Progress Report 1

groveh

February 2019

Problem Description

In order to adequately train a network, a significant amount of data is required that covers all cases that can be seen outside the training set. Even with a large dataset to train on, it may only contain a limited number of real world unforeseen cases. This is the need for data augmentation, which is where the training set is varied in a way to cover cases not yet present in the dataset to increase the number of examples trained on. In the case of orientation estimation, this includes rotation and possibly translation of the examples present in the dataset.

Problem Approach

The initial approach taken was to manually iterate through the data and generate various rotations on each image and save them as an addition to the current dataset. This is an ambiguous task because it isn't clear how many variations are needed on each example to ensure that the training set has covered enough cases. In order to avoid adding added black areas around the edges of the data, it was necessary to implement a maximally inscribed rectangle algorithm to "zoom in" on the images and avoid background added to the edges of an image when it is rotated.

Solution

The solution to this memory intensive approach was to manipulate images at runtime while they were entering the network in batches. This is done through Pytorch transformations for each image grabbed from the loaded training set, therefore these random rotations make up for real world cases outside the known data set. This solution also has no need to create and save some unknown number of optimal variations of each image because each epoch through the data creates a new rotated version of each example. This way, the number of variations is proportional to number of epochs, also creating a regularizing effect as the training set changes each time.

Challenges Faced

Initial challenges include an unfamiliarity with Pytorch syntax and best practices. Syntax of loading and manipulating it was easy, however saving and using

angles as image labels came slower. This was due to the fact that Pytorch transformations are not typically the source of labels, and the values are therefore hidden. This was solved by creating a custom data loader that applied regular transformations in addition to a custom rotate transform that then returns the angle rotation as the image label.

Results

The results are promising, as the manipulation at runtime does not require additional memory usage like saving variations manually, and it means that the probability that the network sees the same example twice is exceedingly low. This means that the longer the network trains, it will theoretically become more accurate (to the point where the probability that numerous of the same examples are trained on).

Future Work

To continue this process further, it would be beneficial to add more variations such as random translations or crops to further vary the data. The main goal of this project, however, is rotation inferencing so different transformations may not help the training process if they aren't related to estimating angles other than to expand the dataset and add regularization.