall Accuracy

For 9 class classification, we have obtained **91.41%** accuracy with the One Vs All U-Net.

5 Conclusion

Our approach towards solving this challenge involved three approaches, with each approach performing better than the last one. The first model involved implementation of a basic U-Net architecture with custom metrics designed for this problem. The second approach involved a P-Net architecture with extensive hyperparamater tuning to improve model accuracy. The third approach utilizes nine distinct U-Net architectures for the segmentation, giving results comparable to state of the art model for multi-class image segmentation.

Although we obtained promising results with the One vs All U-Net architecture, there are many avenues for improvement to the method. One of the most straight forward approach for improvement is to use different hyper-parameters for all of the individual networks.

Further improvements can be made by using generative models such as GANs, where we can construct more data using a GAN, and learn the classification using the discriminator itself. This further solves the problem of paucity of satellite image data.

References

[1] Thomas Brox Olaf Ronneberger, Philipp Fischer. U-Net: Convolutional Networks for Biomedical Image Segmentation. *arXiv e-prints*, May 2015.

[2] G. Wang, W. Li, M. A. Zuluaga, R. Pratt, P. A. Patel, M. Aertsen, T. Doel, A. L. David, J. Deprest, S. Ourselin, and T. Vercauteren. Interactive medical image segmentation using deep learning with image-specific fine tuning. *IEEE Transactions on Medical Imaging*, 37(7):1562–1573, July 2018.

[3] Guotai Wang, Maria A. Zuluaga, Wenqi Li, Rosalind Pratt, Premal A. Patel, Michael Aertsen, Tom Doel, Anna L. David, Jan Deprest, Sebastien Ourselin, and Tom Vercauteren. DeepIGeoS: A Deep Interactive Geodesic Framework for Medical Image Segmentation. *arXiv e-prints*, page arXiv:1707.00652, July 2017.

[4] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. Semantic Image Segmentation with Deep Convolutional Nets and Fully Connected CRFs. *arXiv e-prints*, page arXiv:1412.7062, December 2014.

[5] Sergey Ioffe and Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *arXiv e-prints*, page arXiv:1502.03167, February 2015.

[6] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

[7] James S Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In *Advances in neural information processing systems*, pages 2546–2554, 2011.

[8] James Bergstra, Daniel Yamins, and David Daniel Cox. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. 2013.