CPRE 587

Hardware Design for

Machine Learning

Lab 2

Instructor: Dr. Henry Duwe

Group 9:

Joseph Schmidt

Yi Hang Ang

3.2 Layer Operations

List the different types of layers present.

Conv2D, MaxPooling2D, Flatten, Dense, Softmax

Write the mathematical formulation of each layer type

Conv2D:

$$\mathbf{o}[n][m][p][q] = (\sum_{c=0}^{C-1}\sum_{r=0}^{R-1}\sum_{s=0}^{S-1}\mathbf{i}[n][c][Up+r][Uq+s] \times \mathbf{f}[m][c][r][s]) + \mathbf{b}[m],$$
$$0 \le n < N, 0 \le m < M, 0 \le p < P, 0 \le q < Q,$$
$$P = (H - R + U)/U, Q = (W - S + U)/U.$$

MaxPooling2D:

$$P_{gmax} = \max_{i=0}^{M-1} \max_{j=0}^{N-1} F(i,j)$$

$$P_{gavg} = \frac{1}{MN} \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} F(i,j)$$

Flatten:

$$y_{jk}(x) = f\left(\sum_{i=1}^{n_H} w_{jk} x_i + w_{j0}\right)$$

Dense:

$$\mathbf{O}_{nm} = \sum_{chw} \mathbf{I}_{nchw} \mathbf{F}_{mchw}.$$

Softmax:

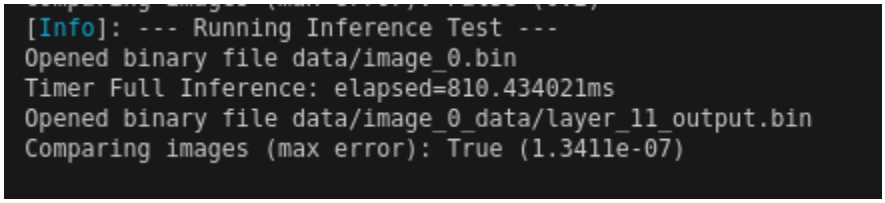
$$s\left(x_i\right) = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}}$$

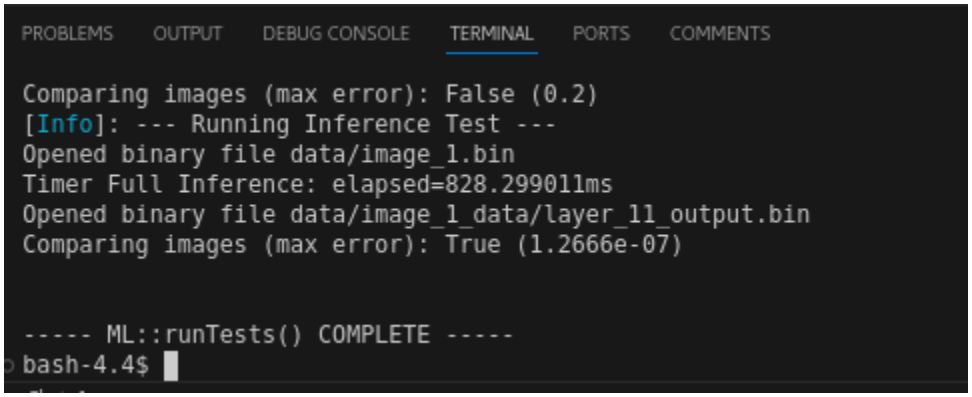
Also, include a brief description of each variable, its use/purpose, and how that is represented in our model.

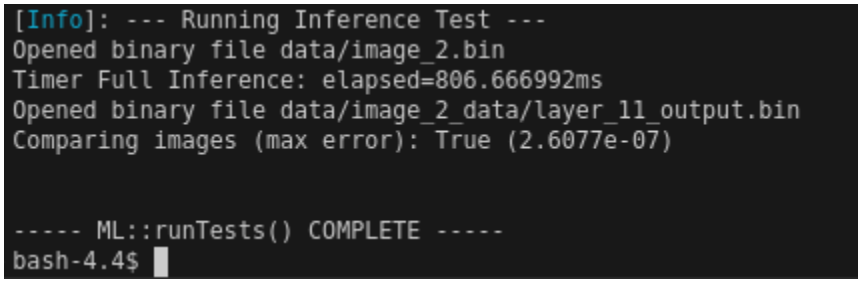
N = Batch Size, multiple input feature maps may be processed as a batch to potentially improve the reuse of weights. Batch size is 1 in our model.
M = Number of filters // Number of output channels, which is equivalent to the depth of the output map, used to get a better average from the layers, represented as numFilters.
C = Number of filter channels // Number of input channels, the purpose of learning more sets of spatial patterns or features from input. It is represented as depth in our model
H/W = Ifmap height/width, used to calculate P and Q, P = (H - R + U) / U, represented as height/width
R/S = Filter height/width, used as condition for the for loops, represented as fill_h and fill_w
P/Q = Ofmap height/width, used as condition for the for loops, represented as outputHeight/outputWidth in our model.

B = Biases, bias represents errors that may be introduced while running inference. Represented as biasData in our model
W = Weights, higher weights mean higher impact on the prediction, and vice versa. Represented as weightData in our model.
X = Inputs, the dataset for training or target class for inference. Model needs inputs to generate predictions or outputs. Represented as image_0_data for layer 0, or layer_0_output.bin for layer 1 etc.
O = Output. Outputs are the predictions made or the features learned that will modify the weights. Represented as layer_0_output.bin for layer 0 etc.

Report the result of appropriate comparison functions and justify that the value is acceptable.







The values are acceptable because they are too small and negligible to affect the predictions, i.e. weight values are too large compared to the error values.

3.3 Analysis

Estimate the size of the memory needed for each layer and how many MAC computations are required for each layer

Conv2D:

First is filter size, second is input channels, third is output channels, and fourth is data size (byte)

Then you add the biases size which is the number of output channels times the data size

Conv1: (5x5) x 3 x 32 x 4 = 9600 bytes

32 x 4 = 128 bytes

9600 + 128 = 9728 bytes

Conv2: (5x5) x 32 x 32 x 4 = 102400 bytes

32 x 4 = 128 bytes

102400 + 128 = 102528 bytes

Conv3: (3x3) x 32 x 64 x 4 = 73728 bytes

64 x 4 = 256 bytes

73728 + 256 = 73984 bytes

Conv4: (3x3) x 64 x 64 x 4 = 147456 bytes

64 x 4 = 256 bytes

147456 + 256 = 147712 bytes

Conv5: (3x3) x 64 x 64 x 4 = 147456 bytes

64 x 4 = 256 bytes

147456 + 256 = 147712 bytes

Conv6: (3x3) x 64 x 128 x 4 = 294912 bytes

128 x 4 = 512 bytes

294912 + 512 = 295424 bytes

MaxPooling2D:

No weights so the size is 0

Flatten:

No weights so the size is 0

Dense:

First is input neuron, second is the output neurons, third is the size of the data

Then you add the biases which is the number of output neurons times the data size

Dense1: 2048 x 256 x 4 = 2097152 bytes

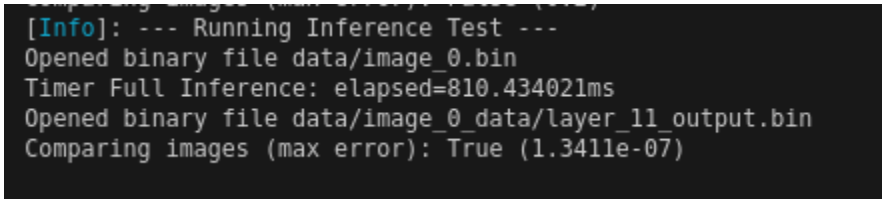
256 x 4 = 1024 bytes

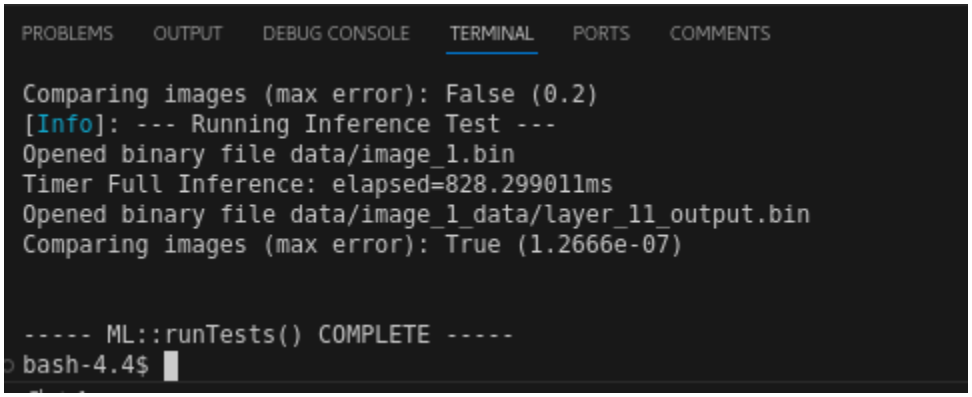
2097152 + 1024 = 2098176 bytes

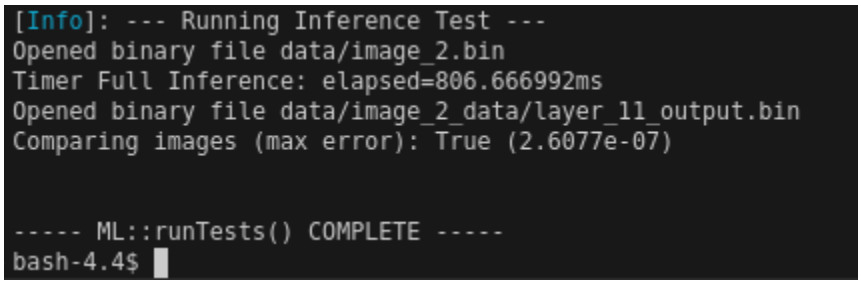
Softmax:

No weights so the size is 0

Verify that your outputs match the expected output sourced from Tensorflow implementation







What was the final maximum error you saw/ the error tolerance you used in your verification process for comparing your output? How did you come to that value and what is a reasonable error range? Where does this error come from and should there be any at all?

The maximum error was 2.60077e-07. We chose our epsilon value to be 0.001. We chose this because it was given. We need it because we are comparing floats which will very rarely ever be exactly the same. We are also doing a lot of floating point calculations which leads to not exact

output values. This means we also need a little bit of leeway on our outputs. We could probably reduce the epsilon based on how small our error was during the inference.

3.4 Profiling C++ Implementation

Report the time taken for a single inference on both your x86 lab machine and the ZedBoard. Compare these to the time taken when a single inference is executed using Tensorflow backend. You should use Tensorboard to measure the performance. If you see a difference in execution time, why do you think there is a difference?

```
xsdb% [Info]: --- Running Basic Test ---
xsdb% Opened binary file ./data/image_0.bin
xsdb% Comparing image 0 to itself (max error): 0
xsdb% Comparing image 0 to itself (T/F within epsilon 0.001): true

Change a value by 0.1 and compare again
xsdb% Comparing images (max error): False (0.1)
Change a value by 0.1 and compare again...
xsdb% Comparing images (max error): False (0.2)
[Info]: --- Running Inference Test ---
xsdb% Opened binary file data/image_2.bin
xsdb% Timer Layer Inference: elapsed=51136.000000ms
Timer Full Inference: elapsed=54484.000000ms
xsdb% Opened binary file data/image_2_data/layer_11_output.bin
xsdb% Comparing images (max error): True (2.6077e-07)
xsdb%

----- ML::runTests() COMPLETE -----


----- STARTING FILE TRANSFER SERVER -----
IP:      192.168.1.2
Netmask: 255.255.255.0
Gateway: 192.168.1.1
MAC:     00:0A:35:00:01:02
xsdb% Start PHY autonegotiation
xsdb% Waiting for PHY to complete autonegotiation.
xsdb% autonegotiation complete
link speed for phy address 0: 1000
xsdb% HTTP file server started!
xsdb%
```

Zedboard: 54484 ms

```
Opened binary file data/model/dense2_biases.bin
[Info]: --- Running Basic Test ---
Opened binary file ./data/image_0.bin
Comparing image 0 to itself (max error): 0
Comparing image 0 to itself (T/F within epsilon 0.001): true

Change a value by 0.1 and compare again
Comparing images (max error): False (0.1)
Change a value by 0.1 and compare again...
Comparing images (max error): False (0.2)
[Info]: --- Running Inference Test ---
Opened binary file data/image_2.bin
Timer Full Inference: elapsed=405.779999ms
Opened binary file data/image_2_data/layer_11_output.bin
Comparing images (max error): True (2.6077e-07)


----- ML::runTests() COMPLETE -----
bash-4.4$ ls
```

x86 machine: 405.779999 ms

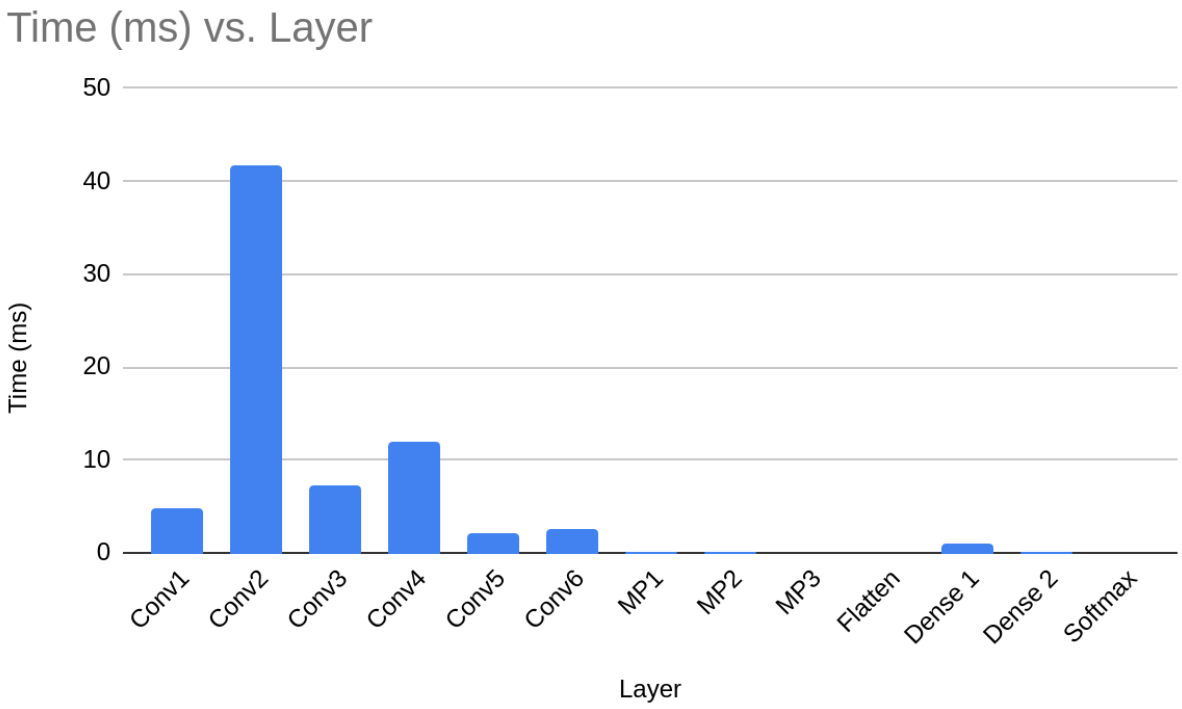
```
[Info]: --- Running Inference Test ---
xsdb% Opened binary file data/image_2.bin
xsdb% Timer Layer Inference: elapsed=51136.000000ms
Timer Full Inference: elapsed=54484.000000ms
xsdb% Opened binary file data/image_2_data/layer_11_output.bin
xsdb% Comparing images (max error): True (2.6077e-07)
xsdb%

----- ML::runTests() COMPLETE -----
```

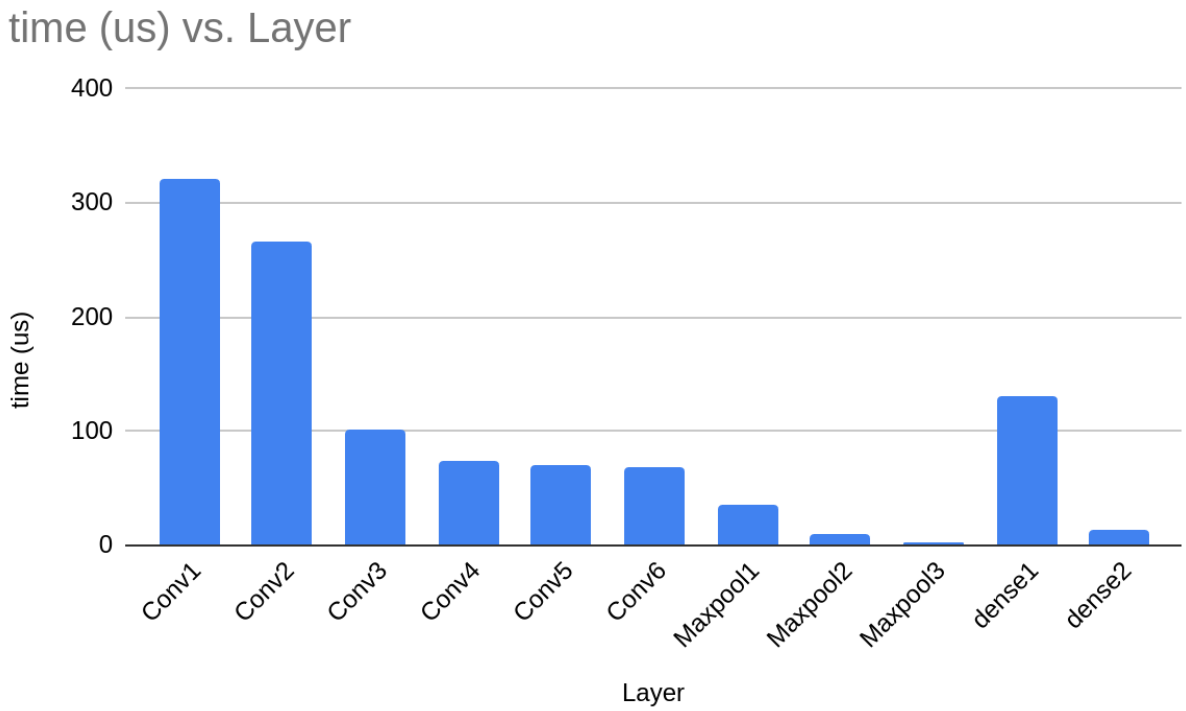
Our implementation took 133 times longer than TensorBoard running layer 1. This is because TensorBoard implements many different efficiencies and algorithms to make inference run faster.

Figure out which layers takes the most if the computation time. Plot the layerwise execution time fraction and report it. State possible reasons why a particular layer takes significant computation time? Does this match your Lab 1 results from Tensorboard? Which layers may be ripe for optimization? Why?

```
Comparing images (max error): False (0.2)
[Info]: --- Running Layer Test ---
Opened binary file data/image_2.bin
Timer Layer Inference: elapsed=4.721000ms
Opened binary file data/image_2_data/layer_0_output.bin
Comparing images (max error): True (5.06639e-07)
[Info]: --- Running Layer Test ---
Opened binary file data/image_2_data/layer_0_output.bin
Timer Layer Inference: elapsed=41.715000ms
Opened binary file data/image_2_data/layer_1_output.bin
Comparing images (max error): True (1.90735e-06)
[Info]: --- Running Layer Test ---
Opened binary file data/image_2_data/layer_1_output.bin
Timer Layer Inference: elapsed=0.096000ms
Opened binary file data/image_2_data/layer_2_output.bin
Comparing images (max error): True (1.17549e-38)
[Info]: --- Running Layer Test ---
Opened binary file data/image_2_data/layer_2_output.bin
Timer Layer Inference: elapsed=7.272000ms
Opened binary file data/image_2_data/layer_3_output.bin
Comparing images (max error): True (7.15256e-07)
[Info]: --- Running Layer Test ---
Opened binary file data/image_2_data/layer_3_output.bin
Timer Layer Inference: elapsed=11.955000ms
Opened binary file data/image_2_data/layer_4_output.bin
Comparing images (max error): True (8.34465e-07)
[Info]: --- Running Layer Test ---
Opened binary file data/image_2_data/layer_4_output.bin
Timer Layer Inference: elapsed=0.039000ms
Opened binary file data/image_2_data/layer_5_output.bin
Comparing images (max error): True (1.17549e-38)
[Info]: --- Running Layer Test ---
Opened binary file data/image_2_data/layer_5_output.bin
Timer Layer Inference: elapsed=2.095000ms
Opened binary file data/image_2_data/layer_6_output.bin
Comparing images (max error): True (1.19209e-06)
[Info]: --- Running Layer Test ---
Opened binary file data/image_2_data/layer_6_output.bin
Timer Layer Inference: elapsed=2.637000ms
Opened binary file data/image_2_data/layer_7_output.bin
Comparing images (max error): True (1.43051e-06)
[Info]: --- Running Layer Test ---
Opened binary file data/image_2_data/layer_7_output.bin
Timer Layer Inference: elapsed=0.006000ms
Opened binary file data/image_2_data/layer_8_output.bin
Comparing images (max error): True (1.17549e-38)
[Info]: --- Running Layer Test ---
Opened binary file data/image_2_data/layer_8_output.bin
Timer Layer Inference: elapsed=0.001000ms
Opened binary file data/image_2_data/layer_9_output.bin
Comparing images (max error): True (0)
[Info]: --- Running Layer Test ---
Opened binary file data/image_2_data/layer_9_output.bin
Timer Layer Inference: elapsed=0.985000ms
Opened binary file data/image_2_data/layer_10_output.bin
Comparing images (max error): True (5.24521e-06)
[Info]: --- Running Layer Test ---
Opened binary file data/image_2_data/layer_10_output.bin
Timer Layer Inference: elapsed=0.025000ms
Opened binary file data/image_2_data/layer_11_output.bin
Comparing images (max error): False (12.3984)
[Info]: --- Running Layer Test ---
Opened binary file data/image_2_data/layer_11_output.bin
Timer Layer Inference: elapsed=0.012000ms
terminate called after throwing an instance of 'std::runtime_error'
  what():  Failed to open binary file: data/image_2_data/layer_12_o
```



The convolutional layer takes the most time. It decreases as the layers and filters get smaller. We differ from lab 1 though, where the second pair in the convolutional layer is higher than the first. The second convolutional in the pair of convolutional takes longer because there are more output weights.



Lab 1

The convolutional layer may need to optimized because it takes up the most runtime in terms of percentage in our CNN.