BİL 442/542 - Deep Learning

Summer 2022

Final Project Report

Your Name and Surname:	Ahmet Selim Pehlivan
Your Student ID:	181101033
Project Title:	Suitable Agriculture Land Detection from Satellite Imaginary
Project Personnel:	Ahmet Selim Pehlivan

1. Project Description and Background

1.1 Problem Description

My problem was that shown arable agricultural lands from satellite imagery. My project has 2 main parts which are semantic segmentation for arable land with supervised learning and arable land detection with unsupervised learning.

1.2 Related Works

Specific to this arable detection with semantic segmentation problem, common sense is supervised learning with specific prepared mask: However, I don't have any labeled data. Extracting label needs to time and knowledge. If we have label for segmentation, state of art this semantic segmentation problem is U-Net and Mask R-CNN architectures.

If we want to unsupervised semantic segmentation, there are some opinions. We can use classic clustering methods or improved clustering methods like deep clustering, and we can segmentate to pixel to pixel all images. Besides, there are some new research about Unsupervised Semantic Segmentation like STEGO which is Unsupervised Semantic Segmentation by Distilling Feature Correspondences. STEGO aim to learn a segmentation that respects the induced correspondences between objects. So, we can say that there is no well-known solution to unsupervised semantic segmentation by there are several new options.

1.3 My Work

My project has two main parts, land detection with an unsupervised learning model and semantic segmentation for arable land using the supervised learning method. Firstly, I used the deep clustering method. Secondly, I extracted annotations from clustered images own, and train the U-Net model for semantic segmentation. In the end, we can show segmented arable land from satellite images.

2. Technical Work

In the first clustering part of my project, I used Deep Clustering that is an unsupervised clustering method created by Facebook AI research Team. The deep clustering method has 3 main parts. These are training encoder, generating the initial labels using means clustering, and deep clustering. In this part, I used keras API. I used combination of two dataset which are Sentinel-2 dataset and Augmented Forest Segmentation dataset which has forest and arable land images. The images are 256x256x3 dimension in dataset.

Before the training, there is pre-processing part of the dataset. In the pre-processing part of the dataset, I used data augmentation to combat overfitting. Satellite images are loaded by applying random horizontal and vertical flips, and a random rotation. These transformations leverage the fact that these different special visualizations remain true to the location's physical representation. Due to the nature of the satellite imagery only cropping, rotations, reflections, and scaling were used for data augmentation because they do not introduce distortion to objects like buildings.

Firstly, I build an encoder-decoder model for feature extraction. The encoder part of the model has pre-trained ResNet-50 architecture which was pre-trained by ImageNet. The model has a flattened layer with uses the last convolutional output from the pre-trained model which has [8*8*2048] dimensional convolutional layer. For the prior layers, we keep the BatchNorm layer that the ResNet-50 architecture uses after each convolution and prior to activation, which has an implicit regularization effect. My model has 3 fully connected parts after the convolutional part. The end of the model has the output layer consisting of 100 neurons. The output layer of the encoder is the "embedding" layer used for pseudo label extraction. The decoder part of my model was just used for training to encoder part of my model with high-dimensional images. I train the model with a train set and test set images that are the same image.

Secondly, I build a K-means clustering model to extract pseudo labels. By training the autoencoder, we have its encoder part learned to compress each image into ten floating point values. I am going to use K-Means to generate the cluster centroids, which are the 100 clusters' centers in the 100-D feature space. But we are also going to build our custom clustering layer to convert input features to cluster label probability. The probability is calculated by t-distribution. T-distribution, as same as used in the t-SNE algorithm, measures the similarity between an embedded point and a centroid. And as you might guess the clustering layer acts like K-means for clustering, and the layer's weights represent the cluster centroids which can be initialized by training a K-means.

Finally, I built a deep clustering model. The deep clustering model has soft labeling, assigning an estimated class to each of the data samples in such a way that can be redefined iteratively. The soft labeling layer which is a clustering layer is added to the end of the encoder model. The main idea behind in deep clustering method is to simultaneously learn the feature representations and cluster assignments using deep neural networks. Therefore, I used the pre-trained autoencoder and K-means model to define a new model that takes the preprocessed resnet50 model which is a trained the imagenet. So, the deep clustering model has a pre-trained input layer and clustering layer as output whose weights are predicted K-means cluster centroids.

The prepared deep clustering model is compiled with a stochastic gradient descent optimizer with 0.01 learning rate using with Kullback-Leibler loss function. After this stage, Model is training as iteratively that refines the clusters by learning from the high confidence assignments with the help of the auxiliary target distribution. The deep clustering model is trained by matching the soft assignment to the target distribution. So, in the training step, the target distribution is redefined one time in every 100 epochs. The Kullback-Leibler loss calculates divergence loss between soft assignments and auxiliary distribution.

After the training part, I predict the images with a deep clustering model. I use the t-SNE to visualize predicted clustering results in two-dimensional space with color coding of the new predicted clustering labels.

In the second part of my project, I used a VGG image annotator and extracted some labels myself. Just I use good clustered arable land images. I extract some image annotations and get an annotation json file. After that, I use a python script and save the mask as a png file. For all, I created a cv file for all annotations. Thus, I used csv file to get image and mask matches. Thereby, I train a supervised semantic segmentation model. However, I have limited available segmentation masks for training. For this reason, I use transfer learning again. I train my model using pre-trained InceptionV3 and ResNet50 which are trained with the imagenet. When building the encoder-decoder part of the model, I use U-Net architecture. I integrated the resnet50 and Inception V3 pre-trained models to U-Net.

Algorithms

t-SNE Algorithm

The t-SNE algorithm calculates a similarity measure between pairs of instances in the high dimensional space and in the low dimensional space. It then tries to optimize these two similarity measures using a cost function with 3 basic steps.

In step 1, measure similarities between points in the high dimensional space. For each data point (xi) we'll center a Gaussian distribution over that point. Then we measure the density of all points (xj) under that Gaussian distribution. Then renormalize for all points. This gives us a set of probabilities (Pij) for all points. Those probabilities are proportional to the similarities. All that means is, if data points x1 and x2 have equal values under this gaussian circle then their proportions and similarities are equal and hence you have local similarities in the structure of this high-dimensional space.

In step 2, we use a student t-distribution with one degree of freedom, which is also known as the Cauchy distribution. This gives us a second set of probabilities (Qij) in the low dimensional space. As you can see the student t-distribution has heavier tails than the normal distribution. The heavy tails allow for better modeling of far apart distances.

In the last step, we want these set of probabilities from the low-dimensional space (Qij) to reflect those of the high dimensional space (Pij) as best as possible. We want the two map structures to be similar. We measure the difference between the probability distributions of the two-dimensional spaces using Kullback-Liebler divergence (KL). KL is an asymmetrical approach that efficiently compares large Pij and Qij values. Finally, we use gradient descent to minimize our KL cost function.

U-Net Architecture

U net model which has a "U" shape. is symmetric and consists of two major parts, the left part is called the encoder part, which is constituted by the general convolutional process; the right part is decoder part, which is constituted by transposed 2d convolutional layers. The U-Net combines the location information from the down-sampling path with the contextual information in the up-sampling path to finally obtain general information combining localization and context, which is necessary to predict a good segmentation map. So, U-net gives to us more power than casual encoder-decoder models when training limited labeled data.

3. Experiments and Results

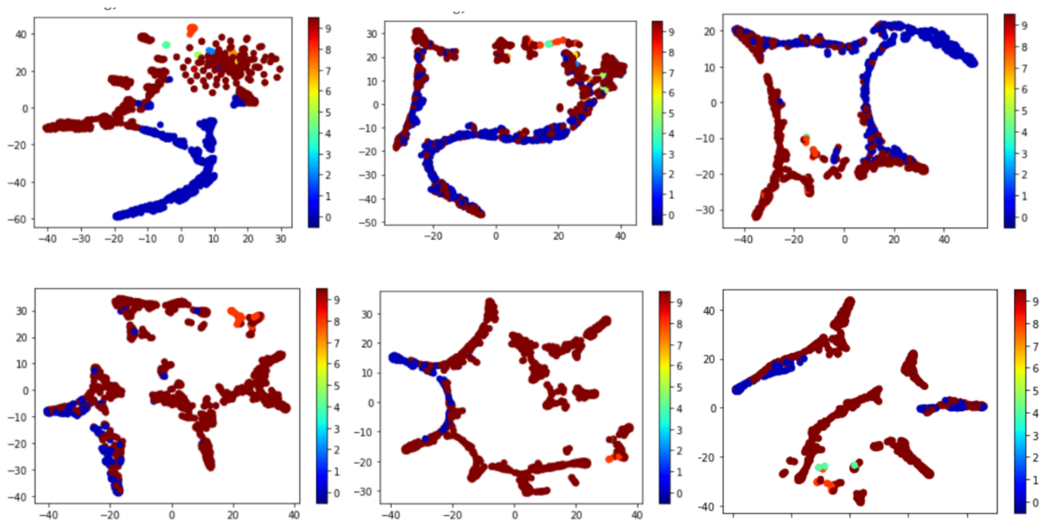
In this project, I used my laptop which has an MX330 graphic process unit, google colab environment, and Kaggle environment. I encountered many problems in the development duration. Firstly, my dataset images have 256*256*3 dimensionality. I need more memory for good training. However, my laptop, colab, and kaggle notebooks do not provide enough memory for my dataset. So, I had to train my model in small batches. As a solution, I want to use transfer learning like ResNet50 and InceptionV3, but I don't have any mask for semantic segmentation, so I don't work with supervised methods. Finally, I decided to use unsupervised methods and get proper images then create proper labels and masks.

Before the training, I use data augmentation methods like rotation, random horizontal and vertical flips, and a random rotation. In the first unsupervised part of my project, I use pre-trained ResNet50 and Vgg16, and I show that ResNet 50 give better result than VGG16. When training the encoder part with images, ResNet50 gives to us better accuracy than ResNet50. In the encoder training part, I work with 16-32 batch sizes because of a lack of memory. I use Adam Optimizer with a 0.001 learning rate. I select my hyperparameters according to common sense of deep clustering research except for batch size. I use the glorot uniform initialization in the fully connected layer.

At the end of the encoder part of the model, it has just one hidden layer and one output layer. More hidden layers increased my memory usage tremendously. Instead of using a more hidden layer. I increase the number of neurons in the hidden layer. Following the decoder part of the model has transpose and a convolutional couple which has the same kernel size, Leaky Relu activation function, and same padding. Besides, there are Batch Normalization layers after the convolutional and transpose layers. Decoder part feature maps size 512-256-128-64-3 respectively. The last convolution part of ResNet 50 has 2048 kernel image, decoder meet to this layer with 512 feature map size.

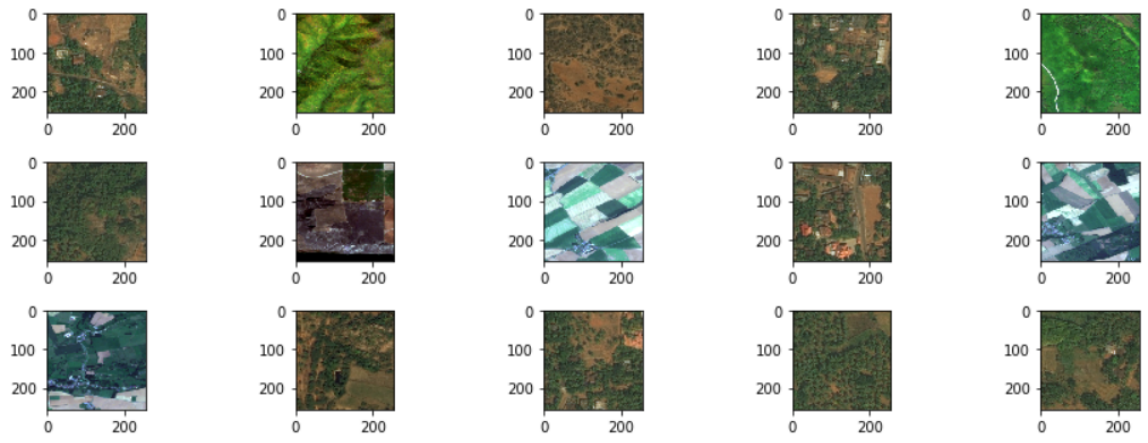
Final Deep clustering training part, I select the batch size as 16, end epoch is 1000. auxiliary target distribution is redefined every 100 epochs. End of the training epoch, I use the garbage collector library. The gc.collect() method decreases the possibility of being the out-of-the-memory. After the training. I selected arable land images from clusters, and I goes to second part of my project.

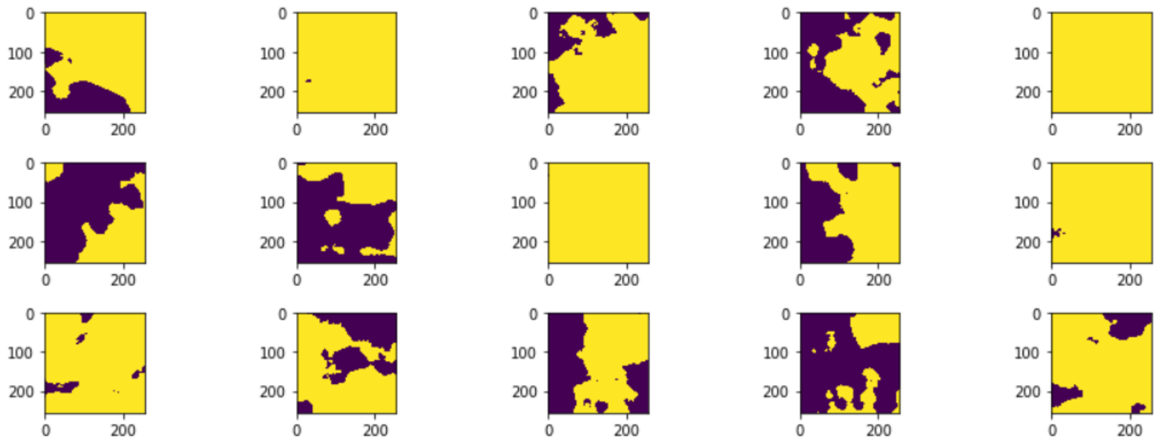
You can see visualizations of predicted clustering results in two-dimensional space 1 to 1000 epochs



In second part of my project, I use U-net architecture with pre-trained InceptionV3 model. In the U net model, the encoder part is the InceptionV3 model connected with decoder part. Following the decoder part of the model has transpose and a convolutional couple which has the different kernel size, Leaky Relu activation function, and same padding. Kernel sizes are differentiated according to the connection layers of the encoder. Besides, there are Batch Normalization layers after the convolutional and transpose layers. Decoder part feature maps size 512-256-128-64-1 respectively. I show the concatenation layer and create the model in my codebase, but I use the segmentation models library. I trained my model with batch size 16, adam optimizer and, binary cross entropy loss function. I train a U-Net model three times these are using ResNet50 and using Inception V3 and non trained model. As a result, Inception V3 is best chose of them. Because it is pre-trained and has an extremely low error rate compared with ResNet50. At the end of the training it catch 0.91 accuracy.

You can see predicted masks of U-Net model





4. Conclusions

In this project, I learned unsupervised clustering applications, methods and clustering of satellite images. I read and understand a AI research article which is Deep Clustering and applicate this article for my problem. I explore the new research about unsupervised learning. I realized that unsupervised learning of images is in its infancy. On the other hand, I learned transfer learning applications on segmentation task. I learn U-net model which is advantages and learning pattern. I used U-Net model with pre-trained models. Shortly, my observation is semantic segmentation become a hard problem, when without labels. So, clustering methods like deep clustering can be solution in nearest future.