A2: CNN

Andrew Vasquez (avasquez56@csuchico.edu)

CSCI 611 - Applied Machine Learning Summer 2025

Bo Shen

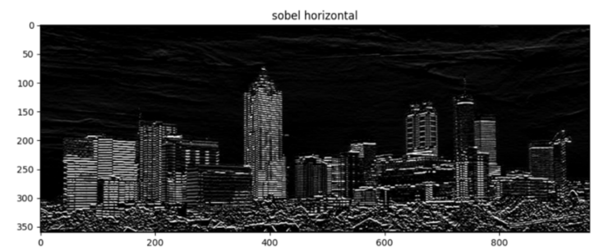
2025-Jun-15

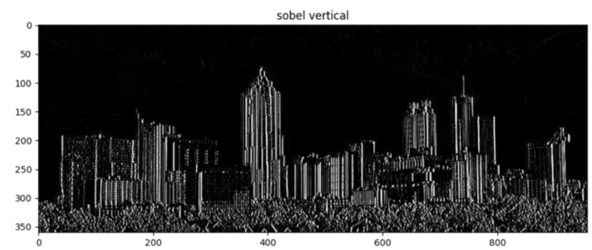
Part 1

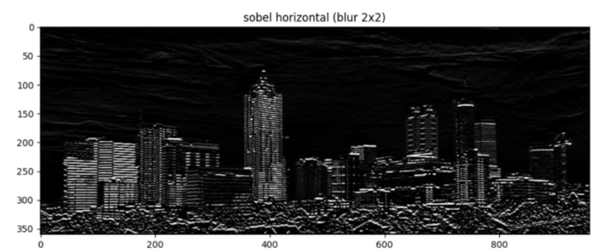
As instructed the original image was sampled with the Sobel operator to detector both vertical and horizontal edges. Implemented the code defines the following kernels:

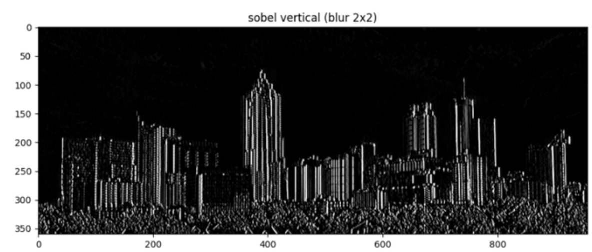
```
# Sobel operator
shorizon = np.array([[ -1, -2, -1],
                     [  0,  0,  0],
                     [  1,  2,  1]])
svertica = np.array([[ -1, 0, 1],
                     [ -2, 0, 2],
                     [ -1, 0, 1]])
```

Against the 2x2 blurred image, resulting in these images:







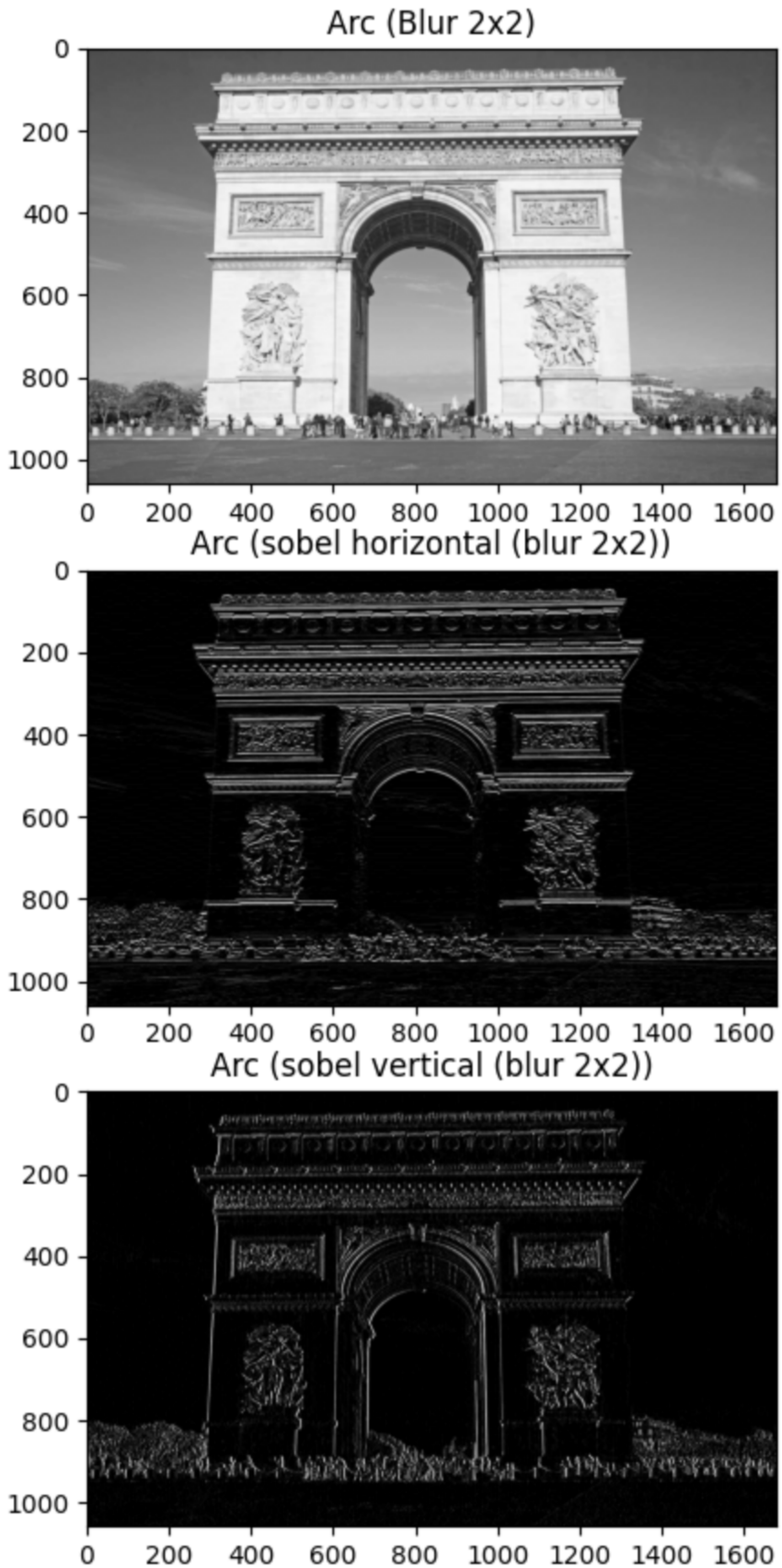


An additional image, arc.jpg was tested against the 2x2 and 4x4 blur resulting in the following results:

```
# Read in the Arc image
arc = mpimg.imread('arc.jpg')
…
```

2x2

```
# Blur image with previously defined 2x2 kernel
ablur22 = cv2.filter2D(agray, -1, S2x2/4.0)
fig.add_subplot(5,1,3)
plt.imshow(ablur22, cmap='gray')
plt.title('Arc (Blur 2x2)')
```

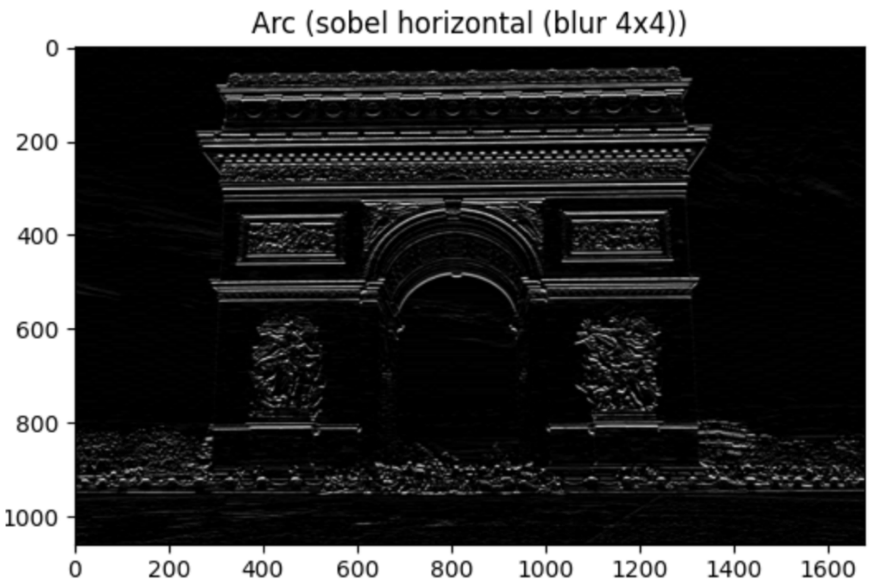


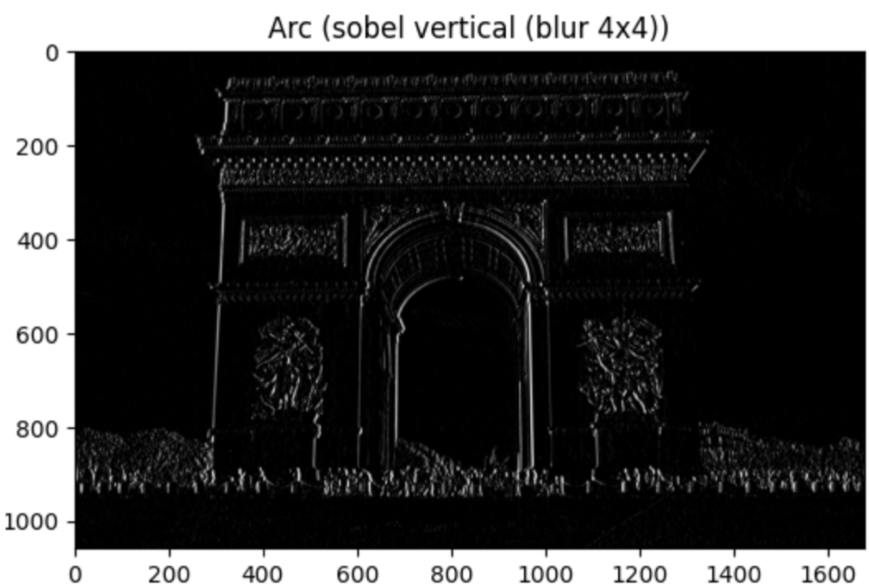
Arc (Blur 2x2)
0
200
400
600
800
1000
0 200 400 600 800 1000 1200 1400 1600
Arc (sobel horizontal (blur 2x2))
0
200
400
600
800
1000
0 200 400 600 800 1000 1200 1400 1600
Arc (sobel vertical (blur 2x2))
0
200
400
600
800
1000
0 200 400 600 800 1000 1200 1400 1600

4x4

```
# Blur with a 4x4 kernel
S4x4 = np.ones((4, 4), np.float32)

ablur44 = cv2.filter2D(agray, -1, S4x4/16.0)
fig.add_subplot(4,1,2)
plt.imshow(ablur44, cmap='gray')
plt.title('Arc (Blur 4x4)')
```





Part 2

After several iterations of results, final results with 66% resulted with the following CNN, a dual convolution mapping the 3-segment image to 32 regions, pooling, then mapping the resultant segments to 16, both against a 5x5 kernel. Two RELU first map the resultant to 320 datasets, then to 128 datapoints. Then final activation function then performs a Sigmoid transformation against a final 10 with a dropout of 0.25, resulting in:

Net definition

```
Net(
  (conv1): Conv2d(3, 32, kernel_size=(5, 5), stride=(1, 1))
  (pool): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
ceil_mode=False)
  (conv2): Conv2d(32, 16, kernel_size=(5, 5), stride=(1, 1))
  (fc1): Linear(in_features=400, out_features=320, bias=True)
  (fc2): Linear(in_features=320, out_features=128, bias=True)
  (fc3): Linear(in_features=128, out_features=10, bias=True)
  (dropout): Dropout(p=0.25, inplace=False)
)
```

Loss Optimizer

```
optimizer = optim.Adam(model.parameters(),lr=0.001)
```

Training

```
Epoch: 1     Training Loss: 1.513372   Validation Loss: 1.314815
Validation loss decreased (inf --> 1.314815).  Saving model ...
Epoch: 2     Training Loss: 1.214979   Validation Loss: 1.126815
Validation loss decreased (1.314815 --> 1.126815).  Saving model ...
Epoch: 3     Training Loss: 1.062462   Validation Loss: 1.091137
Validation loss decreased (1.126815 --> 1.091137).  Saving model ...
Epoch: 4     Training Loss: 0.953082   Validation Loss: 1.040521
Validation loss decreased (1.091137 --> 1.040521).  Saving model ...
Epoch: 5     Training Loss: 0.875411   Validation Loss: 1.024898
Validation loss decreased (1.040521 --> 1.024898).  Saving model ...
Epoch: 6     Training Loss: 0.798122   Validation Loss: 0.978446
Validation loss decreased (1.024898 --> 0.978446).  Saving model ...
Epoch: 7     Training Loss: 0.737508   Validation Loss: 0.985200
Epoch: 8     Training Loss: 0.680161   Validation Loss: 1.032954
Epoch: 9     Training Loss: 0.625557   Validation Loss: 1.053352
Epoch: 10    Training Loss: 0.569876   Validation Loss: 1.071555
```

Accuracy

```
Test Loss: 1.011217

Test Accuracy of airplane: 64% (646/1000)
Test Accuracy of automobile: 79% (795/1000)
Test Accuracy of  bird: 56% (569/1000)
Test Accuracy of   cat: 48% (484/1000)
```

```
Test Accuracy of  deer: 45% (459/1000)
Test Accuracy of   dog: 49% (497/1000)
Test Accuracy of  frog: 80% (801/1000)
Test Accuracy of horse: 69% (693/1000)
Test Accuracy of  ship: 84% (848/1000)
Test Accuracy of truck: 73% (738/1000)

Test Accuracy (Overall): 65% (6530/10000)
```

ADAM optimizer enabled results beyond 60%+, but beyond 10 epoch iterations, additional computations did not assist in identification.

Summary

Overall, this was an effort in guessing and experimentation. Pragmatic analysis was minimal, at this point, based on known forms and instruction.