A study on object detection with deep neural network trained with low quality images

Yiming Bian

Team Number 18

March 10, 2022

Project Proposal for

HCI 575: Computational Perception

Iowa State University

Spring 2022

ABSTRACT

Adopting deep neural networks to object detection and image recognition is a great success. Many architectures are introduced and proved to be very good at recognizing different categories of objects. In early years, neural networks train on simple low-resolution images such as number digits. As the objects to recognize becoming more and more complicated, the size of training images increases inevitably. However, in most real-life scenarios, it is more often the case that only low-quality images are accessible such as those captured by surveillance cameras. In this project, we apply some image transformation technologies such as image blur, crop, color conversion and noise addition based on existing datasets to generate the training and testing data. After training a model using low quality images, we test with both low-quality and high-quality images. Then compare those two accuracies together with the accuracy of pretrained model on high-quality images. In this proposal, we show some preliminary experiment results that prove the need of training a new model using low-quality images.

# 1 Introduction

Image process and computer vision are two of the most popular research areas nowadays. Object detection is one of the cross areas that draws great attention because its applications are ubiquitous. For example, face detection and recognition enable us to unlock smartphones with faces and help the police to find criminals using surveillance camera. Object segmentation and recognition boost the development of self-driving systems in car industry.

After many years of research, computer scientists and mathematicians found multiple ways to realize object classification in the given image or video. For non-neural approaches, one can define features using feature descriptor such as Haar-like feature[1] and histogram of oriented gradients(HOG)[2]. Then use support vector machine(SVM)[3] to do the classification. With the development of computational capabilities, convolutional-neural-network(CNN)-based approaches are gaining more and more popularities. Some widely acknowledged methods are R-CNN[4], Single Shot MultiBox Detector(SSD)[5] and You Only Look Once(YOLO)[6].

In 1989, Yann LeCun et al. proposed a CNN structure called LeNet-5[7]. This network takes a handwriting image of number digit whose size is , thus a black and white image with 32 pixels in both width and height. After the computation of two convolutional layers and three fully-connect layers, it outputs a recognition result from 0 to 9. LeNet-5 only has five layers, which is extremely simple compared to networks nowadays, but it greatly promoted the development of deep learning.

Recognizing handwriting digits is a good start of object recognition. In 2017, Fei-Fei Li, a professor in Standford University, and Christiane Fellbaum, Princeton professor and one of the creators of WordNet, began to work on a project called ImageNet. They presented their ImageNet database for the first time as a poster at the 2009 Conference on Computer Vision and Pattern Recognition(CVPR) in Florida[8]. From 2010 to 2017, the ImageNet Large Scale Visual Recognition Challenge(ILSVRC) was annually held. In this challenge, the validation and test set have a size of 150,000 and in 1000 non-overlapped object categories. In Fig.1, it shows the top-5 error of each annual winner network of ILSVRC. One noticeable achievement is that beginning from 2015, the network has a better error rate than average human being. The competition discontinued after 2017, one of the reasons, I guess, is saving human from being shamed by computers and the appearance of the Terminator.

Chart, bar chart, waterfall chart

Description automatically generated

Figure Top-5 error of each annual winner of ILSVRC and human error rate[9]

In Fig. 2, it shows the depth of network of each winner of the year. The network going deeper and deeper is a very obvious trend. Therefore, deep neural networks(DNN) are under the spotlight of the stage when doing object recognition. In this project, we will first modify existing successful DNNs and do the training. If the results are very poor and hard to converge, we will try to do neural architecture search(NAS) and construct a new network architecture.

Chart, bar chart

Description automatically generated

Figure Depth of each annual winner of ILSVRC[10]. Image credit: https://www.paddlepaddle.org.cn/documentation/docs/en/user\_guides/cv\_case/image\_classification/README.html

We are interested in the performance on high quality image test set of a model trained with low quality images. There are few off-the-shelf low quality image databases, so we decide to create distorted images based on a subset of ImageNet. We define low quality here beyond low resolution. If an image only reflects part of the whole object, provides wrong color information, or contains a lot of noise pixels or even chunks, it is identified as a low-quality image. Details of generating low-quality images are introduced in Section 3.

# 2 Related Work

In 2016, Samuel Dodge et al. reveals some insights on how image quality affects deep neural networks. They considered five types of quality distortions: blur , noise, contrast, JPEG, and JPEG2000 compression[10]. They concluded that all the neural networks tested, including VGG16, GoogleNet, VGG-CNN-S and Caffe Reference, are susceptible to blur and noise distortions, while being resilient to compression artifacts and contrast[10].

In recently year, more attention was drawn to face recognition in low quality images, such as [11] and [12].

# 3 Experimental Platform

To generate low quality images, we apply the following modifications to a given high-quality image.

## 3.1 Decrease resolution

To adjust the resolution of an image, one can call .save() function in Python PIL(pillow) library and change the value of the option quality ranging from 0 to 100.

Text

Description automatically generated

Figure Code segment to change resolution of an image

In the above code segment, the values of quality are set to 50, 10 and 1 respectively and results are shown below in Fig. 4.

Application

Description automatically generated

Figure Same image with different resolutions

## 3.2 Image blurring

In OpenCV, there is a built-in image blurring function cv.blur(), which convolves the image with a filter kernel. The size of the kernel determines the smoothness of the output image. In our experiment, we chose 10 different kernel sizes. A sample code segment is provided below. One can also choose other image blurring functions such as cv.GaussianBlur() and cv.medianBlur().

Graphical user interface, text

Description automatically generated

Figure Code segment to blur an image

In Fig. 6, we show the different levels of blurring on the sample image.

Graphical user interface, application, PowerPoint

Description automatically generated

Figure Same image in different blur levels

## 3.3 Image cropping

When an image only shows a part of the object, it can also be identified as low-quality. There are many ways of cropping image. We generate this kind of image by cropping the out tiers of the original image. In our experiment, we introduce a variable proportion as shown in the code segment below. It is the edge length proportion of the original image. For example, proportion=80 means trimming off 10% of the original length on both end of each edge and getting a cropped image whose area is 64% of the original image, as shown in Fig. 8.

Text

Description automatically generated

Figure Code segment to crop an image

Shape, rectangle

Description automatically generated

Figure Image crop method in our experiment

In the Fig. 9 below, we show the five different cropped images.

Graphical user interface, application

Description automatically generated

Figure Different crop proportions on the same image

## 3.4 Converting Color

For some objects such as car, clothes and shoes, the color does not affect the accuracy of recognition. While other object such as sky and road, the color could be recognized as one of the key features to help the recognition. In this subsection, we convert the normal RGB image to different color system such as grayscale, binary and LAB as shown in Fig. 10. The conversion can be realized by calling cv.cvtColor() from OpenCV library.

A picture containing text, sign

Description automatically generated

Figure Same image in different color systems

There are many other ways to define and generate low-quality images for both training and testing purpose. With the low-quality images, we are ready to pass them through a DNN and get some results. In this proposal, we obtain some preliminary results with ResNet-18. It has a depth of 18 that includes 17 convolutional layers and 1 fully connected layer as shown in Fig. 11. Detailed implementation of ResNet-18 can be found in [13]. There is a pretrained ResNet-18 model in PyTorch, it is adopted in our preliminary experiments.

Table

Description automatically generated

Figure Implementation details of ResNet suite[13]

# 4 Methodology

Before training a model using low-quality images, we first test a pretrained model using some low-quality images to justify that the existing models that trained on high-quality datasets may not perform well when tested with low-quality images. Therefore, it is necessary to explore how a “poorly-trained” model perform on images of any quality. Here poorly-trained means that a model is trained with poor quality images.

We first picked three random pictures of a Samoyed, a crayfish, and a bike for two. Feed them to the pretrained ResNet-18 model and get the performance baseline. Then apply some modifications to the original images and generate a set of low-quality images. We then pass these low-quality images to the same model and obtain some interesting preliminary results. Lastly, we observe the results and provide reasonable interpretations.

For the main part of this project, we plan to first, generate a systematic dataset of low-quality images based on existing dataset such as ImageNet. Then train several existing networks with low-quality image data such as AlexNet, VGG, ResNet etc. If convergence is not observed in the existing networks, we will do a NAS for an architecture that does the object recognition in low-quality images. After that, we will compare the “poorly-trained” model with the pretrained model regarding the accuracy on our test set, which includes images of all qualities. The results will either prove there is no need to train the network with high-quality images or prove it is necessary to have a network for low-quality image processing.

# 5 Preliminary Results

We first generate some blurry images as shown in Fig. 12. Blur kernel sizes are ,,,,,,,,, and.

Graphical user interface

Description automatically generated

Figure Test images with different blur levels

Then we crop the original images with a proportion of 90%, 80%, 70% 60% and 50%. The results are shown in Fig. 13.

A picture containing mammal

Description automatically generated

Figure Test images with different crop proportions

Lastly, we change the original images from RGB to grayscale and binary as shown in Fig. 14. These color conversions cause the dimension of images reduces by a factor of 3. When passed to the network, it causes a dimension mismatch. This issue will be addressed later when introducing the neural network.

A cat sitting on a bicycle

Description automatically generated with low confidence

Figure Test images in different color systems

We first pass three original images to a pretrained NresNet-18 model, provided by PyTorch, and get the following results, as shown in Table 1.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Samoyed** | | **Crayfish** | | **Bike-for-two** | |
| Samoyed | 0.8846215605 | crayfish | 0.95281046628 | bike-for-two | 0.997031569480 |
| Arctic fox | 0.0458053462 | American lobster | 0.03504079207 | mountain bike | 0.002437431132 |
| white wolf | 0.0442763194 | spiny lobster | 0.00815820135 | stretcher | 0.000208732293 |

Table 1. Object recognition results of ResNet-18 on original images

In the above table, the results of ResNet-18 contain several predictions. Top-3 predictions of each are listed here. The value indicates the confidence of the prediction. For example, 0.8846215605 following Samoyed means the model is about 88.5% confident that the object in the input image is Samoyed. For the further results, we only show the confidence of the right prediction, if the value is “NA”, it indicates that the right prediction is not among the top 10 predictions, thus the model does not do a great job recognizing the input image.

For the blurry images, results are shown in Table 2. It has a very clear trend of decrease in confidence of correct prediction as the blur kernel size increases, thus the image getting blurrier. Bike-for-two has a relatively simpler structure. Therefore, when it is blurred, it is extremely hard to recognize. On the other hand, crayfish has very iconic shape, texture, and color. As a result, even it is blurred with a kernel size of 50 by 50. It still has a confidence of almost 70% to be recognized as a crayfish.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| blur kernel | (5, 5) | (10, 10) | (15, 15) | (20, 20) | (25, 25) |
| Samoyed | 0.846616029 | 0.762731850 | 0.645088195 | 0.48983588 | 0.38073784 |
| Crayfish | 0.95000362 | 0.94415336 | 0.925047159 | 0.9045510888 | 0.89052999 |
| Bike-for-two | 0.990263223 | 0.982426881 | 0.9796100854 | 0.9749803543 | 0.918024539 |
| blur kernel | (30, 30) | (35, 35) | (40, 40) | (45, 45) | (50, 50) |
| Samoyed | 0.40070894 | 0.311370581 | 0.25738039 | 0.172615528 | 0.093404002 |
| Crayfish | 0.894245743 | 0.8673642873 | 0.81843727 | 0.728033661 | 0.6971022486 |
| Bike-for-two | 0.423929959 | 0.0480785891 | NA | NA | NA |

Table 2. Object recognition results of ResNet-18 on blurry images

For the cropped images, results are shown in Table 3. Intuitively, as the image getting smaller, the worse accuracy should it be. However, in the example of crayfish, the original image can be correctly recognized with a confidence of about 95%, while when cropped with a proportion of 90, 80, 70 even 50, the confidence is higher than that. In other words, when the proportion equals to 50, the real size of the cropped image is 25% of the original image, but it achieves a better confidence of correct recognition. To explain this, we can take a closer look at Fig. 13. The cropped part in the crayfish image actually removes the background rather than crayfish itself, which decreases the noise in the recognition process and increases the weight of crayfish features. In the other two examples, important features of the object were removed as the crop proportion increases. As a result, the confidence of correction recognition decreases sharply, especially in the Samoyed image.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Proportion | 90 | 80 | 70 | 60 | 50 |
| Samoyed | 0.64993494 | 0.430918306 | 0.45336005 | 0.035013571 | 0.028830636 |
| Crayfish | 0.964680373 | 0.985844851 | 0.97845822 | 0.952149569 | 0.958202421 |
| Bike-for-two | 0.98704564 | 0.988181531 | 0.959052383 | 0.78438401 | 0.496588349 |

Table 3. Object recognition results of ResNet-18 on cropped images

For the color-converted images, results are shown in Table 4. As mentioned earlier in the section. There is a dimension mismatch issue after converting RGB image to either grayscale or binary image. We fix this issue by padding two additional channels duplicating the pixel values in the existing channel. The results in the table provide the same indication: if the feature loss is huge, the recognition fails. For example, when converting from RGB to binary, the fur texture of Samoyed is gone, and it is one of the reasons why it is not recognized as it is even in the top 10 results. On the other hand, bike-for-two does not have many features to lose when it is converted to other colors. As a result, the confidence does not drop significantly.

|  |  |  |
| --- | --- | --- |
| Color | Grayscale | Binary |
| Samoyed | 0.8207725882530212 | NA |
| Crayfish | 0.7709838151931763 | 0.18149273097515106 |
| Bike-for-two | 0.9997569918632507 | 0.991424560546875 |

Table 4. Object recognition results of ResNet-18 on color-converted images

**REFERENCES**

[1] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001, 2001, pp. I-I, doi: 10.1109/CVPR.2001.990517.

[2] McConnell, Robert K. "Method of and apparatus for pattern recognition." U.S. Patent No. 4,567,610. 28 Jan. 1986.

[3] Cortes, Corinna, and Vladimir Vapnik. "Support-vector networks." Machine learning 20.3 (1995): 273-297.

[4] Girshick, Ross, et al. "Rich feature hierarchies for accurate object detection and semantic segmentation." Proceedings of the IEEE conference on computer vision and pattern recognition. 2014.

[5] Liu, Wei, et al. "Ssd: Single shot multibox detector." European conference on computer vision. Springer, Cham, 2016.

[6] Redmon, Joseph, et al. "You only look once: Unified, real-time object detection." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

[7] LeCun, Yann, et al. "Backpropagation applied to handwritten zip code recognition." Neural computation 1.4 (1989): 541-551.

[8] Gershgorn, Dave. "The data that transformed AI research—and possibly the world." Quartz, July 26 (2017): 2013-2017.

[9] Kang, Dae-Young, Hieu Pham Duong, and Jung-Chul Park. "Application of deep learning in dentistry and implantology." (2020): 148-181.

[10] Dodge, Samuel, and Lina Karam. "Understanding how image quality affects deep neural networks." 2016 eighth international conference on quality of multimedia experience (QoMEX). IEEE, 2016.

[11] Zhou, Yuqian, Ding Liu, and Thomas Huang. "Survey of face detection on low-quality images." 2018 13th IEEE international conference on automatic face & gesture recognition (FG 2018). IEEE, 2018.

[12] Li, Pei, et al. "Face recognition in low quality images: a survey." arXiv preprint arXiv:1805.11519 (2018).

[13] He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.