CSC490 PartB

Chi Zhang, Artur Kuramshin

March 2022

1 Introduction

In the real world, precise object detection could improve the security of self-driving car. In part A project, there is still room for the average precision to be improved. We changed the loss function and gaussian kernel type to generate the target heatmap and predicted heatmap. Recently, focal loss function [1][2] made great achievements in the field of object detection. We evaluated focal loss and alpha-balanced focal loss for heatmap building in the application of object detection, as well as its influence on average precision of object detection. In addition, we explored to applying gaussian kernels indicating the size and heading of cars for each car's heatmap representation.

2 Related work

2.1 loss function

We explored to apply focal loss function and α-balanced focal loss function to replace MSE loss function in the process of heatmap building

2.1.1 focal Loss

$$FL(p, y, \gamma) = -y(1-p)^{\gamma} log(p) - (1-y)p^{\gamma} log(1-p)$$

$$\frac{\partial FL}{\partial p} = y\gamma(1-p)^{\gamma-1} log(p) - y(1-p)\gamma\frac{1}{p} - (1-y)\gamma p^{y-1} log(1-p) + (1-y)p^{\gamma}\frac{1}{1-p}$$

In theory, as the γ increase, model will take more care of hard negative samples. Next, do a simple test. Assume two points (p_1, y_1) are (0.4, 1) and (0.1, 1). When $\gamma = 0$, the $\frac{\partial FL}{\partial p}$ is 1.66 and 1.11. When γ increases up to 2, $\frac{\partial FL}{\partial p}$ is 0.67 and 0.032 respectively. Obviously, the gradient ratio between the samples difficulty to be classified (p = 0.4) and those easy to be classified (p=0.1) sharply increases, which makes the model take more attention to hard negative samples

2.1.2 α-balanced Focal Loss

$$\alpha - FL(p, \alpha, \gamma) = -\alpha y(1-p)^{\gamma} log(p) - (1-\alpha)(1-y)p^{\gamma} log(1-p)$$

In some samples, the number of negative and positive samples might be imbalanced. α is added to help balance the bias caused by such case, which helps improve the performance of the model.

2.2 Generalized Gaussian Filter

Let us define the Multivariate Gaussian filter as:

$$\begin{aligned} G(\mathbf{x}, \mu) &= \frac{exp(-\frac{1}{2}(\mathbf{x}-\mu)^T \Sigma^{-1}(\mathbf{x}-\mu))}{\sqrt{(2\pi)^k |\Sigma|}} \\ &\propto exp(-\frac{1}{2}(\mathbf{x}-\mu)^T \Sigma^{-1}(\mathbf{x}-\mu)) \end{aligned}$$

Where $\mathbf{x} = [x, y]^T$, $\mu = [\mu_x, \mu_y]^T$ and k is the number of variables in x

An isotropic Gaussian kernel is symmetrical and has a covariance matrix that can be represented as $\Sigma = \sigma^2 I$. The covariance matrix dictates the shape of our Gaussian kernel. Our target detections differ in bounding box size and yaw, so we need to match our Gaussian kernel according to those parameters.

Since cars are not symmetric, we can define an anisotropic Gaussian filter according to detection shape using a covariance matrix as follows:

$$\Sigma = \begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix}$$

Let $\texttt{scale} = \frac{\texttt{x\_size}^2 + \texttt{y\_size}^2}{\eta}$ where `x_size` and `y_size` are the respective dimensions of a detection bounding box and η is the heatmap norm scale. We will define a and b as follows: $a = \frac{\texttt{x\_size}}{\texttt{y\_size}} \cdot \texttt{scale}$ and $b = \texttt{scale}$.

One problem with this, is that all of our kernels will be scaled in the same direction, regardless of their yaw direction. To better match our detection targets, we will now take our anisotropic Gaussian kernel and will rotate it to match the yaw of the label. We will use a rotation matrix define as follows (where θ is the detection yaw angle):

$$\mathbf{T} = \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix}$$

We will now define our new rotate anisotropic Gaussian kernel as:

$$
\begin{aligned}
E(\mathbf{x}, \mu) &= G(\mathbf{T}\mathbf{x}, \mathbf{T}\mu) \\
&= \frac{exp(-\frac{1}{2}(\mathbf{T}\mathbf{x} - \mathbf{T}\mu)^T \Sigma^{-1}(\mathbf{T}\mathbf{x} - \mathbf{T}\mu))}{\sqrt{(2\pi)^k |\Sigma|}} \\
&\propto exp(-\frac{1}{2}(\mathbf{T}\mathbf{x} - \mathbf{T}\mu)^T \Sigma^{-1}(\mathbf{T}\mathbf{x} - \mathbf{T}\mu)) \\
&= exp(-\frac{1}{2}(\mathbf{x} - \mu)^T \mathbf{T}^T \Sigma^{-1} \mathbf{T}(\mathbf{x} - \mu)) \\
&= exp(-\frac{1}{2}(\mathbf{x} - \mu)^T \hat{\Sigma}^{-1}(\mathbf{x} - \mu)) \qquad\qquad (\hat{\Sigma}^{-1} = \mathbf{T}^T \Sigma^{-1} \mathbf{T})
\end{aligned}
$$

We see that after the rotation, we simply get a new Gaussian kernel.

3 Methodology

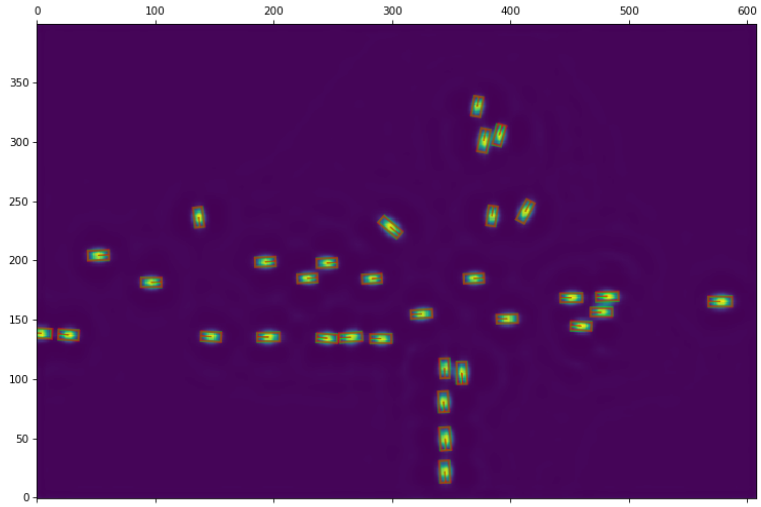
Based on the project in part A, we add two more arguments for the command-line *–loss_func=* and *–kernel=* to config loss function and kernel for train and overfit . For more details, please check README file.

- 1. Firstly, we tune the hyperparameters respectively inside MSE, Focal and α-balanced loss functions with the isotropic gaussian kernel, and then find the optimized loss hyperparameters to applying into the following step.
- 2. Then, apply isotropic, anisotropic and rotated gaussian kernel respectively on 3 kinds of loss functions mentioned above.

4 Evaluation

4.1 Heatmap Visualizing

We add the label bounding boxes into the target heatmap picture (in the case of rotated kernel). The picture below shows the heading and size of target labels **matches** bounding box better than an anisotropic Gaussian kernel, which indicates the correctness of heatmap representation with rotated kernel .



4.2 Loss Function Hyperparameter Tuning

We apply average precision, with threshold 2, 4, 8, 16 and mean average precision, to evaluate different models.

4.2.1 Focal Loss Function

γ	AP_2	AP_4	AP_8	AP_{16}	mean
1	0.22	0.37	0.40	0.40	0.34
2	0.23	0.37	0.40	0.41	0.35
3	0.23	0.36	0.38	0.38	0.34
5	0.22	0.31	0.33	0.33	0.30

4.2.2 alpha-balanced Focal Loss Function

α	γ	AP_2	AP_4	AP_8	AP_{16}	mean
0.75	0	0.25	0.41	0.48	0.49	0.41
0.75	0.1	0.28	0.45	0.51	0.53	0.44
0.75	0.2	0.26	0.41	0.45	0.46	0.39
0.25	0.75	0.17	0.25	0.26	0.29	0.24
0.5	0.5	0.25	0.40	0.43	0.44	0.38

For alpha-balanced focal loss, hyperparameters $\alpha = 0.75$ and $\gamma = 0.1$ where used. For focal loss, the value of $\gamma = 2$ was used.

4.3 Applying various Gaussian kernels

We apply average precision, with threshold 2, 4, 8, 16 and mean average precision, to evaluate different models.

loss	kernel	AP_2	AP_4	AP_8	AP_{16}	mean
mse	isotropic	0.26	0.41	0.45	0.45	0.40
mse	anisotropic	0.27	0.41	0.45	0.46	0.40
mse	rotated	0.27	0.44	0.48	0.49	0.42
focal	isotropic	0.23	0.37	0.40	0.41	0.35
focal	anisotropic	0.23	0.36	0.39	0.39	0.34
focal	rotated	0.22	0.37	0.39	0.40	0.35
abfocal	isotropic	**0.28**	**0.45**	**0.51**	**0.53**	**0.44**
abfocal	anisotropic	0.23	0.36	0.40	0.40	0.35
abfocal	rotated	0.27	0.43	0.48	0.49	0.42

5 Observation

From the experiments result, compared with isotropic kernel, anisotropic and rotated kernel slightly improves the average precision in the cases of mse loss function while when the loss function is focal or α-balanced focal loss function, anisotropic and rotated filters slightly decreased the average precision. Compared with anisotropic filter, rotated kernel always perform better, which accords with that rotated kernel takes both scale and headings of labels into consideration. Generally, MSE performs worse than α-balanced focal loss function but better than focal loss function. Hence, from the observations, rotated gaussian kernel and α-balanced focal loss function cannot help improve average precision at the same time. The model with isotropic gaussian kernel and α-balanced focal loss function where $\alpha = 0.75$ and $\gamma = 0.1$ achieves best performance.

6 Limitations

A limitation of our focal loss and alpha-balanced focal loss approach is related to hyper-parameter tuning. In order to evaluate the performance of a chosen γ and α, the model has to be trained from scratch which is time consuming. In our approach we used manual search which might not have produced optimal results. For future work, algorithmic techniques for hyperparameter optimization such as grid search, Bayesian optimization or neural networks could be used.

For the Generalized Gaussian Filter, there is room for future improvement in terms of tuning the scale parameter. The scale parameter controls the size of our Gaussian kernel and it is something that we need to tune. Future work could look into the effect of voxelization step size on the scale, and comparing different candidate scale calculations on performance.

Reference

[1] Xingyi Zhou, Dequan Wang, Philipp Krahenb. Objects as Point. Computer Vision and Pattern Recognition 2019
[2] Tsung-Yi Lin, Priya Goyal, Ross B. Girshick, Kaiming He, and Piotr Doll´ar. Focal loss for dense object detection. In IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017, pages 2999{3007. IEEE Computer Society, 2017.