Predicting vehicle angle using camera images

Tian Sang, Zhaoxi Wang

11/30/2019

1 Summary:

In this project, we will focus on predicting vehicle position & pose information from a real-world traffic image (Figure 1). That is, given this image, we will first locate all vehicles in this image, then output poses information including x, y, z, yaw, pitch, and roll.

Figure 1

This research is lead by Baidu Autonomous Driving Car. Self-driving cars have come a long way in recent years, but they're still not flawless. Consumers and lawmakers remain wary of adoption, in part because of doubts about vehicles' ability to accurately perceive objects in traffic.

The goal of this project is that by making an accurate estimation of cars' pose information, we'll help improve computer vision by adding useful features. That, in turn, will bring autonomous vehicles a big step closer to widespread adoption, so they can help reduce the environmental impact of our growing societies.

2 Dataset:

Our dataset consists of 4 parts: first is 4000 traffic images just like Figure 1 with a total size of 3 GB; the second is a big table of pose information (x, y, z, yaw, pitch, roll) of every vehicle in every image, which will be our training label; third is the camera intrinsic matrix, which is about camera angle settings and can be used to project 3D points to 2D later; the last one is the car model information, which store car's dimension into a JSON file.

This dataset is provided by Baidu. Traffic images are taken from the streets in Beijing with masks on vehicles' plates.

3 Data preprocessing:

3.1 3D to 2D

One of our challenges is to understand the x, y, z coordinates. It's very confusing to apply 3D coordinates on a 2D image. We assume the x, y, z coordinates are in the 3D world, with the camera as the center. So to locate vehicles on the image, we need to project 3D points to 2D.

The solution is to apply the camera intrinsic. Figure 2 shows that a camera is a mapping between the 3D world and a 2D image. So by applying the camera intrinsic matrix (equation 1), we can get 2D coordinates on the image taken by this camera.

$$\underset{\text{2D image point}}{\boldsymbol{x}} = \underset{\text{camera matrix}}{\mathbf{P}} \underset{\text{3D world point}}{\mathbf{X}}$$

Equation 1

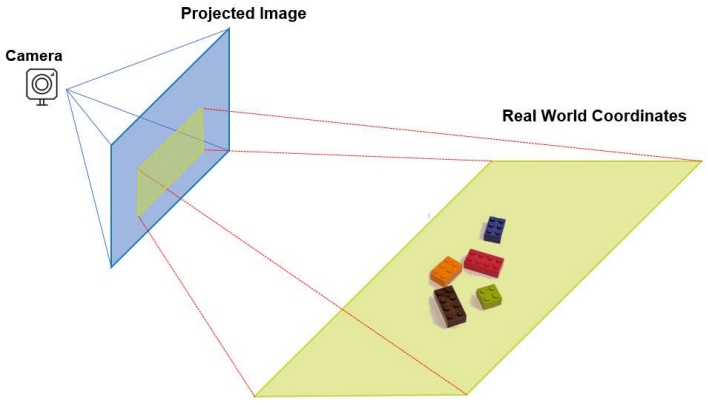


Figure 2

After the projection, now we are able to locate vehicles' positions on the image (red dots in Figure 3).

Figure 3

3.2 Bounding box

The bounding box is a frame around an object. In 2D it can be a rectangular and in 3D it can be a cuboid. The reason we construct bounding boxes for all vehicles is that we will use CenterNet as our model. It will output the bounding box for an object. So by visualizing the bounding box for the ground truth, we can compare with our model's to see the performance.

The idea is that first, we construct a 3D bounding box for all vehicles. We can use the car model information to calculate an average x, y, z dimension for all vehicles (visualized in Figure 4), then construct a cuboid around the vehicle's position point (red dots in Figure 2) with the size of 2x, 2y, 2z. Now we know the coordinates of all 8 points of this cuboid. So we apply the camera transformation again. Finally, we can get the bounding box on the 2D image (Figure 5).

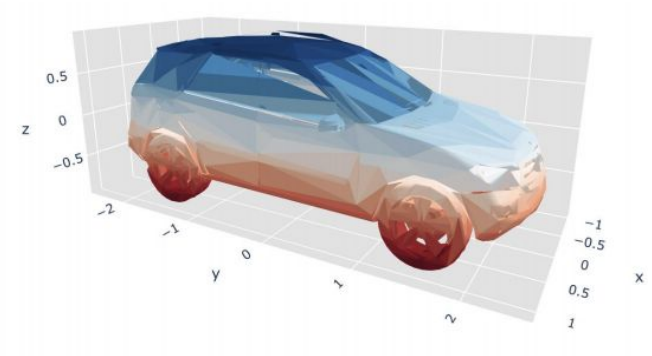


Figure 4

Figure 5

Noticed that in Figure 5, bounding boxes are all rectangular. That is because if we project all 8 points to 2D and draw lines between them, the result is really messy. Instead, we just project 4 points in the bottom to 2D then draw lines between them. Now the visualization looks clearer than before and is easier to identify.

3.3 Data Augmentation

Overfitting is a common problem for deep neural networks. Neural networks are extremely powerful in learning graphical features; however, they are often overparameterized given the sizes of common datasets, i.e. when we don't have enough images, a model would learn the "noise" within the dataset instead of the actual "signal". They can memorize unintended properties of the dataset instead of learning meaningful, general information about the world. As a result, overfit networks fail to yield useful results when given new, real-world data.

Compared to well-known benchmark image dataset, for example COCO dataset which contains 164k images, HOTELS-50K which has 50k images, and etc, our dataset is actually pretty small. Thus,in order to address overfitting, we can "augment" our training data. Common methods for augmenting visual data include randomly flipping images horizontally (flip), shifting their hues (shift) or zooming at random levels (zoom).

For our project we first randomly select 50% images from our initial 4 thousands training dataset, then apply one of flipping, shifting and zooming augmentation. We do this for all three augmentation techniques which bring us additional 6 thousands "generated" new image.

	5-layer Unet	6-layer Unet with cropping(image embedding)	34-layer resnet
Test Precision on 4k image	56%	61%	63%
Test Precision on 10k image	63%	64%	67%

Figure 6 above are experiments we run to compare model performance on original dataset versus augmented dataset, we see a simple data augmentation could increase model performance for neural networks on a relatively small dataset. We didn't exhaustively run experiments on data augmentation to find the best approach; the above test is simply to show data augmentation could benefit neural network's performance.

4 Model:

Computer vision is an interdisciplinary scientific field that deals with how computers can be made to gain high-level understanding from digital images. From the perspective of engineering, it seeks to automate tasks that the human visual system can do, for example objection detection, image classification, object segmentation are some classic computer vision tasks. Convolutional neural networks are powerful methods in learning graphic features as it in some sense mimic human's eye in learning low-level features (edgers, colors, gradients, etc) to high level features[1]. For our project we would mainly focus on implementing various classic CNN architectures including Unet and ResNet. Our prediction variable is a 1*7 vector which contains [car model, x, y, z, yaw, pitch, row] at each image pixel.

We would measure our model performance mainly focused on [x,y,z] by precision score as we can simplify it as a semantic segmentation problem. For yaw pitch row, we can measure it by distance, namely average residual error.

4.1 Baseline Model - Unet

The typical use of convolutional networks is on classification tasks, where the output to an image is a single class label. However, in many visual tasks, especially in the application of auto-driving, the desired output should include both object detection and object localization, i.e., a class label is supposed to be assigned to each pixel.

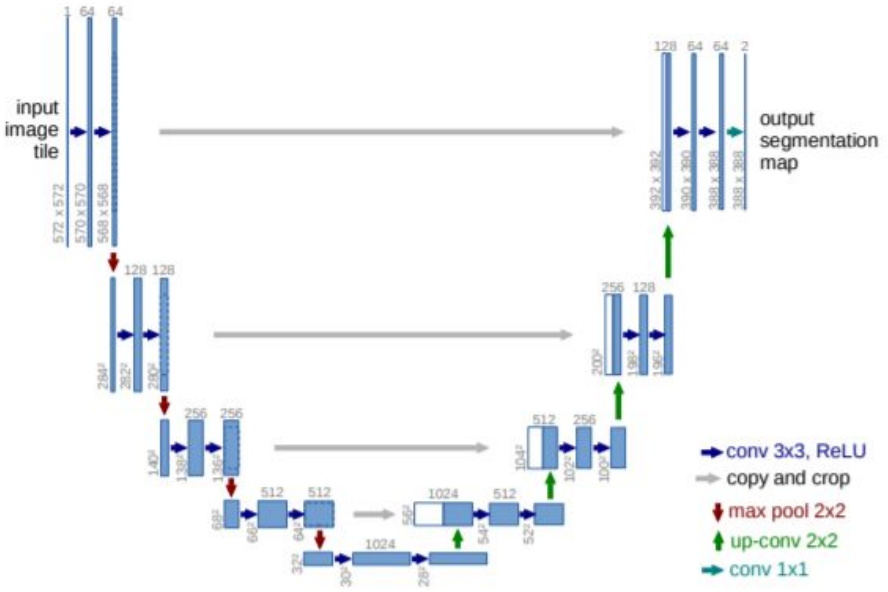


figure 7

Unet, as shown in figure7, is a network architect that apply both down-sampling convolution and upsampling transposed convolution to obtain both the “what” and “where” information. [2]. The left hand side of the network are combination of convolution layers, cropping layers(image embedding) and max pooling layers to reduce a high resolution image into low resolution image to learn the “what” information, while on the right hand side it uses up-convolution and up-sampling layers to reconstruct a high resolution image from the learned low-resolution image to obtain the “where” information.

For our project, due to equipment limitations, we implemented a relatively shallow 5-layer Unet and followed the paper’s setting: cross-entropy as loss function and use Adam as gradient leaning optimizer. The training process takes about 10 hours to converge and we get a test average precision of 63%.

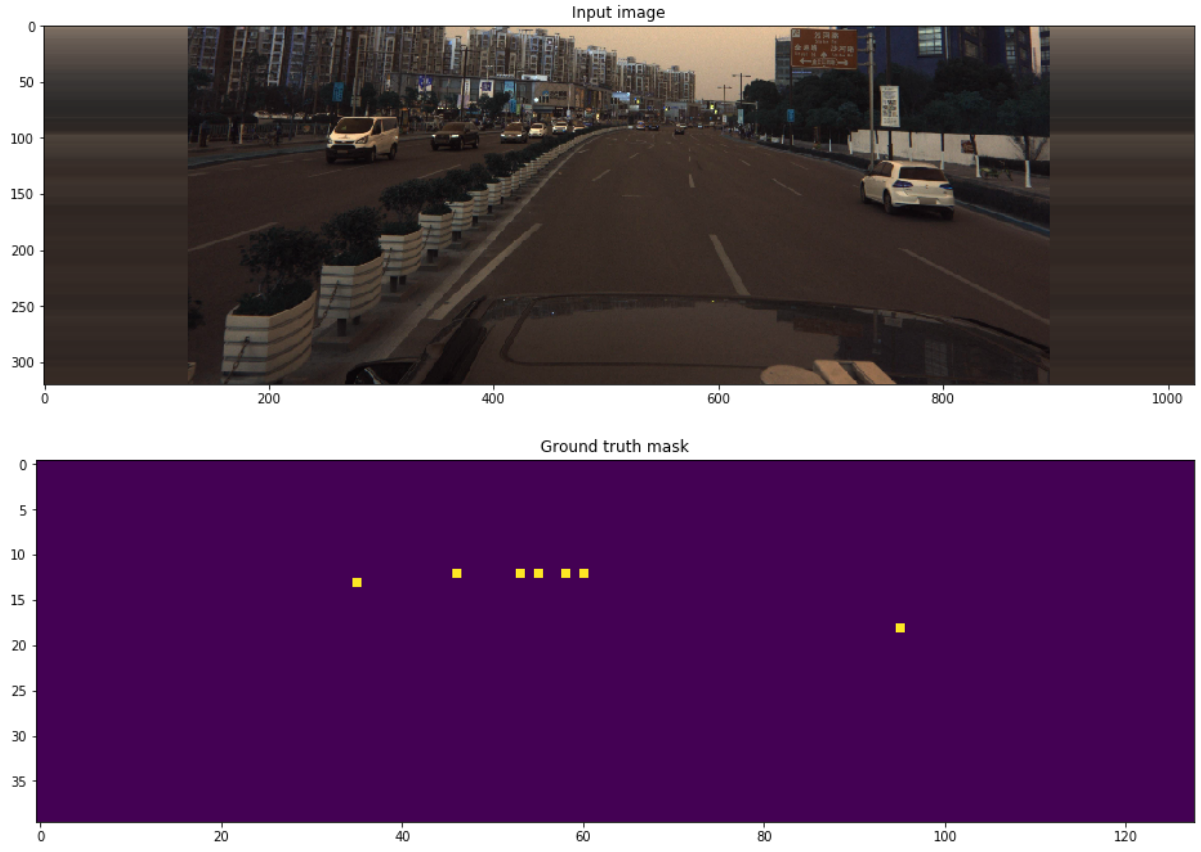


Figure 8

Figure 8 is a visualization of a test image, here we are showing the real input image and a mask of the [x,y,z] coordinates. Ideally our prediction pixel map would reflect the masked image.

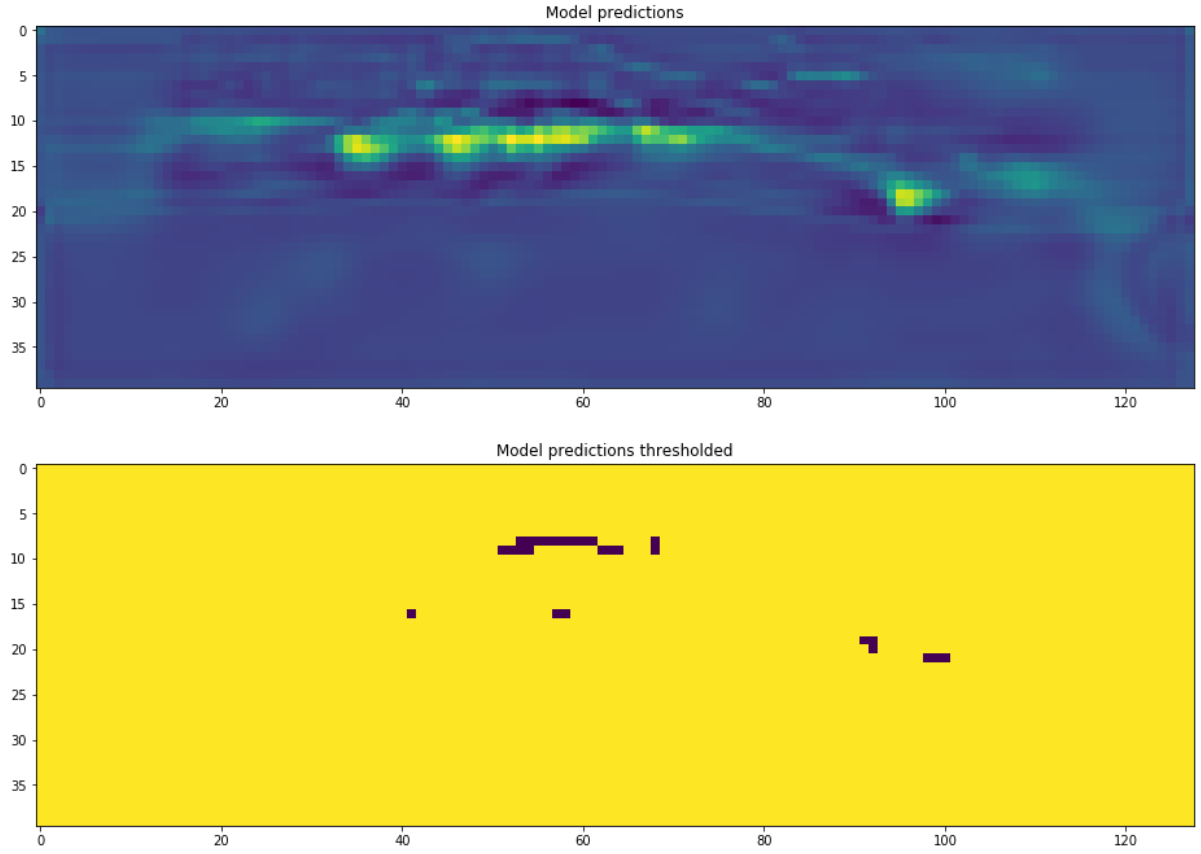
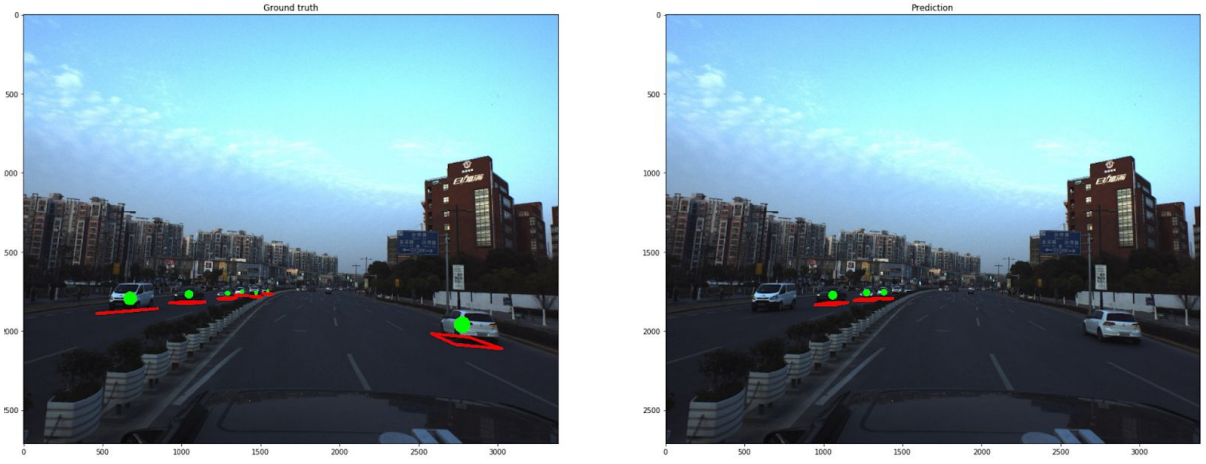


Figure 9

Figure 9 is our prediction. As we are learning the image on pixel level, we have one prediction vector on each pixel. We make a “threshold” that says only when the nearby 3*3 pixels are all predicted as positive(in our case predicted as “there is a car”), we flag the pixel as a positive prediction. Below in figure 10 is a visualization, we see among the 6 cares in the image, we only detect 3 of them correctly. (we are not drawing those miss localized boxes in the image to make it clean)



4.2 ResNet

As the test precision in not high in implementing the 5-layer Unet, we decided to learn the images on a deeper network. However, as we stack more convolution layers on the network, the training process is taking more and more time, which forces us to make additional adjustment beyond simply stacking network layers.

Since AlexNet, state-of-the-art CNN architecture is going deeper and deeper. While AlexNet and Unet both had only 5 convolutional layers, the VGG network [3] and GoogleNet (codenamed Inception_v1) [4] had 19 and 22 layers respectively.

However, increasing network depth does not work by simply stacking layers together like we initially tried. Deep networks are hard to train because of the notorious gradient vanishing problem — as the gradient is back-propagated to earlier layers, repeated multiplication may make the gradient infinitely small. As a result, as the network goes deeper, its learning power gets stuck - or even starts to degrade because the gradient cannot be passed by the propagation chain. ResNet [5] suggested identity shortcut connection" that skips one or more layers if the gradient barely change(no new useful information get learned), as shown in the following figure 10:

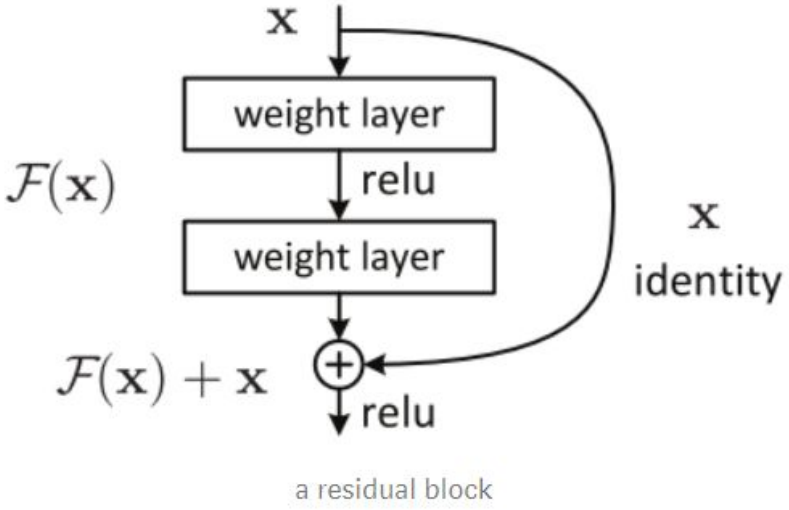
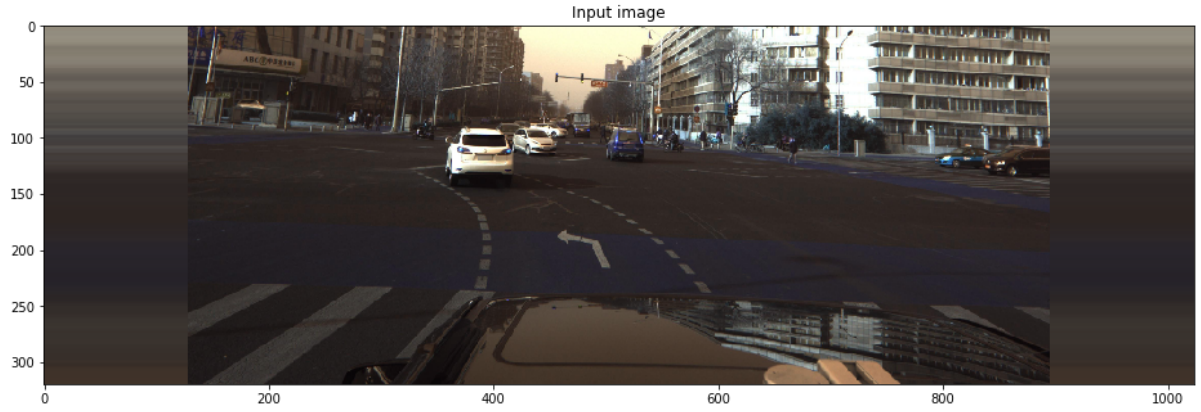


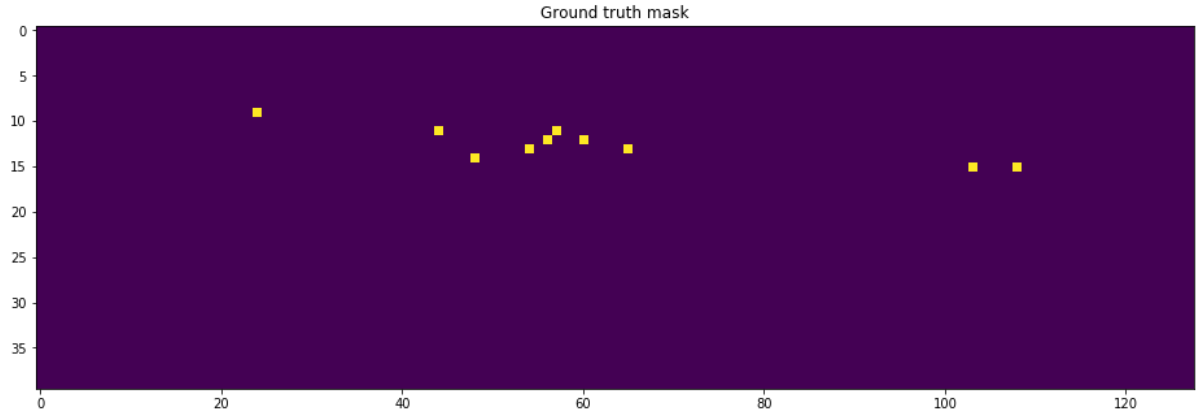
Figure 10

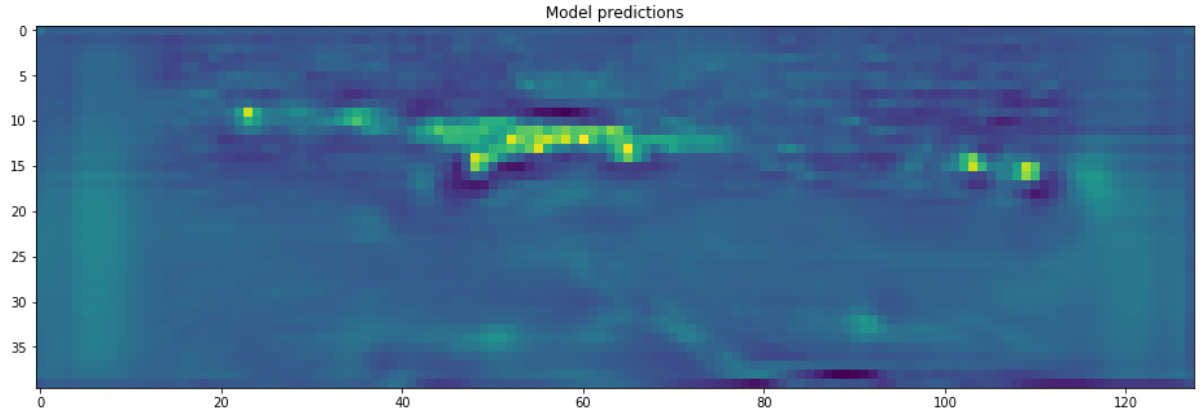
The author argues that stacking layers shouldn't degrade the network performance, because we could simply stack identity mappings (layer that doesn't do anything) upon the current network, and the resulting architecture would perform the same. This indicates that the deeper model should not produce a training error higher than its shallower counterparts. They hypothesize that letting the stacked layers fit a residual mapping is easier than letting them directly fit the desired underlying mapping. And the residual block above explicitly allows it to do precisely that.

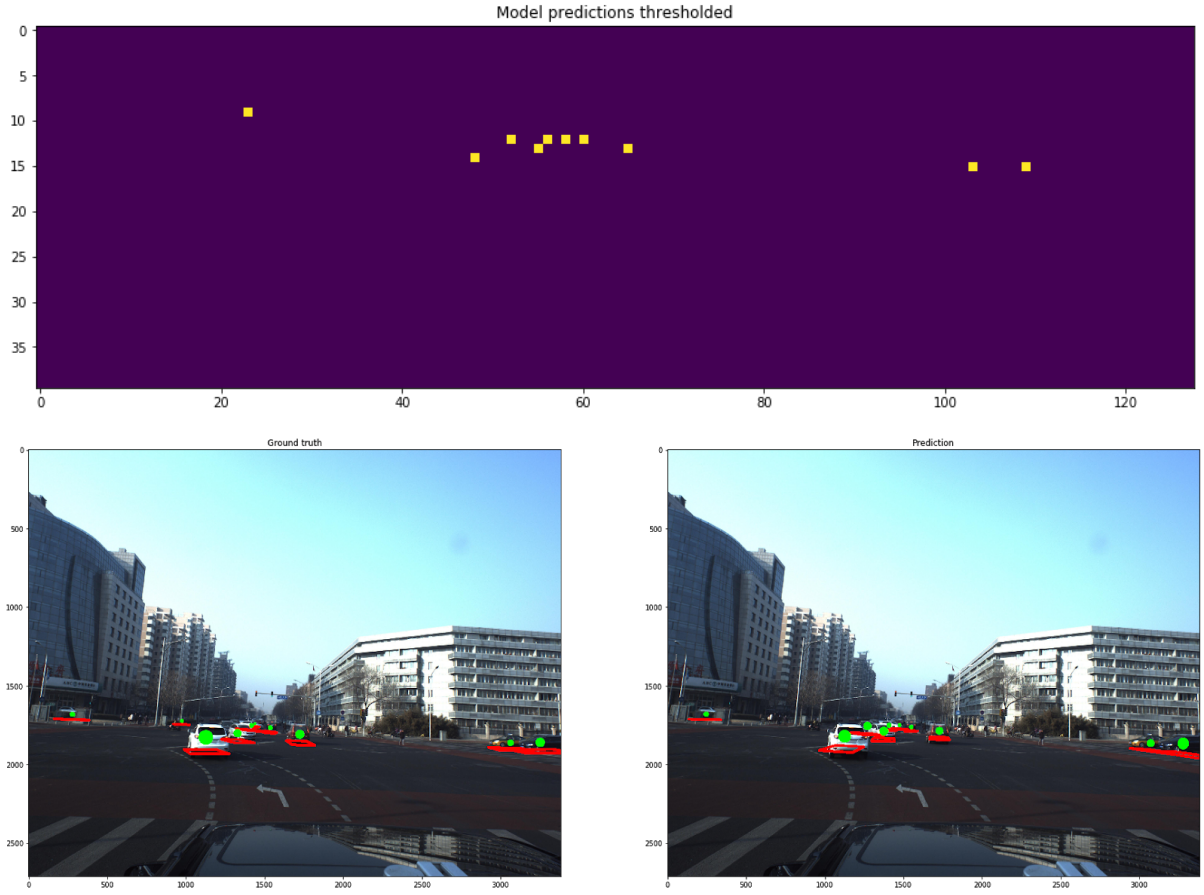
Followed the paper, we have trained a 34-layer CNN network by stacking convolutional layers with residual blocks; loss function and optimizer are the same as our implement of

Unet, namely cross entropy and Adam. The training process took 30 hours on our personal PC and still doesn't converge. We took an early stop at 30 hour and the test precision is 67%, which is already better than the previous 5-layer Unet. This suggests increase network complexity could boost learning power and our project model could get better performance by investing more computational power.









5. Discussion and Future Works:

From the project we have a chance to learn some computer vision applications, including 2D-3D transformation, image augmentation, CNN and etc. During the process of the project, we find that computational power becomes on of our major constraints in testing different network architecture. Being specific, due to the nature of deep neural network's heavy parameterizing and gradient propagation in training, we need lots of RAM and CPU/GPU power to train the model, especially for today's high-resolution images(which means more pixels and thus higher data dimensions).

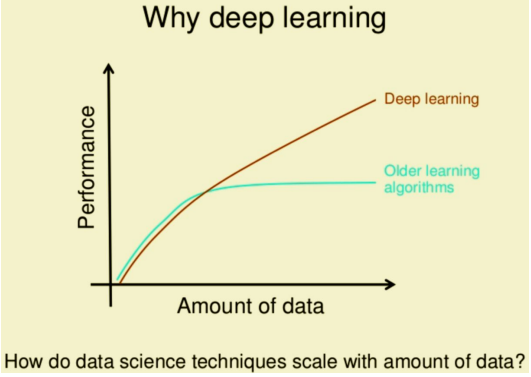


Figure 11

Figure 11 illustrates that as we get more data (in our case more images, and some image augmentation) and more computational power, we are almost guaranteed to get better model performance. However, in reality we cannot afford unlimited machines and training times, and thus we feel we haven't really exposed all those images and could get better model performance. There is indeed a tradeoff between model complexity, model performance and training time, and we might have to make some compromise based on different environments, for example in industry we want models to be iterated fast while in academic we might want to pursue the best model performance.

For future work, besides reading new neural network papers, modifying network architecture and tuning parameters, we are also very interested in learning more about how self-driving is getting evolved in reality. That is, industrial wise, how machine learning and deep learning techniques are really being applied; what are some main challenges in self-driving, and maybe guess when we can see the real level-5 autonomy self-driving.

Reference:

[1]Sumit Saha, A Comprehensive Guide to Convolutional Neural Networks
https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

[2] Olaf Ronneberger, Philipp Fischer, and Thomas Brox
U-Net: Convolutional Networks for Biomedical Image Segmentation, 2015

[3]. K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. 2014.

[4]. C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich.
Going deeper with convolutions. pages 1–9,2015.

[5] Kaiming He Xiangyu Zhang Shaoqing Ren Jian Sun,
Deep Residual Learning for Image Recognition, 2015