Training Requirements Report Chapter

TR.1 Our Model Results

For our model without Tensor Contraction Layer (TCL) to train to 80% accuracy, the results are:

- Number of Training Images: ~3500.
- Latency to Classify an Image: 3.2 ms.
- Amount of Memory: 2.744 MB.

For our model with Tensor Contraction Layer (TCL) to train to 80% accuracy, the results are:

- Number of Training Images: ~30000.
- Latency to Classify an Image: 110 ms.
- Amount of Memory: 23.52 MB.

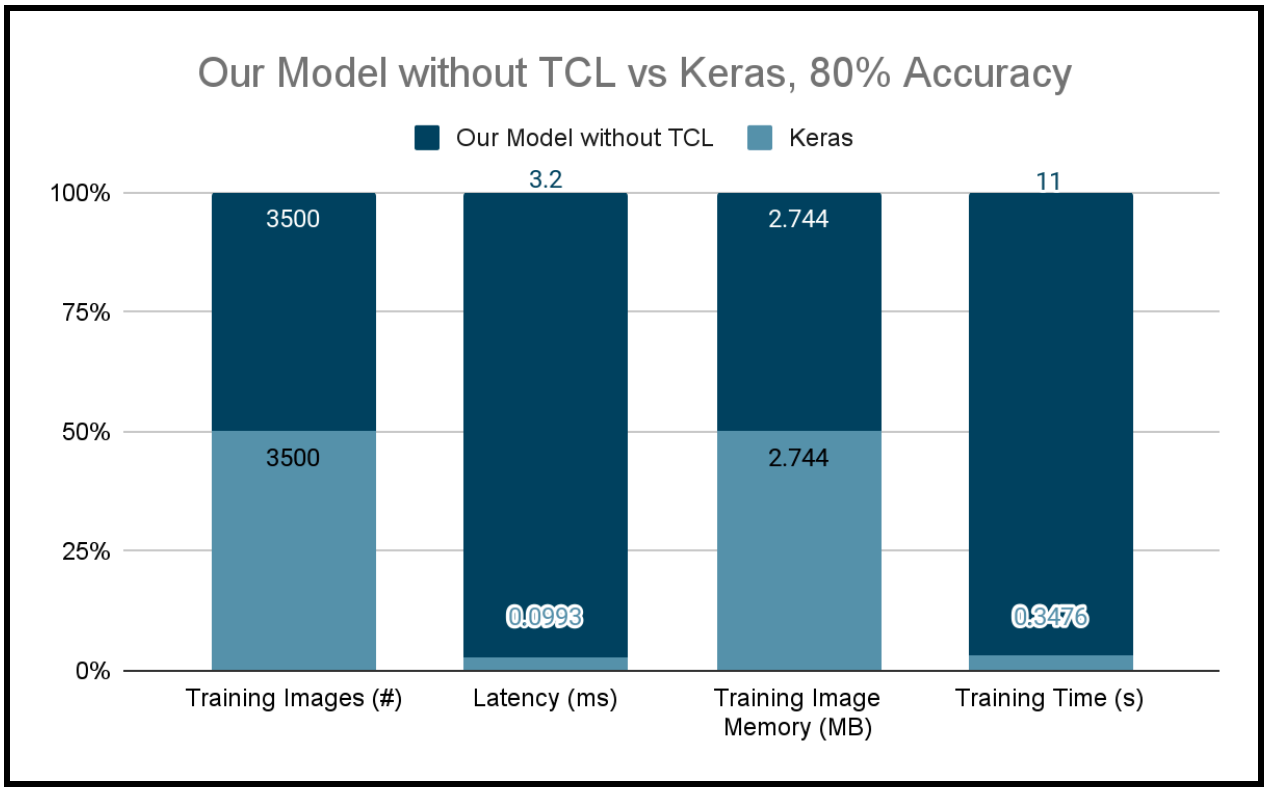
TR.2 Ideal, Popular Model Results

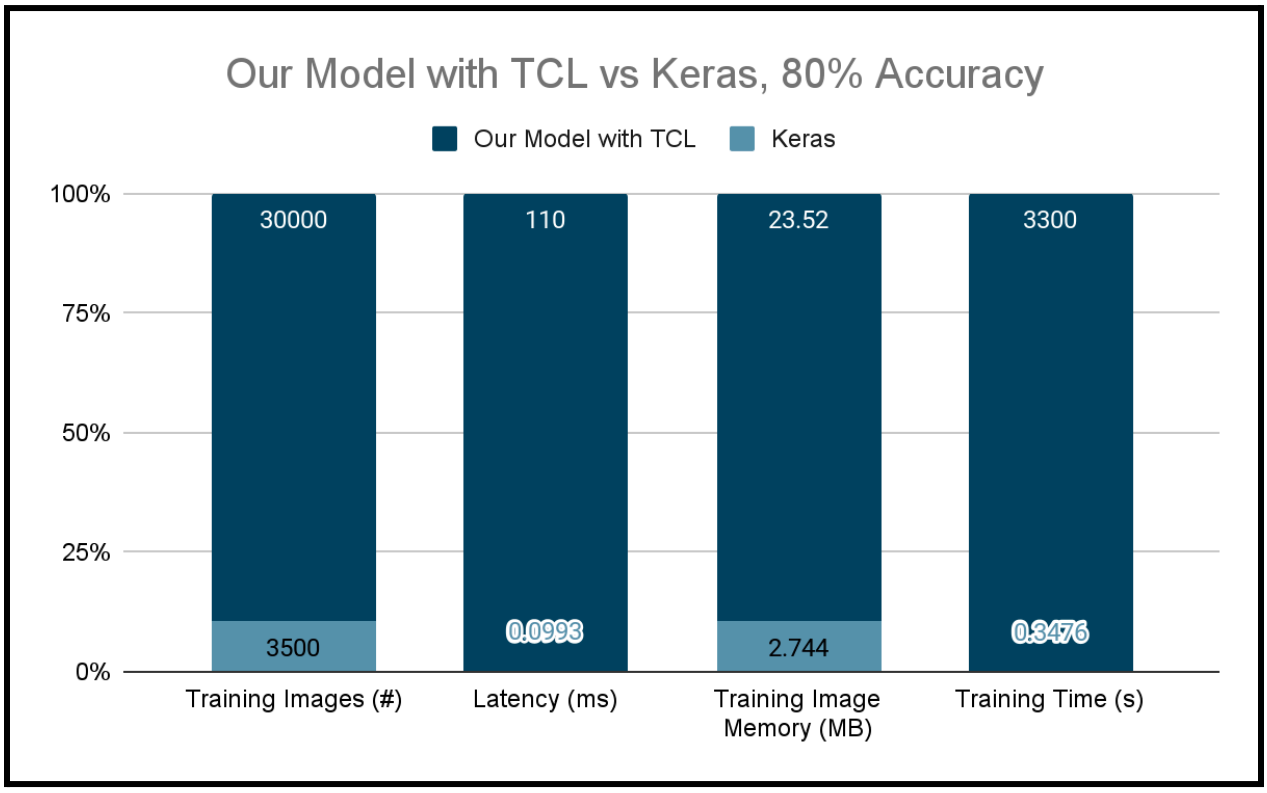
For an ideal, popular model (Keras) to train to 80% accuracy, the results are:

- Number of Training Images: ~3500.
- Latency to Classify an Image: 0.0993 ms.
- Amount of Memory: 2.744 MB.

TR.3 Summary of Model Results

The two graphs below illustrate the comparison between our model with and without TCL vs Keras. The specified accuracy we were interested in achieving is 80%, this number is arbitrary and could have been different, but 80% accuracy is a system level requirement for our project. To be clear, on the graph less is better; the less area that model takes up on a column, the better it performed compared to the other model.


Our Model without TCL vs Keras, 80% Accuracy
Our Model without TCL
Keras
100%
75%
50%
25%
0%
3500
3500
3.2
0.0993
2.744
2.744
11
0.3476
Training Images (#)
Latency (ms)
Training Image Memory (MB)
Training Time (s)


Our Model with TCL vs Keras, 80% Accuracy
Our Model with TCL
Keras
100%
75%
50%
25%
0%
30000
3500
110
0.0993
23.52
2.744
3300
0.3476
Training Images (#)
Latency (ms)
Training Image Memory (MB)
Training Time (s)

Looking at the first graph (without TCL), we can see that each model took the same number of training images, but that our model's latency was higher. Because each model had the same number of training images, the amount of memory was also the same. Because our model had a higher latency, the training time was also higher. Overall, the Keras model performed better.

Looking at the second graph (with TCL), we can see that our model took a higher number of training images, and that our model's latency was higher. Because our model had a higher number of training images, the amount of memory was also higher. Because our model had a higher latency, the training time was also higher. Overall, the Keras model performed better.

Comparing both our models, one with TCL and one without, and ignoring Keras, we can see that the model without TCL performed better. It needed a lower number of training images, which led to a lower amount of training time. Overall, the model without TCL performed better.

TR.4 Summary of Methodology and Result Reproducibility

Our results are determined by using a model we created from C++. Ideal results are determined by using an ideal, popular model (Keras) that will be trained and tested on the same dataset (MNIST) as our model. The architecture of both models is the same, but the implementation is different, so specifically it is the difference between implementations that is being tested. By ideal, we mean that it is well-developed and refined. By popular, we mean it is widely accepted and used by reputable organizations.

Explanation for each of the four statistics measured:

- Number of training images is determined by rerunning the model training code until it reaches the desired 80% accuracy for the first time, incrementing the number of training images after each run that it did not reach 80%.
- Latency to classify an image is determined by recording the time a model takes to perform a certain number of trials and dividing by the number of trials.
- Amount of memory is determined by multiplying the number of images times the image size. For the MNIST dataset, all images are 28x28 pixels in black and white, and a black and white pixel only needs 1 byte to store a [0,255] grayscale value, thus:
    - (28*28 pixels/image) * (1 / byte/pixel) = 784 bytes/image.
    - (# images) * (784 bytes/image) = # bytes.
- Amount of training time is determined by multiplying the number of training images by the latency to classify an image.

To review the code or to reproduce these results, the files for each model can be found in the File Links section below. Our model is in "cppcnn.zip", and the Keras model is in "tensorflo.zip". They will require C++ and Python, respectively, along with required libraries and modules that can be found near the beginning of the code files. Some basic programming literacy is required and assumed that a reader would know or be able to learn.

TR.5 References and File Links

TR.5.1 File Links

[1] “Our Model: cppcnn.zip.”
https://drive.google.com/file/d/1S3dmeaw7qcR1s2fVVnikuXwkPbFE4jqS/view?usp=drive_link

[2] “Keras Model: tensorflo.zip.”
https://drive.google.com/file/d/12uVjtEuIQOitpUm4AfSmCDZjIkl5mobL/view?usp=drive_link