Sneha Bandi
6557689144

Problem 2

Convolutional Layer Output size calculation

```
Received [32, 16, 16, 16] and expected [32, 16, 16, 16]
```

1. **Convolutional Layer**
   a. **Forward :**
   ```
   Received output shape: (1, 4, 4, 2), Expected output shape: (1, 4, 4, 2)
   Difference:  5.110565335399418e-08
   ```
   b. **Backward**
   ```
   dimg Error:  1.6670134158491315e-08
   dw Error:  1.056723541487401e-08
   db Error:  1.1157795914175144e-10
   dimg Shape:  (15, 8, 8, 3) (15, 8, 8, 3)
   ```

2. **Pooling Layer (Max)**
   a. **Forward**
   ```
   Received output shape: (1, 3, 3, 1), Expected output shape: (1, 3, 3, 1)
   Difference:  1.8750000280978013e-08
   ```
   b. **Backward**
   ```
   dimg Error:  3.276186843072994e-12
   dimg Shape:  (15, 8, 8, 3) (15, 8, 8, 3)
   ```

3. **Test Small Convolutional Neural Network**
```
Testing initialization ...
Passed!
Testing test-time forward pass ...
Passed!
Testing the loss ...
Passed!
Testing the gradients (error should be no larger than 1e-6) ...
conv1_b relative error: 1.01e-09
conv1_w relative error: 1.05e-09
fc1_b relative error: 3.65e-10
fc1_w relative error: 3.95e-07

Param names : Conv1_w, Conv1_b, fc1_w, fc1_b
```

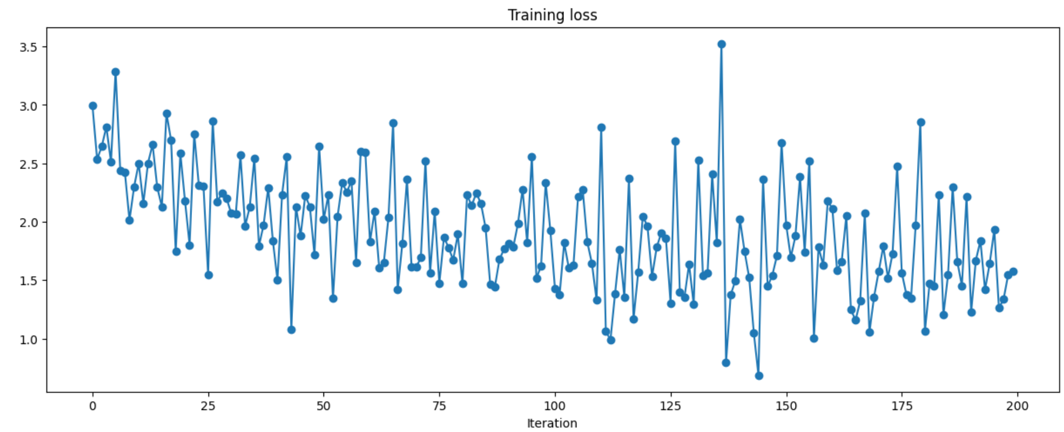
4. **Train a Network : Conv->Max Pool->flatten->FC->GeLU->FC**

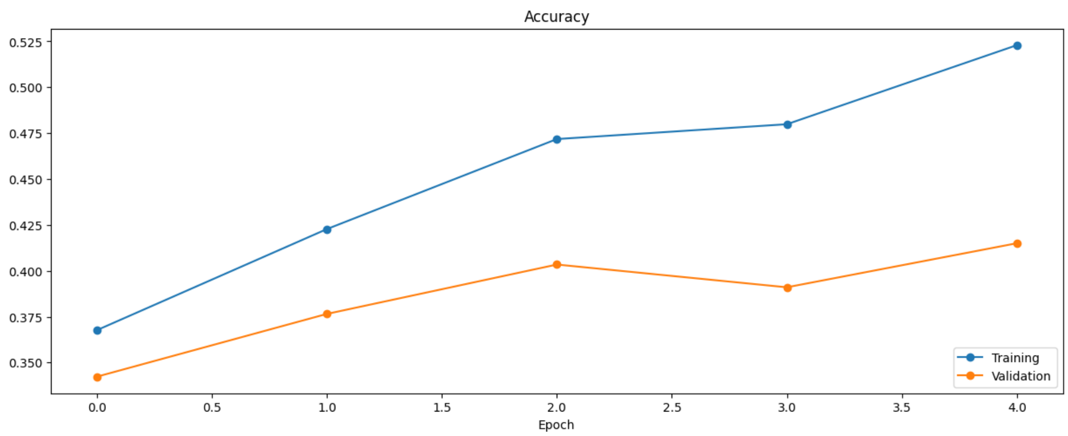
Input Data :

```
Data shape: (40000, 32, 32, 3)
Flattened data input size: 3072
Number of data classes: 20

Layers defined:
```

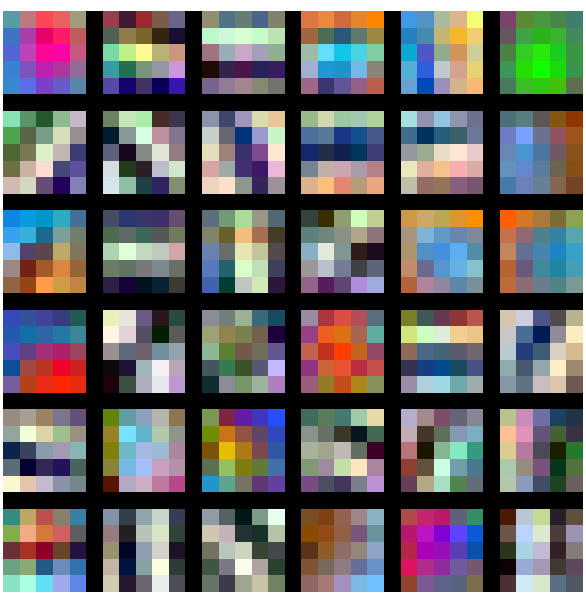
```
ConvLayer2D(input_channels=3, kernel_size=5, number_filters=36, stride=2,
padding=1, name="conv1", init_scale=.02)
MaxPoolingLayer(pool_size=2, stride=2, name="maxpool1"),
flatten(name="flatten"),
fc(1764, 48, 2e-2, name="fc1"),
gelu(name="gelu1"),
fc(48, 20, 2e-2, name="fc2")
```

```
(Iteration 1 / 20000) Average loss: 3.001034925522668

100%|██████████| 4000/4000 [2:04:13<00:00,  1.86s/it]

(Epoch 1 / 5) Training Accuracy: 0.36765, Validation Accuracy: 0.3423

  0%|          | 1/4000 [00:01<2:00:46,  1.81s/it]
(Iteration 4001 / 20000) Average loss: 2.3154061446075054

 40%|████      | 1584/4000 [1:11:03<1:08:09,  1.69s/it]
(Epoch 2 / 5) Training Accuracy: 0.422725, Validation Accuracy: 0.3765

  0%|          | 1/4000 [00:01<1:56:45,  1.75s/it]
(Iteration 8001 / 20000) Average loss: 1.9999308835295233

100%|██████████| 4000/4000 [1:50:08<00:00,  1.65s/it]

(Epoch 3 / 5) Training Accuracy: 0.471775, Validation Accuracy: 0.4034

  0%|          | 1/4000 [00:01<1:44:44,  1.57s/it]
(Iteration 12001 / 20000) Average loss: 1.867676979601489

100%|██████████| 4000/4000 [1:44:30<00:00,  1.57s/it]

(Epoch 4 / 5) Training Accuracy: 0.47985, Validation Accuracy: 0.391

  0%|          | 1/4000 [00:01<1:47:02,  1.61s/it]
(Iteration 16001 / 20000) Average loss: 1.769121363689662

100%|██████████| 4000/4000 [1:49:07<00:00,  1.64s/it]

(Epoch 5 / 5) Training Accuracy: 0.522925, Validation Accuracy: 0.415
```





5. Visualization Layer



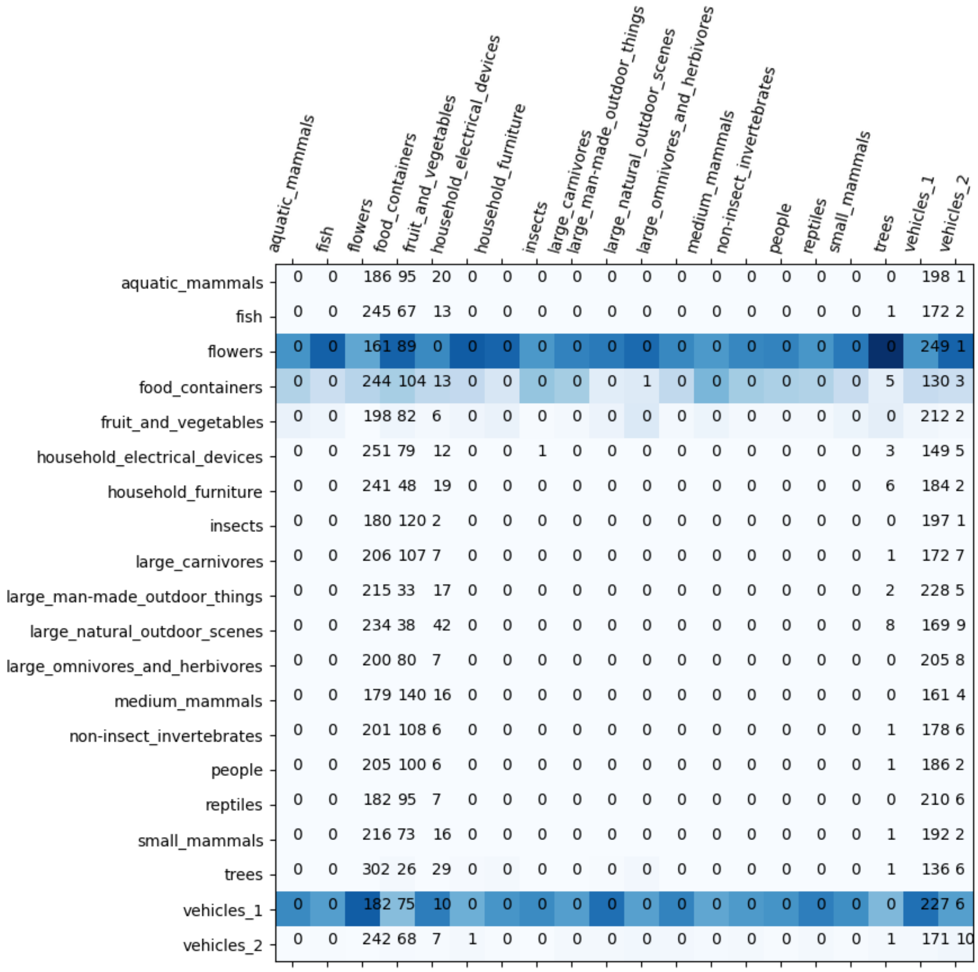
6. Visualization Inline Question :

```
The first layer of the model is defined as follows :
- (C) input_channels = 3
```

```
- (K) kernel_size = 5
- (N) number_filters=36
- (S) stride=2
- (P) padding=1
- Input image size = (32,32,3) => (input_height, input_width, channels)
- filter size = (5,5,3,36) => (K,K,C,N)

The above visualization depicts :
- 6x6 grid representing 36 filter
- Each small grid i.e. 5 x 5 map represents a color (r,g,b) matrix applied to the
input image.
- One square represents

Analyses of the above visualization :
- We can see that in some cases, the filter is the similar across the channels
(the first row), and in others, the filters differ (the last row)
- The darker squares indicate smaller weights and the lighter squares represent
larger weights.
- We can notice the top left denotes the color filter RED (r,g,b) = (0-255,0,0),
top right GREEN (r,g,b) = (0,0-255,0) and similarly BLUE.
- Hence, the filters on the first row detect a gradient from redder shades in the
top left to dark blacker in the bottom right.
```

7. Extra Credit

```
### EXTRA CREDIT Inline Answer:

- Plotted the [confusion matrix](https://en.wikipedia.org/wiki/Confusion_matrix)
of model's predictions on the test set.
- Justified different trends to see which classes are frequently misclassified as
other classes (e.g. are the two vehicle superclasses frequently confused with each
other?)
    - We can see in the below graph x-axis denotes expected classes and y denotes
predicted classes.
    - Square which are darker are the ones that hold more classification weight
for that particular (row-col) predicted-expected mapping. In ideal case along the
diagonal the values would be the darkest meaning correct classifications.
    - We mark the trend where there are darker blocks in the heat map of confusion
matrix, accounting for misclassifications. Examples of misclassification trends
are as follows :
        - 2, 18 i.e trees and flower (Because of the possible colow matrix being
similar)
        - Similarly 18,19 classes i.e. vehicle_1 and vehicle_2 are often mixed and
misclassified by the model, possibly because of object similarity
        - Similarly large_natural_outdoor_scenes and aquatic mammals are
misclassified, possibly because of close color patterns and arrangements (Attached
example after confusion matrix)


- Also plotted a pair plot between samples to check for any correlation between
them.
```


aquatic_mammals fish flowers food_containers fruit_and_vegetables household_electrical_devices household_furniture insects large_carnivores large_man-made_outdoor_things large_natural_outdoor_scenes large_omnivores_and_herbivores medium_mammals non-insect_invertebrates people reptiles small_mammals trees vehicles_1 vehicles_2
aquatic_mammals 0 0 186 95 20 0 0 0 0 0 0 0 0 0 0 0 0 0 198 1
fish 0 0 245 67 13 0 0 0 0 0 0 0 0 0 0 0 0 1 172 2
flowers 0 0 161 89 0 0 0 0 0 0 0 0 0 0 0 0 0 0 249 1
food_containers 0 0 244 104 13 0 0 0 0 0 1 0 0 0 0 0 0 5 130 3
fruit_and_vegetables 0 0 198 82 6 0 0 0 0 0 0 0 0 0 0 0 0 0 212 2
household_electrical_devices 0 0 251 79 12 0 0 1 0 0 0 0 0 0 0 0 0 3 149 5
household_furniture 0 0 241 48 19 0 0 0 0 0 0 0 0 0 0 0 0 6 184 2
insects 0 0 180 120 2 0 0 0 0 0 0 0 0 0 0 0 0 0 197 1
large_carnivores 0 0 206 107 7 0 0 0 0 0 0 0 0 0 0 0 0 1 172 7
large_man-made_outdoor_things 0 0 215 33 17 0 0 0 0 0 0 0 0 0 0 0 0 2 228 5
large_natural_outdoor_scenes 0 0 234 38 42 0 0 0 0 0 0 0 0 0 0 0 0 8 169 9
large_omnivores_and_herbivores 0 0 200 80 7 0 0 0 0 0 0 0 0 0 0 0 0 0 205 8
medium_mammals 0 0 179 140 16 0 0 0 0 0 0 0 0 0 0 0 0 0 161 4
non-insect_invertebrates 0 0 201 108 6 0 0 0 0 0 0 0 0 0 0 0 0 1 178 6
people 0 0 205 100 6 0 0 0 0 0 0 0 0 0 0 0 0 1 186 2
reptiles 0 0 182 95 7 0 0 0 0 0 0 0 0 0 0 0 0 0 210 6
small_mammals 0 0 216 73 16 0 0 0 0 0 0 0 0 0 0 0 0 1 192 2
trees 0 0 302 26 29 0 0 0 0 0 0 0 0 0 0 0 0 1 136 6
vehicles_1 0 0 182 75 10 0 0 0 0 0 0 0 0 0 0 0 0 0 227 6
vehicles_2 0 0 242 68 7 1 0 0 0 0 0 0 0 0 0 0 0 1 171 10