Project 2

CS 525: Deep Learning

Jerry Duncan
jdunca51@vols.utk.edu

February 23, 2020

1 Introduction

For this project, we're extending our previous project to now work on image-like data. More specifically, we've added Convolutional, Flatten, and Max Pooling layers so that we can create a Convolutional Neural Network. This means that we're not able to solve a larger range of problems with our library than when it was just a Fully Connected Neural Network. We can now tackle image data as well as unique problems that benefit from convolving over the input.

2 Assumptions / Choices Made

The only differences from the stated requirements and my code is as follows:

- My addLayer function has been renamed add
- The shape of my weights has been finagled to match those of Keras

3 Problems / Issues

Nothing to add.

4 Running My Code

Run "python main.py <example#>". That's the driver file, the rest is the library.

5 Weight Comparison

In Figure 1, 2, and 3 are my weight comparisons. On the left is the weights before backprop, showing that they've been set to be equal between both models. The right is after backprop to show they're still the same, except for some small rounding errors between when and where the two models truncate their values. This also helps show that my model is my own code since they're not quite exactly equal. This is especially noticeable in the biases).

```
Example 1 Before Backprop
NN Layer 1 Kernel (Conv2D)
[[-1.0833623  -1.4984878  -0.5366443 ]
 [-0.50480884 -1.273768   -0.38621536]
 [ 0.3911237  -1.2641213   1.1121072 ]]
Keras Layer 1 Kernel (Conv2D)
[[-1.0833623  -1.4984878  -0.5366443 ]
 [-0.50480884 -1.273768   -0.38621536]
 [ 0.3911237  -1.2641213   1.1121072 ]]
NN Layer 1 Bias (Conv2D)
[0.66047106]
Keras Layer 1 Bias (Conv2D)
[0.6604711]
NN Layer 3 Weights (Dense)
[[ 0.43606091]
 [-0.52013448]
 [ 0.22527573]
 [ 1.0518723 ]
 [-0.86437522]
 [-0.29550921]
 [ 0.88504199]
 [ 0.56854734]
 [ 0.62414245]]
Keras Layer 3 Weights
[[ 0.4360609 ]
 [-0.5201345 ]
 [ 0.22527573]
 [ 1.0518723 ]
 [-0.86437523]
 [-0.29550922]
 [ 0.885042  ]
 [ 0.5685473 ]
 [ 0.62414247]]
NN Layer 3 Bias (Dense)
[0.00958371]
Keras Layer 3 Bias (Dense)
[0.00958371]
```

```
Example 1 After Backprop
NN Layer 1 Kernel (Conv2D)
[[-1.0825901  -1.4976834  -0.53526086]
 [-0.50653726 -1.2745653  -0.38645843]
 [ 0.39128095 -1.2640626   1.1121176 ]]
Keras Layer 1 Kernel (Conv2D)
[[-1.0825901  -1.4976834  -0.53526086]
 [-0.50653726 -1.2745653  -0.38645843]
 [ 0.39128095 -1.2640626   1.1121176 ]]
NN Layer 1 Bias (Conv2D)
[0.66182407]
Keras Layer 1 Bias (Conv2D)
[0.6618241]
NN Layer 3 Weights (Dense)
[[ 0.43669759]
 [-0.52009726]
 [ 0.22758374]
 [ 1.05195494]
 [-0.86421859]
 [-0.29409906]
 [ 0.88769053]
 [ 0.5689954 ]
 [ 0.62554939]]
Keras Layer 3 Weights
[[ 0.43669757]
 [-0.5200973 ]
 [ 0.22758374]
 [ 1.0519549 ]
 [-0.8642186 ]
 [-0.29409906]
 [ 0.88769054]
 [ 0.56899536]
 [ 0.6255494 ]]
NN Layer 3 Bias (Dense)
[0.01300284]
Keras Layer 3 Bias (Dense)
[0.01300284]
```

Figure 1: Weight comparison for example 1.

```
Example 2 Before Backprop
NN Layer 1 Kernel (Conv2D)
[[-1.1205736   0.2903458   0.09528793]
 [ 1.6349801   0.6784291   0.98520786]
 [-1.7223295  -1.378373    0.49927405]]
Keras Layer 1 Kernel (Conv2D)
[[-1.1205736   0.2903458   0.09528793]
 [ 1.6349801   0.6784291   0.98520786]
 [-1.7223295  -1.378373    0.49927405]]
NN Layer 1 Bias (Conv2D)
[1.61310414]
Keras Layer 1 Bias (Conv2D)
[1.6131041]
NN Layer 2 Kernel (Conv2D)
[[-1.2035196   1.3658456   1.1663582 ]
 [-0.79283404 -0.6173138   1.4708627 ]
 [ 1.041199    0.46550116  0.0999433 ]]
Keras Layer 2 Kernel (Conv2D)
[[-1.2035196   1.3658456   1.1663582 ]
 [-0.79283404 -0.6173138   1.4708627 ]
 [ 1.041199    0.46550116  0.0999433 ]]
NN Layer 2 Bias (Conv2D)
[-1.24537419]
Keras Layer 2 Bias (Conv2D)
[-1.2453742]
NN Layer 4 Weights (Dense)
[[2.02287626]]
Keras Layer 4 Weights
[[2.0228763]]
NN Layer 4 Bias (Dense)
[0.01785838]
Keras Layer 4 Bias (Dense)
[0.01785838]
```

```
Example 2 After Backprop
NN Layer 1 Kernel (Conv2D)
[[-1.1198242   0.29057083  0.09529074]
 [ 1.634868    0.6785071   0.98532534]
 [-1.7225615  -1.3788362   0.49922964]]
Keras Layer 1 Kernel (Conv2D)
[[-1.1198242   0.29057083  0.09529074]
 [ 1.634868    0.6785071   0.98532534]
 [-1.7225615  -1.3788362   0.49922964]]
NN Layer 1 Bias (Conv2D)
[1.61279287]
Keras Layer 1 Bias (Conv2D)
[1.6127928]
NN Layer 2 Kernel (Conv2D)
[[-1.20591     1.3635455   1.1662606 ]
 [-0.79489404 -0.619655    1.4686    ]
 [ 1.0388285   0.46329924  0.09930083]]
Keras Layer 2 Kernel (Conv2D)
[[-1.20591     1.3635455   1.1662606 ]
 [-0.79489404 -0.619655    1.4686    ]
 [ 1.0388285   0.46329924  0.09930083]]
NN Layer 2 Bias (Conv2D)
[-1.24777229]
Keras Layer 2 Bias (Conv2D)
[-1.2477723]
NN Layer 4 Weights (Dense)
[[2.01973637]]
Keras Layer 4 Weights
[[2.0197363]]
NN Layer 4 Bias (Dense)
[0.01281392]
Keras Layer 4 Bias (Dense)
[0.01281392]
```

Figure 2: Weight comparison for example 2.

```
Example 3 Before Backprop
NN Layer 1 Kernel 1 (Conv2D)
[[-0.97741055 -0.44822663  0.4270769 ]
 [-0.45838857  0.22617835  0.41849762]
 [-1.1198361   0.99857247 -1.734879  ]]
Keras Layer 1 Kernel 1 (Conv2D)
[[-0.97741055 -0.44822663  0.4270769 ]
 [-0.45838857  0.22617835  0.41849762]
 [-1.1198361   0.99857247 -1.734879  ]]
NN Layer 1 Kernel 2 (Conv2D)
[[-0.04933351  0.9276853  -0.72543037]
 [ 0.73875856  0.6632397  -0.8408866 ]
 [-0.16628678  0.20209618 -0.37767538]]
Keras Layer 1 Kernel 2 (Conv2D)
[[-0.04933351  0.9276853  -0.72543037]
 [ 0.73875856  0.6632397  -0.8408866 ]
 [-0.16628678  0.20209618 -0.37767538]]
NN Layer 1 Bias (Conv2D)
[1.83307462 0.4757754 ]
Keras Layer 1 Bias (Conv2D)
[1.8330746 0.4757754]
NN Layer 4 Weights (Dense)
[[-1.8392117 ]
 [ 0.34865464]
 [-0.22153025]
 [-0.04874927]
 [ 0.71899062]
 [ 1.00271691]
 [-0.19397537]
 [-1.31264242]]
Keras Layer 4 Weights
[[-1.8392117 ]
 [ 0.34865466]
 [-0.22153024]
 [-0.04874927]
 [ 0.7189906 ]
 [ 1.0027169 ]
 [-0.19397536]
 [-1.3126425 ]]
NN Layer 4 Bias (Dense)
[1.26580133]
Keras Layer 4 Bias (Dense)
[1.2658013]
```

```
Example 3 After Backprop
NN Layer 1 Kernel 1 (Conv2D)
[[-0.9780513  -0.44897848  0.42726558]
 [-0.4581885   0.22551183  0.4182107 ]
 [-1.1201798   0.9986849  -1.7344742 ]]
Keras Layer 1 Kernel 1 (Conv2D)
[[-0.9780513  -0.44897848  0.42726558]
 [-0.4581885   0.22551183  0.4182107 ]
 [-1.1201798   0.9986849  -1.7344742 ]]
NN Layer 1 Kernel 2 (Conv2D)
[[-0.04744424  0.92940146 -0.7250064 ]
 [ 0.73584354  0.66450065 -0.8427918 ]
 [-0.16584022  0.20105258 -0.3777557 ]]
Keras Layer 1 Kernel 2 (Conv2D)
[[-0.04744424  0.92940146 -0.7250064 ]
 [ 0.73584354  0.66450065 -0.8427918 ]
 [-0.16584022  0.20105258 -0.3777557 ]]
NN Layer 1 Bias (Conv2D)
[1.83355079 0.47478265]
Keras Layer 1 Bias (Conv2D)
[1.8335507  0.47478265]
NN Layer 4 Weights (Dense)
[[-1.84571416]
 [ 0.34219744]
 [-0.22832243]
 [-0.05300528]
 [ 0.71235587]
 [ 0.99866591]
 [-0.20081305]
 [-1.31889462]]
Keras Layer 4 Weights
[[-1.8457142 ]
 [ 0.34219745]
 [-0.22832242]
 [-0.05300529]
 [ 0.71235585]
 [ 0.9986659 ]
 [-0.20081304]
 [-1.3188946 ]]
NN Layer 4 Bias (Dense)
[1.25894831]
Keras Layer 4 Bias (Dense)
[1.2589483]
```

Figure 3: Weight comparison for example 3.