Introduction to Machine Learning

NYU K12 STEM Education: Machine Learning

Department of Electrical and Computer Engineering,
NYU Tandon School of Engineering
Brooklyn, New York

Course Details

- Course Website
- Instructors:

Rugved Mhatre
rugved.mhatre@nyu.edu

Akshath Mahajan
akshathmahajan@nyu.edu

NYU

Outline

1. Review
2. Working with Images
3. Convolution Neural Networks
4. Data Augmentation
5. Normalization
6. Dropout
7. Transfer Learning

Outline

Grayscale Images

- Images are stored as arrays of quantized numbers in computers

Grayscale Images

- Images are stored as arrays of quantized numbers in computers
- 2D matrices with each entry specifying the intensity (brightness) of a pixel

Grayscale Images

- Images are stored as arrays of quantized numbers in computers
- 2D matrices with each entry specifying the intensity (brightness) of a pixel
- Pixel values range from 0 to 255, 0 being the darkest, 255 being the brightest

```
[[255 255 255]
 [255   0 255]
 [255 255 255]]
```

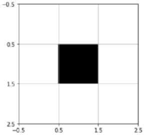

Figure 1: A 3x3 Grayscale Image

Color Images

- Color (RGB) images have an extra dimension for color (3D array)
- Imagine three 2D matrices stacked together
- Each 2D matrix specifies the amount of color for Red, Green, and Blue at each pixel

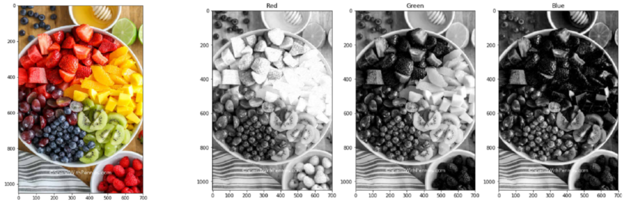

Figure 2: RGB Images

Color Images

- Color (RGB) images have an extra dimension for color (3D array)
- Imagine three 2D matrices stacked together
- Each 2D matrix specifies the amount of color for Red, Green, and Blue at each pixel

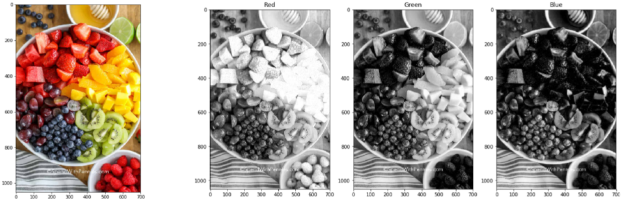

Figure 2: RGB Images

- Shape - (1050, 700, 3)

Images and Neural Networks

- How to feed Images in a Fully Connected Network?

Images and Neural Networks

- How to feed Images in a Fully Connected Network?
- Flatten the image!

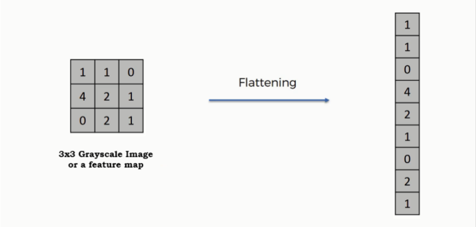

Figure 3: Flattening an Image

Images and Neural Networks

- How to feed Images in a Fully Connected Network?
- Flatten the image!

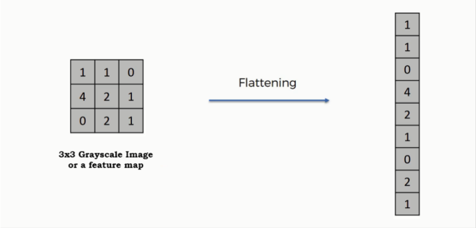

Figure 3: Flattening an Image

- Does this make sense? Is this how we see images?

Images and Neural Networks

- How to feed Images in a Fully Connected Network?
- Flatten the image!

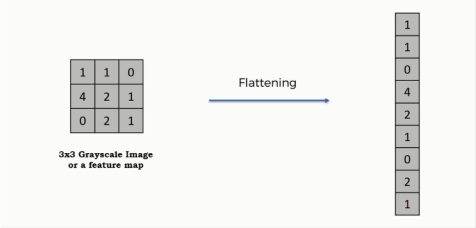

Figure 3: Flattening an Image

- Does this make sense? Is this how we see images?
  - No consideration for spatial positions!!
  - How many input neurons for 1024x1024 image?
  - What about slightly rotated photographs?

Outline

The Convolution Operation

- All these problems are solved by Convolutions!
- Convolution operation is applied on an image matrix X with a kernel W

$$Z = X \circledast W$$

The Convolution Operation

- All these problems are solved by Convolutions!
- Convolution operation is applied on an image matrix X with a kernel W

$$Z = X \circledast W$$

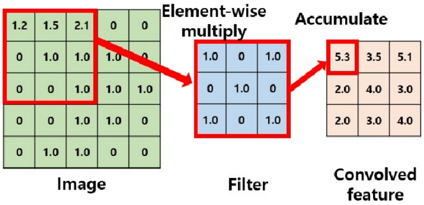

Figure 4: Convolution Operation

The Convolution Operation

- Let's see some visualizations!

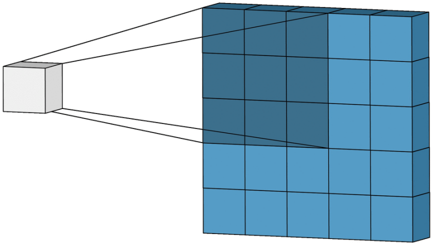

Figure 5: Standard Convolution Operation

The Convolution Operation

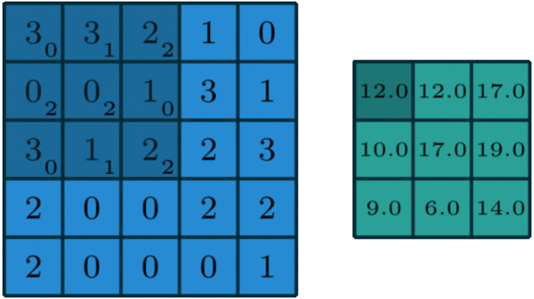

Figure 5: Standard Convolution with Numbers

Padding and Stride

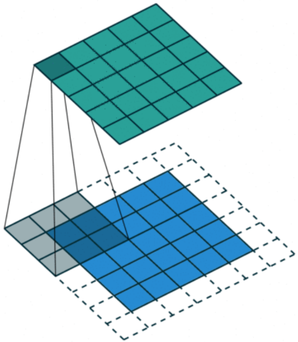

Figure 6: Convolution with Padding

Why Convolution?

- With convolution, each output pixel depends on only the neighboring pixels in the input

Why Convolution?

- With convolution, each output pixel depends on only the neighboring pixels in the input
- This allows us to learn the positional relationship between pixels

Why Convolution?

- With convolution, each output pixel depends on only the neighboring pixels in the input
- This allows us to learn the positional relationship between pixels
- Different kernels capture different features from the image

Convolution for Multiple Channels

- There is a single kernel for each channel

Convolution for Multiple Channels

- There is a single kernel for each channel
- Each kernel performs a 2D convolution a its respective channel

Convolution for Multiple Channels

- There is a single kernel for each channel
- Each kernel performs a 2D convolution a its respective channel
- The results are then summed

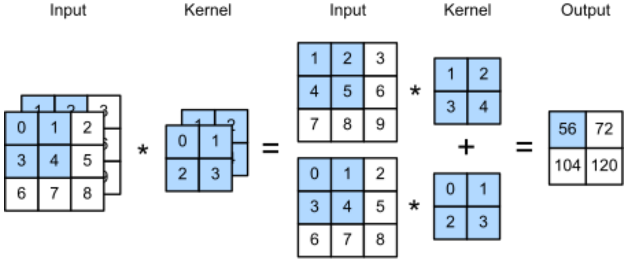

Figure 7: Convolution Across Channels
Source dl2.ai

Max Pooling

- It is a down-sampling technique in Convolutional Neural Networks
- Reduces the dimensions of intermediate network results

Max Pooling

- It is a down-sampling technique in Convolutional Neural Networks
- Reduces the dimensions of intermediate network results
- It provides "translational invariance". Why?

Max Pooling

- It is a down-sampling technique in Convolutional Neural Networks
- Reduces the dimensions of intermediate network results
- It provides "translational invariance". Why?
  - Most prominent feature in every local region is preserved
  - Focuses on the presence of features rather than their precise location

Max Pooling

- Let's see an example!

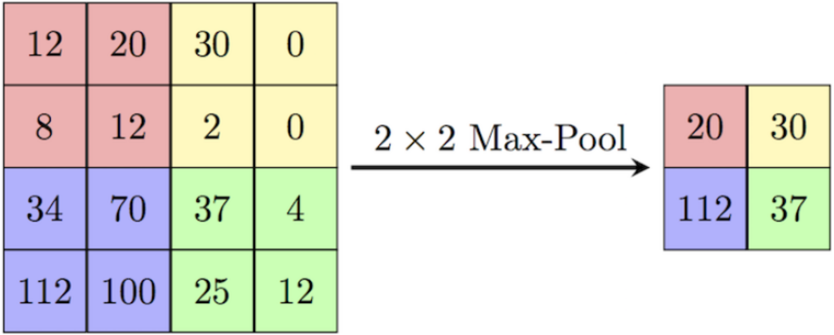

Figure 8: Max Pooling Example

Outline

Scarcity of training data

- Large-scale deep learning models are extremely data hungry

Scarcity of training data

- Large-scale deep learning models are extremely data hungry
- We don't always have enough data to train the model
- Labelling data is expensive and time-consuming

Scarcity of training data

- Large-scale deep learning models are extremely data hungry
- We don't always have enough data to train the model
- Labelling data is expensive and time-consuming
- What can we do now?

Scarcity of training data

- Large-scale deep learning models are extremely data hungry
- We don’t always have enough data to train the model
- Labelling data is expensive and time-consuming
- What can we do now?
- Create new images!

Data Augmentation

- Key Idea: Augment existing images from the original dataset

Data Augmentation

- Key Idea: Augment existing images from the original dataset
- Similar enough to contain the same Subject as the original
- Different enough to prove meaningful for training

Data Augmentation

- Key Idea: Augment existing images from the original dataset
- Similar enough to contain the same Subject as the original
- Different enough to prove meaningful for training
- Let's look at some techniques for Data Augmentation

Mirroring

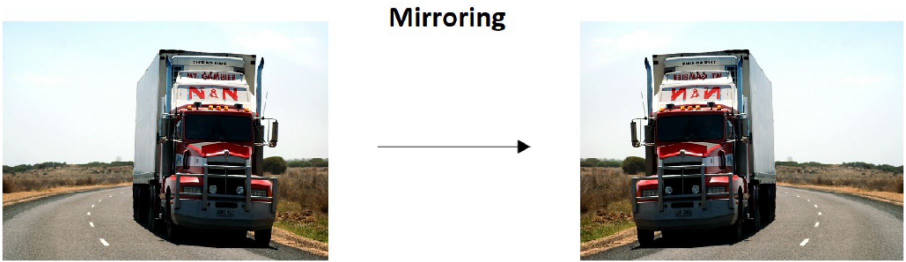

Figure 9: Mirroring

Rotation and Translation

Figure 10: Rotation and Translation

Random Cropping

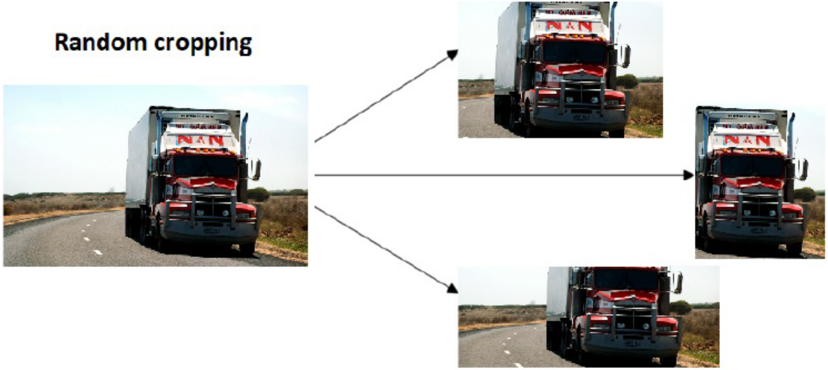

Figure 11: Random Cropping

Color Shifting

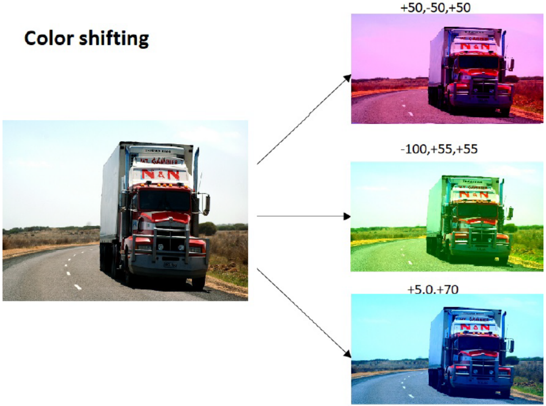

Figure 12: Color Shifting

Outline

1. Review
2. Working with Images
3. Convolution Neural Networks
4. Data Augmentation
5. Normalization
6. Dropout
7. Transfer Learning

Data Normalization

- Consider the dataset $(x_i, y_i)\ \forall i \in \{1, 2, 3..., N\}$

Data Normalization

- Consider the dataset $(x_i, y_i)\ \forall i \in \{1, 2, 3..., N\}$
- Mean $\bar{x} = \dfrac{1}{N} \sum x_i$
- Variance $\sigma^2 = \dfrac{1}{N} \sum (x_i - \bar{x_i})^2$

Data Normalization

- Consider the dataset $(x_i, y_i)\ \forall i \in \{1, 2, 3..., N\}$
- Mean $\bar{x} = \frac{1}{N}\sum x_i$
- Variance $\sigma^2 = \frac{1}{N}\sum(x_i - \bar{x_i})^2$
- Normalization: Replace each x_i with x_i', where:

$$x_i' = \frac{x_i - \bar{x}}{\sigma}$$

- The new dataset will have a mean of 0 and a variance of 1

Data Normalization

- Consider a single weight w and bias b
- The contours in the plot represents the value of the loss function for the given w and b

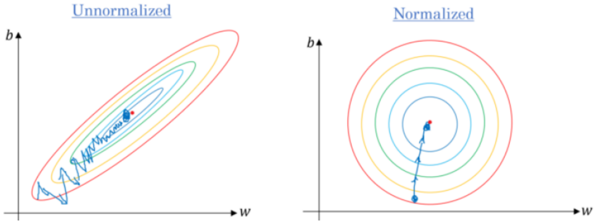

Figure 13: Unnormalized vs Normalized Descent
Source: TowardsDataScience

Batch Normalization

- We can normalize inputs to the network. Why not do that to the intermediate layer outputs

Batch Normalization

- We can normalize inputs to the network. Why not do that to the intermediate layer outputs
- Batch Normalization involves normalizing the inputs to each layer within each mini-batch

Batch Normalization

- We can normalize inputs to the network. Why not do that to the intermediate layer outputs
- Batch Normalization involves normalizing the inputs to each layer within each mini-batch
- Batch normalization is applied before activation

```
model = models.Sequential()

model.add(layers.Conv2D(64,(3,3)))
model.add(layers.BatchNormalization())
model.add(layers.Activation('relu'))


model.add(layers.Flatten())
model.add(layers.Dense(64))
model.add(layers.BatchNormalization())
model.add(layers.Activation('relu'))

model.add(layers.Dense(1, activation =
'sigmoid'))
```

Figure 14: Batch Normalization

Outline

Dropout

- This techinque is patented by Google

Dropout

- This techinque is patented by Google
- Randomly disable neurons and their connections between each other

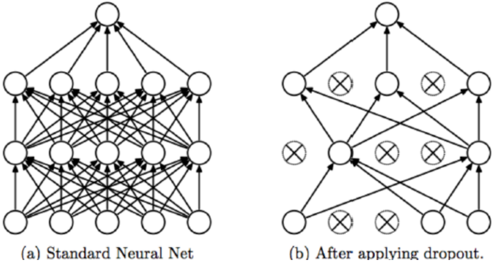

Figure 15: Dropout

Dropout

- This techinque is patented by Google
- Randomly disable neurons and their connections between each other
- Without dropout, neurons can become too reliant on the outputs of specific other neurons, leading to overfitting

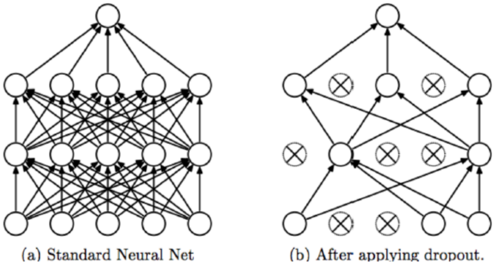

Figure 15: Dropout

Dropout

- This is the same as using a neural network with the same amount of layers but less neurons per layer.

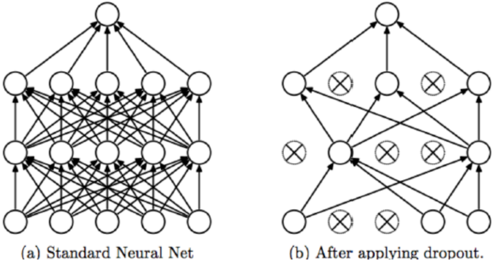

Figure 16: Dropout

Dropout

- This is the same as using a neural network with the same amount of layers but less neurons per layer.
- The more neurons the more powerful the neural network is, and the more likely it is to overfit.

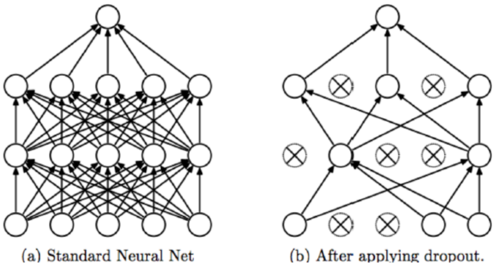

Figure 16: Dropout

Dropout

- This is the same as using a neural network with the same amount of layers but less neurons per layer.
- The more neurons the more powerful the neural network is, and the more likely it is to overfit.
- This also means that the model can not rely on any single feature, therefore would need to spread out the weights

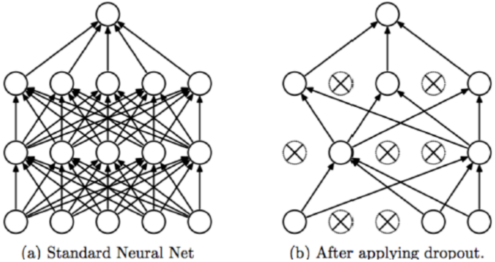

Figure 16: Dropout

Outline

Transfer Learning

- Key Idea: "Standing on the shoulder of Giants"

Transfer Learning

- Key Idea: "Standing on the shoulder of Giants"
- Training large computer vision models requires extensive hyperparameter search and multiple GPU running for weeks!

Transfer Learning

- Key Idea: "Standing on the shoulder of Giants"
- Training large computer vision models requires extensive hyperparameter search and multiple GPU running for weeks!
- Solution: Transfer Learning

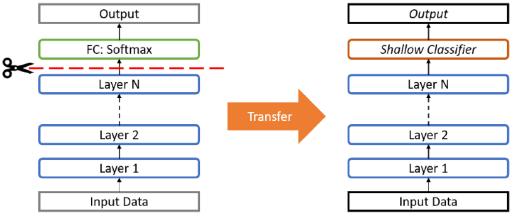

Figure 17: Transfer Learning
Source: Oreilly

Transfer Learning

- Researchers now open-source their model weights, which can be a great initialization point for your applications

Transfer Learning

- Researchers now open-source their model weights, which can be a great initialization point for your applications
- Often in practice, people preserve the feature extractor and re-train the classification head

Transfer Learning

- Researchers now open-source their model weights, which can be a great initialization point for your applications
- Often in practice, people preserve the feature extractor and re-train the classification head
- Freeze the early layers and replace the last few to match your needs. Only train the replaced layers

Transfer Learning

- Researchers now open-source their model weights, which can be a great initialization point for your applications
- Often in practice, people preserve the feature extractor and re-train the classification head
- Freeze the early layers and replace the last few to match your needs. Only train the replaced layers
- This is similar to transferring the knowledge from one network to another, thus the name transfer learning.