Compositional Data

Convolutional Neural Networks

10.18.17

Jinghu Lei

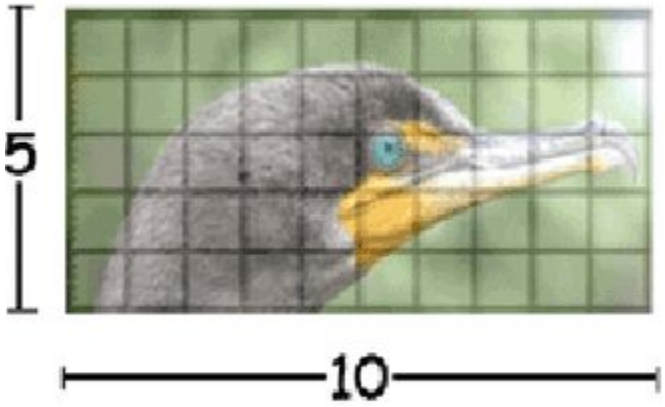
Compositional Data

- Compositional data is the **relative, qualitative aspects** of some entirety.
- They provide purely relative information and can be represented by positive vector components summing to a whole constant.
- For example, A population with 30% children, 20% adults, and 50% elderly could be represented as [0.3, 0.2, 0.5].

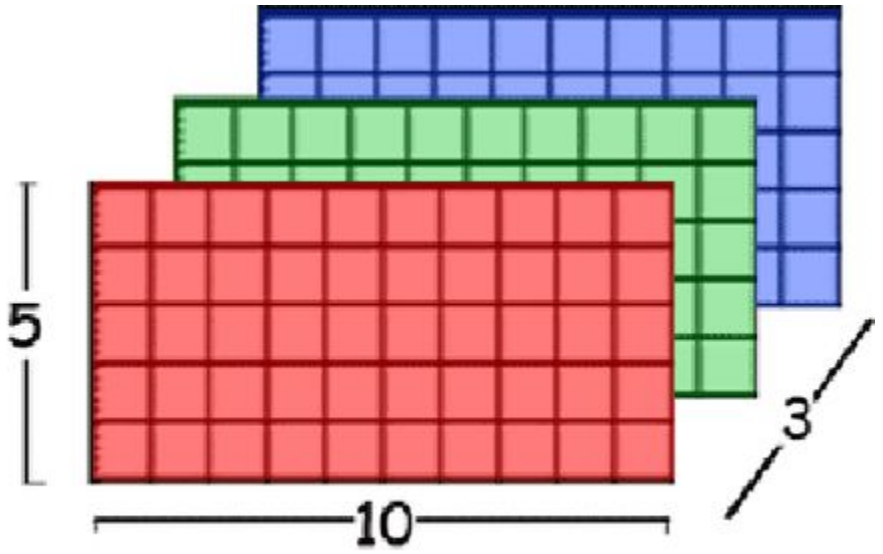
Compositional Data in Machine Learning

BOSTON UNIVERSITY
MACHINE INTELLIGENCE COMMUNITY

- When looking at real world objects, it can be difficult at first to represent it simply ‘on paper’.
- However, we can extract key qualitative values to represent the object. The qualities we look at will depend on what we need to implement a given function.
- We can implement images as a 3-D matrix of Red, Green, and Blue colors.



Original Color Image

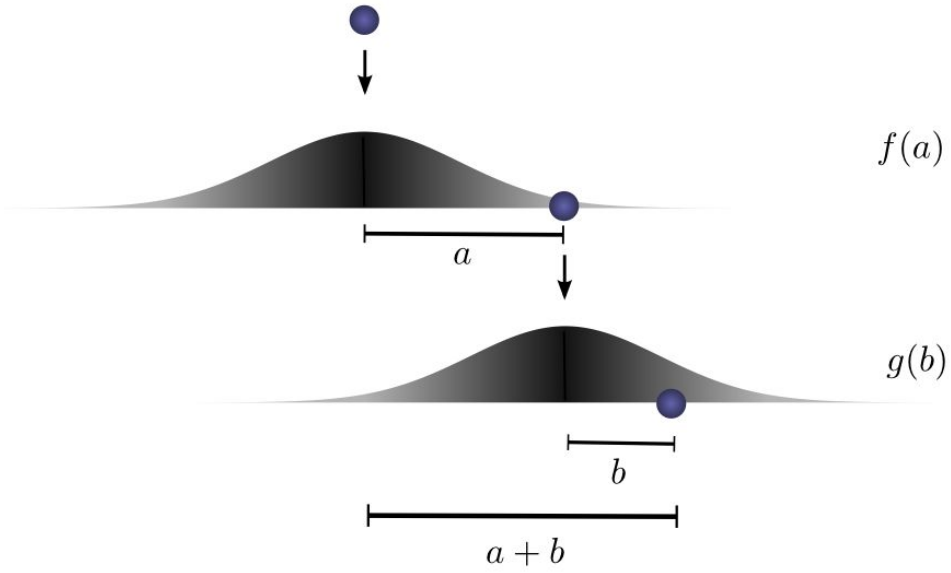


Matlab RGB Matrix

Convolutions

BOSTON UNIVERSITY
MACHINE INTELLIGENCE COMMUNITY

- Convolutions, in terms of mathematics, are the combining of two functions to form a third, resulting function.
- Consider the analogy of predicting the location of a ball after a drop followed by another drop from its initial landing location.

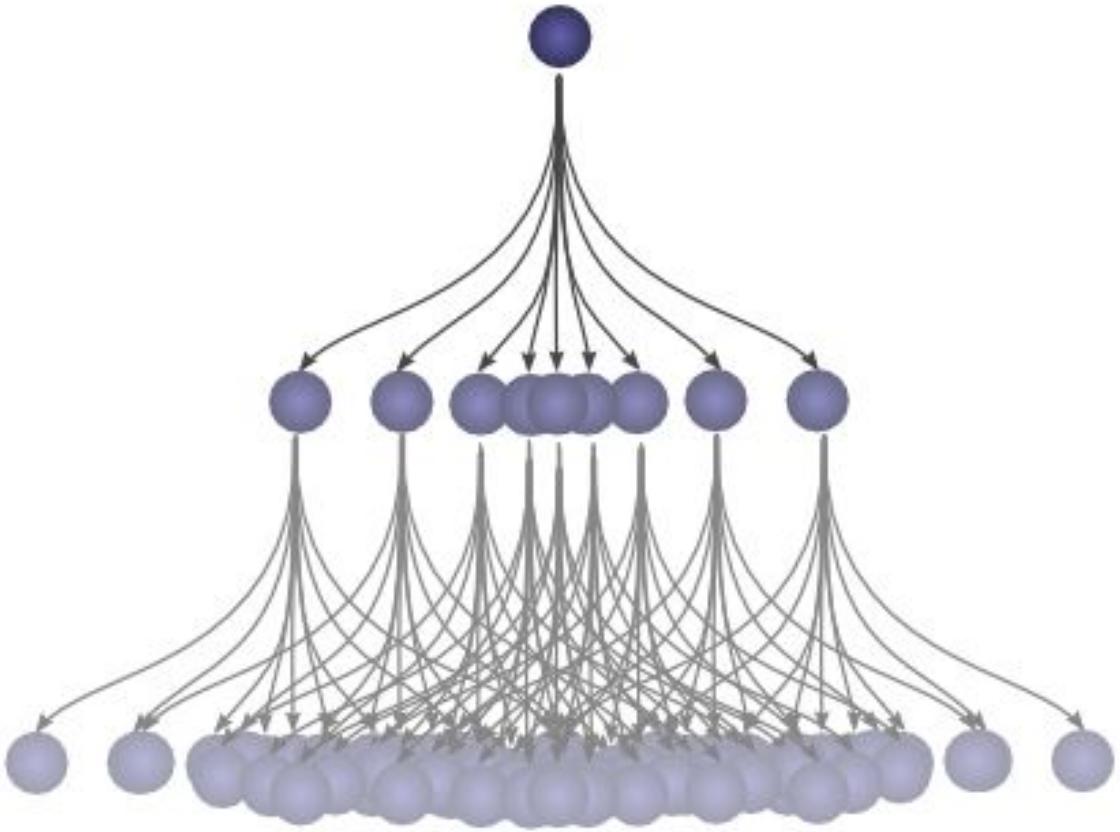


$$(f * g)(c) = \sum_a f(a) \cdot g(c-a)$$

Convolutions - 2

BOSTON UNIVERSITY
MACHINE INTELLIGENCE COMMUNITY

- Now consider a graph representing the different possibilities of outcomes:

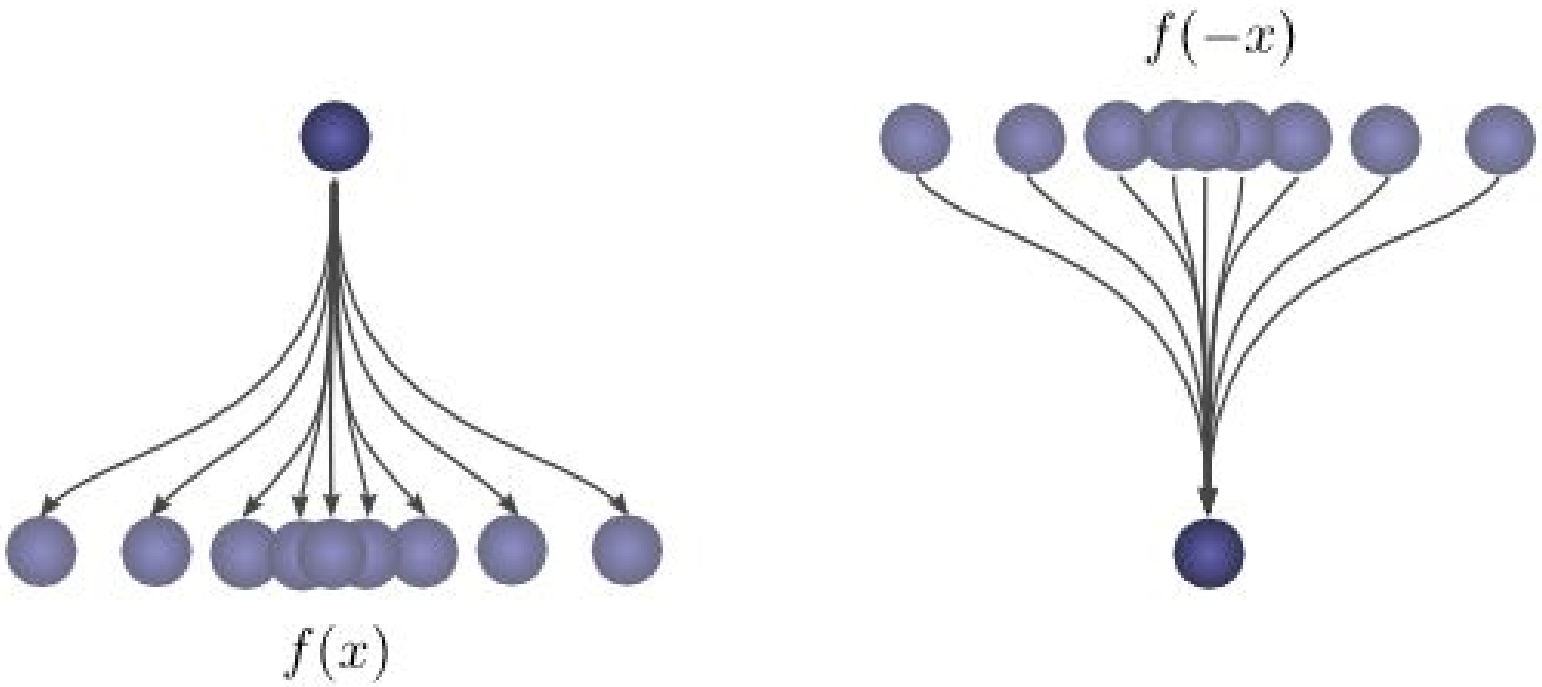


$f(a)$

$$\sum_a f(a)g(c-a)$$

Convolutions - 3

- Finally consider an inverse function to find the starting point of a dropped ball.



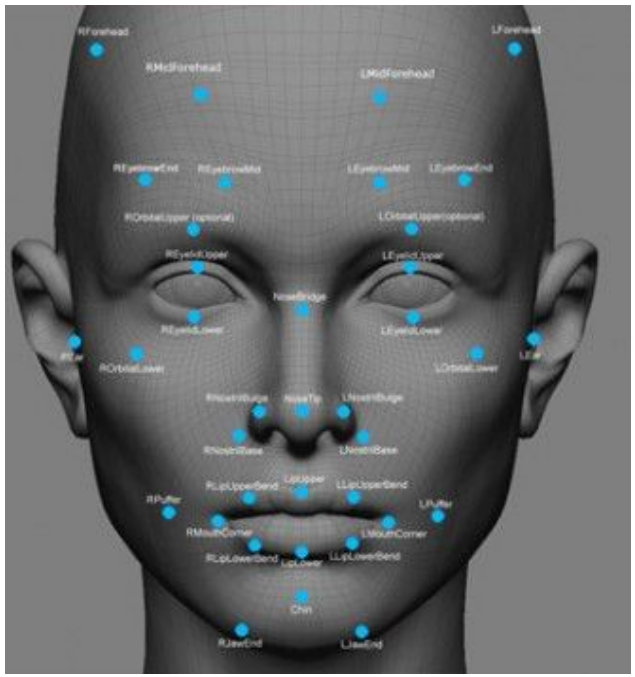
- Let *f(x)* be the probability function of landing in a location, then *f(-x)* is the probability, given an landing location, of the starting point.

Convolutional Neural Networks

BOSTON UNIVERSITY
MACHINE INTELLIGENCE COMMUNITY

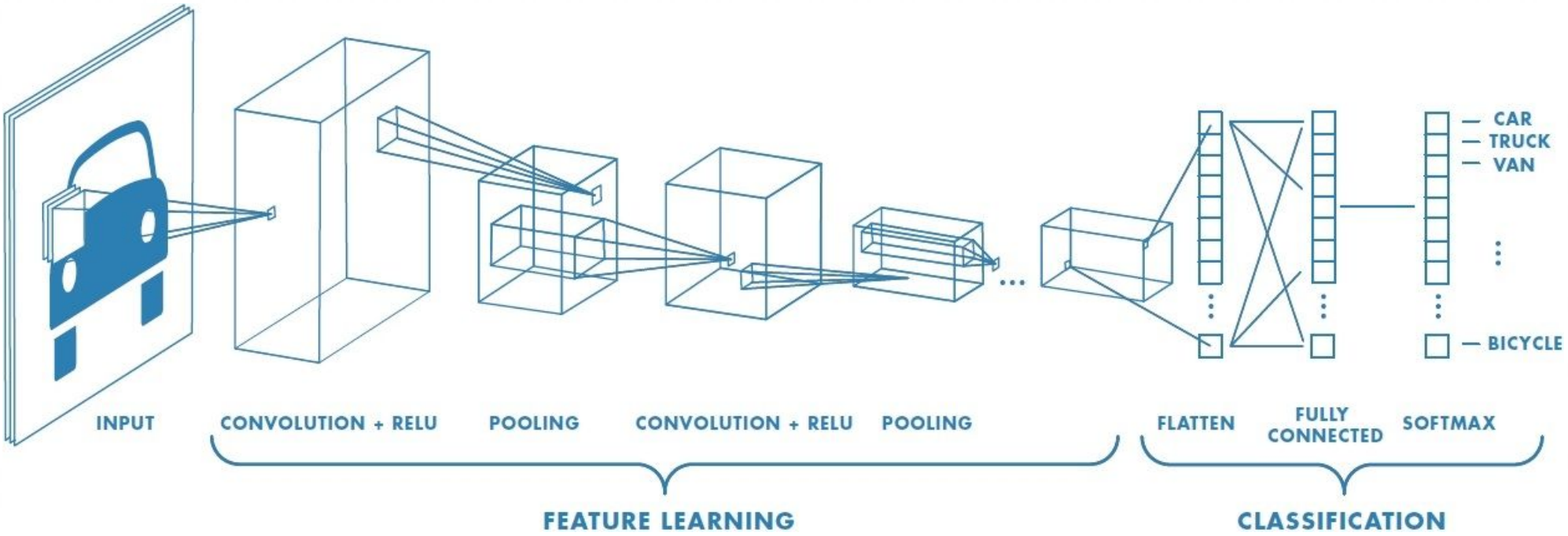
- As the name implies, it is a Neural Network that utilizes **convolutions**.
- The main input for CNNs is **compositional data**.
- What are possible uses by combining compositional data and convolutions?

1. Object/feature recognition
2. Self-driving cars
3. Game mastery (AlphaGo, Mario)



CNN Structure

BOSTON UNIVERSITY
MACHINE INTELLIGENCE COMMUNITY



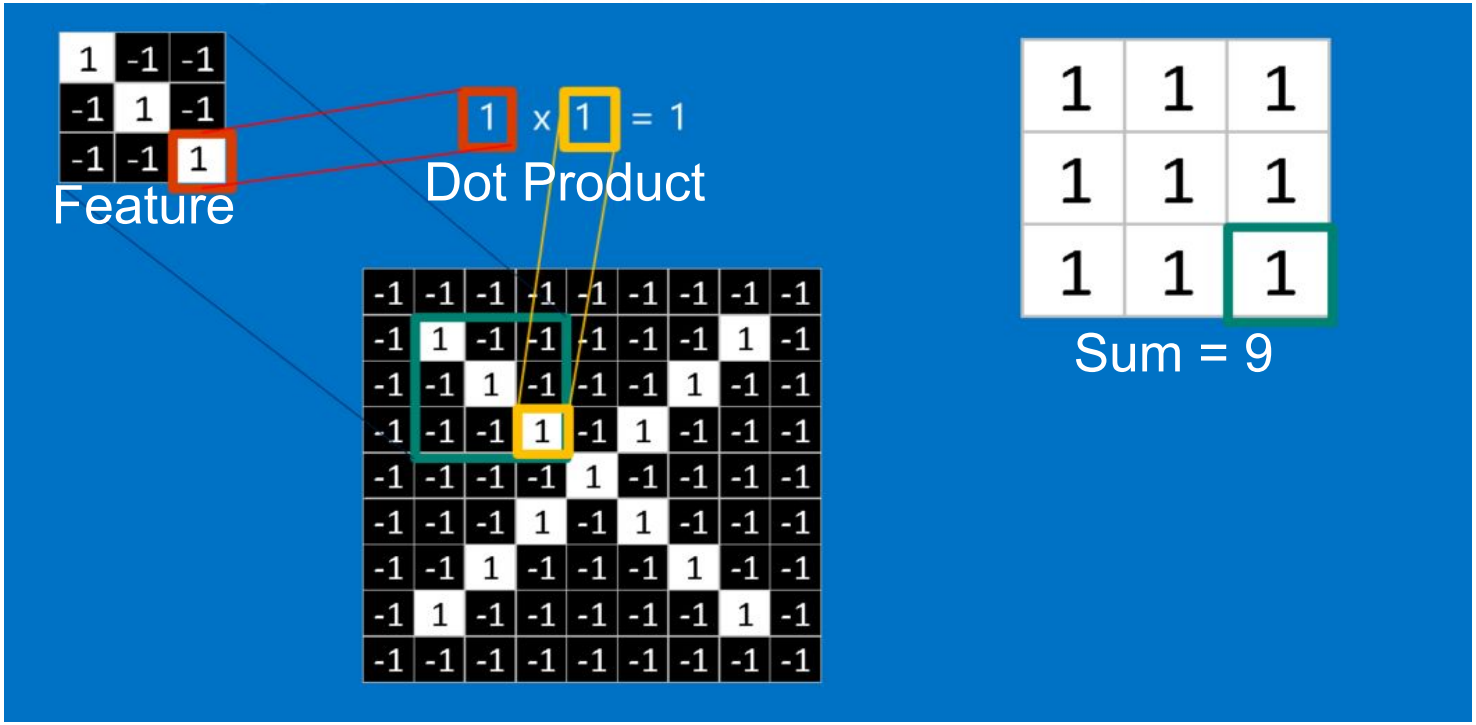
- Convolutional Layer
- ReLU
- Pooling Layer
- Fully Connected Layer
- Softmax

Convolutional Layer

BOSTON UNIVERSITY
MACHINE INTELLIGENCE COMMUNITY

- **Features** are parts of an expected class that is split into different characteristics to compare to an input image.
  - Defined as a **filter** which is iterated through the image to produce an **activation map.**
  - The **filter size** depends on the representation of the feature.

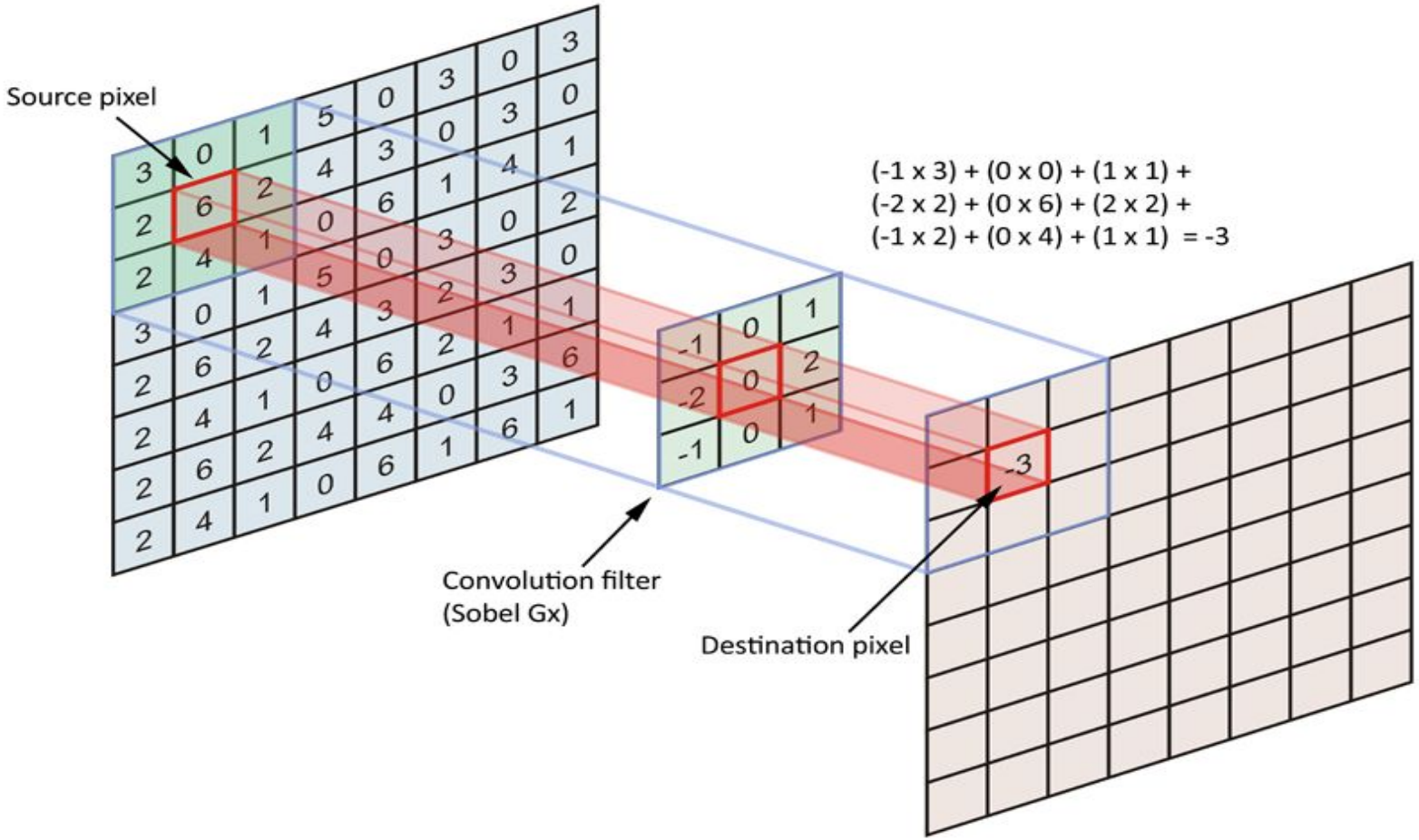
- For each iteration a dot product is performed with the input region and filter.



Convolutional Layer - 2

BOSTON UNIVERSITY
MACHINE INTELLIGENCE COMMUNITY

- The result of the dot product is then mapped to the specific pixel in the output map.
- The map fills up as the filter moves across the input.

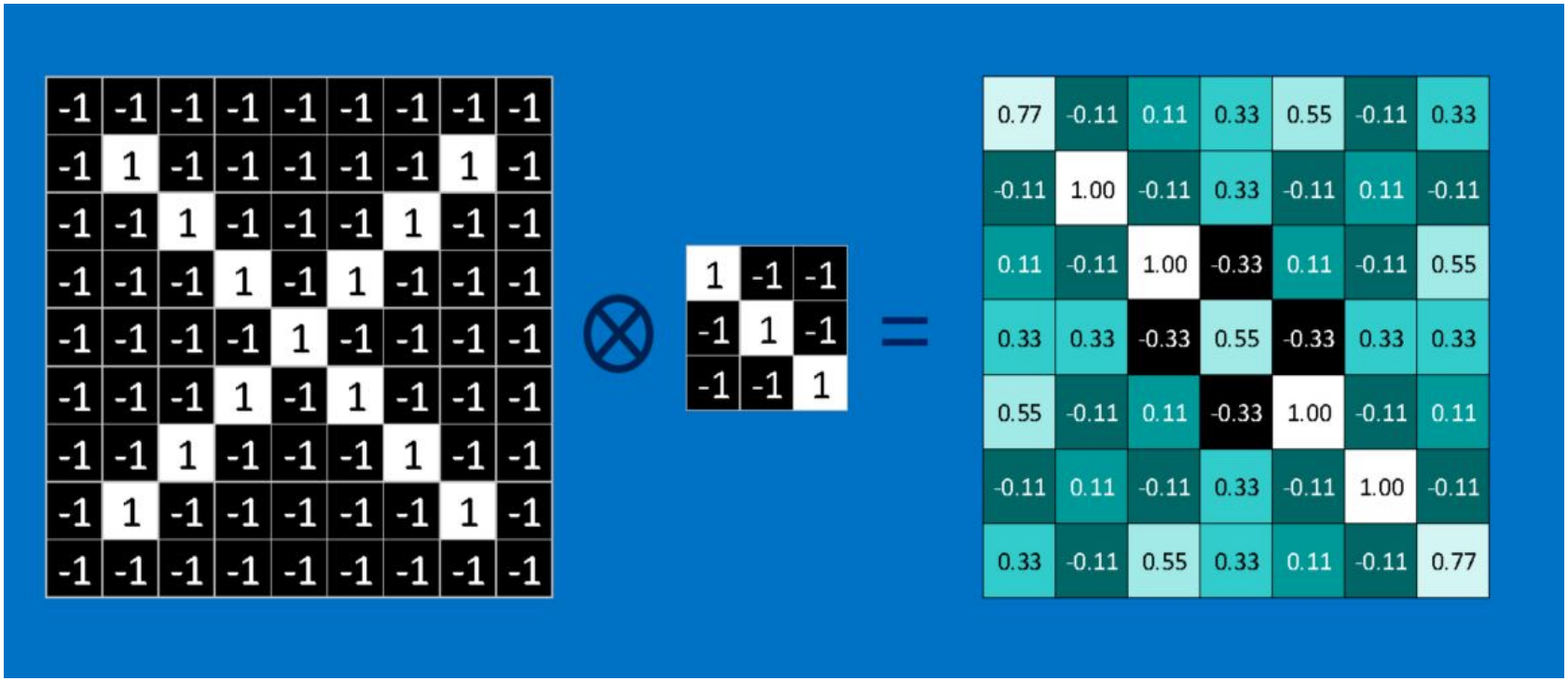


Normalization

BOSTON UNIVERSITY
MACHINE INTELLIGENCE COMMUNITY

- The resulting raw output will produce a diverse amount of numbers that may be hard to interpret.

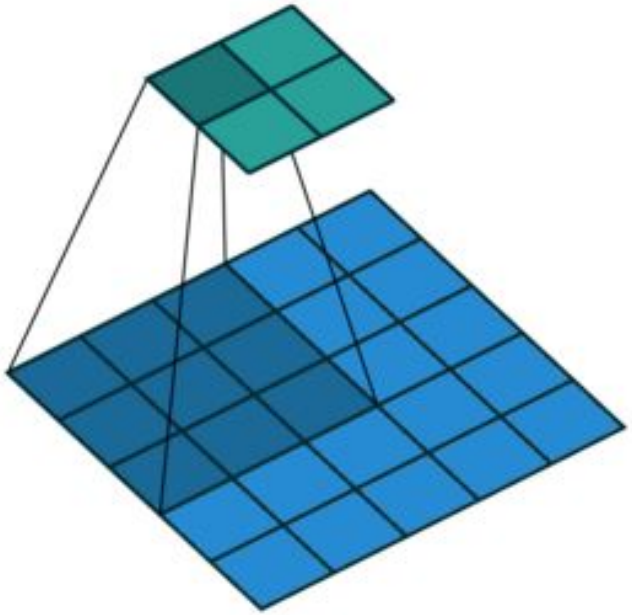
- Normalization occurs by dividing each element by the maximum obtainable number, bringing the range to [0-1].



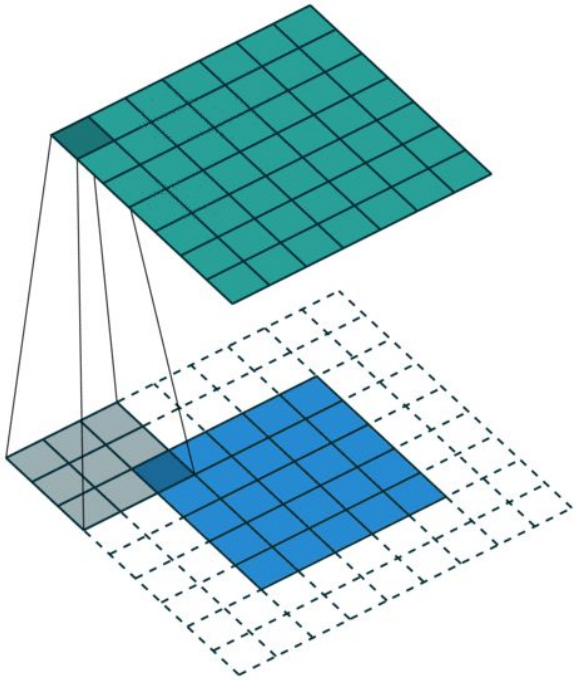
Convolutional Layer - 3

BOSTON UNIVERSITY
MACHINE INTELLIGENCE COMMUNITY

- There are three more hyper parameters which can control the output:
  - **Stride**
  - **Padding**
  - **Depth**



Stride 2, No padding,



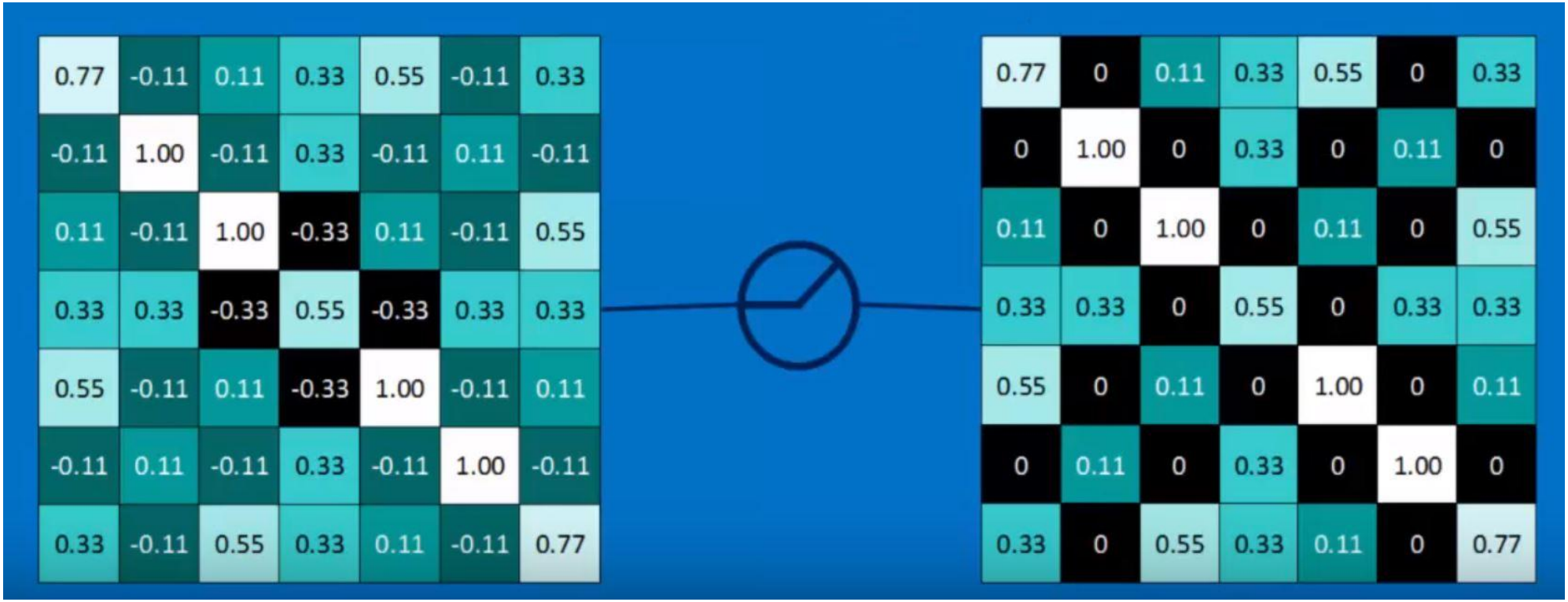
Stride 1, Full (2) Padding

ReLU

BOSTON UNIVERSITY
MACHINE INTELLIGENCE COMMUNITY

- ReLU has been previously introduced as an element-wise activation function.
  - Although others may be used, ReLU is the most prevalent for CNNs.

- Performs the max(0, x) function on all the resulting elements.



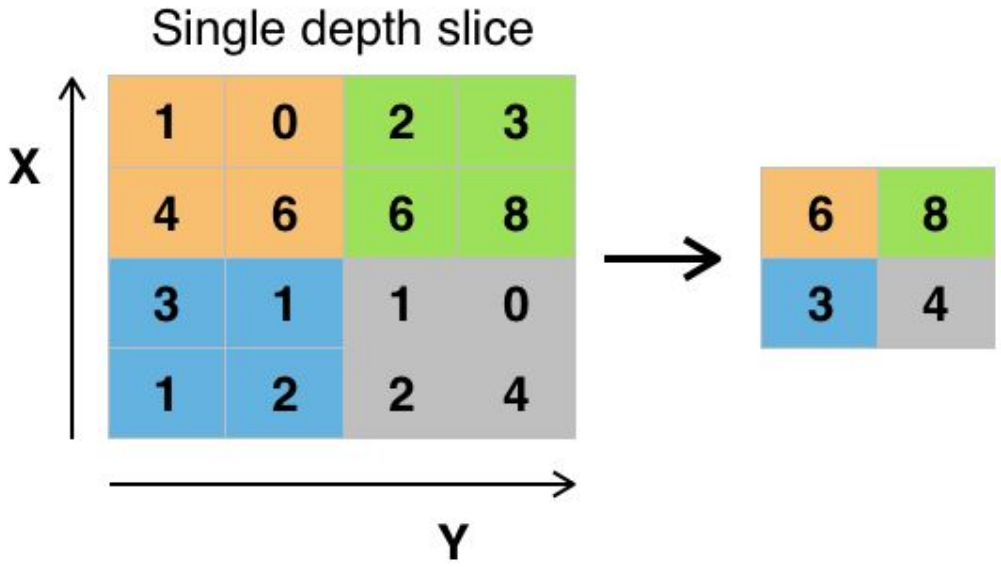
Pooling Layer - Downsampling

BOSTON UNIVERSITY
MACHINE INTELLIGENCE COMMUNITY

- The pooling layer is meant to shrink down the current representation and lower the amount of parameters (to prevent overfitting).

- It iterates similarly to the convolutional layer, applying a function on the selection.
  - It maintains the hyper parameter stride, but filter size can be specified as the kernel.

- Maxpooling uses the max(..) function on the selected region.



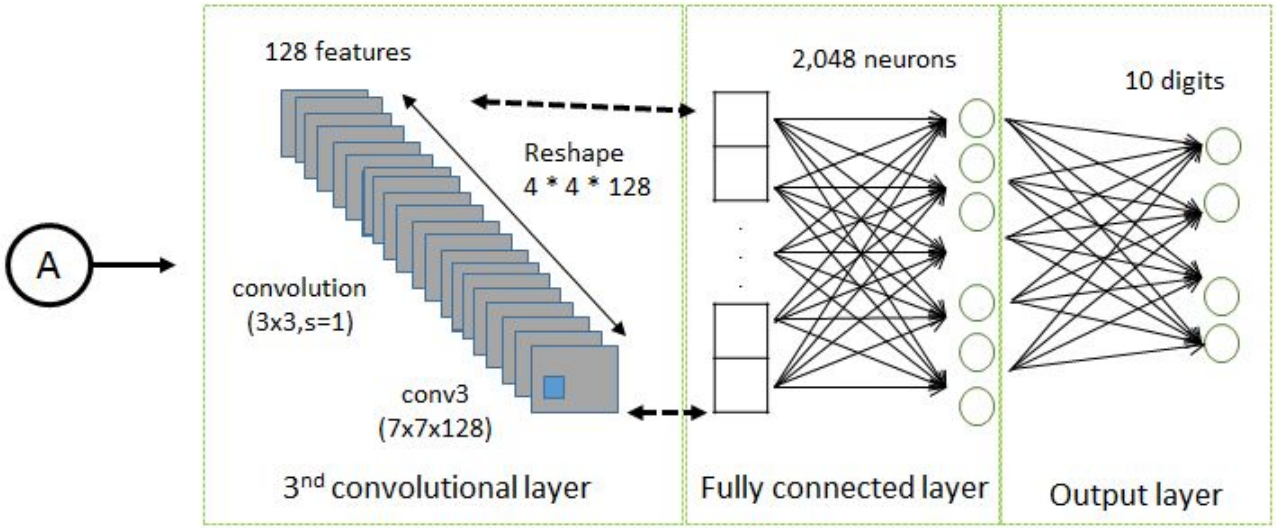
Fully Connected Layers

BOSTON UNIVERSITY
MACHINE INTELLIGENCE COMMUNITY

- As the name indicates, fully connected layers are connected to all activated neurons in the previous layer.

- It flattens the previous layer into a one dimensional vector, for which it outputs another vector containing the class scores.
  - FC layers can be stacked on top of each other because of matching dimensions.

- The largest score in the output layer is the chosen class for the network input.



Softmax - The Score Function

BOSTON UNIVERSITY
MACHINE INTELLIGENCE COMMUNITY

- Normalizes the output values to add up to 1.

- The highest value gets the most weight and the rest of the values are reduced.
  - The goal is to make the highest value reach 1, since it is the correct result, and the remaining values are errors.

	Scoring Function	Unnormalized Probabilities	Normalized Probabilities	Negative Log Loss
Dog	-3.44	0.0321	0.0006	
Cat	1.16	3.1899	0.0596	
Boat	-0.81	0.4449	0.0083	
Airplane	3.91	49.8990	0.9315	**0.0709**

Try It Out in PYTORCH

BOSTON UNIVERSITY
MACHINE INTELLIGENCE COMMUNITY

github link

Try It Out in PYTORCH

The Imports:

```
import torch
import torch.nn as nn
import torchvision.datasets as dsets
import torchvision.transforms as transforms
from torch.autograd import Variable
```

The Hyper parameters:

```
epochs = 5
batch_size = 100
learning_rate = 0.001
```

Try It Out in PYTORCH

BOSTON UNIVERSITY
MACHINE INTELLIGENCE COMMUNITY

Load datasets:

```
train_dataset = dsets.MNIST(root='./data/',
                            train=True,
                            transform=transforms.ToTensor(),
                            download=True)

test_dataset = dsets.MNIST(root='./data/',
                           train=False,
                           transform=transforms.ToTensor())

# Data Loader (Input Pipeline)
train_loader = torch.utils.data.DataLoader(dataset=train_dataset,
                                           batch_size=batch_size,
                                           shuffle=True)

test_loader = torch.utils.data.DataLoader(dataset=test_dataset,
                                          batch_size=batch_size,
                                          shuffle=False)
```

Try It Out in PYTORCH

Define CNN by Layers:

```
class CNN(nn.Module):
    def __init__(self):
        super(CNN, self).__init__()

        # Convolutional Layer 1 (2d)
        self.layer1 = nn.Sequential(
            nn.Conv2d(1, 16, kernel_size=5, padding=2),
            nn.BatchNorm2d(16),
            nn.ReLU(),
            nn.MaxPool2d(2))

        # Convolutional Layer 2 (2d)
        self.layer2 = nn.Sequential(
            nn.Conv2d(16, 32, kernel_size=5, padding=2),
            nn.BatchNorm2d(32),
            nn.ReLU(),
            nn.MaxPool2d(2))
        self.fc = nn.Linear(7*7*32, 10)

    def forward(self, x):
        out = self.layer1(x)
        out = self.layer2(out)
        out = out.view(out.size(0), -1)
        out = self.fc(out)
        return out

cnn = CNN()
```

Try It Out in PYTORCH

BOSTON UNIVERSITY
MACHINE INTELLIGENCE COMMUNITY

Training:

```
optimizer = torch.optim.Adam(cnn.parameters(), lr=learning_rate)

# Training
for epoch in range(epochs):
    for i, (images, labels) in enumerate(train_loader):
        images = Variable(images)
        labels = Variable(labels)

        # Forward -> Backwards -> Optimization
        optimizer.zero_grad()
        outputs = cnn(images)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()

        if (i+1) % 100 == 0:
            print ('Epoch [%d/%d], Iter [%d/%d] Loss: %.4f'
                   %(epoch+1, epochs, i+1, len(train_dataset)//batch_size, loss.data[0]))
```

Try It Out in PYTORCH

BOSTON UNIVERSITY
MACHINE INTELLIGENCE COMMUNITY

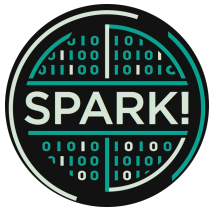
Sample Output:

Anaconda Prompt

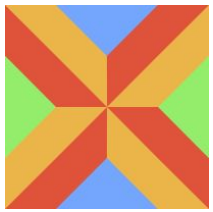
```
Downloading http://yann.lecun.com/exdb/mnist/train-images-idx3-ubyte.gz
Downloading http://yann.lecun.com/exdb/mnist/train-labels-idx1-ubyte.gz
Downloading http://yann.lecun.com/exdb/mnist/t10k-images-idx3-ubyte.gz
Downloading http://yann.lecun.com/exdb/mnist/t10k-labels-idx1-ubyte.gz
Processing...
Done!
Epoch [1/5], Iter [100/600] Loss: 0.2137
Epoch [1/5], Iter [200/600] Loss: 0.1685
Epoch [1/5], Iter [300/600] Loss: 0.0689
Epoch [1/5], Iter [400/600] Loss: 0.0372
Epoch [1/5], Iter [500/600] Loss: 0.0334
Epoch [1/5], Iter [600/600] Loss: 0.0806
Epoch [2/5], Iter [100/600] Loss: 0.0124
Epoch [2/5], Iter [200/600] Loss: 0.0187
Epoch [2/5], Iter [300/600] Loss: 0.0474
Epoch [2/5], Iter [400/600] Loss: 0.0207
Epoch [2/5], Iter [500/600] Loss: 0.0719
Epoch [2/5], Iter [600/600] Loss: 0.0148
Epoch [3/5], Iter [100/600] Loss: 0.0769
Epoch [3/5], Iter [200/600] Loss: 0.0102
Epoch [3/5], Iter [300/600] Loss: 0.0208
Epoch [3/5], Iter [400/600] Loss: 0.0428
Epoch [3/5], Iter [500/600] Loss: 0.0790
Epoch [3/5], Iter [600/600] Loss: 0.0285
Epoch [4/5], Iter [100/600] Loss: 0.0240
Epoch [4/5], Iter [200/600] Loss: 0.0041
Epoch [4/5], Iter [300/600] Loss: 0.0729
Epoch [4/5], Iter [400/600] Loss: 0.0105
Epoch [4/5], Iter [500/600] Loss: 0.0040
Epoch [4/5], Iter [600/600] Loss: 0.0657
Epoch [5/5], Iter [100/600] Loss: 0.0074
Epoch [5/5], Iter [200/600] Loss: 0.0008
Epoch [5/5], Iter [300/600] Loss: 0.0211
Epoch [5/5], Iter [400/600] Loss: 0.0408
Epoch [5/5], Iter [500/600] Loss: 0.0196
Epoch [5/5], Iter [600/600] Loss: 0.0009
Test Accuracy of the model on the 10000 test images: 98 %
```

Shoutout to our Sponsors

Boston University Computer Science



Boston University SPARK!



Boston University Software Application and Innovation Lab

MACHINE INTELLIGENCE COMMUNITY

References

Papers:

[1] Dumoulin, Vincent, and Francesco Visin. “A Guide to Convolution Arithmetic for Deep Learning.” Arxiv, vol. 1603, 23 Mar. 2016

[2] Krizhevsky, Alex, et al. “ImageNet Classification with Deep Convolutional Neural Networks.” Neural Information Processing Systems,

[3] "Convolutional Neural Network." Convolutional Neural Network - MATLAB & Simulink. N.p., n.d. Web.

[4] Rohrer, Brandon. "How Do Convolutional Neural Networks Work?" N.p., 18 Aug. 2016. Web.

[5] Rosebrock, Adrian. "Softmax Classifiers Explained." PyImageSearch. N.p., 12 Sept. 2016.

[6] Zaccone, Giancarlo. "CNN Architecture." PACKT Books. N.p., n.d. Web.