A Study on Multi-Exit Deep Neural Network for Real-time Processing

1810209 LI NAQIN

The deep neural network has been wildly used in various fields in recent years, since it has excellent performance and is easy to use in classification and prediction tasks. It is also applied in real-time systems, such as self-driving and object tracking systems, etc. However, the problem of it is the huge amount of multiplication and accumulation operations make a neural network task become a heavy task in real-time systems. This may bring a huge impact on the performance of the system in the real-time property. Because of this, most of the real-time systems which include the neural network task must make the compromise to the accuracy of the result of the neural network task and the real-time requirement of the system.

BranchyNet and MSDNet were proposed in 2017. These two neural networks are different from the typical neural network model. They have more than one exit-point instead of only one exit-point. Different exit-points are inserted into different places in the neural network model. Since the insertion locations in the model are different from each other, the execution times from input-point to each exit-point are also different. The later exit-point will take more time to finish compared with earlier exit-points. Each of the exit-points in the multiple exit-point models gives the output of the classification result on one input picture.

In Chapter 2, we briefly introduce the CNN(Convolutional neural network), and how the feature map flows as the input and output data between each layer. For the BranchyNet, it was proposed for the fast inference via finishing its execution from earlier exit-points if the system already has enough confidence with the generated classification results. One of our proposals utilizes a BranchyNet model based on the VGG-16 for Cifar-10 dataset. This VGG-16 based BranchyNet has 3 exit-points. From the entry of the model to the first exit-point "EXIT1", it has 9 layers. In addition, it has 12 layers from entry-point to second exit-point "EXIT2", and 16 layers from entry-point to the third exit-point "EXIT3". The accuracy for "EXIT1" on Cifar-10 is 87.22%, for "EXIT2" is 88.51%, and for "EXIT3" is 88.57%. MSDNet, which was proposed for the similar purpose to the BranchyNet considers the impact of the accuracy of the final exit-point by inserting early exit-points into the model. The MSDNet for Cifar-10 has 24 layers, and an exit-point is inserted after every 2 layers. The accuracy for "EXIT1" in MSDNet is 84.45%, for "EXIT2" is 86.55%, "EXIT3" is 87.95%, "EXIT4" is 88.93%, "EXIT5" is 90.11%, "EXIT6" is 90.17%, "EXIT7" is 90.28%, "EXIT8" is 90.55%, "EXIT9" is 90.6%, "EXIT10" is 90.61%, and "EXIT11" is 90.77%.

In Chapter 3, we regard the neural network task which is implemented with the multiple exit-points model in real-time systems as the imprecise computation task. We show three scheduling methods for real-time systems that have one neural network task in their taskset. The first scheduling method is the typical real-time scheduling algorithm with the single exit-point neural network task. The second method "IC" includes the neural network task with a multiple exit-points model and regard this task as the imprecise computation task. In this scheduling, we decide the exit-point to finish the execution of the neural network based on the system load, instead of choosing by the confidence of classification results of each exit-point. If the time resource that the neural network task received from the system is long enough to exit from the later exit-points, then the system uses the classification result of the later exit-point. The later exit-point will give a result with higher accuracy. The third method "SIC" is a server-based imprecise computation scheduling method. It is proposed as an enhanced imprecise computation scheduling. With "SIC", the response time of each exit-point will be improved. Since "SIC" is a server-based method, we present a way to decide the server's priority in the systems, a way to compute the budget of the server, and when this server will be released to the system.

In Chapter 4, we show experiments to evaluate these 3 scheduling methods. We compare the accuracy of the classification result of the task among a single exit-point model, multiple exit-points model with "IC", and multiple exit-points model with "SIC". In addition, we compare the response time of each exit-point in "IC" and "SIC". In experiments, we implemented the multiple exit-points model for neural network task with the VGG-16 based BranchyNet and the MSDNet model for Cifar-10 dataset. The basic scheduling algorithms we used are EDF(earliest deadline first) and RM(rate-monotonic).

In Chapter 5, we implemented the binarized VGG-16 based BranchyNet on FPGA with VHDL. We show the structure of the neural network model and the way we are using it to implement the batch normalization operation. Then, we compare the execution result between the binarized BranchyNet and its software implementation, for checking the correctness of the hardware implementation. Furthermore, we show the accuracy and execution of it.

In Chapter 6, we draw a conclusion on our research. From our research, we can know the efficiency of applying the multiple exit-points neural network models and treating the neural network task as the imprecise computation task. It can improve the accuracy of the neural network task without making other tasks miss their deadline.