Road Segmentation Using Fully Convolutional Networks

Rebecca Cheng, Eddie Wang, Sandra Yang
EPFL CS-433 Machine Learning Project 2

***Abstract*—The goal of this project is to build an image semantic segmentation model that classifies every patch of 16x16 pixels in sets of satellite images as road or not road. Due to the computer vision nature of the task, convolutional neural networks (CNNs) were used to solve the problem. More specifically, we use an architecture of fully convolution networks (FCNs) as they tackle the problem of semantic segmentation with great results.**

I. Introduction

Semantic segmentation in computer vision is the task of classifying an image pixel by pixel of belonging to a particular class. This problem is interesting because it has many applications such as medical imaging, recognition tasks, and photography [1]. For example, iPhone and Android ”portrait mode” on their cameras use this technique to segment the point of interest (foreground) from the background for the application to perform its blurring of the backdrop. In our task, we are given satellite images collected from Google Maps and need to classify the road areas in the image.

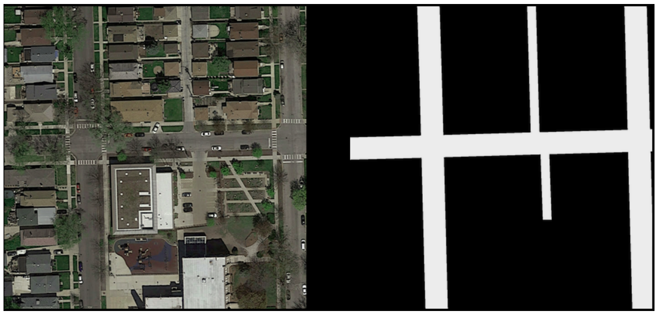

Fig. 1: example of training data

We are given 100 training examples with their associated ground truth values (as above) of size 400 x 400. The test set contains 100 images of size 608 x 608, in which we predict every 16x16 pixel patch as either a road or not. This report will describe the exploratory analysis, data pre-processing, model configuration, and evaluation techniques we used to achieve a 0.78 F1-score on CrowdAI.

II. Exploratory Analysis

Upon first inspection of the training data along with their ground truths, we notice certain complications that may arise that can cause difficulty for the machine learning model to predict accurately. Firstly, roads may be blocked by trees or cars for many patches at a time which can confuse the model to predict false negatives. Furthermore, parking lots and pedestrian walkways have similar colors resembling road pixels and thus these patches may cause the model to predict false positives. The distribution of road pixels to non-roads pixels is skewed (in relation to the dataset size of 100) as well, as around 25% of pixels are roads and 75% of pixels are non roads. Thus, models that do not take in account of overfitting may predict more pixels of the non-road class because of the somewhat skewed data. Lastly, images with diagonal roads and ground truth values are not as common as images with purely horizontal and vertical roads. Thus the model may have a hard time predicting diagonal pixels correctly, especially with a limited dataset of 100 training images.

III. Data Pre-processing

Due to the dataset being very limited in size, image augmentation is applied to increase the dataset size. The strategy we used was to rotate the images by 90°, 180°, and 270°. The original image was also flipped vertically and horizontally to produce two new images. In total, each image is transformed into five additional images which increases the dataset size by a factor of six.

Furthermore, upon inspection of preliminary model results, we found that border pixels were not classified correctly. Thus for each image, we expand the border by 32 pixels in all directions using a *mirror reflection*. Then for prediction, we run the augmented image with mirrored borders into the model and post-prediction, re-crop the photo back to its original size but with a more accurate boundary classification. All of these functions were implemented with the help of the **Pillow** Python library.

Finally, to address the issue of class imbalance, John et al [2] states in their paper of fully convolutional networks that they did not find class balancing necessary (even though they had similar class imbalance split between 75% background and 25% foreground).

IV. Model Configuration

A. Baseline Model

We commenced the project by setting up a baseline evaluation model so it can be used to compare its performance vs a fully convolutional network. The baseline evaluation model we chose was the already given 2-layered CNN model as described in the *tf_aerial_images.py* file. When trained on 600 examples the given model had a performance of around 0.67 F1-Score on CrowdAI.

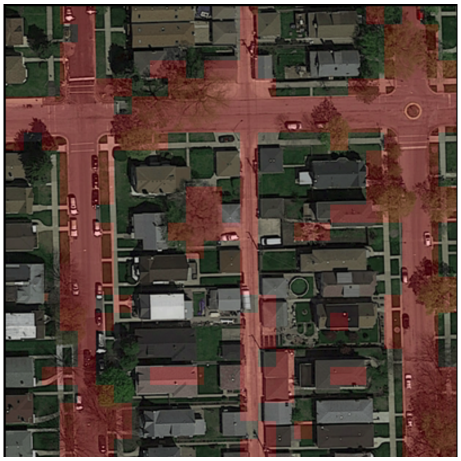

Fig. 2: Baseline CNN model output example

B. FCN Description

Typically in a traditional CNN, there are a series of convolutional layers followed by a final dense layer usually found at the end of a network. These networks also take fixed-size inputs and produce a non-spatial output, such predicting a singular class value for the image in the case of image recognition/classification. However in a fully convolutional network, there exist no dense layers in the entire architecture, hence its name of being ”fully convolutional” as the entire neural network is composed of only convolutional layers. The lack of dense layers plus the fact that there are ground truth values for every pixel also greatly increases the speed of computation [2]. Furthermore, fully convolutional networks can take in input of arbitrary size and outputs a segmented image with the same dimensions, which is favorable for our segmentation task as training and test images are shaped differently.

C. Downsampling and Upsampling

Classical CNNs downsample the input image usually through means of pooling into a smaller ”compressed” image that lacks every pixel-wise detail but extracts the most important features in the image to perform efficient image classification. This is a problem for image segmentation however, as the output of the model must be of the same dimensions of the input image. Thus we achieve a technique known as *upsampling* through the use of *inverse convolutional layers*. The method to construct these layers are trivial because it is just the reverse of regular forward and backward convolutional layers. For example, a technique called *max unpooling* is demonstrated in the Figure 3 [3].

In max pooling of the left side, the highest value element in each window is stored into the subsequent output. For max unpooling on the right side, we remember which element was the maximum from the max pooling step and artificially construct a new output with greater dimensions using the previous layer and stored max indices. The remarkable part of this architecture is that the inverse convolutional layer filters can be learned just like regular convolutional filters. By utilizing this technique, any arbitrary sized image can be fed into the neural network and consequently generate an output of the same dimensions with image classes segmented.

D. Our FCN Architecture

Our FCN adopts a similar architecture as VGG-16 net except with the last fully connected layer converted to convolution layers as well as inverse convolutional layers inserted into the model. Our full model summary is described similarly as the image below [3].

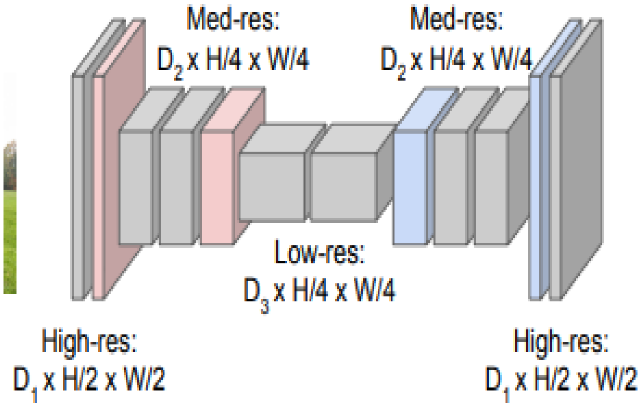

Figure 4: Similar model architecture

V. Training and evaluation

A. Training and validation

To train the baseline CNN model, we used CPU to train and k-fold cross validation to select hyperparameters, as the model is relatively simple and fast to compute. To train the FCN model, we used a NVIDIA Tesla K80 GPU on Google Cloud Platform, which took around 2 hours to train. Due to the high computational resources and time required for training,f we chose to only perform simple cross validation using a validation set within the training data (80/20 random split). However this method does have its drawbacks, namely if all the images in the training set are similar or simple (for example all having only vertical and horizontal roads). By not randomly shuffling randomly the training and validation sets as in k-fold cross validation, our model risks not having optimal performance, however the trade-off is much faster computation which we deemed well worth the sacrifice.

B. Evaluation metrics

During training we log three main evaluation metrics across the training and validation sets: accuracy, F1-score, and intersection over union (IoU). Note that for the final output value for each metric is calculated by using the mean across all training and validation examples. The three measurements are briefly described below.

1) **Accuracy** is described as the percentage of number of patches predicted correctly divided by the total number of patches. It is a simple measurement which is used to

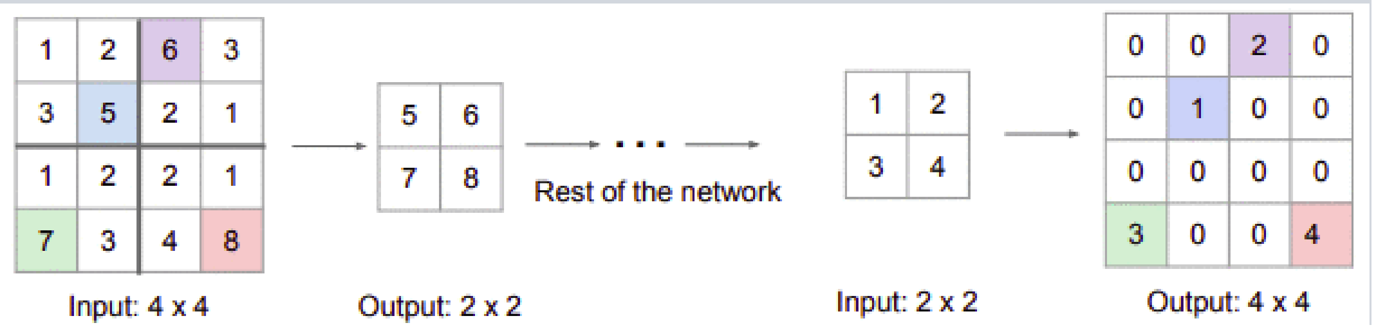

Fig. 3: Max unpooling example

serve as a baseline evaluation metric.

2) **F1-score** is described by the following equation:

$$F1 - score = \frac{2 \times P \times R}{P + R}$$

Where P is precision and R is recall. The F1-score is a good option for evaluation due to the fact that road vs non-road classes are somewhat skewed in terms of proportion. For example, since on average 75% of patches are non-road, a naive algorithm that purely predicts non-road for every patch can have around 75% accuracy. However if the naive model was assessed with the F1-score, it would not perform well, thus F1-score is a highly regarded metric in our evaluation.

3) **Intersection over union**: is described by the following equation:

$$IoU = \frac{Area\ of\ Overlap}{Area\ of\ Union}$$

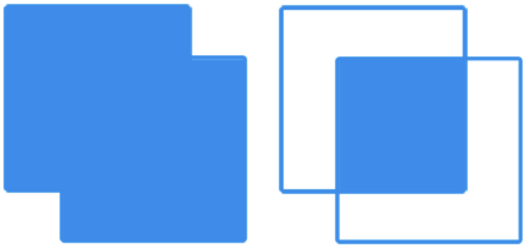

Figure 5: Depiction of overlap vs. union

This evaluation metric is very suitable for the problem of image segmentation, as overlaps and unions of predicted images can be easily calculated by referencing ground truth images. For IoU, a higher value indicates better performance.

VI. Summary

Overall, the FCN model performed relatively well at 0.78 F1-Score on CrowdAI. Upon inspection, we realize that the model has the most error when predicting roads that are covered by objects such as trees, cars, etc.

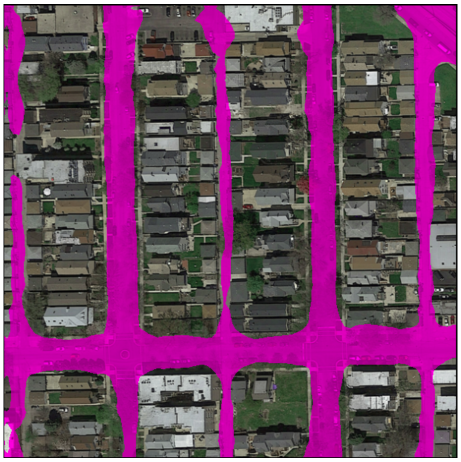

Figure 7: Example of final FCN model prediction

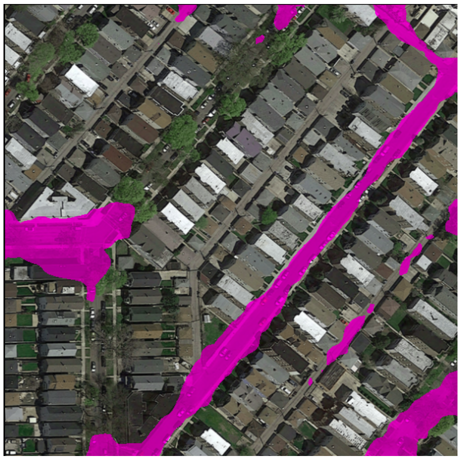

Figure 8: Tree covered roads not classified properly

This may be because the model is overfitting the data since there are not as many training examples of roads covered by trees that are supplied to the model. Solutions to this problem (next time) may include trying L1 or L2 regularization or increasing probability in dropout layers so that more neurons will be dropped during the training process. Overall, below is a graph depicting cross entropy loss and a table summarizing the significant evaluation metrics for the FCN model.

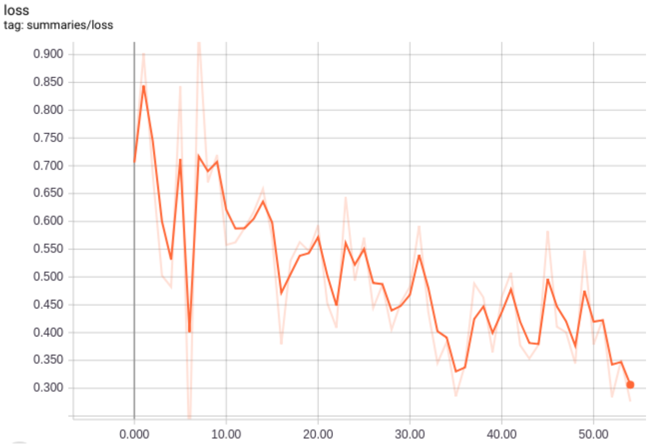

Figure 9: Cross Entropy Loss for FCN vs Iterations

FCN Model	Accuracy	F1-score	IOU
Training	0.86	0.81	0.83
Validation	0.82	0.74	0.73

TABLE I: Summarized results

REFERENCES

[1] Michel Goossens, Frank Mittelbach, and Alexander Samarin. Xiaoyong Shen, Aaron Hertzmann, Jiaya Jia, Sylvain Paris, Brian Price, Eli Shechtman, Ian Sachs *Automatic Portrait Segmentation for Image Stylization*. The Chinese University of Hong Kong, 2016.

[2] Jonathan Long, Eva Shelhamer, Trevor Darrell. *Fully Convolutional Networks for Semantic Segmentation*. UC Berkely, 2014.

[3] Fei-Fei Li, Justin Johnson, Serena Yeung, CS231n: Convolutional Neural Networks for Visual Recognition