Convolutional Neural Network (CNN) using PyTorch – Part 1 of 2

John Wong, 3 Jan 2024

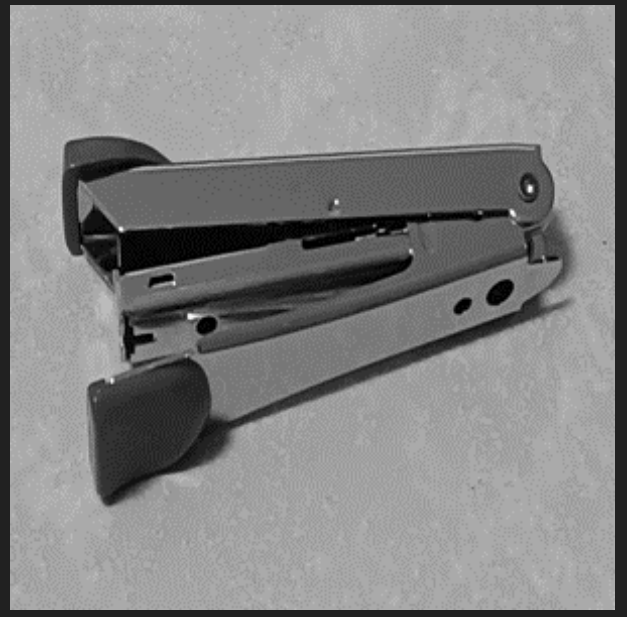
Context

High-level view on connecting steps in CNN through:

- Convolution, Rectified Linear Unit Function, Max Pooling, Flattening, Fully Connected Layer, Linear Transformation, Sigmoid Function
- Colour, grayscale, channel, size, and tensor form of image

Colour and Grayscale

Colour
Open = “PIL.Image.open”
Resize = “PIL.Image.resize”



Grayscale
Transform = “torchvision.transforms.Grayscale”

Tensor Form, Channel, Size

```
[8]: to_tensor = transforms.ToTensor()

[9]: image_colour_tensor = to_tensor(image_colour)
     image_colour_tensor

[9]: tensor([[[0.6745, 0.6824, 0.6784,  ..., 0.6627, 0.6588, 0.6627],
              [0.6667, 0.6706, 0.6706,  ..., 0.6706, 0.6627, 0.6667],
              [0.6745, 0.6667, 0.6784,  ..., 0.6863, 0.6745, 0.6706],
              ...,
              [0.6667, 0.6588, 0.6510,  ..., 0.6667, 0.6745, 0.6902],
              [0.6549, 0.6549, 0.6549,  ..., 0.6627, 0.6745, 0.6824],
              [0.6510, 0.6549, 0.6549,  ..., 0.6588, 0.6706, 0.6745]]])

[10]: image_colour_tensor.size()

[10]: torch.Size([3, 300, 300])
```

You will see 3 channels

Colour Tensor Form
Transform = "torchvision.transforms.ToTensor"
3 Channels of Size 300 x 300

```
[13]: image_grayscale_tensor = to_tensor(image_grayscale)
      image_grayscale_tensor

[13]: tensor([[[0.6706, 0.6784, 0.6745,  ..., 0.6588, 0.6549, 0.6588],
               [0.6627, 0.6667, 0.6667,  ..., 0.6667, 0.6588, 0.6627],
               [0.6706, 0.6627, 0.6745,  ..., 0.6824, 0.6706, 0.6667],
               ...,
               [0.6667, 0.6588, 0.6510,  ..., 0.6706, 0.6824, 0.6980],
               [0.6549, 0.6549, 0.6549,  ..., 0.6706, 0.6824, 0.6902],
               [0.6510, 0.6549, 0.6549,  ..., 0.6667, 0.6784, 0.6824]]])

[14]: image_grayscale_tensor.size()

[14]: torch.Size([1, 300, 300])
```

You will see 1 channel

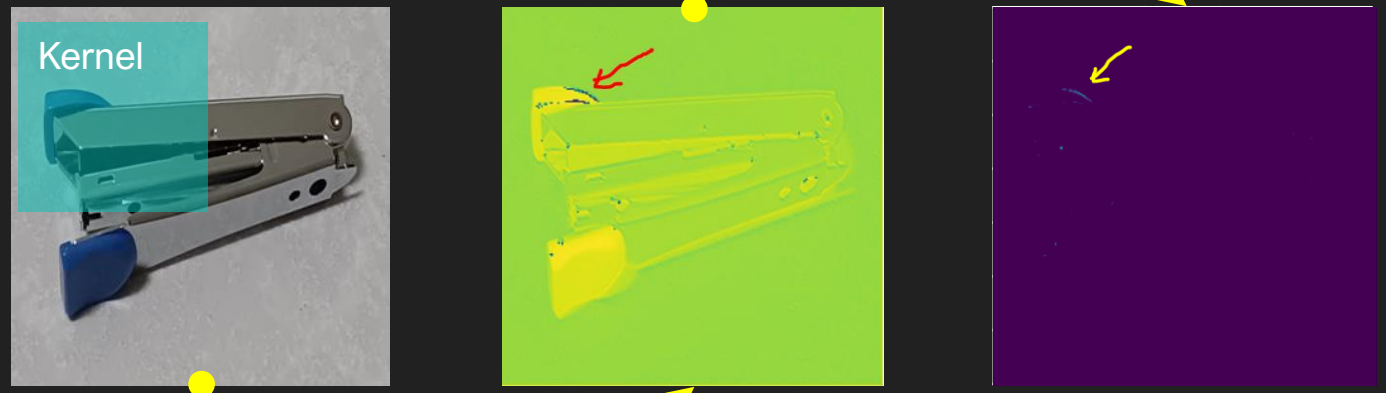
Grayscale Tensor Form
Transform = "torchvision.transforms. ToTensor"
1 Channel of Size 300 x 300

Convolution

Convolution
Use Kernel to get feature

Rectified Linear Unit (ReLU) Function

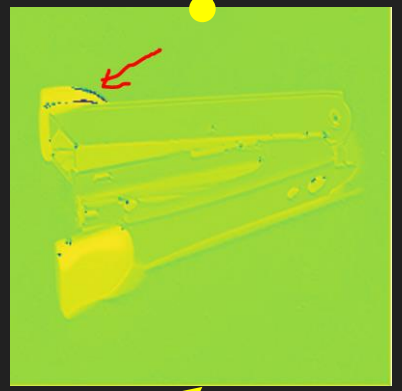
ReLU
Reduce gradation
Create non-linearity to get complex feature

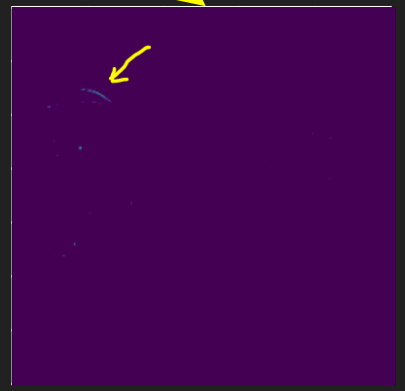


Convolution
Use Kernel to
get feature

Max Pooling

ReLU
Reduce gradation
Create non-linearity to get complex feature



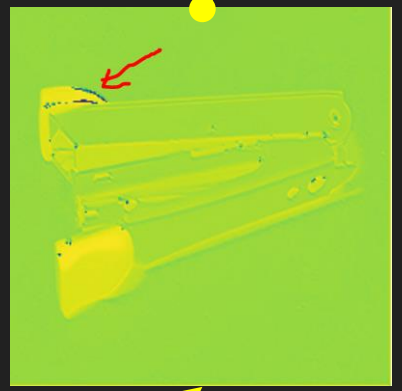


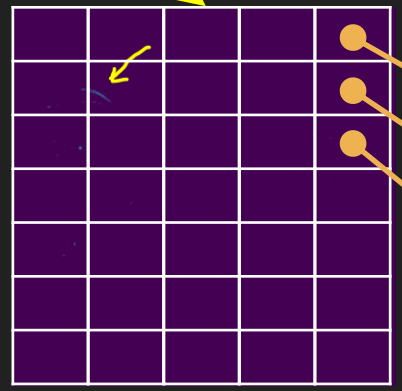
Convolution
Use Kernel to
get feature

Max Pooling
Re-size from 300 x 300
to 150 x 150

Flattening (To 1-Dimensional Tensor)

ReLU
Reduce gradation
Create non-linearity to get complex feature



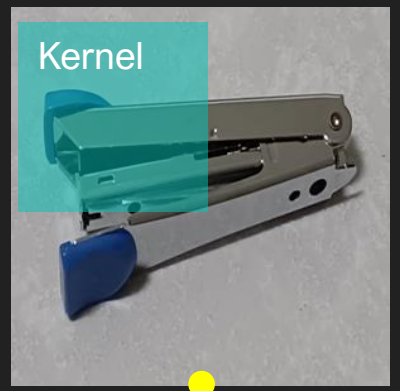


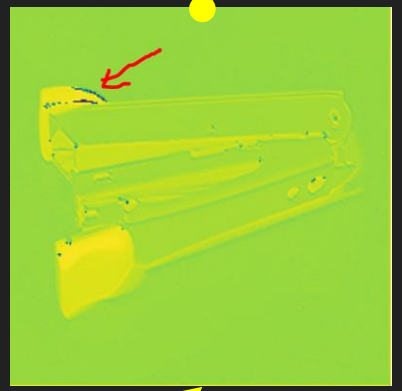
Convolution
Use Kernel to
get feature

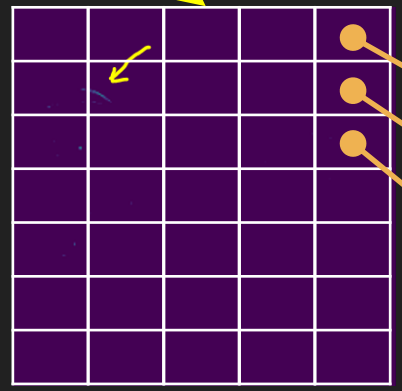
Max Pooling
Re-size from 300 x 300
to 150 x 150

Fully Connected Layer (Linear Transformation + ReLU)

ReLU
Reduce gradation
Create non-linearity to get complex feature

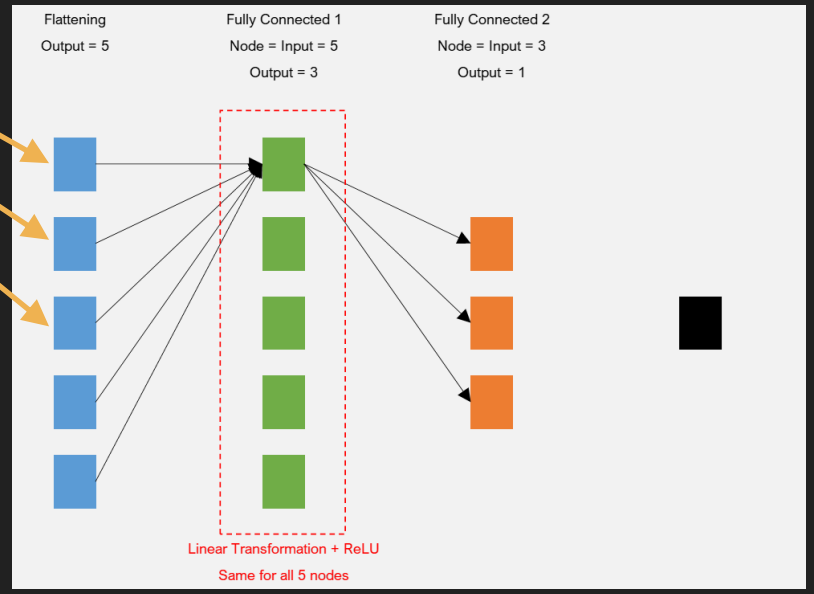






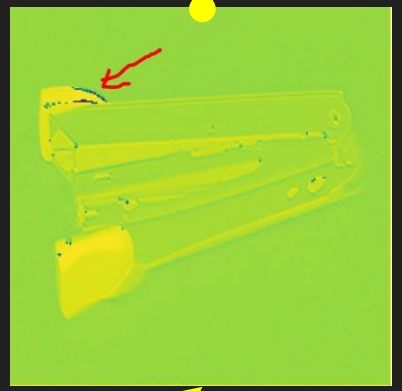
Convolution
Use Kernel to
get feature

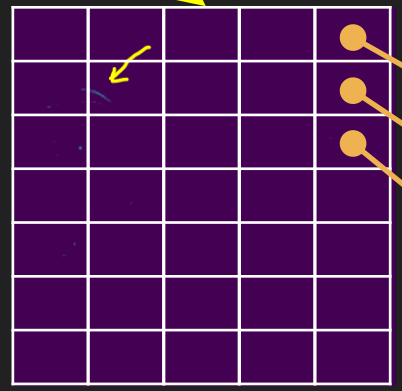
Max Pooling
Re-size from 300 x 300
to 150 x 150



Fully Connected Layer (Linear Transformation + Sigmoid) to Classify

ReLU
Reduce gradation
Create non-linearity to get complex feature

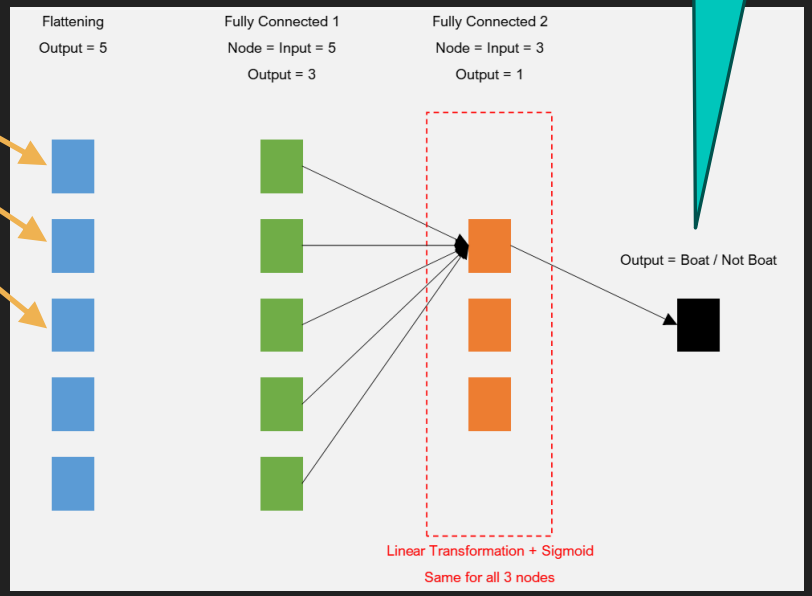




Convolution
Use Kernel to get feature

Max Pooling
Re-size from 300 x 300 to 150 x 150

Classification
Boat or Not Boat



The End

Hope you find this useful

Visit my GitHub for the codes

https://github.com/johnwck/my_da_ds_work/tree/master/my_projects_github_pages/pytorch_convolutional_neural_network_part_1