

## 1. INTRODUCTION

Building off the artificial neural network library created in Project 1, several modifications are introduced to build a convolutional neural network. Specifically, several classes are introduced, namely

`ConvolutionalLayer`

`MaxPoolingLayer`

`FlattenLayer`

An additional method `addLayer` is introduced in the `NeuralNetwork` class to allow for a variable number of layers in the neural network with different layer types.

In this project, the convolution layer is restricted to 2-dimensional convolution with a stride of 1 and padding is set to 'valid'. The max pooling layer is also restricted to 2-dimensional with the stride is always the same as the filter size and no padding is needed.

The options for activation function are logistic function (sigmoid) and linear function. The options for loss function are mean squared error and binary cross entropy.

Code examples are run using the TensorFlow with the keras API (`tf.keras`) to validate the correctness of the convolutional neural network library. The input, output, weights and bias are generated using `parameter.py` with a random seed specified so the results are reproducible.

## 2. ASSUMPTIONS/CHOICES MADE

- The input for each layer type (except for the fully connected layer) has a dimension of  $(w, h, c)$  where  $w$  is the width of the input,  $h$  is the height of the matrix and  $c$  is the number of input channels.
- The fully connected layer accepts an input with a dimension of  $(n, m)$ , where  $n$  is the number of neurons in the fully connected layer and  $m$  is the size of the flattened input.
- The kernel for each convolutional layer has a dimension of  $(k, kw, kh)$  where  $k$  is the number of kernel,  $kw$  is the width of the kernel and  $kh$  is the height of the kernel.
- The argument of the `addLayer` method accepts 4 layer types, namely:
  - "Conv2D" : 2-dimensional convolutional layer
  - "Dense" : fully connected layer
  - "MaxPooling2D" : 2-dimensional max pooling layer
  - "Flatten" : flatten layer

## 3. PROBLEMS/ISSUES

There is a slight discrepancy between the solution of the project code and the solution of the TensorFlow keras API. This may be due to round-off errors.

## 4. HOW TO RUN THE CODE

You can run the project code with examples using the following command:

```
$ python project2.py [example1|example2|example3]
```

You can choose one example to run with either example1, example2 or example3.

To run the code examples with the TensorFlow keras API, use the following commands:

```
$ python tensorflowtest_example1.py
```

```
$ python tensorflowtest_example2.py
```

```
$ python tensorflowtest_example3.py
```

## 5. RESULTS

The results from each run of one-step backpropagation using the project code are compared to the results from one run of Tensorflow keras API with the same input paramters.

The code examples are:

- Example 1: Run a network with a  $5 \times 5$  input, one  $3 \times 3$  convolutional layer with a single kernel, a flatten layer and a single neuron for the output.
- Example 2: Run a network with a  $7 \times 7$  input, one  $3 \times 3$  convolutional layer with two kernels, another  $3 \times 3$  convolutional layer with a single kernel, a flatten layer and a single neuron for the output.
- Example 3: Run a network with an  $8 \times 8$  input, one  $3 \times 3$  convolutional layer with two kernels, a  $2 \times 2$  max pooling layer, a flatten layer and a single neuron for the output.

```
((base) suannchong@Su-Anns-MacBook-Pro code % python project2.py example1
-----
EXAMPLE 1
-----
conv2d weights, num_kernels: (1, 2) 1
model output before:
[0.9758]
model output after
[0.97251]
loss: 0.09609416349119977
1st convolutional layer, 1st kernel weights:
[[0.37354 0.94967 0.73034]
 [0.59744 0.15461 0.15435]
 [0.05721 0.8651 0.59871]]
1st convolutional layer, 1st kernel bias:
0.7045946404142284
fully connected layer weights:
[0.007419985687200998 0.9565426348619112 0.8181649701462376
 0.19884386167081952 0.16817186588397992 0.16954039486719252
 0.29099048154164586 0.5111478005915057 0.41757290790107326]
fully connected layer bias:
0.2764318209365368

((base) suannchong@Su-Anns-MacBook-Pro code % python tensorflowtest_example1.py
2021-03-06 22:42:13.328685: I tensorflow/core/platform/cpu_feature_guard.cc:143
] Your CPU supports instructions that this TensorFlow binary was not compiled t
o use: AVX2 FMA
2021-03-06 22:42:13.358170: I tensorflow/compiler/xla/service/service.cc:168] X
LA service 0x7faee4684d0 initialized for platform Host (this does not guarante
e that XLA will be used). Devices:
2021-03-06 22:42:13.358201: I tensorflow/compiler/xla/service/service.cc:176]
StreamExecutor device (0): Host, Default Version
example1 output before
[0.66252228]
-----
EXAMPLE 1
-----
model output before:
[[0.9758]]
1/1 [=====] - 0s 349us/step - loss: 0.0981 - accuracy:
0.0000e+00
model output after:
[[0.97251]]
1st convolutional layer, 1st kernel weights:
[[0.37354 0.94967 0.73034]
 [0.59744 0.15461 0.15435]
 [0.05721 0.8651 0.59871]]
1st convolutional layer, 1st kernel bias:
0.7045947
fully connected layer weights:
[0.00742 0.95654 0.81816 0.19884 0.16817 0.16954 0.29099 0.51115 0.41757]
fully connected layer bias:
0.2764318
```

FIGURE 1. Screenshots of the project solution (on left) and the TensorFlow keras API solution (on right) for Example 1.

```
((base) suannchong@Su-Anns-MacBook-Pro code % python project2.py example2)

=====
EXAMPLE 2
=====

conv2d weights, num_kernels: (2, 2) 2
conv2d weights, num_kernels: (1, 2) 1
model output before:
[0.99653]
model output after
[0.99636]
loss: 0.4598039079374514
1st convolutional layer, 1st kernel weights:
[[0.77132 0.02075 0.63365]
 [0.7488 0.49851 0.2248 ]
 [0.19806 0.76053 0.16911]]
1st convolutional layer, 1st kernel bias:
0.9177707984641161
1st convolutional layer, 2nd kernel weights:
[[0.08834 0.68536 0.95339]
 [0.00395 0.51219 0.81262]
 [0.61252 0.72175 0.29187]]
1st convolutional layer, 2nd kernel bias:
0.7145724593488633
2nd convolutional layer weights:
[[0.54254 0.14216 0.37334]
 [0.67413 0.44183 0.43401]
 [0.61776 0.51313 0.65039]]
[[0.60103 0.80522 0.52164]
 [0.90864 0.31923 0.09045]
 [0.30069 0.11398 0.82868]]
2nd convolutional layer bias:
0.046890298029266794
fully connected layer weights:
[0.6215941307767051 0.5428931905412971 0.81459389551866
 0.19425447032487714 0.8521572816271012 0.3469595071752548
 0.7499546157784657 0.2912685994295055 0.8792433289612503]
fully connected layer bias:
0.3208174387801674
```

```
((base) suannchong@Su-Anns-MacBook-Pro code % python tensorflowtest_example2.py
2021-03-06 22:40:45.875074: I tensorflow/core/platform/cpu_feature_guard.cc:143
] Your CPU supports instructions that this TensorFlow binary was not compiled t
o use: AVX2 FMA
2021-03-06 22:40:45.889029: I tensorflow/compiler/xla/service/service.cc:168] X
LA service 0x7fa7ee469490 initialized for platform Host (this does not guarante
e that XLA will be used). Devices:
2021-03-06 22:40:45.889078: I tensorflow/compiler/xla/service/service.cc:176]
StreamExecutor device (0): Host, Default Version
weights for two kernels: (3, 3, 2, 1)

=====
EXAMPLE 2
=====

model output before:
[[0.99653]]
1/1 [=====] - 0s 364us/step - loss: 0.4600 - accuracy:
0.0000e+00
model output after:
[[0.99636]]
1st convolutional layer, 1st kernel weights:
[[0.77132 0.02075 0.63365]
 [0.7488 0.49851 0.2248 ]
 [0.19806 0.76053 0.16911]]
1st convolutional layer, 1st kernel bias:
0.9177725
1st convolutional layer, 2nd kernel weights:
[[0.08834 0.68536 0.95339]
 [0.00395 0.51219 0.81262]
 [0.61253 0.72175 0.29188]]
1st convolutional layer, 2nd kernel bias:
0.71457416
2nd convolutional layer weights:
[[0.54254 0.14216 0.37334]
 [0.67413 0.44183 0.43401]
 [0.61776 0.51313 0.65039]]
[[0.60103 0.80522 0.52164]
 [0.90864 0.31923 0.09045]
 [0.30069 0.11398 0.82868]]
2nd convolutional layer bias:
0.0468903
fully connected layer weights:
[0.62159 0.54289 0.81459 0.19425 0.85216 0.34696 0.74995 0.29127 0.87924]
fully connected layer bias:
0.3208175
```

FIGURE 2. Screenshots of the project solution (on left) and the TensorFlow keras API solution (on right) for Example 2.

```
((base) suannchong@Su-Anns-MacBook-Pro code % python project2.py example3)

=====
EXAMPLE 3
=====

model output before:
[0.99971]
model output after
[0.99971]
loss: 0.24121534402723444
1st convolutional layer, 1st kernel weights:
[[0.37448 0.95065 0.73194]
 [0.59859 0.15596 0.15593]
 [0.05801 0.86611 0.60106]]
1st convolutional layer, 1st kernel bias:
0.4318100814515682
1st convolutional layer, 2nd kernel weights:
[[0.70801 0.02052 0.96985]
 [0.83238 0.21228 0.18176]
 [0.18333 0.30417 0.5247 ]]
1st convolutional layer, 2nd kernel bias:
0.29109420300749433
fully connected layer weights:
[0.6116021278858128 0.13922343611700336 0.2918731901145876
 0.36609833112006496 0.4557963969769421 0.7848963787908407
 0.19941938158281974 0.5139723750418981 0.5921499641127843
 0.046184228224449744 0.6072775915403131 0.170249153141332
 0.0647803171856808 0.9486099365902736 0.9653647507281036
 0.8081240173260219 0.3043779432267761 0.09740270845806717]
fully connected layer bias:
0.6839471885830961
```

```
((base) suannchong@Su-Anns-MacBook-Pro code % python tensorflowtest_example3.py
2021-03-06 23:00:55.751012: I tensorflow/core/platform/cpu_feature_guard.cc:143
] Your CPU supports instructions that this TensorFlow binary was not compiled t
o use: AVX2 FMA
2021-03-06 23:00:55.767988: I tensorflow/compiler/xla/service/service.cc:168] X
LA service 0x7fca11915ba0 initialized for platform Host (this does not guarante
e that XLA will be used). Devices:
2021-03-06 23:00:55.768020: I tensorflow/compiler/xla/service/service.cc:176]
StreamExecutor device (0): Host, Default Version

=====
EXAMPLE 3
=====

model output before:
[[0.99971]]
1/1 [=====] - 0s 337us/step - loss: 0.2412 - accuracy:
0.0000e+00
model output after:
[[0.99971]]
1st convolutional layer, 1st kernel weights:
[[0.37451 0.95068 0.73196]
 [0.59862 0.156 0.15597]
 [0.05807 0.86615 0.60109]]
1st convolutional layer, 1st kernel bias:
0.43188888
1st convolutional layer, 2nd kernel weights:
[[0.70801 0.02055 0.96987]
 [0.83239 0.2123 0.1818 ]
 [0.18338 0.30422 0.52473]]
1st convolutional layer, 2nd kernel bias:
0.29115108
fully connected layer weights:
[0.61159 0.13924 0.29187 0.3661 0.45579 0.78491 0.19941 0.51398 0.59215
 0.04618 0.60727 0.17025 0.06478 0.94863 0.96536 0.80813 0.30435 0.09741]
fully connected layer bias:
0.6839487
```

FIGURE 3. Screenshots of the project solution (on left) and the TensorFlow keras API solution (on right) for Example 3.