Implementing a U-Net Architecture in MagmaDNN

Students: Chow Tsz Ching, Edward Karak, Spencer Smith

Chinese Univ. of Hong Kong, Baruch College at City Univ. of New York, Univ. of North Texas

Mentors: Dr. Kwai Wong, Dr. Stan Tomov

Abstract

MagmaDNN is an open-source deep-learning library written in C++. It is based on Magma, a linear algebra package and is designed to handle supervised problems. MagmaDNN is unique in that it is tailored for parallel computing and, consequently, supercomputing applications.

A U-Net is a convolutional neural network developed originally for biomedical image segmentation to detect tumors. It can be defined in terms of down-sampling and up-sampling layers. Our U-Net implementation is called semantic segmentation. It aims to learn the classifications of individual pixels in an image.

Introduction

- A U-Net is a convolutional neural network developed originally for biomedical image segmentation to detect tumours. It can be defined in terms of down-sampling and up-sampling layers. Our U-Net implementation is called semantic segmentation. It aims to learn the individual pixels in an image.
- Similar to PyTorch and Keras, MagmaDNN is a machine learning package. It is still in the development phase, so it is very limited in its scope. Take the loss function as an example, MagmaDNN only supports categorical cross-entropy loss and MSE. MagmaDNN can only do classification but not segmentation and hence the Output Layer of a neural network must be a flattened two-dimensional tensor or else errors will occur. Therefore, the main task of our research is to implement segmentation in MagmaDNN.

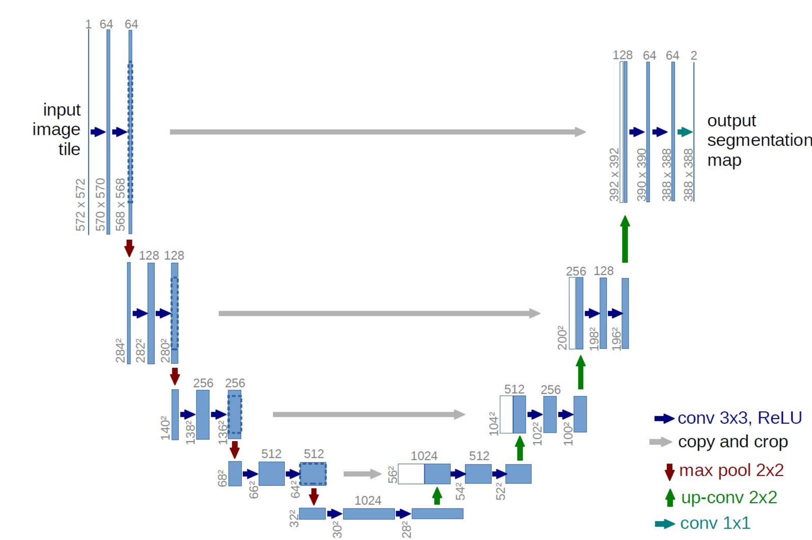
Methodology

Data

We will train our neural network with the Oxford-IIIT Pet Dataset. The dataset contains around 7000 cats and dogs classified by breed. Each image has its corresponding masking. The masking is in trimap format. Trimap is in three colors only: 1 for foreground, 2 for background and 3 for not classified.

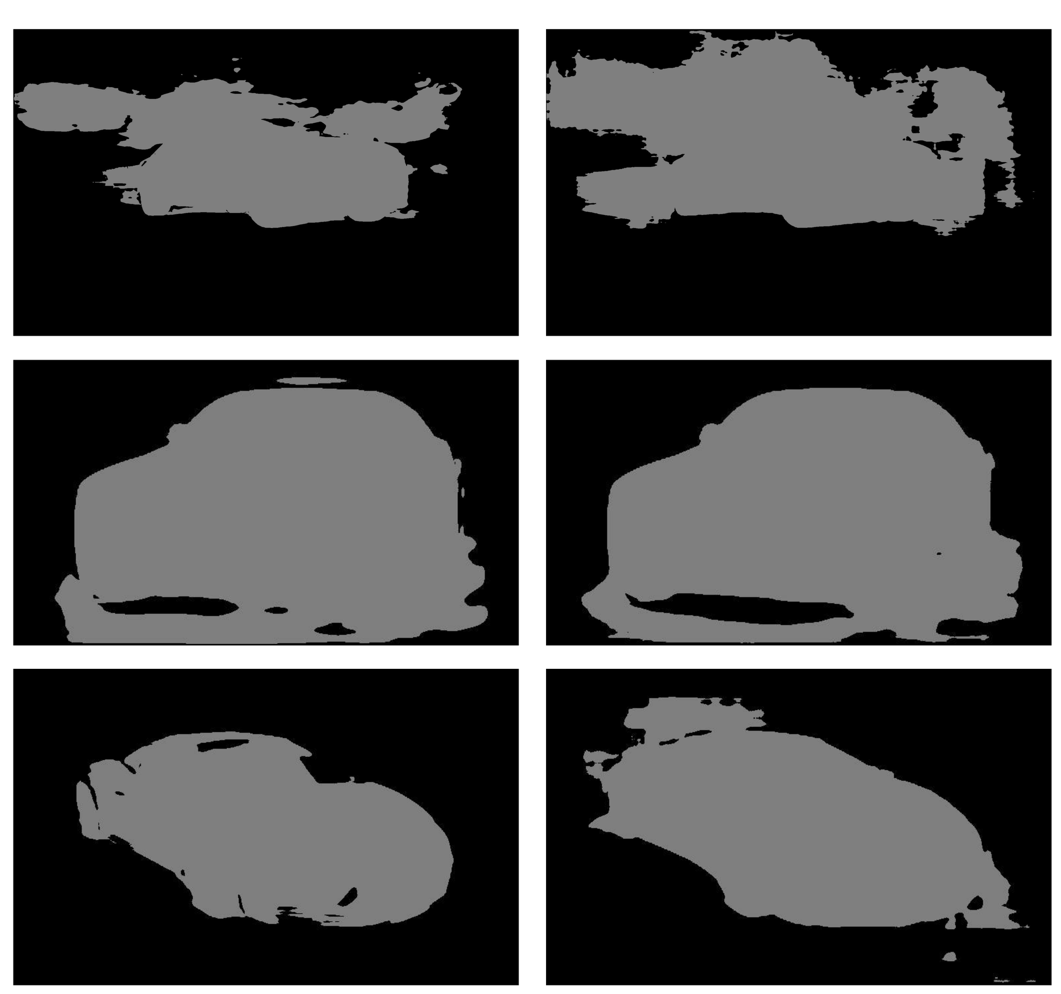
Data Extraction

MagmaDNN was only able to input MNIST, CIFAR10, CIFAR100 data and one-hot encoded ground-truth data. Moreover, MagmaDNN is only able do classification instead of image segmentation. Therefore, we need to use a custom dataset for the training and testing for the U-Net. After, we can integrate OpenCV with MagmaDNN so it can input data from ImageNet and Oxford-IIIT Pet Dataset.



Model

We have adopted convolution transpose instead of using bilinear interpolation for upsampling in the U-Net. Convolution transpose is better than bilinear interpolation because convolution transpose will learn when it is training. However, up-sampling using bilinear interpolation will consume less resources Still, we still adopt the convolution transpose because it overperform bilinear interpolation theoretically.

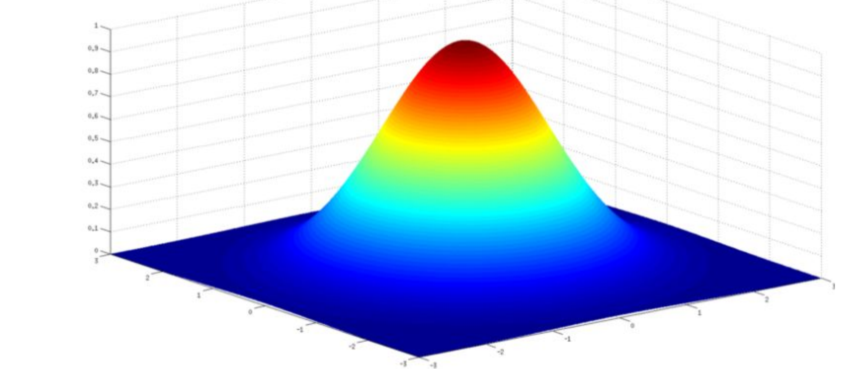


Results

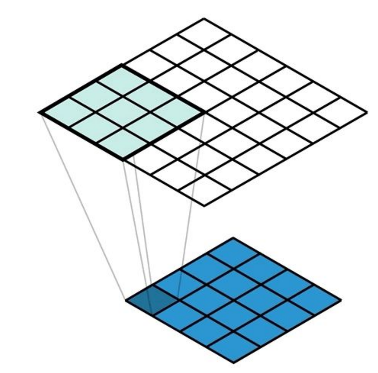
```
Name                          Output Shape        # Params
==========================================================
InputLayer                  (32, 3, 32, 32)              0
Conv2d                     (32, 16, 32, 32)            432
BatchNormLayer             (32, 16, 32, 32)             32
RELU                       (32, 16, 32, 32)              0
Conv2d                     (32, 16, 32, 32)           2304
BatchNormLayer             (32, 16, 32, 32)             32
RELU                       (32, 16, 32, 32)              0
Pooling                    (32, 16, 16, 16)              0
Conv2d                     (32, 32, 16, 16)           4608
BatchNormLayer             (32, 32, 16, 16)             64
RELU                       (32, 32, 16, 16)              0
Conv2d                     (32, 32, 16, 16)           9216
BatchNormLayer             (32, 32, 16, 16)             64
RELU                       (32, 32, 16, 16)              0
Pooling                      (32, 32, 8, 8)              0
Conv2d                       (32, 64, 8, 8)          18432
BatchNormLayer               (32, 64, 8, 8)            128
RELU                         (32, 64, 8, 8)              0
Conv2d                       (32, 64, 8, 8)          36864
BatchNormLayer               (32, 64, 8, 8)            128
RELU                         (32, 64, 8, 8)              0
Conv2dTranspose            (32, 64, 16, 16)          36864
Conv2d                     (32, 32, 16, 16)          18432
Concat                     (32, 64, 16, 16)              0
Conv2d                     (32, 32, 16, 16)          18432
BatchNormLayer             (32, 32, 16, 16)             64
RELU                       (32, 32, 16, 16)              0
Conv2d                     (32, 32, 16, 16)           9216
BatchNormLayer             (32, 32, 16, 16)             64
RELU                       (32, 32, 16, 16)              0
Conv2dTranspose            (32, 32, 32, 32)           9216
Conv2d                     (32, 16, 32, 32)           4608
Concat                     (32, 32, 32, 32)              0
Conv2d                     (32, 16, 32, 32)           4608
BatchNormLayer             (32, 16, 32, 32)             32
RELU                       (32, 16, 32, 32)              0
Conv2d                     (32, 16, 32, 32)           2304
BatchNormLayer             (32, 16, 32, 32)             32
RELU                       (32, 16, 32, 32)              0
Conv2d                      (32, 1, 32, 32)             16
SOFTMAX                     (32, 1, 32, 32)              0
OutputLayer                 (32, 1, 32, 32)              0
==========================================================
Total number of params : 176192
n_samples: 199
loss = 0.840868
loss = 0.960302
loss = 0.919261
loss = 1.059976
loss = 1.191010
loss = 1.152314
Epoch (1/10): accuracy=0 loss=1.295e-10 time=3
```

Implementation

$$L_{det} = \frac{-1}{N}\sum_{c=1}^{C}\sum_{i=1}^{H}\sum_{j=1}^{W}\begin{cases}(1-p_{cij})^{\alpha}\log(p_{cij}) & \text{if } y_{cij}=1\\ (1-y_{cij})^{\beta}(p_{cij})^{\alpha}\log(1-p_{cij}) & \text{otherwise}\end{cases}$$



We used a modified cross-entropy loss function to calculate the loss of the network. It uses a 2-d gaussian distribution (pictured above) to calculate help the model converge on the predictions of the foreground pixels.



Our upsampling method of choice in transposed convolution. We implemented it using the cuDNN C++ API.

```
Input Tensor:
Tensor size of {1, 1, 4, 4}
{
  {
    | 1.00000,   1.00000,   1.00000,   1.00000, |
    | 1.00000,   1.00000,   1.00000,   1.00000, |
    | 1.00000,   1.00000,   1.00000,   1.00000, |
    | 1.00000,   1.00000,   1.00000,   1.00000, |
  }
}
```

```
Output Tensor:
Tensor size of {1, 1, 8, 8}
{
  {
    |-0.04657,  -0.23533,  -0.04657,  -0.23533,  -0.04657,  -0.23533,  -0.04657,   0.11967, |
    | 0.00806,   0.56428,   0.00806,   0.56428,   0.00806,   0.56428,   0.00806,   0.32512, |
    |-0.04657,  -0.23533,  -0.04657,  -0.23533,  -0.04657,  -0.23533,  -0.04657,   0.11967, |
    | 0.00806,   0.56428,   0.00806,   0.56428,   0.00806,   0.56428,   0.00806,   0.32512, |
    |-0.04657,  -0.23533,  -0.04657,  -0.23533,  -0.04657,  -0.23533,  -0.04657,   0.11967, |
    | 0.00806,   0.56428,   0.00806,   0.56428,   0.00806,   0.56428,   0.00806,   0.32512, |
    |-0.04657,  -0.23533,  -0.04657,  -0.23533,  -0.04657,  -0.23533,  -0.04657,   0.11967, |
    | 0.13881,   0.22965,   0.13881,   0.22965,   0.13881,   0.22965,   0.13881,   0.44798, |
  }
}
```

Next Steps

- Finish the implementation of U-Net
- HDF5 implementation in MagmaDNN
- ResU-Net implementation

Acknowledgements/References

This project was sponsored by the National Science Foundation through Research Experience for Undergraduates (REU) award, with additional support from the Joint Institute for Computational Sciences at University of Tennessee Knoxville. This project used allocations from the Extreme Science and Engineering Discovery Environment (XSEDE), which is supported by the National Science Foundation. In addition, the computing work was also performed on technical workstations donated by the BP High Performance Computing Team.

Ronneberger, Fischer, P., & Brox, T. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. In Medical Image Computing and Computer-Assisted Intervention MICCAI 2015 (pp. 234241). Springer International Publishing.

Nichols, Wong, K., Tomov, S., Ng, L., Chen, S., & Gessinger, A. (2019). MagmaDNN: Accelerated Deep Learning Using MAGMA. Proceedings of the Practice and Experience in Advanced Research Computing on Rise of the Machines (learning), 16. https://doi.org/10.1145/3332186. 3333047

Dumoulin, & Visin, F. (2016). A guide to convolution arithmetic for deep learning.