# Chinese Word Detection

Namrata Nadagouda

## 1 Introduction

Since AlexNet took the research world by storm at the 2012 ImageNet Large-Scale Visual Recognition Challenge (ILSVRC), deep learning has become the go-to method for image recognition tasks, far surpassing more traditional computer vision methods used in the literature. In the field of computer vision, convolution neural networks excel at image classification, which consists of categorising images, given a set of classes (e.g. cat, dog), and having the network determine the strongest class present in the image.

Nowadays, deep learning networks are better at image classification than humans, which shows just how powerful this technique is. However, we as humans do far more than just classify images when observing and interacting with the world. We also localize and classify each element within our field of view. These are much more complex tasks which machines are still struggling to perform as well as humans. In fact, I would argue that object detection when performed well, brings machines closer to real scene understanding.

The problem of automatic text detection and recognition has been extensively studied. Given an image, the goal of the character detection task is to detect the bounding boxes of all character instances and also recognize each character instance, i.e., predict its character category. The problem can be broadly divided into two categories of images: document images and natural images. The former is less challenging and many commercial tools are already available. However, this task in the context of natural images is very challenging. This is due to the different appearances of the characters in different images due to style, font, resolution, or illumination differences; characters may also be partially occluded, distorted, or have complex background, which makes detection and recognition even harder. The approaches can be classified as those approaches that use hand-crafted features, and those approaches that use automatically learned features (as deep learning does). In this task, the YOLO framework, a deep neural network, was applied on the Chinese Text in the Wild dataset to detect the Chinese characters in the images.

## 2 Algorithm description

The YOLO (You Only Look Once) model treats the object recognition task as a unified regression problem, different from the models in the R-CNN family which learn to solve a classification task. In the meantime, YOLO sees the entire image during training and thus

it has better performance in recognizing the background with the knowledge of the full context. Compared to other region proposal classification networks (fast RCNN) which perform detection on various region proposals and thus end up performing prediction multiple times for various regions in a image, YOLO architecture is more like FCNN (Fully Convolutional Neural Network) and passes the image (nxn) once through the FCNN and output is (mxm) prediction. This the architecture is splitting the input image in mxm grid and for each grid generation 2 bounding boxes and class probabilities for those bounding boxes. Note that bounding box is more likely to be larger than the grid itself.

The model workflow is as follows:

- Pre-train a CNN network on image classification tasks.
- Split an image into S x S cells. Each cell is responsible for identifying the object (if any) with its center located in this cell. Each cell predicts the location of B bounding boxes and a confidence score, and a probability of object class conditioned on the existence of an object in the bounding box.
  - A bounding box is defined by a tuple of (center x, center y, width, height) — (x,y,w,h). x and y are normalized to be the offsets of a cell location; w and h are normalized by the image width and height, and thus between (0, 1].
  - A confidence score is: probability(containing an object) x IoU(pred, truth).
  - If the cell contains an object, it predicts a probability of this object belonging to one class $C_i$, i=1,2,..., K: probability(the object belongs to the class Ci | containing an object). At this stage, the model only predicts one set of class probabilities per cell, regardless of the number of boxes B.

    In total, one image contains S x S x B bounding boxes, each box corresponding to 4 location predictions, 1 confidence score, and K conditional probability for object classification. The total prediction values for one image is S x S x (5B + K).
- The final layer of the pre-trained CNN is modified to output a prediction tensor of size S x S x (5B + K).

The YOLO network is supposed to be as accurate as SSD but much faster.

# 3 Experimentation

## 3.1 Dataset Description

The Chinese Text in the Wild (CTW) dataset has been used in this project. It is a dataset of Chinese text with about 1 million Chinese characters annotated by experts in over 30 thousand street view images. This is a challenging dataset with good diversity. It contains planar text, raised text, text in cities, text in rural areas, text under poor illumination, distant text, partially occluded text, etc. For each character in the dataset, the annotation includes its underlying character, its bounding box, and 6 attributes. The attributes indicate whether

it has complex background, whether it is raised, whether it is handwritten or printed, etc. It consists of 32,285 high resolution images, 1,018,402 character instances and 3,850 character categories.

## 3.2 Implementation

I followed the CTW detection tutorial given at
https://github.com/yuantailing/ctw-baseline/blob/master/tutorial/3-detection.ipynb

Brief Outline

### 3.2.1 Darknet compilation

The YOLO network is built upon the darknet implementation framework. Darknet has to be downloaded and compiled on the machine. Darknet19, a model pretrained for Imagenet classification is downloaded.

### 3.2.2 Decide categories

A majority of the character categories are rarely-used Chinese characters, which have very few samples in the training data and also have very rare usage in practice. Only the top 1000 frequent observed character categories are consider in this task. The detections are classified into 1001 categories in total, the top 1000 categories and an 'others' category comprising of the rest of the characters. This step produces a file containing information about the frequency of observation of the different characters.

### 3.2.3 Data Pre-processing

This step crops each image into smaller regions containing the characters and saves it to .jpg file. The corresponding text annotations are also produced. The configuration file for training the YOLO model is also an output of this step.

### 3.2.4 Training

This step trains the YOLO network to perform the character detection task and outputs a weights file. Since the training of the model is computationally very expensive because of the large dataset, the model trained on the training data (available on the website), is used for this project. The model is trained by batch gradient descent using the Momentum Optimization algorithm.

### 3.2.5 Testing

The trained model is used to detect characters on the validation set. The testing is done on the validation set instead of the testing set since annotations aren't available for the latter. The testing data is pre-processed and and the model is used to do the predictions. Given an image, the output of the model is a list of recognized character instances, each is associated with a character category, a bounding box, and a confidence score in [0, 1]. Sample detections are in the file sample_detections.json.

# 4 Evaluation

## 4.1 Detections on images

A few sample evaluations are included. The bounding boxes along with the character detected is displayed. Correct detections are shown in green while wrong detections are shown in yellow.


Figure 1

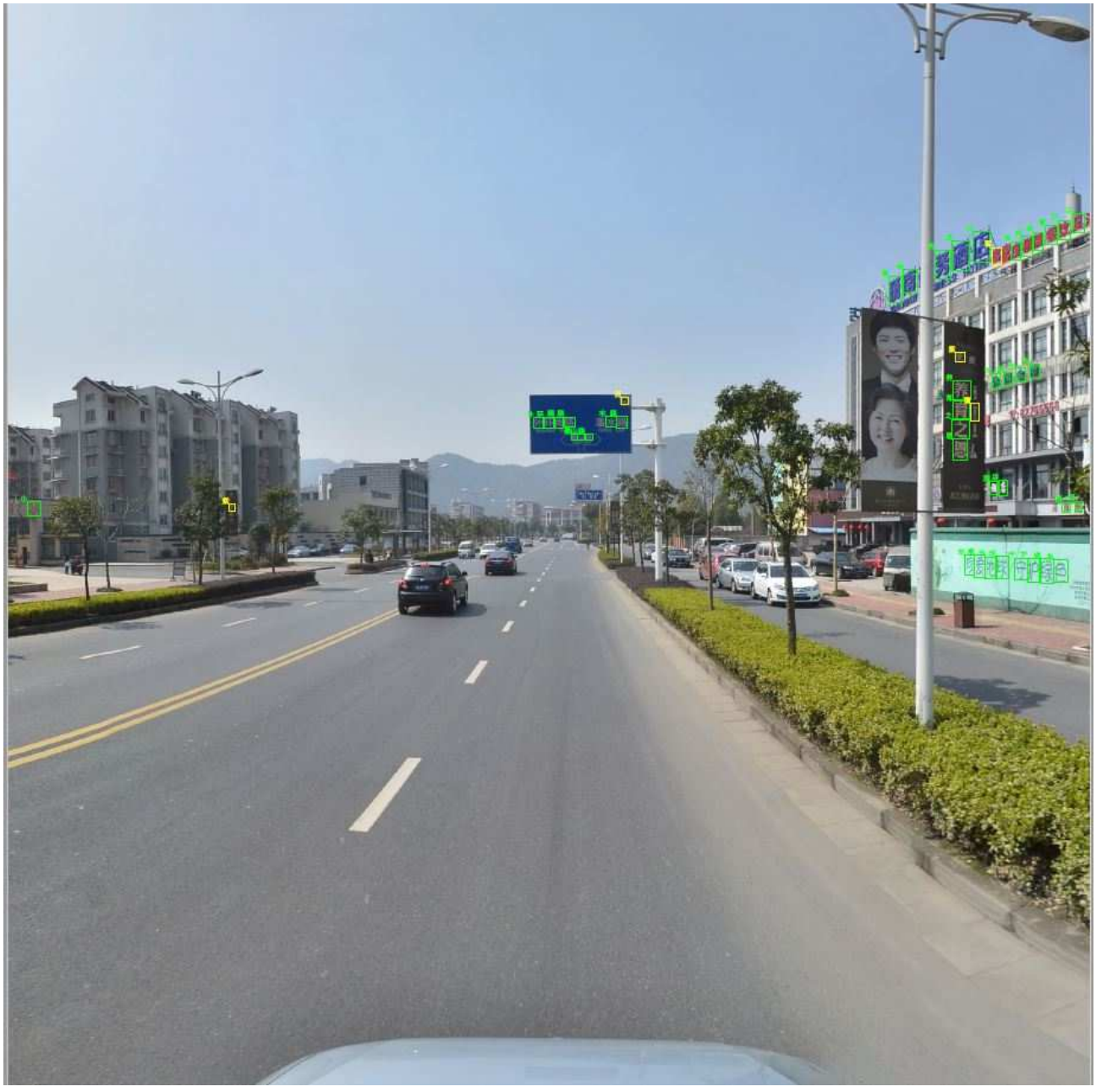


Figure 2

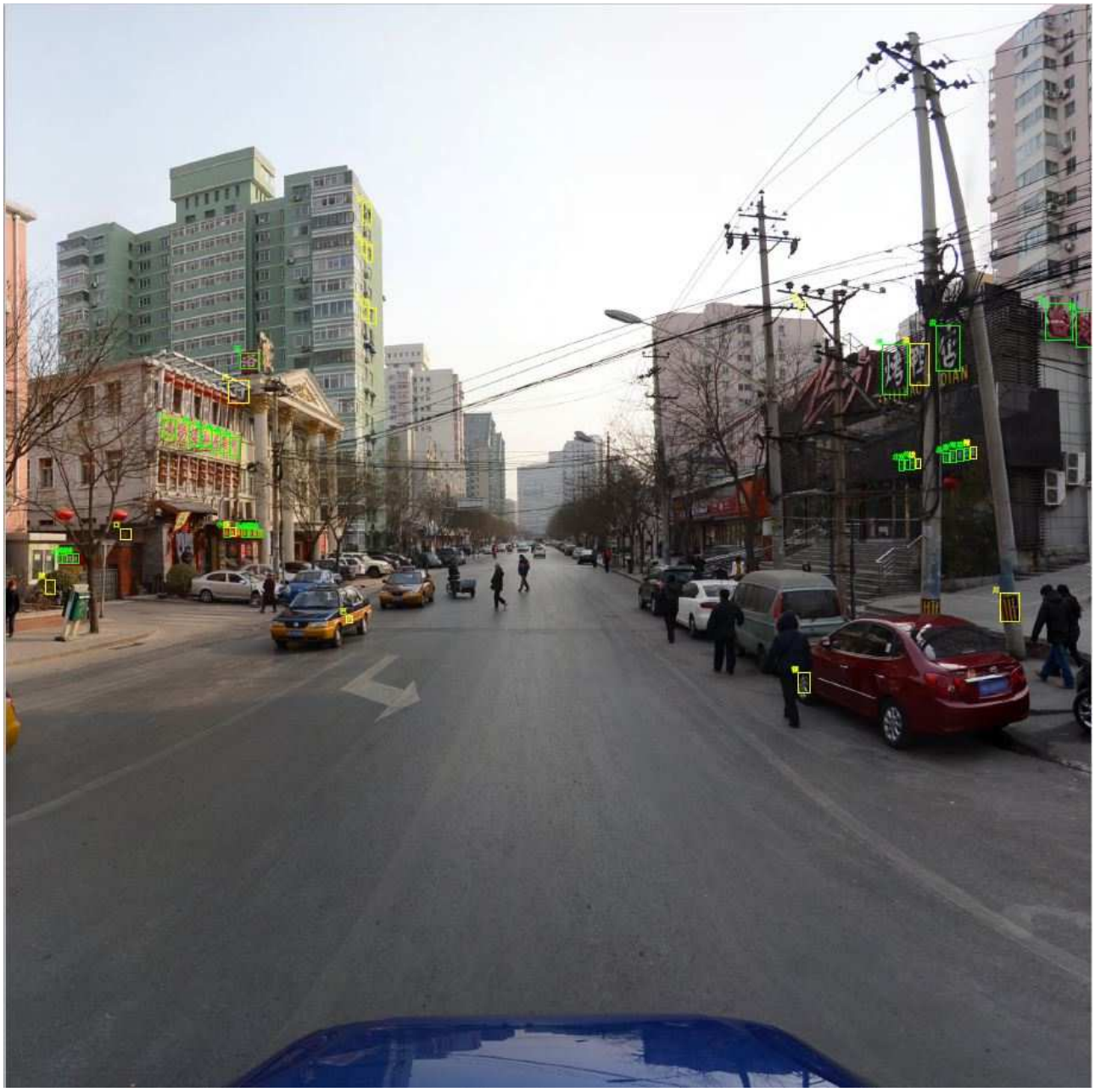


Figure 3

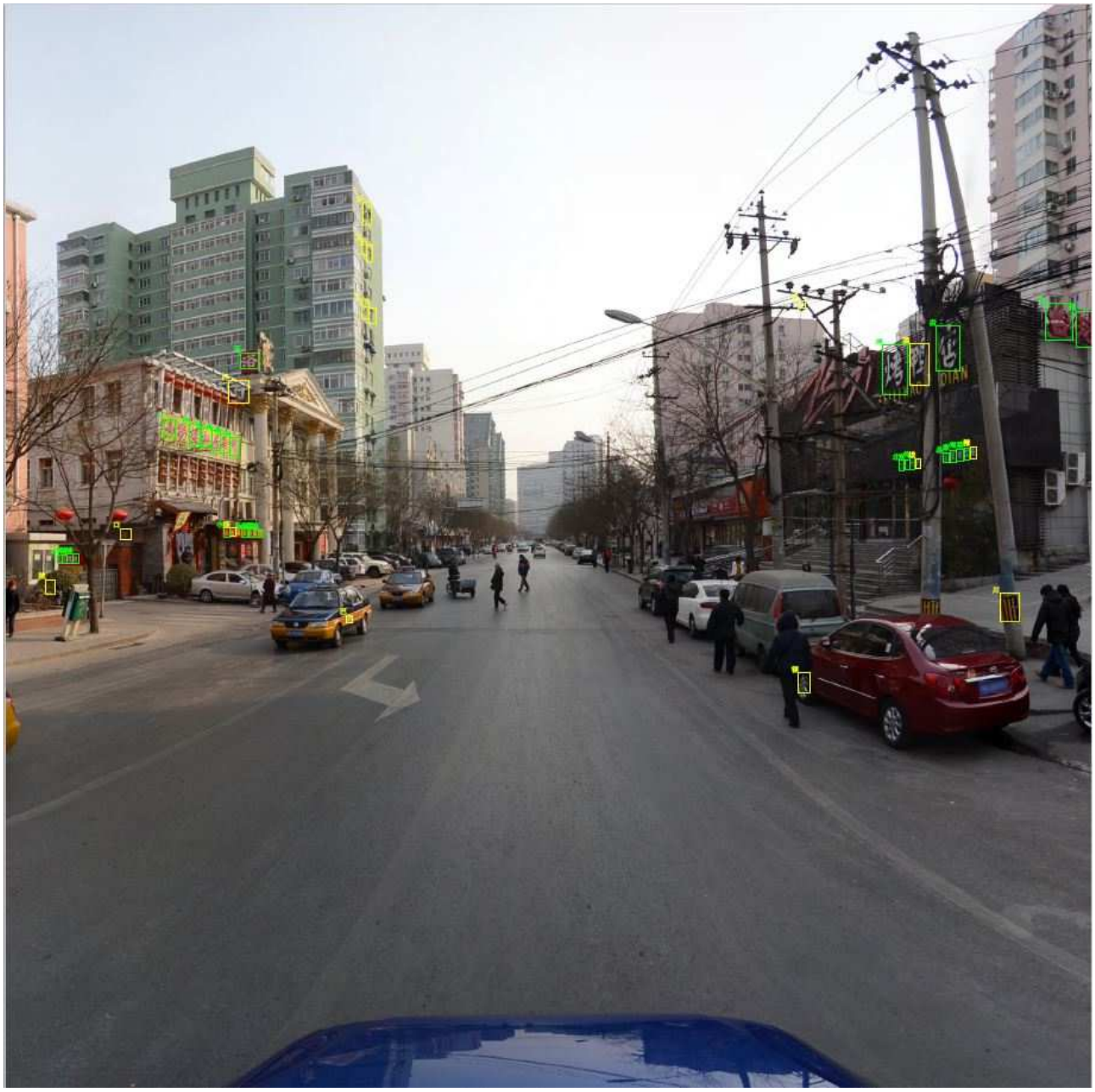


Figure 4

## 4.2 Statistics

According to the paper, the model achieves an mAP of 71.0% on the detection set. In this project, since the validation set is used as the testing set (due to lack of annotations for the testing set) and the trained model used is trained using the training data consisting of training set and validation set, performance is better than that in the paper. The overall mAP is 77.0%. The values of mAP for large, medium and small images are 80.8%, 80.9% and 68.4% respectively.

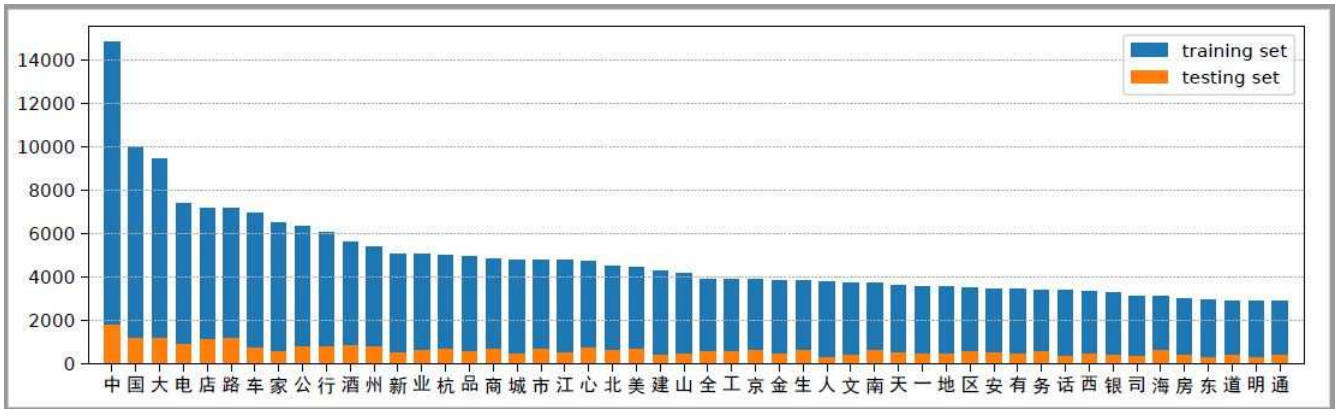
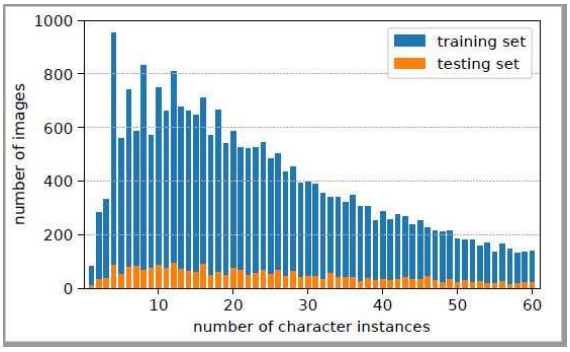
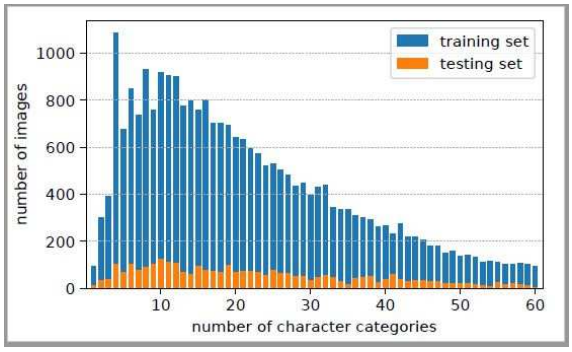



Figure 5: Number of character instances for the 50 most frequent observed character categories in our dataset

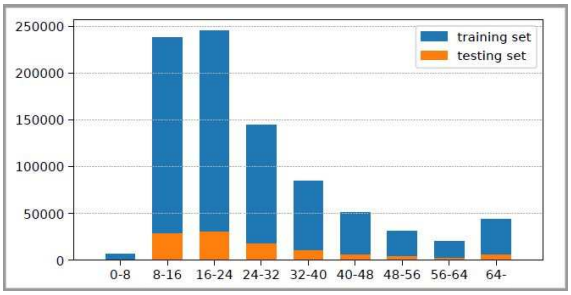



(a) Number of images containing specific number of character instances.

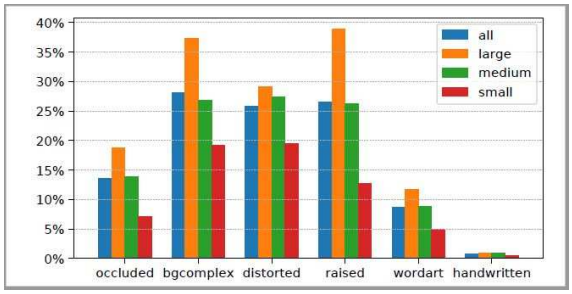



(b) Number of images containing specific number of character categories.

Figure 6

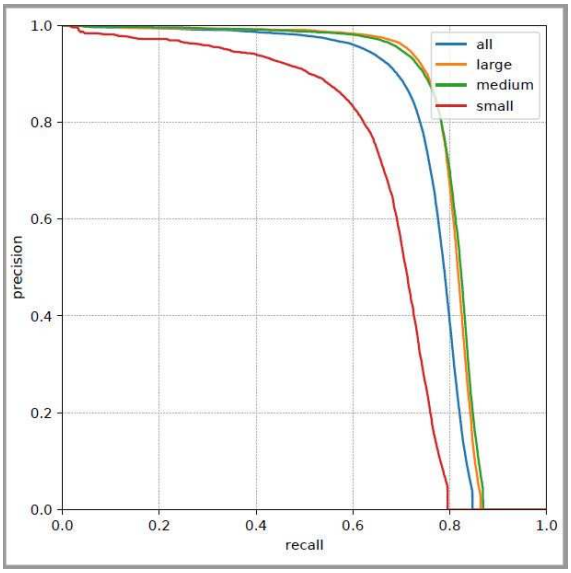



(a) The number of character instances with different sizes. The size is measured by the long side of its bounding box in pixels.

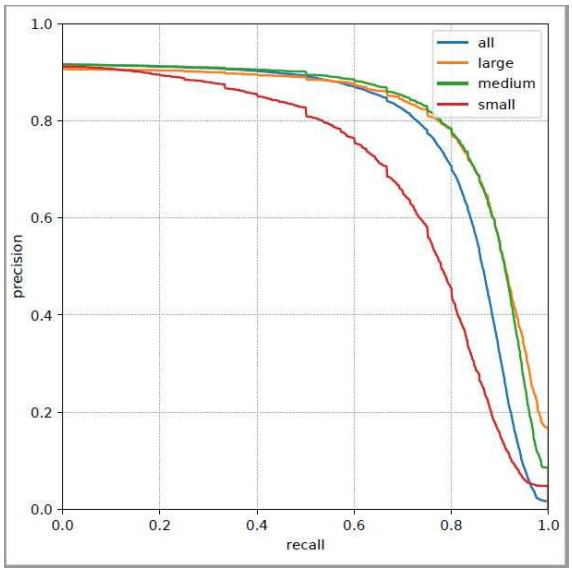



(b) The percentage of character instances with different attributes in all/large/medium/small character instances, respectively. Small, medium, and large refer to character size $< 16, \in [16, 32) and \geq 32$, respectively.

Figure 7




(a) AP curve




(b) mAP curve

Figure 8: Precision-recall curves

# 5 Future Work

## 5.1 Hyperparameter Tuning

Tuning hyperparameters for deep neural network is difficult as it is slow to train a deep neural network and there are numerous parameters to configure. YOLO consists of a convolution neural network (CNN/convnet) and the different hyperparameters are described below.

### 5.1.1 Learning rate

Learning rate controls how much to update the weight in the optimization algorithm. We can use fixed learning rate, gradually decreasing learning rate, momentum based methods or adaptive learning rates, depending on our choice of optimizer such as SGD, Adam, Adagrad, AdaDelta or RMSProp.

### 5.1.2 Number of epochs

Number of epochs is the the number of times the entire training set pass through the neural network. We should increase the number of epochs until we see a small gap between the test error and the training error.

### 5.1.3 Batch size

Mini-batch is usually preferable in the learning process of convnet. A range of 16 to 128 is a good choice to test with. We should note that convnet is sensitive to batch size.

### 5.1.4 Activation function

Activation function introduces non-linearity to the model. Usually, rectifier (ReLU) works well with convnet. Other alternatives are sigmoid, tanh and other activation functions depending on the task.

### 5.1.5 Number of hidden layers and units

It is usually good to add more layers until the test error no longer improves. The trade off is that it is computationally expensive to train the network. Having a small amount of units may lead to underfitting while having more units are usually not harmful with appropriate regularization.

### 5.1.6 Weight initialization

We should initialize the weights with small random numbers to prevent dead neurons, but not too small to avoid zero gradient. Uniform distribution usually works well.

### 5.1.7 Dropout for regularization

Dropout is a preferable regularization technique to avoid overfitting in deep neural networks. The method simply drops out units in neural network according to the desired probability. A default value of 0.5 is a good choice to test with.

#### 5.1.8 Grid search or randomized search

Manually tuning hyperparameters is painful and also impractical. There are two generic approaches to sampling search candidates. Grid search exhaustively search all parameter combinations for given values. Random search sample a given number of candidates from a parameter space with a specified distribution.

With more time and better computing resources, we could experiment with the hyperparameters until we get the desired accuracy.

### 5.2 Data augmentation

Data augmentation is a method to increase the size of training data by generating more images based on the existing training data. Methods such as horizontal flipping, vertical flipping and slight rotation are employed. Like in many other deep learning applications, this method has proven to be crucial in teaching the network to become more robust to various object sizes in the input.

### 5.3 Real time detection

The trained model can be saved and interfaced with cameras on laptops/mobile phones for real time detection. For this, darknet has to be compiled with CUDA and OpenCV.

## References


[1] T. Yuan, Z. Zhu, K. Xu, C. Li, and S. Hu, "Chinese text in the wild," *CoRR*, vol. abs/1803.00085, 2018.

[2] J. Redmon and A. Farhadi, "Yolo9000: Better, faster, stronger," *arXiv preprint arXiv:1612.08242*, 2016.

[3] J. Redmon, "Darknet: Open source neural networks in c." http://pjreddie.com/darknet/, 2013–2016.

[4] L. Weng, "Object recognition for dummies part 3: R-cnn and fast/faster/mask r-cnn and yolo." https://lilianweng.github.io/lil-log/2017/12/31/object-recognition-for-dummies-part-3.html.

[5] S. Lau, "A walkthrough of convolutional neural network[U+200A]—[U+200A]hyperparameter tuning." https://towardsdatascience.com/a-walkthrough-of-convolutional-neural-network-7f474f91d7bd.


## 6 Github profile

https://github.com/nnadagouda95