Training custom image classification and object detection models

1   Introduction

Convolutional neural networks (CNNs) have been actively researched and adopted as large computation power and large datasets are becoming available. Numerous CNN architectures and models have been developed. CNN applications such as autonomous car, medical image analysis, face recognition and scene labelling are becoming mature [1]. Papers and practical guides give out huge amount of knowledge about CNNs. Nonetheless, as there are numerous CNN models, use cases and theories, beginners often encounter a difficulty of knowing what techniques they need to understand and implement and how to kick-start their practical projects.

The aim and contribution of this project is to provide one example of image classification and one example of object detection to demonstrate the methodology and pipeline of classifying and detecting user-defined objects using CNNs. The underlying concepts and implementation details are explained. It is hoped that readers can grasp these concepts, utilize the provided source codes to overcome any technical challenges they encounter and intelligently select the right architecture for their real-life applications.

Live demo: https://cnn-repo.s3.ap-east-1.amazonaws.com/build/index.html

2   Concepts

Building a CNN model from scratch with random weight initialization has two major shortcomings [2]. First, large amount of data are required to collect and process to achieve high accuracy. Second, training from scratch is computationally expensive and memory-demanding. Fine-tuning, which is a type of transfer learning, can significantly reduce the number of training data and computational resources required. Data augmentation further lowers the barrier of preparing for a sufficiently large and representative dataset. Therefore, it is crucial to understand transfer learning, fine-tuning and data augmentation.

2.1   Transfer Learning

Transfer learning takes a network pre-trained on a dataset and transfers knowledge learnt from the previous training to the new task. Transfer learning is proven to help improve the accuracy and training time of CNNs [2, 3]. For startup companies and individual learners, it is too demanding to train their models from scratch. Therefore, they typically repurpose pre-trained models such as VGG19, ResNet50, Faster R-CNN and YOLO built by tech giants and star researchers to solve their specific tasks.

2.2   Fine-tuning

There are two approaches to transfer learning. They are feature extraction and fine-tuning. Feature extraction in transfer learning is the process of using pre-trained network as a feature extractor to extract feature vectors and then feeding these feature vectors together with optional labels to a machine learning algorithm for training. Fine-tuning in transfer learning is the process of re-training the CNN in which at least some of the parameters are added or changed. It has been demonstrated that fine-tuning can boost the task accuracy when the dataset is small and different from the pre-trained model's dataset [4]. Intuitively, since the dataset is small, it must rely on the pre-trained model to learn more basic geometric shapes. When the dataset is different from the pre-trained model's dataset, for example, knowing whether an object is a clothes is not directly helpful for differentiating clothes of different styles without adding more knowledge on top of the previous visual cognition.

2.3   Data Augmentation

Data augmentation is a process of applying random transformations, such as flipping, shifting, rotation, zooming, changing brightness and changing colours, to increase the diversity of training data to reduce overfitting. Indeed, the original batch is replaced by the new randomly transformed batch. As real-world data can have great variance in terms of transformation, training data need to capture them. Since it is difficult and expensive to obtain training data under a comprehensive set of transformations, data augmentation allows the training data to generalize better and easier.

3   Implementations

In this project, an image classification example and an object detection example will be developed. For image classification example, 99% validation accuracy and 98% testing accuracy are already

achieved and prediction of each image is completed within 0.07 second using tensorflow model in python and a desktop with i7-10700 CPU and 32GB RAM. Using tensorflow model in javascript and the same desktop, prediction time is approximately doubled. Since image classification tasks are generally more analytical rather than about real-time monitoring, such latency and testing time are sufficient. It is believed that 98% testing accuracy is likely sufficient for readers to persuade themselves, their bosses, their colleagues and even investors to kick-start their project, while keeping this example simple enough for readers to get started. Testing accuracy can be further improved by larger dataset, better hyper-parameter tuning and re-training more layers.

On the other hand, object detection tasks generally emphasize on latency and frames per second (FPS) to accomplish real-time monitoring. This example has not been implemented yet. The model is aimed to achieve 30 FPS and 10% mean average precision (mAP) measured at 50% intersection over union (IOU) using the same desktop with i7-10700 CPU and 32GB RAM [5]. Since it is expensive for startup companies and individual learners to set up high-end GPUs, it is difficult to achieve both high FPS and high mAP at the same time. It is believed that for a deployable solution, low mAP can still deal with typical cases well and can be compensated by other parts of the application, while a fairly good FPS should not be compromised. Otherwise, real-time user interactions and real-time control systems become infeasible as the whole pipeline is bottlenecked by this object detection task. It is hoped that readers succeed to develop real-time applications such as games and autonomous robot plugins.

3.1 Image Classification

hello

3.2 Object Detection

hello

4 Conclusion

5 References

[1] https://ijcsit.com/docs/Volume%207/vol7issue5/ijcsit20160705014.pdf [2] https://openreview.net/pdf?id=
[3] https://www.researchgate.net/profile/Jordan-Bird/publication/325803364_A_Study_on_CNN_Transfer_Lea
Study-on-CNN-Transfer-Learning-for-Image-Classification.pdf [4] https://arxiv.org/pdf/1406.2952.pdf
[5] https://arxiv.org/pdf/1907.11093.pdf