Road Segmentation

Meet Vora, Rubin Deliallisi, Rupanshu Ganvir, Vignesh Ram
Group Name: Yahoo
Department of Computer Science, ETH Zurich, Switzerland

***Abstract*—In this work, we demonstrate that road segmentation from aerial images can be effectively carried out with an end-to-end convolutional neural network. We attempt the problem from basic approaches, inspired by computer vision approaches, and then proceed to scale the models with recent deep learning approaches. We see that fully convolutional networks outperform our computer-vision approaches and help us achieve a F1 score of 0.90519 and 0.89024 on public and overall leaderboard respectively.**

I. Introduction

Enormous amounts of overhead images are captured daily by satelites and airborne cameras. These can be leveraged not only for building digital terrestrial maps but also for generating accurate maps of transportation systems and lane based models. Data gathered is not limited to optical images and includes data from remote sensing methods such as multispectral imaging and LIDAR.

A major hurdle is automating the process of identifying different features from raw data so that the information can be used to enrich existing maps. Hence, there has been considerable interest in pixel-wise labelling of aerial images.

In this report, we restrict ourselves to the specified task. In particular, we have a hundred 400×400 RBG images from GoogleMaps with each pixel labelled as road (1) or background (0). The goal is to train a classifier which can annotate a label (road or background) to each pixel in a given image. The classifier is then evaluated on the basis of prediction accuracy measured over 16×16 patches of test images.

We include a short survey of state-of-the-art in labelling of aerial images in Section **??**, before describing our solution.

II. Related Work

Deep learning solutions have been state of the art in image analysis tasks since 2012 [**?**]. VGG nets [**?**] and ResNets [**?**] are network architectures which benefit from multiple layers stacked successively learning increasingly abstract features from input images. While ResNets can be $8\times$ deeper than VGG nets, the key idea which counters problems such as vanishing/exploding gradients associated with increasing depth is the "skip connection"

? [**?**] developed fully convolutional networks (FCN) as an extension to CNNs to perform semantic segmentation which have become the de-facto standard in semantic egmentation. Combinations of FCNs with VGG nets and ResNets have also been used to address segmentation in aerial images [**?**] [**?**] [**?**].

U-Net is another network architecture built on top of FCNs. The main characteristic of U-Net is the symmetric contracting (downsampling) and expansive (upsampling) path composed with "skip connections" which concatenate features from downsampling path to the equivalent upsampling layers. A significant advantage of U-Net is less dependence on huge amounts of data in contrast to other data hungry networks. Though originally developed for segmentation task in biomedical imaging, these have been used for automatic road detection [**?**] [**?**].

III. Models and Methods

In this section we describe data pre-processing techniques, model architectures and the post-processing methods that we apply to our model outputs.

A. Baseline

For our baseline model, we split each of our images and masks in disjoint patches of size 16×16. We then normalize each image patch by subtracting the mean and dividing by the standard deviation across our training set. To compute the patch label, we compute the mean across the mask patch and set the label to 1 (road) if the mean is bigger than 0.25 and 0 (not road) otherwise.

These patches are used to train a convolutional neural network. The exact architecture is described in **??**.

Conv(filters=32, kernel=5×5)	MaxPool(kernel=2×2)	BatchNorm
Conv(filters=64, kernel=5×5)	MaxPool(kernel=2×2)	BatchNorm
Dense(units=512, activ.=ReLU)	BatchNorm	
Dense(units=1, activ.=Sigmoid)		

Table I. Architecture of our baseline CNN model.

B. Improved Baseline

Here we borrow from the ideas presented in [**?**]. We use a much bigger patch of size 64×64 to predict the middle 16×16 patch. This gives our model more information and hence a better prediction is expected. Similarly to **??**, we split each image in disjoint patches of size 64×64, using 0 padding on the edges. Image patch normalization, mask patch extraction and label computation are identical to what we did in **??**.

To predict the labels, we use a convolutional neural network. The exact architecture is described in **??**.

Conv(filters=32, kernel=3 × 3)	MaxPool(kernel=2 × 2)	BatchNorm
Conv(filters=64, kernel=3 × 3)	MaxPool(kernel=2 × 2)	BatchNorm
Conv(filters=64, kernel=3 × 3)	MaxPool(kernel=2 × 2)	BatchNorm
Conv(filters=128, kernel=3 × 3)	MaxPool(kernel=2 × 2)	BatchNorm
Dense(units=512, activ.=ReLU)	BatchNorm	
Dense(units=1, activ.=Sigmoid)		

Table II. Architecture of our improved baseline CNN model.

After obtaining the results from the CNN we tried incorporating the continuous road structure in our prediction. We group our labels into 7×7 patches and use these groupings to refine the prediction of the middle label.

We use an SVM with a radial basis function kernel as our post processing model. This different from what was done in [**?**] where a CNN was used.

C. Fully Convolutional Neural Network (UNet)

Our best performing models in this task are fully convolutional networks, which make use of only convolutional and sampling (down & up) layers to obtain outputs with same spatial dimensions as the input. This makes FCN an ideal candidate for tasks like segmentation. First, we try UNet as introduced in [**?**]. One of the advantages of these type of networks is that they don't need enormous amounts of data to yield excellent results.

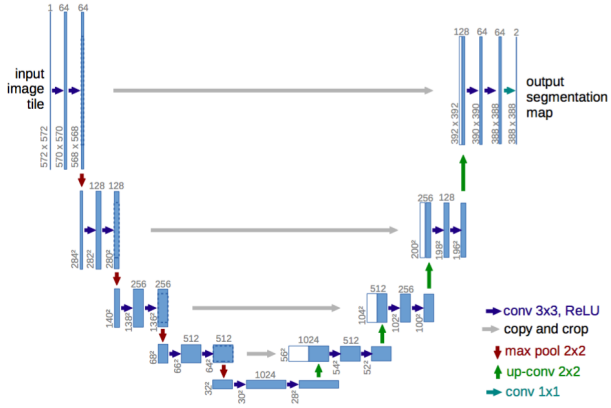

Figure 1. Auto-encoder approach using UNet. Image taken from [**?**]

Contrary to the baseline models, we can feed the entire image to the network and backpropagate the loss from the entire mask. Nevertheless, some pre-processing is necessary. We scale all our images to the $[0, 1]$ interval and normalize the data with the channel-wise mean and standard deviation of the ImageNet [**?**] dataset. This type of normalization is necessary because we use a model pre-trained on this particular dataset. The data is further augmented with random rotations, flips, crops and scaling. Finally, the entire model is trained end-to-end using binary cross-entropyloss at each pixel.

Being highly flexible, UNet allows us to try different CNN architectures in its contracting and expanding paths. Among the ones we experimented with were: VGG [**?**] and ResNet [**?**].

The masks generated by the network needed further processing in order to convert them into 16×16 patch predictions. We tried averaging the model output directly in each patch and applying a threshold or applying the threshold twice, once on the pixel level and once on the mask patch average level.

D. Fully Convolutional Neural Network (FCN)

The most successful model among the ones we experimented with was the FCN based on [**?**]. Though it is slightly older than UNet, it shares most of its characteristics.

The preprocessing part is identical to **??**. The model we use, is pre-trained on the MS-COCO dataset. Additionally, we augment the images as we observe that FCN is prone to overfitting. We apply random rotations, horizontal and vertical flipping, as well as random jittering in the color space, contrast, brightness etc.

FCN is also very flexible. We could choose from several CNN architectures in its contracting and expanding paths, pre-trained on different datasets. However, we experimented only with the ResNet variants as they proved easier to train.

E. Deeplabv3

Amongst the most recent models proposed for semantic segmentation, DeepLabV3 has shown promising results. DeepLab proposes to use Atrous Convolution, which helps us maximize the field of view for any given kernel without increasing the number of parameters. Using atrous convolution, the model is able to extract output features at different output strides during training and evaluation, which efficiently enables training with batch normalization. Additionally, the model also uses multiscale processing to create image pyramid along with parallel atrous convolutions. Finally, the model also makes use of a fully connected Conditional Random Fields to structure its predictions.

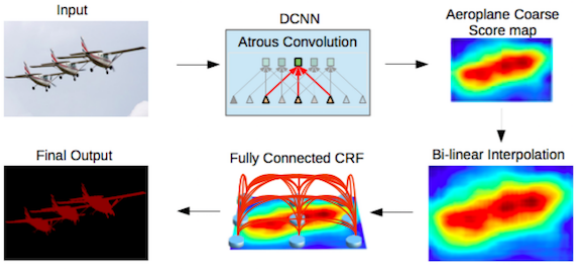

Figure 2. A simplistic overview of DeepLab architecture. Image taken from DeepLabv2. [**?**]

We use a publically available implementation of the model in PyTorch, which makes use of a ResNet backbone. We finetune an already pretrained model on MS-COCO using the augmentation techniques discussed above.

F. FCN Ensemble

In order to further improve our performance, we finally create a simple ensemble using FCN and DeepLabV3 with ResNet 101 backbones each. Similar to previous setups, we simply augment our data and train with cross entropy loss. However, we merge the final output predictions of the two models as a linear combination and normalize them, which is then trained against the target masks.

IV. Training and Results

The metric that we use to evaluate our models is the F1 score.

A. Baseline models

We trained our baseline models from scratch. After extracting the patches and corresponding labels we had 62500 training data points. We tried data augmentation procedures such as rotations and adding noise, but these procedures did not help in this case. We use Adam with a 0.001 learning rate and no decay as our optimizer. The batch size is fixed at 32.

One important property of our data is class imbalance. We have roughly 3 times more background patches than road patches. In the current literature, there are two common ways to solve this issue, modifying the training dataset or using weighted loss functions. We opted for the latter and used weighted binary cross-entropy as our loss function.

These models always achieved the best results within the first 10 epochs of training. The simpler baseline model managed to score 0.79 on the test set. The better baseline model scored 0.84 when no post-processing was applied. When SVM post-processing was applied, we achieved a slightly increased score of 0.85. A summary of these finding can be found in **??**.

Model	F1 score
Simple Baseline (no augmentation)	0.79
Simple Baseline (with augmentation)	0.79
Improved Baseline (no post-processing)	0.84
Improved Baseline (SVM post-processing)	0.85

Table III. F1 score of the various baseline models.

B. Fully convolutional neural networks

The models used for this part are pre-trained on the ImageNet & MS-COCO dataset. We attempted to train VGG based models from scratch, but this resulted in significantly lower scores. For this type of models we have only 100 images available so data augmentation was essential. The optimizer choice was Adam with a learning rate of 0.001 and no weight decay. The batch size is fixed at 16 for UNet and only 2 for the FCN base models (because of GPU memory limitations).

As we did in **??**, we use the loss function to combat class imbalance. We experimented with weighted/unweighted binary cross entropy and the Jaccard loss. However, we found that the type of loss did not affect the prediction quality. Here we report only the results obtained by using binary cross-entropy.

We train these models for 125 epochs each. The UNet (VGG) model trained from-scratch managed to score 0.85 on the test set and 0.87 when using a pre-trained model. When using a pre-trained UNet (ResNet) model we were able to achieve an F1 score of 0.89. Our FCN (ResNet) pre-trained on MS-COCO achieved a score of 0.904 on the test set. We achieve similar scores with DeepLabv3 (0.902) and ensemble (0.904) as well. A summary of these finding can be found in **??**. Surprisingly, we see that using Deeplabv3 in our setup doesn't improve our score. We hypothesize that complexity of the model (in comparision to FCN) is a factor responsible for the slight underperformance. With limited data, we expect simpler models to train easily, which is not the case with Deeplabv3.

We also see that our ensemble model slightly outperforms DeepLabV3 but again slightly underperforms in comparision to our FCN implementation. We again attribute this to the increased complexity of the model.

The variant of ResNet used throughout was the 101 layer one. We experiment with 34 or 50 layer networks, but found that the performance gain by using deeper networks was worth the extra training time.

Model	Backbone	Pretrained	F1 score
Unet	VGG	No	0.843
Unet	VGG	Yes	0.871
Unet	ResNet	Yes	0.892
FCN	ResNet	Yes	0.905
DeepLabv3	ResNet	Yes	0.902
Ensemble (FCN + DeepLabv3)	ResNet	Yes	0.904

Table IV. F1 score of the various fully convolutional networks.

V. Discussion

The baseline CNN model is by far the worst model for this task. The 16×16 patches used with this architecture contain far too little information to make an educated guess. This is true even for the trained human eye. In **??** its clear that this model suffers from an abundance of false positives and false negatives.

Using 64×64 patches as we did in our second baseline model helped a tremendous amount. The larger patch size provides the model more context. Post-processing the input, whether with an SVN or a CNN helped but not significantly. We noticed that improvements of the initial predicting model

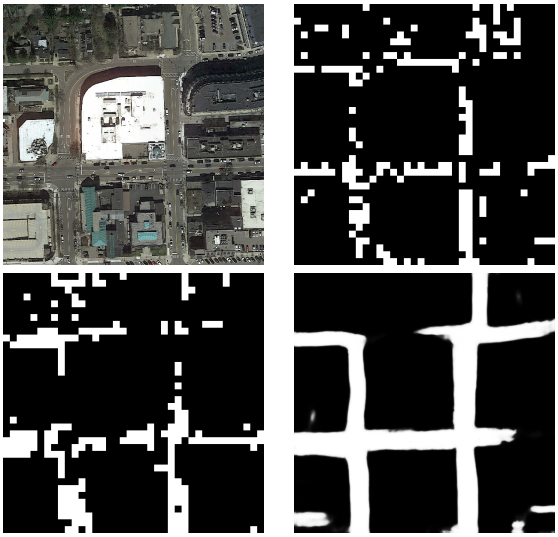

Figure 3. Top left original image. Top right baseline prediction. Bottom left better baseline prediction. Bottom right FCN prediction.

impacted the performance of the post-processing model, but not vice-versa. In **??** the results are slightly better than the baseline model. There is more structure to the data and less false positives. However, there are big structural gaps and a large amount of false negatives.

Fully convolutional architectures proved to be much more powerful than traditional CNN approaches. Even though we are not directly optimizing patch label prediction, better per pixel label prediction led to a better patch label prediction. As for the network type, Unet or FCN, we see very similar results at the high end and further data is needed to make a thorough comparison. The results of this model are shown in **??**.

Pre-trained networks on much bigger datasets, such as ImageNet, significantly improved performance. These networks allowed us to use much deeper architectures, like the 101 layer ResNet, which would have been impossible to train with our tiny dataset.

VI. Summary

In conclusion, we presented new algorithms to extract roads from aerial images and compared them to existing baselines. We use 2 different fully convolutional architectures, Unet and FCN, which have delivered state of the art results in several semantic segmentation tasks.

Our models handily outperform the baselines and achieve an F1 score over 0.9. We showed that pre-trained models, with possibly unrelated datasets, yield excellent results in the road extraction task.

In the future, we would like to experiment with other types of fully convolutional architectures. The models we used are fairly outdated and current state of the art models will almost certainly produce better results.