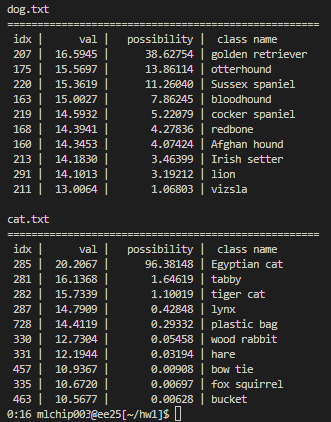
Mlchip003 109511105 謝宗霖 HW1 Report

1. Simulation results demonstrate the predicted output for the provided input data.



1. Implementation Approach:
   1. 一張含有 文字, 功能表, 螢幕擷取畫面, 字型 的圖片

      自動產生的描述AlexNet Structure Implementation in C++:

Initially, the logic and structure of AlexNet, including convolution layers, fully connected layers, ReLU activation, and max pooling, were implemented in C++.

* 1. Weight Loading:

It was crucial to ensure that the order in which weights were read matched the order used by a pre-trained model in PyTorch.

* 1. Parameter Estimation:

Since stride and padding parameters for each convolution and max pooling layer were not provided in the homework, they were estimated.

Below are the final guesses:

Conv1 = Conv(11, 11, 3, 64, 4, 2)

Conv2 = Conv(5, 5, 64, 192, 1, 2)

Conv3 = Conv(3, 3, 192, 384, 1, 1)

Conv4 = Conv(3, 3, 384, 256, 1, 1)

Conv5 = Conv(3, 3, 256, 256, 1, 1)

MaxPool = MaxPool(3, 3, 2)

* 1. Layer Parameter Verification:

Ensured that each layer's dimensions, parameters, and multiplication-addition counts matched those specified in "HW1.pdf."

* 1. Adjustments for Specific Weights:

Modified the order of reading and counting to accommodate the specific weights used in this implementation of AlexNet.

* 1. Conversion to SystemC:

The code was then translated from C++ to SystemC. This process involved code restructuring, including file segmentation (.h and .cpp) and modularization to enhance clarity and organization.

1. Challenges Faced:

* Padding Discrepancy:

An initial misunderstanding regarding the padding in Conv1 led to incorrect output dimensions ([54, 54]). This issue was resolved after a detailed comparison with the multiplication-addition counts provided in the homework document.

* Weight and Input Loading:

Given that the assignment focused solely on inference, correct ordering of weights and inputs was paramount. Deviations in this ordering could significantly affect the outcomes. A debugging strategy was devised by implementing AlexNet in Python with PyTorch for direct comparison of layer outputs, facilitating the identification and correction of discrepancies.

* Transition to SystemC:

The conversion of the implementation from C++ to SystemC presented challenges, particularly in adapting to SystemC's modeling paradigms and ensuring the preservation of functional correctness throughout the process.

1. Other Observations:

* The exercise provided deep insights into the intricacies of deep learning model implementation, particularly the critical importance of precision in parameters and weight management.
* Debugging strategies, especially the comparative analysis with a PyTorch model, proved invaluable in understanding and rectifying issues that arose during implementation.
* The conversion to SystemC underscored the importance of clean code practices and modular design in facilitating readability, maintainability, and scalability of complex systems.