Image Segmentation using a Transformer

Background

Image segmentation is a fundamental task in computer vision that involves partitioning an image into meaningful and distinct regions. This process plays a pivotal role in extracting valuable insights from visual data by assigning each pixel or region of an image to a specific class or category. Image segmentation is a crucial foundation for more advanced visual analysis and interpretation by outlining objects or areas of interest within an image. In medical imaging, image segmentation finds application in precisely identifying organs or anomalies, enabling accurate diagnosis and treatment planning. Moreover, autonomous driving systems aid in identifying pedestrians, vehicles, and road boundaries, contributing to safer and more efficient navigation.

Problem Specification

Deep Learning is widely used for image segmentation. More specifically, Convolutional Neural Networks (CNNs), play a pivotal role in image segmentation due to their ability to automatically learn relevant features from raw image data, leading to significant improvements in accuracy and efficiency. However, most deep learning architectures (including recent solutions) are computationally expensive. For example, the Segmentation Transformer (SegFormer) [1] requires up to 84.7 million parameters while achieving 51.8% mean intersections over union accuracy on the ADE20K dataset [2]. Such architecture requires long inference time and demands power-consuming computational resources. Can we compress these state-of-the-art deep learning while achieving comparable results (without performance degradation) so that they can be deployed to mobile devices? What about the trade-off between computational complexity and model performance for image classification tasks?

Suggested Method

TensorFlow-Lite [3] library is widely used to compress and optimize the deep learning model. It generates the compressed TensorFlow-Lite model file, which can be instantly deployed to the mobile device. Different compression methods (i.e., Pruning, Quantization, Clustering, and Weight Sharing) are available and documented on the TFLite website. These methods can be applied to state-of-the-art solutions to reduce computational complexities. This research will select the best-performing recent image classification architecture and aim at compressing it using the TensorFlow-Lite library. The compressed architecture can be compared with the original architecture by presenting quantitative results on training and validation accuracy, the total number of parameters, and inference time to predict one output.

Difficulty Level

This project has INTERMEDIATE difficulty level. The student will run the publicly available Python source code of any chosen state-of-the-art architecture and pass it to the Tensorflow Lite library to compress it. The student will adjust the TFLite's hyper-parameters and architectures to optimize it. At the end of this project, the student will be able to demonstrate the trade-off between computational cost (i.e., # parameters and model file size) and model performance (i.e., output accuracy).

Pre-requisite

The student should have basic knowledge of Deep Learning and understand Python coding with TensorFlow libraries.

Relevant Articles

[1] Xie, Enze, Wenhai Wang, Zhiding Yu, Anima Anandkumar, Jose M. Alvarez, and Ping Luo. "SegFormer: Simple and efficient design for semantic segmentation with transformers." Advances in Neural Information Processing Systems 34 (2021): 12077-12090.

[2] Zhou, Bolei, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. "Scene parsing through ade20k dataset." In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 633-641. 2017.

[3] Li Shuangfeng. TensorFlow Lite: On-Device Machine Learning Framework[J]. Journal of Computer Research and Development, 2020, 57(9): 1839-1853. doi: 10.7544/issn1000-1239.2020.20200291

Useful Tools

i. Google CoLaboratory for running Python code and training deep learning model: https://colab.research.google.com/

ii. Getting started with Segmentation Transformer using TensorFlow Keras Library: https://keras.io/examples/vision/segformer/

iii. Tensorflow Model Optimization Toolkit: https://www.tensorflow.org/lite/performance/model_optimization

iv. Recent Image segmentation architectures: https://paperswithcode.com/task/semantic-segmentation

v. Image segmentation databases: https://paperswithcode.com/datasets?task=semantic-segmentation

vi. An example of Tensorflow Lite Image Classification Mobile Application for Deployment. https://www.tensorflow.org/lite/examples/image_classification/overview

Project proposed by Ali Hassan (ali.hassan@miun.se)