Matthew Walters

Steven Koprowicz

Convolutional Neural Networks

1. In this project, we were tasked with modifying our previous neural network code so that we can add layers to the network after it's initialization, and to add convolutional, max-pooling, and flattening layers to the network, not just fully connected layers. Once we have this code, we need to compare its outputs to a network of the same construction made through TensorFlow's Keras library.

2. In making our code for convolutional and max-pooling layers, we figured it would be easier to implement everything together if the convolutional layers were treated as 2-dimensional in their own functions, but from any other object's perspective, the inputs and outputs of the convolutional layer were in 1-dimensional lists. This did make some things easier and possibly more efficient, but it made flatten layer a little complicated.

3. We did not finish the Max Pooling implementation and we had issues with translating between convolutional layers and fully connected layers. We didn't quite finish any of the examples with our own code, they don't have the same outputs as the ones in Keras.

4. Our code works like the write-up describes, except that you flatten the weights and append the bias to the weights before passing them to a convolutional layer in addLayer.

5. Example 1 from Keras

```
model output before:
1/1 [==============================] - 0s 75ms/step
[[0.99239]]
WARNING:absl:`lr` is deprecated, please use `learning_rate` instead, or use the
model output after:
1/1 [==============================] - 0s 45ms/step
[[0.99238]]
1st convolutional layer, 1st kernel weights:
[[0.61776 0.51313 0.65039]
 [0.60103 0.80521 0.52164]
 [0.90864 0.31923 0.09045]]
1st convolutional layer, 1st kernel bias:
0.30067843
fully connected layer weights:
[0.1139  0.8286  0.04682 0.62621 0.54751 0.81921 0.19887 0.85677 0.35157]
fully connected layer bias:
0.75456333
```

Example 2 from Keras

```
model output before:
1/1 [==============================] - 0s 109ms/step
[[0.99653]]
WARNING:absl:`lr` is deprecated, please use `learning_rate` instead, or use
model output after:
1/1 [==============================] - 0s 49ms/step
[[0.99653]]
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:
0.91777414
1st convolutional layer, 2nd kernel weights:
[[0.08834 0.68536 0.95339]
 [0.00395 0.51219 0.81262]
 [0.61253 0.72176 0.29188]]
1st convolutional layer, 2nd kernel bias:
0.71457577
2nd convolutional layer weights:
[[0.54254 0.37334 0.44183]
 [0.61777 0.6504  0.80522]
 [0.90865 0.09046 0.11398]]
[[0.14217 0.67413 0.43401]
 [0.51314 0.60104 0.52165]
 [0.31924 0.3007  0.82868]]
2nd convolutional layer bias:
0.04689626
fully connected layer weights:
[0.62624 0.54754 0.81924 0.1989  0.8568  0.35161 0.7546  0.29591 0.88389]
fully connected layer bias:
0.3254647
```

Example 3 from Keras

```
model output before:
1/1 [==============================] - 0s 109ms/step
[[0.99965]]
WARNING:absl:`lr` is deprecated, please use `learning_rate` instead, or use
model output after:
1/1 [==============================] - 0s 48ms/step
[[0.99965]]
1st convolutional layer, 1st kernel weights:
[[0.35718 0.07961 0.30546]
 [0.33072 0.77383 0.03996]
 [0.42949 0.31493 0.63649]]
1st convolutional layer, 1st kernel bias:
0.34634447
fully connected layer weights:
[0.02539 0.30306 0.24207 0.55757 0.5655  0.47513 0.29279 0.06424 0.97881
 0.3397  0.49504 0.97707 0.44077 0.31827 0.51979 0.57813 0.85393 0.06809]
fully connected layer bias:
0.46452412
```

Example 1 from Our Program

```
Running Example 1
BEFORE TRAINING
network.layers[0].weights=[[0.2209216157140511, 0.4072915589537597, 0.855408569791041, 0.8901003799214978, 0.46504363767582724, 0.7608982750741918, 0.3010993073656475, 0.5778973185847075, 0.8222654082653019, 0.5
276463590414521]]

network.layers[2].weights=[0.5721610903883384, 0.1314254137803099, 0.38574540690762904, 0.16369047212477134, 0.34436793467425464, 0.36457544958570565, 0.7190642369684767, 0.13196736695198183, 0.4435937427453107,
 0.3519414974741507]

network.calculate(input)=[[0.9696328281793758]]
POST TRAINING
network.layers[0].weights=[[0.0015975701925638097, 0.0019838434726651088, 0.0018215191789455552, 0.0016637634828186281, 0.0015401158778622819, 0.001641788050269391, 0.0014387256564367483, 0.0015578656941190093,
0.0010922396257299278, 0.0040111759130787O9]]

network.layers[2].weights=[array([-1.28569174]), array([-1.78817697]), array([-1.56133321]), array([-1.66255636]), array([-1.56105469]), array([-1.55115682]), array([-1.20027777]), array([-1.71034983]), array([-
1.46912623]), array([-1.68038486])]

network.calculate(input)=[[array([0.00018169])]]
```