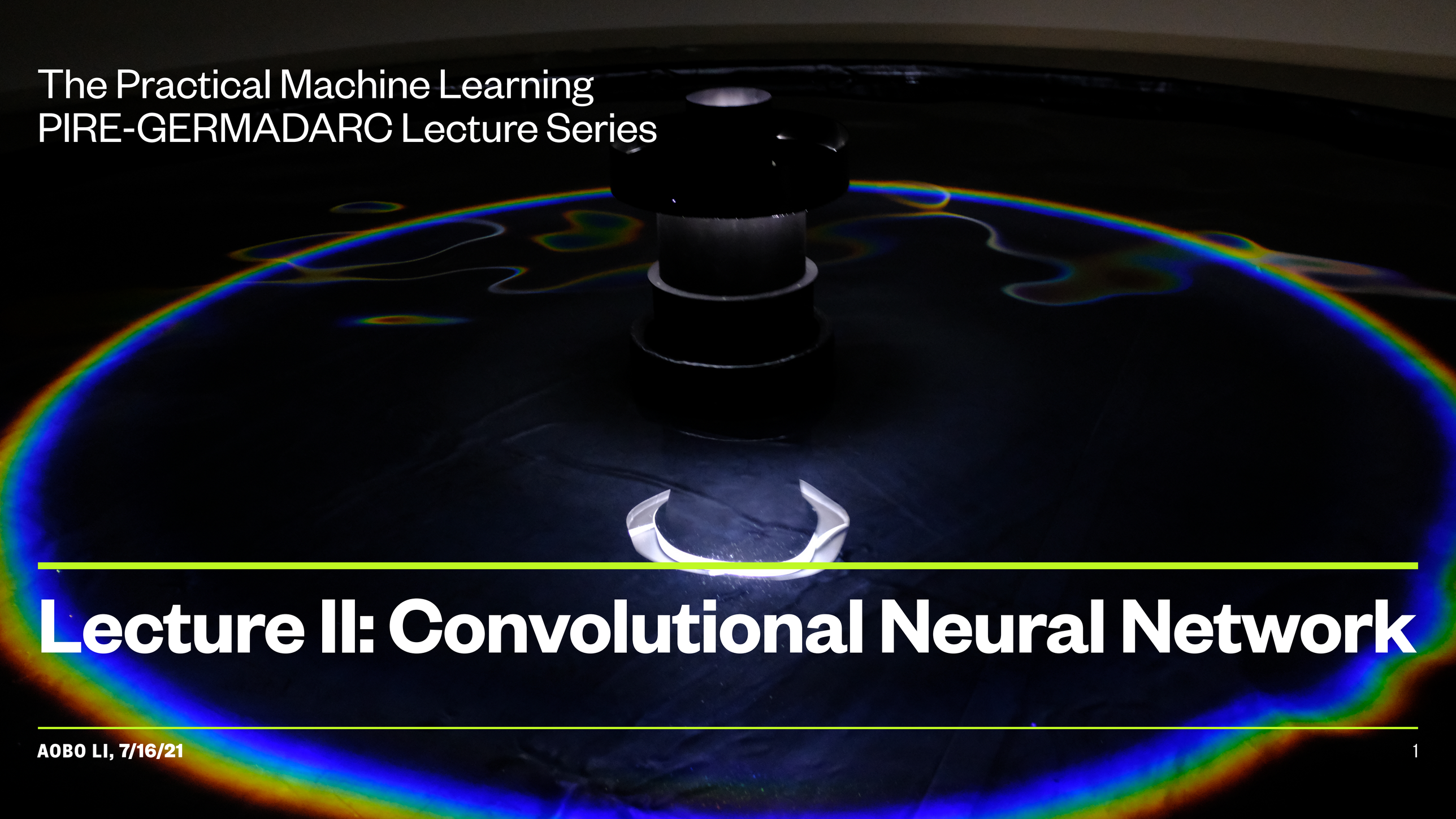
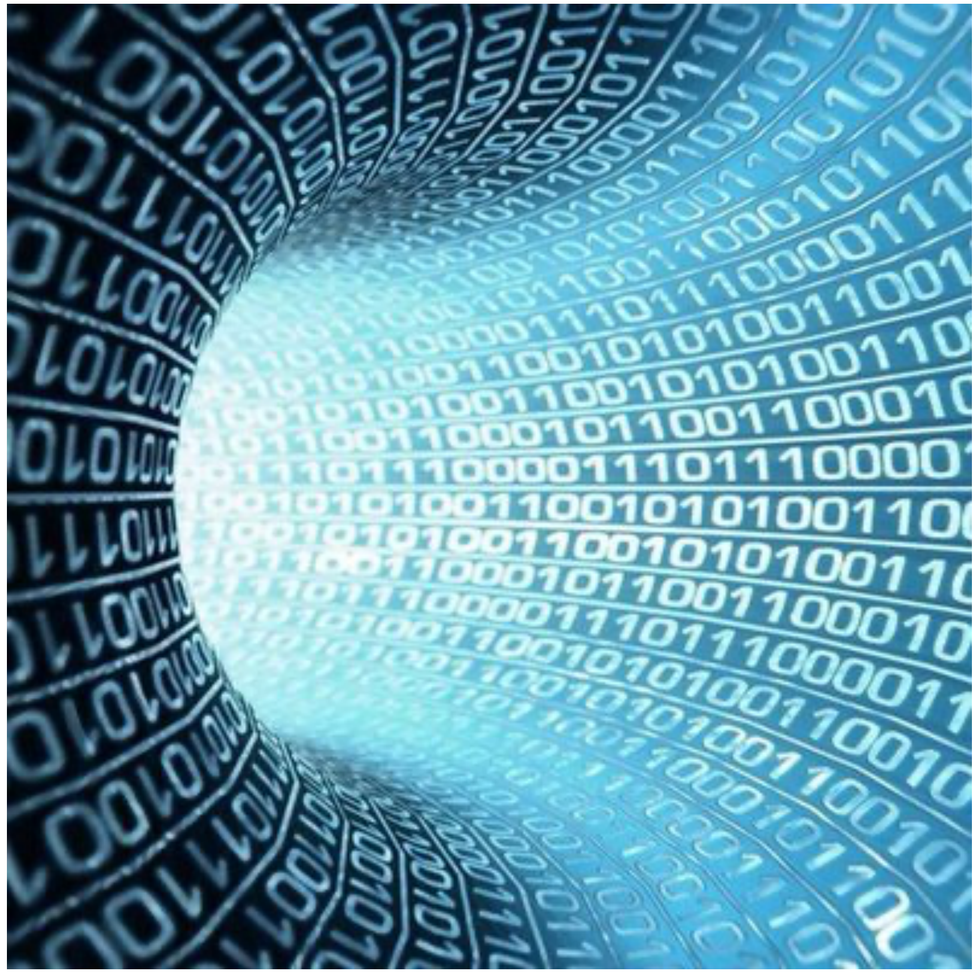


The Practical Machine Learning
PIRE-GERMADARC Lecture Series

Lecture II: Convolutional Neural Network

AOBO LI, 7/16/21

Outline



- Understanding Image Data
- Convolutional Neural Network
  - Convolutional Layer
  - Pooling Layer
  - Batch Normalization
- Extended CNN Models:
  - Regional CNN
  - Spherical CNN
  - CNN for videos

No Free Lunch

- No Free Lunch theorem: *Averaging over all possible data generating distributions, every classification algorithms has the same error rate when classifying previously unobserved points* (Wolpert, 1996)
- There is no omnipotent model that performs better than every other models, we have to design specific models that perform well on specific task.
- We need rethink about the attributes of image:
  - What level of information is important in images?
  - What symmetry is embedded in images?
- Learning from data, we will be able to understand why such a model is used for this type of data

Learning From Data

Pixel Level Info:
Not that useful

Learning From Data

Pixel Level Info:
Not that useful

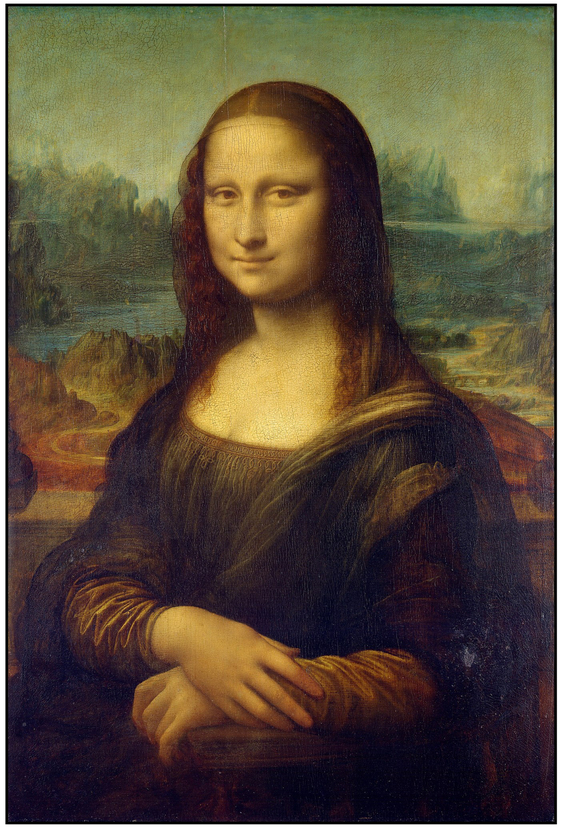
Local level Info:
Useful, but not enough

Learning From Data

Pixel Level Info:
Not that useful

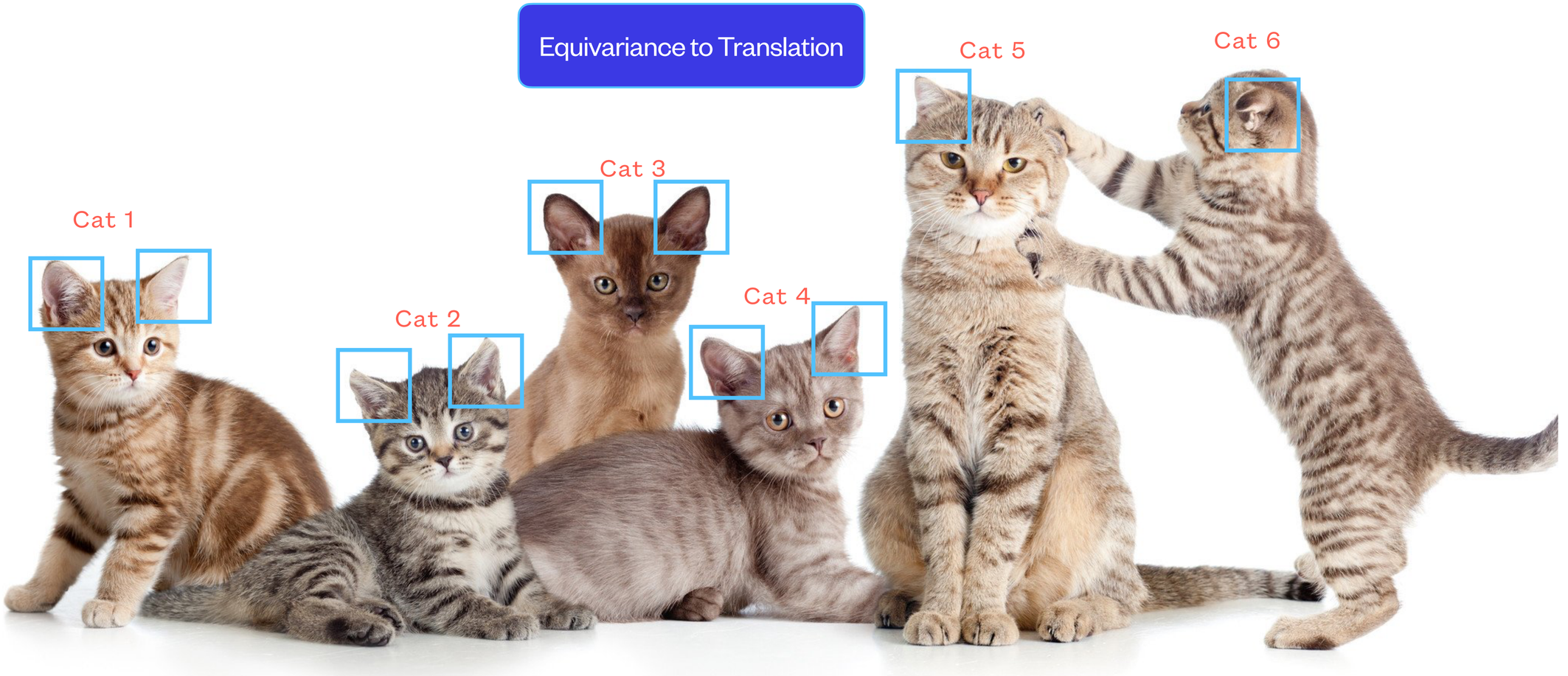
Local level Info:
Useful, but not enough

Global Info:
Enough

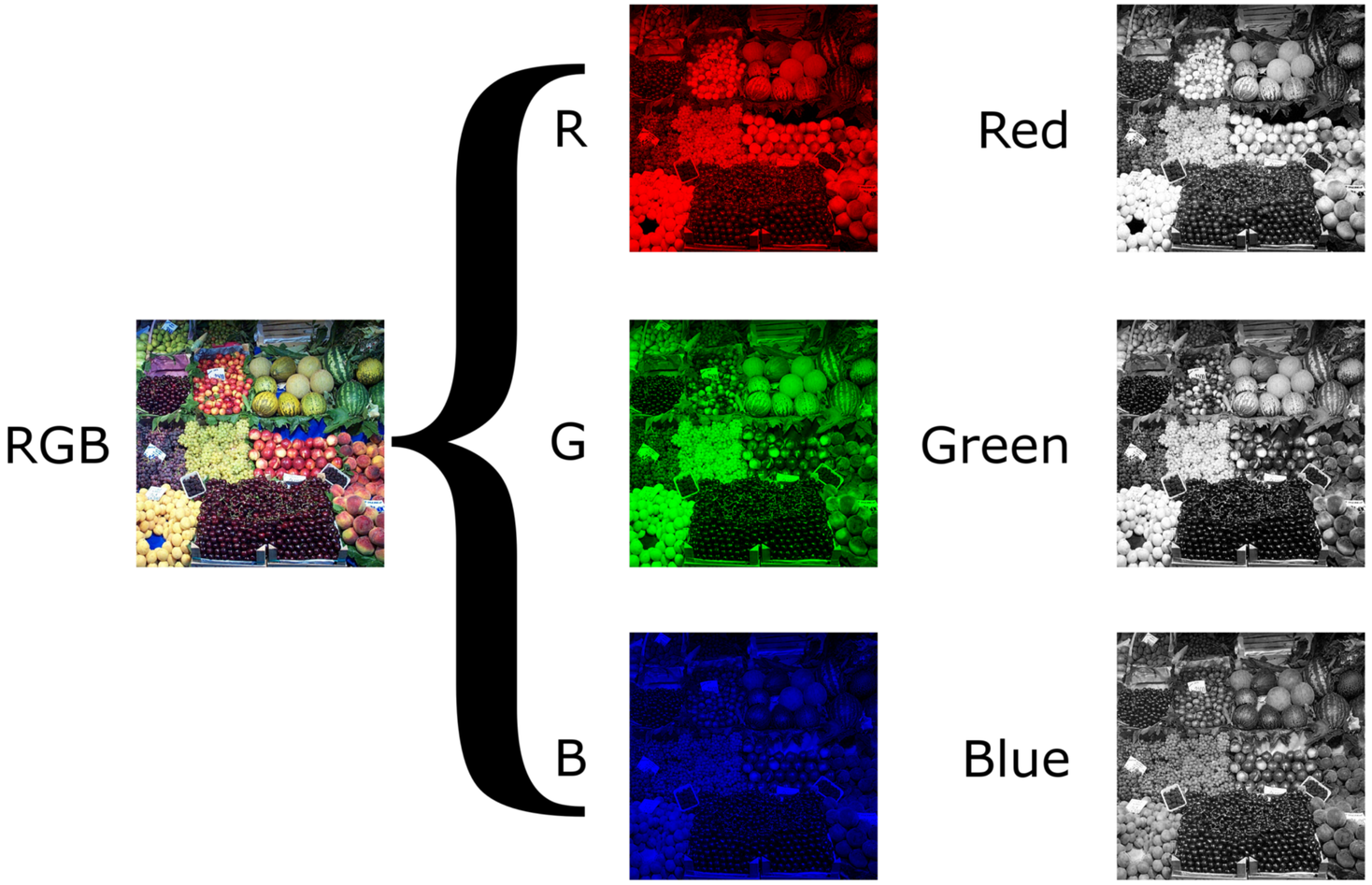


Mona Lisa, by Leonardo da Vinci

Learning From Data



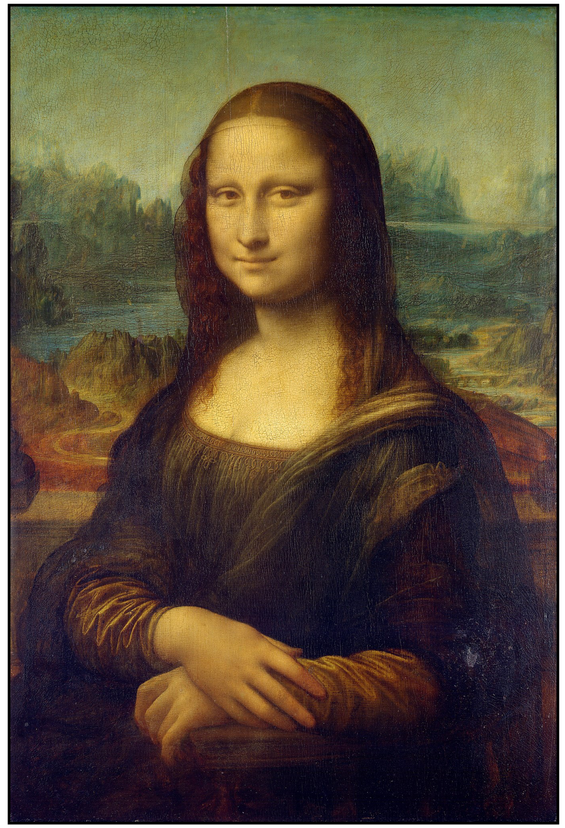
Learning from Data



Multiple Channel per Image
(Usually RGB + Grayscale)

What We Learned from Images

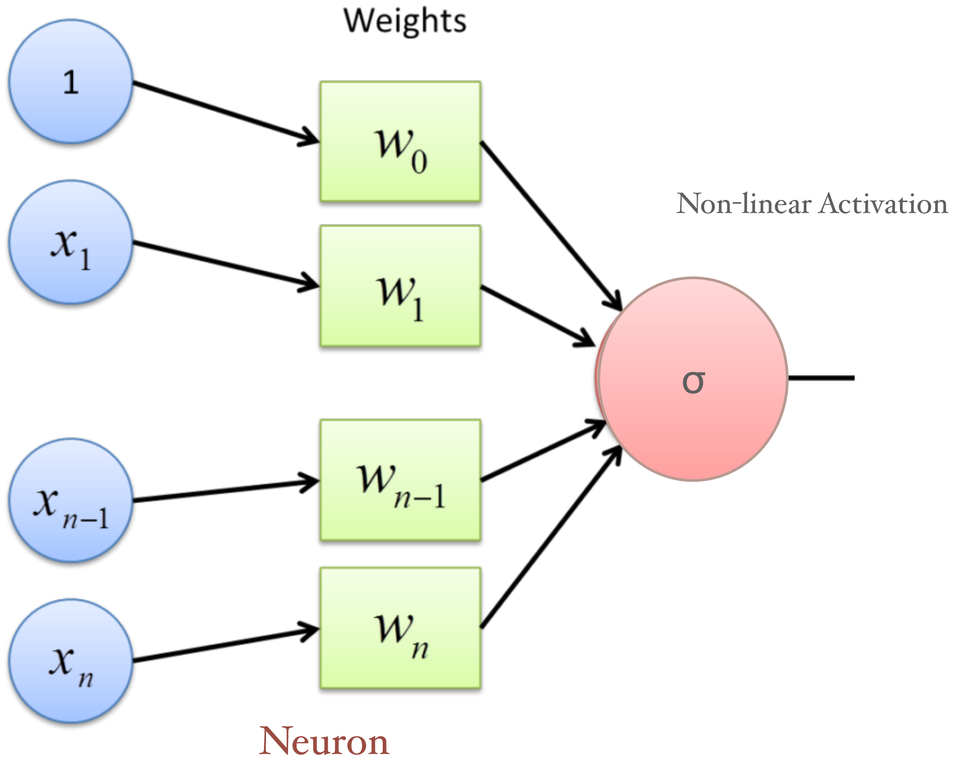
- Each image contains a large amount of float point input:
  - An 1024×768 image contains $786{,}432$ inputs per channel
- Pixel level information alone is usually not that useful (sparsity of image)
- Local level information is useful but usually not enough to make recognition
- Global level awareness arises by combining local level information
- Equivariance of translation is important for image data
- Image data are usually multi-channel



Mona Lisa, by Leonardo da Vinci

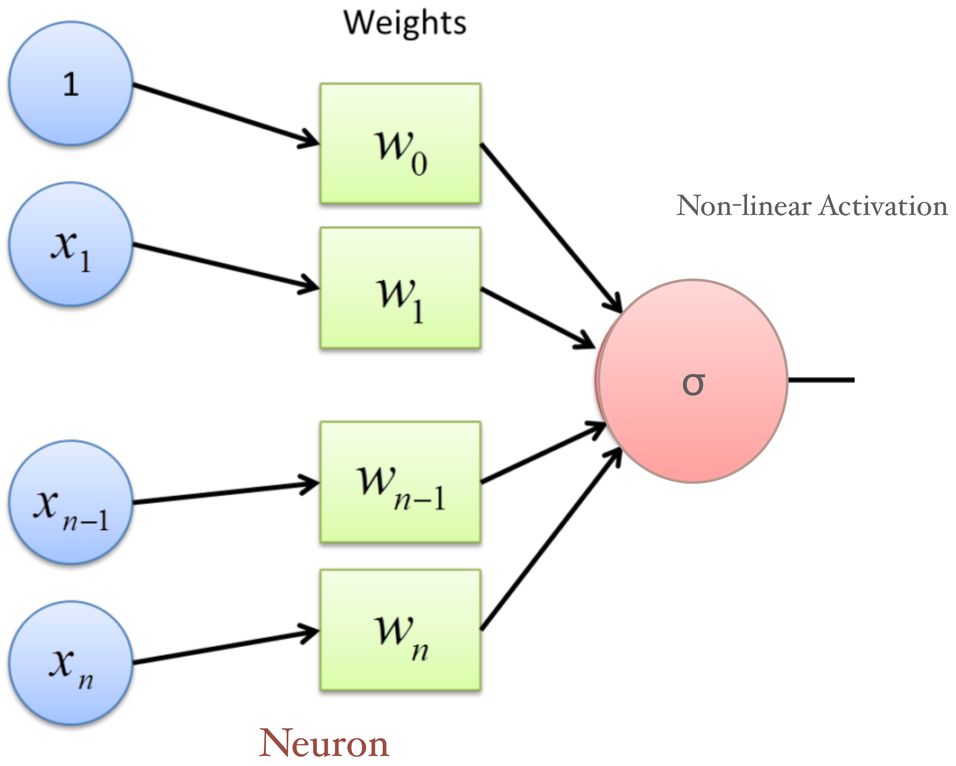
Using Fully Connected NN to Analyze Image

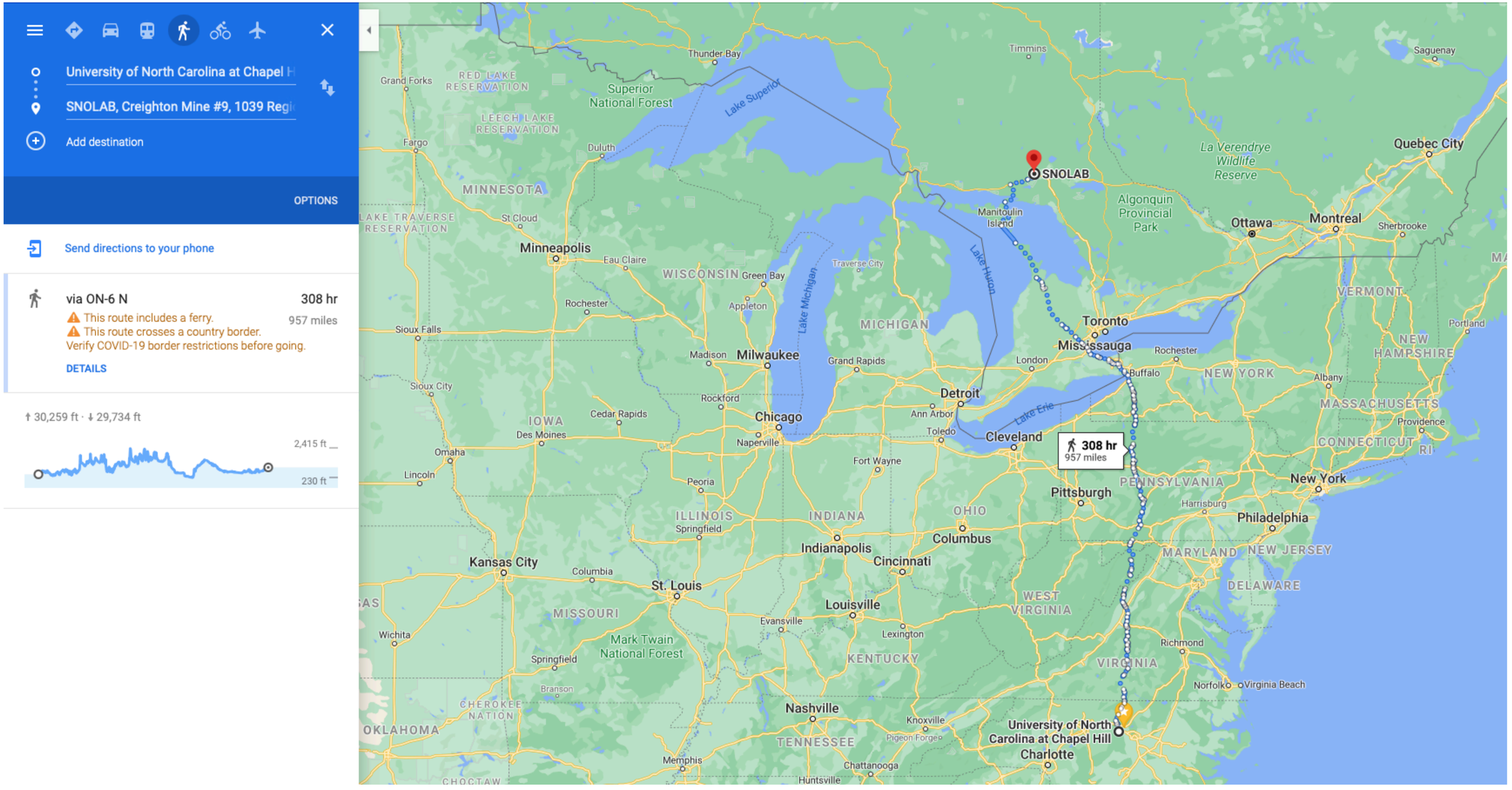
- Universal Approximation Theorem(UAT) states that it is possible to use a 3-layer fully connected neural network to approximate any function, given mild restrictions on activation function
- If we design FC NN to analyze 1024×768 image:
  - Each neuron needs $(786{,}432 + 1) \times 4$ weights in kernel
  - If we have 100 neurons in the first layer, then the parameters of NN easily exceeds 78 million



Using Fully Connected NN to Analyze Image

- If we attempt to train this NN:
  - Fully connected NN assign a weight to every single pixel of the image, and treat them with equal footing
  - Since pixel level information is usually not that useful, the FC NN can easily get lost in the sea of pixels
  - It will be extremely hard to optimize the FC NN to recognize local level information and further gain global awareness
  - FC NN is not equivariance to translation
- The UAT model for image might exist, but it's not possible to train

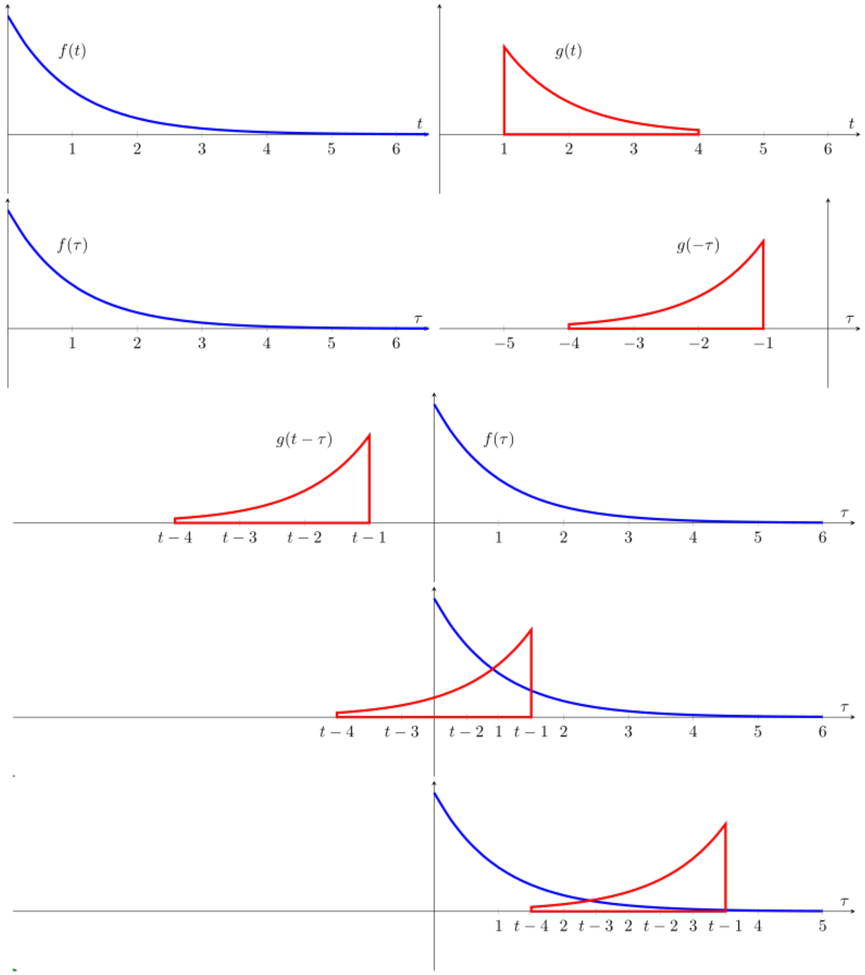



University of North Carolina at Chapel H
SNOLAB, Creighton Mine #9, 1039 Regi
Add destination
OPTIONS
Send directions to your phone
via ON-6 N
308 hr
957 miles
This route includes a ferry.
This route crosses a country border.
Verify COVID-19 border restrictions before going.
DETAILS
↑ 30,259 ft · ↓ 29,734 ft
2,415 ft
230 ft
SNOLAB
308 hr
957 miles
University of North Carolina at Chapel Hill
Thunder Bay
Timmins
Saguenay
Quebec City
Grand Forks
RED LAKE RESERVATION
Superior National Forest
Lake Superior
LEECH LAKE RESERVATION
Fargo
Duluth
MINNESOTA
La Verendrye Wildlife Reserve
Algonquin Provincial Park
Manitoulin Island
Ottawa
Montreal
Sherbrooke
LAKE TRAVERSE RESERVATION
St Cloud
Minneapolis
Eau Claire
WISCONSIN
Green Bay
Traverse City
Lake Michigan
Lake Huron
Rochester
Appleton
MICHIGAN
Toronto
Mississauga
VERMONT
Portland
NEW HAMPSHIRE
Sioux Falls
Madison
Milwaukee
Grand Rapids
London
Rochester
Buffalo
NEW YORK
Albany
Sioux City
Rockford
Detroit
Ann Arbor
Lake Erie
MASSACHUSETTS
Providence
IOWA
Cedar Rapids
Chicago
Des Moines
Toledo
Cleveland
CONNECTICUT
RI
Omaha
Naperville
Fort Wayne
Lincoln
Peoria
Pittsburgh
PENNSYLVANIA
New York
Harrisburg
ILLINOIS
INDIANA
OHIO
Springfield
Philadelphia
Columbus
Indianapolis
MARYLAND
NEW JERSEY
Kansas City
Columbia
Cincinnati
St. Louis
DELAWARE
WEST VIRGINIA
MISSOURI
Louisville
Wichita
Evansville
Lexington
Mark Twain National Forest
Springfield
Richmond
KENTUCKY
VIRGINIA
Branson
Norfolk
Virginia Beach
CHEROKEE NATION
Nashville
Knoxville
OKLAHOMA
Pigeon Forge
TENNESSEE
Charlotte
Memphis
Chattanooga
Huntsville
CHOCTAW


Q: Are all the local level information
equally important?

Convolution

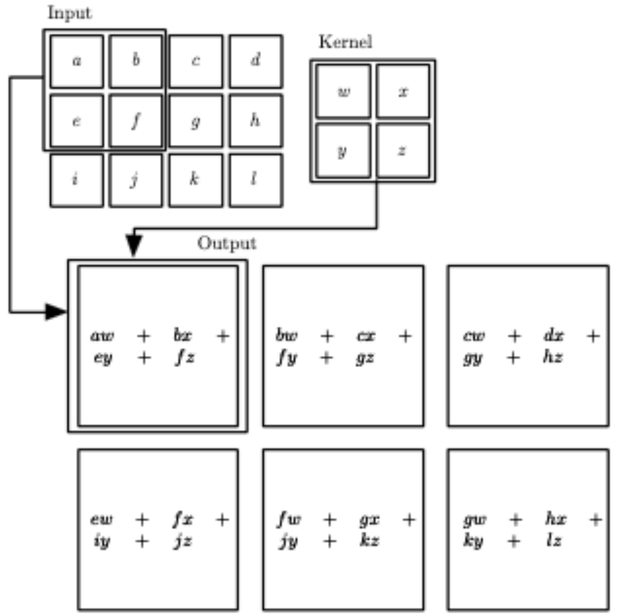
- Convolution: $s(t) = (x * w)(t) = \int x(\tau)w(t-\tau)d\tau$
- Discrete Convolution: $s[n] = (x * w)[n] = \sum x[m]w[n-m]$
- Discrete convolution is commonly used in digital signal processing
  - The trapezoidal filter to get Ge detector energy
- Extending discrete convolution to 2D:
  - $s(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n)$

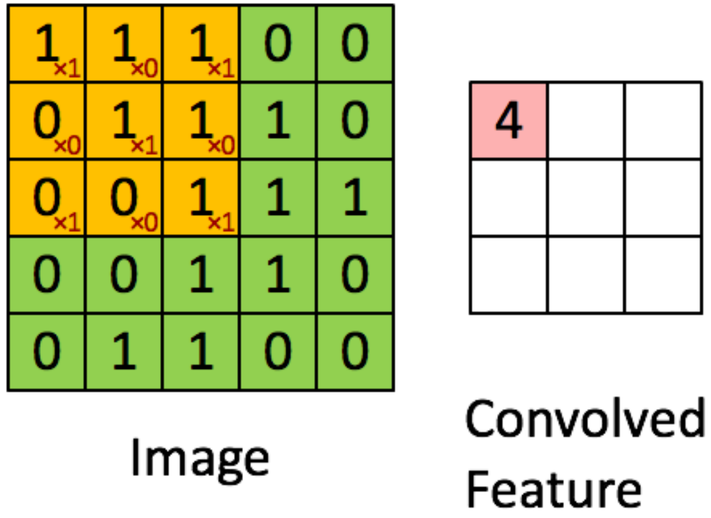


Wikipedia

Convolutional Layers

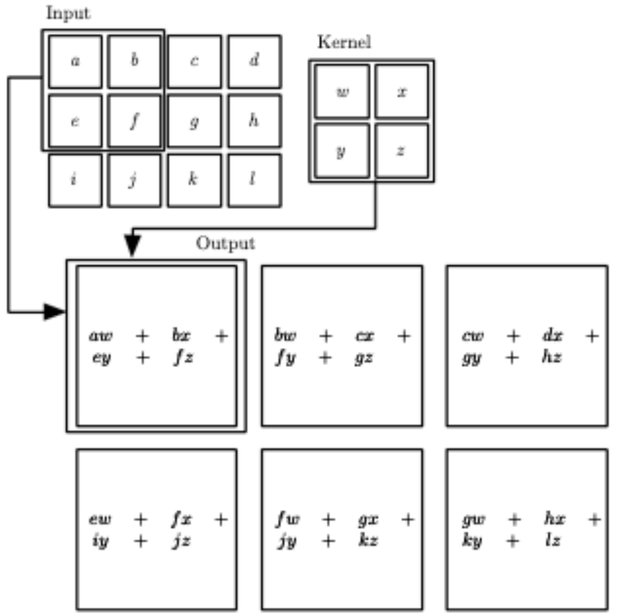
- Each convolutional layer contains a set of kernels, or sometime called filters
  - The values within kernel changes during the training process
- The kernels are sliced over the surface of input images
- For each step, a weighted sum is calculated between the kernel weights and input values at corresponding location
- The weighted sum is then placed in the output feature maps
- This process resembles 2D discrete convolution





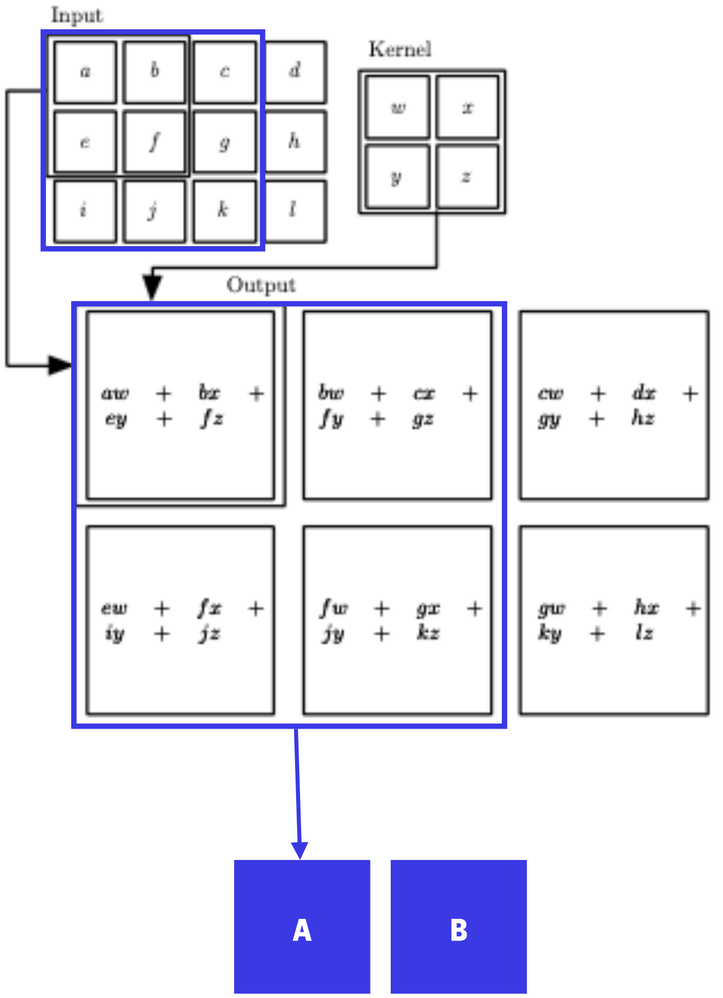
Why Convolution Works?

- Convolution is the same as the filtering process in Ge detector, except that the shape of filter changes w.r.t. data
- Convolutional layers solves a set of key problem we brought up in previous sections:
  - Image data size is usually large → Kernel size of convolutional layers is much smaller than the image
  - Pixel level information is not important → Output feature map contains only summed information under each kernel step
  - Equivariance to translation → After training, the weight of kernel is fixed. Thus even if the local level feature is at a different location, its value in the feature map will be the same
- We are still missing the process of assembling local level information to global level awareness, is this achievable in convolutional layer?



Reception Field

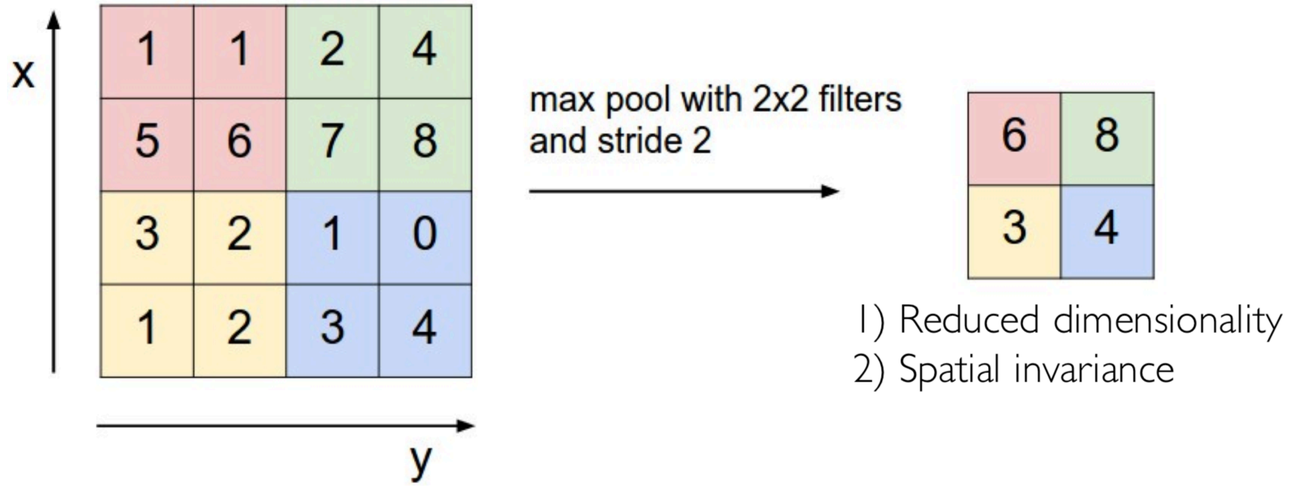
- The answer is yes, but not with a single convolutional layer
- Reception Field of a convolutional neural network (CNN) is the area CNN can perceive in a single feature map cell
- If the CNN is only one layer, then the reception field is simply the Kernel size
- However, if we stack another convolutional layers on the previous layer, taking the output feature map from previous layer as input, then we can increase the reception field of CNN
- Larger reception field helps the CNN to gain global awareness, thus Deep CNN is always preferred



Feature Map of Second Convolutional Layer

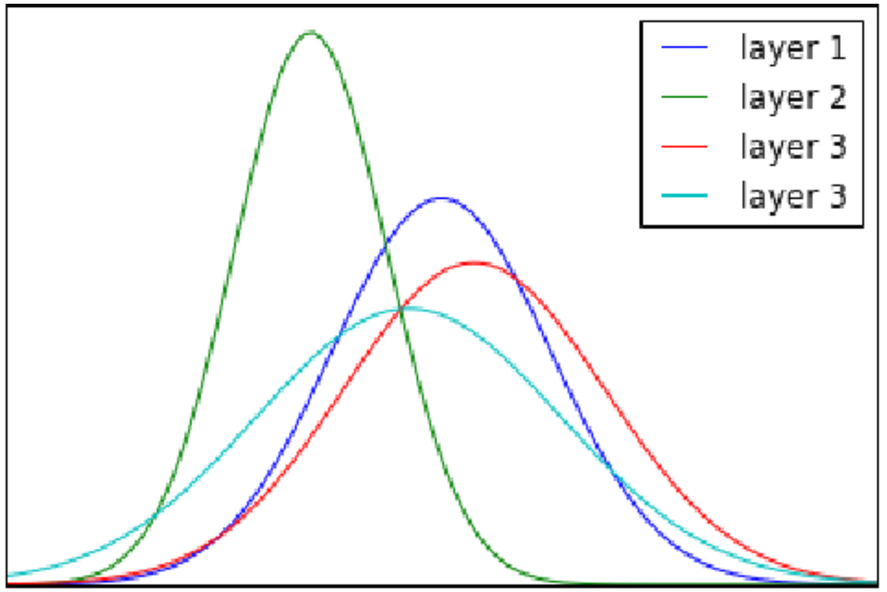
Pooling Layer

- Even after several layers of convolution, the size of input might still be two large to gain global awareness
- Pooling layer is applied to the intermediate feature maps of CNN to solve this.
- Advantages of pooling layer:
  - Reduce dimensionally of feature maps
  - Preserve the spatial invariance of input images
  - Introduce additional non-linearity
  - Greatly increase the reception field of CNN
- Disadvantage is the information loss
- But as long as we do not apply pooling layer to the input, because *life will find a way*
  - Analogy: election



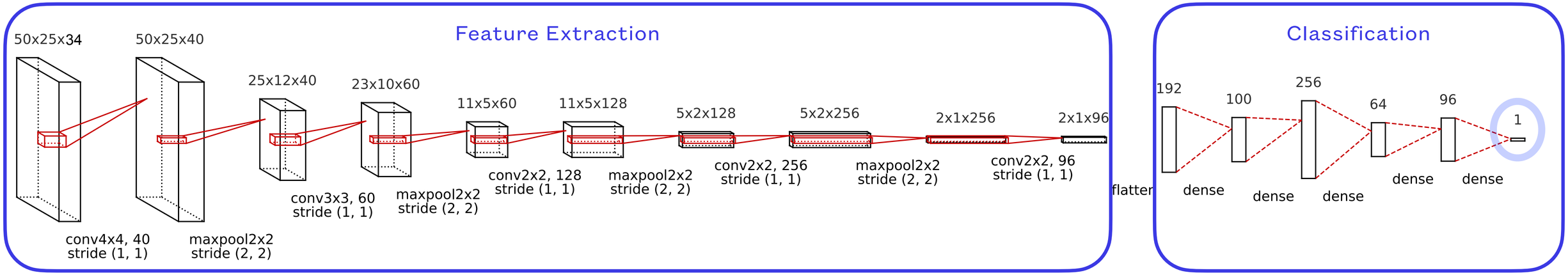
Internal Covariant Shift and Batch Normalization

- As neural network get deeper and deeper, the output distribution of each layer tends to shift
- This internal covariant shift slow down the convergence of training
- Batch Normalization is a layer in deep NN to compensate the internal covariant shift
  - $\hat{x} = \frac{x - E[x]}{\sqrt{\text{Var}[x]}} * \beta + \gamma$, (β, γ) are kernel parameters
  - Shifting the feature map after each layer, so that they have 0 mean and 1 standard deviation
- Without BN, the training of deep CNN can sometimes be quite painful



(b) Internal covariate shift

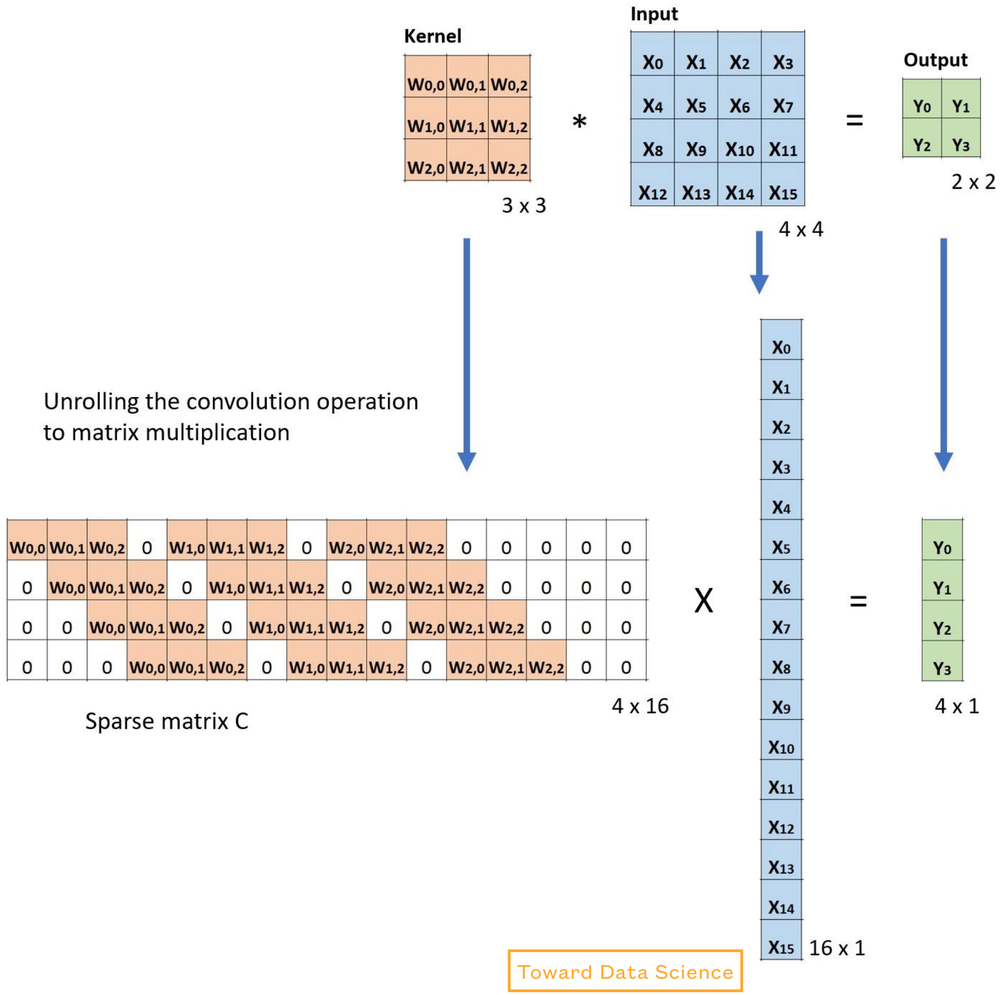
Convolutional Neural Network



- Input images:
  - 50(H)×25(W)×34(Ch)
- Feature Extraction:
  - Convolutional Module: ConvLayer→MaxPooling→BatchNorm→Activation
  - 5 repetition of Convolutional Modules form the feature extraction part of CNN
- The last feature map is flattened into a 192-dimensional feature vector
- Fully connected NN makes classification decision based on feature vectors

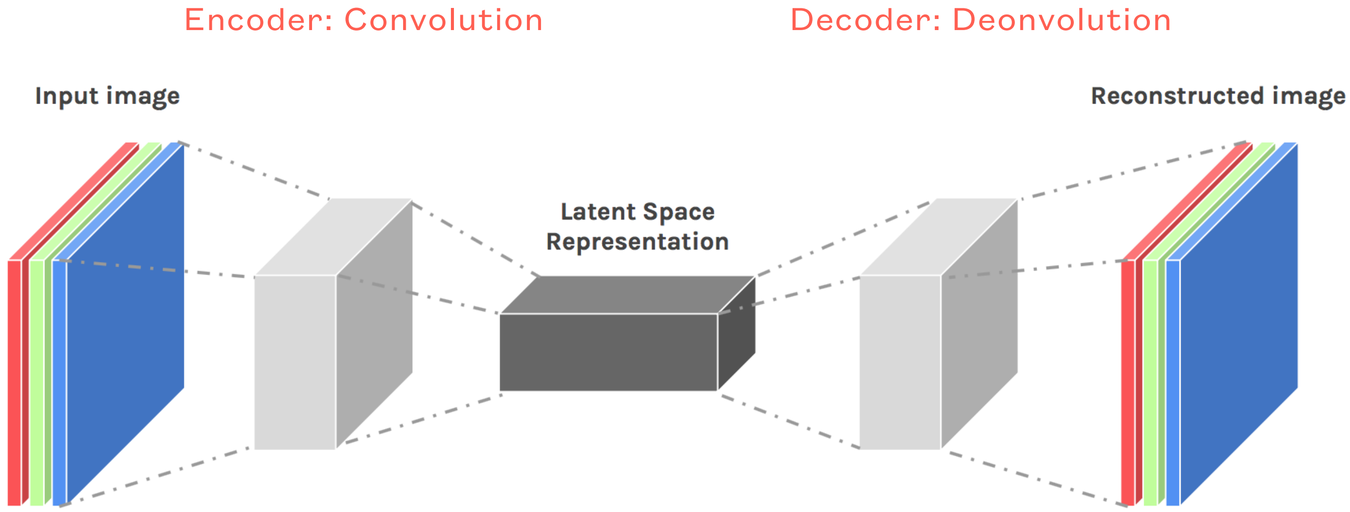
Convolution in PyTorch

- In PyTorch, convolution is not performed by actually sliding the kernel over the entire surface
  - Kernel sliding is a serial operation, which is the advantage of CPU but not GPU
- Unroll kernel and input into a matrix and a vector
- Perform a single matrix multiplication to convolve
- Vectorized approach, can be significantly accelerated by GPU
- Allows the inverse operation: Deconvolution



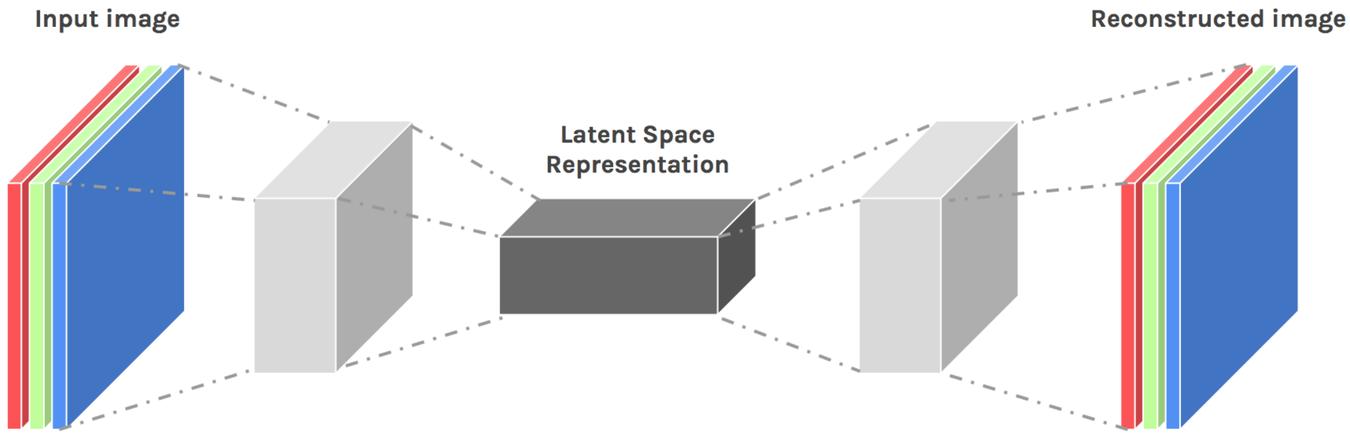
Convolutional Autoencoder

- Concatenating two CNNs back-to-back
  - Encoder: stacked convolutional layers
  - Decoder: stacked deconvolutional layers
- Train to reconstruct the input image
  - Minimize the MSE loss between input images and reconstructed images
- Each image is encoded into a latent space vector by the encoder
- The image can be reconstructed from the latent space vector via the decoder



Convolutional Autoencoder

- Convolutional Autoencoder is an unsupervised learning algorithm
  - No label is needed
- The latent space representation contains information suffice to reconstruct the image
- Thus, CAE can be treated as an image suppression technique
- We can perform traditional unsupervised learning algorithms on the latent space vector of each image



Extended CNN Models

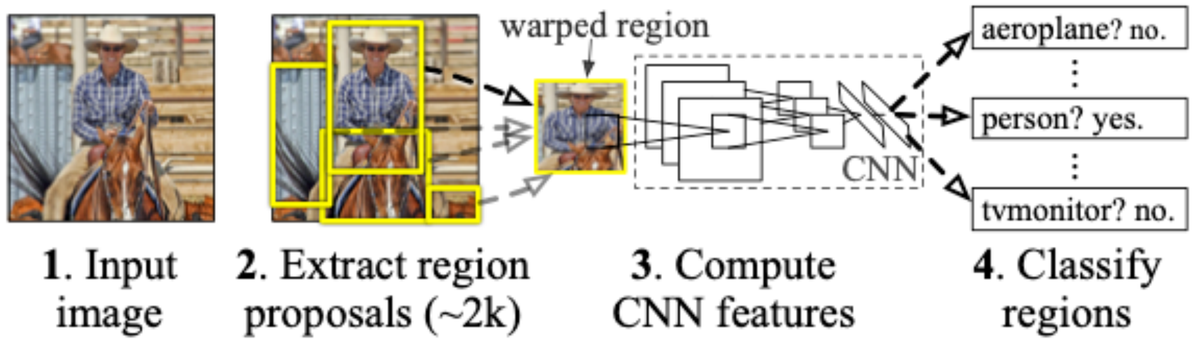
- After learning the vanilla CNN model, we will extend our reach to some more complicated CNN-based models:
  - How to let CNN extract location information of objects?
  - How to deal with non-flat images?
  - How to deal with videos?

CNN with Location Info

- In many situations, we not only want the type of objects in an image, but also where they are located at
- Pictures on the right not only contains object labels, but also a “box” containing the object
- To achieve this, we need to extract both classification information and location information from CNN feature extractor

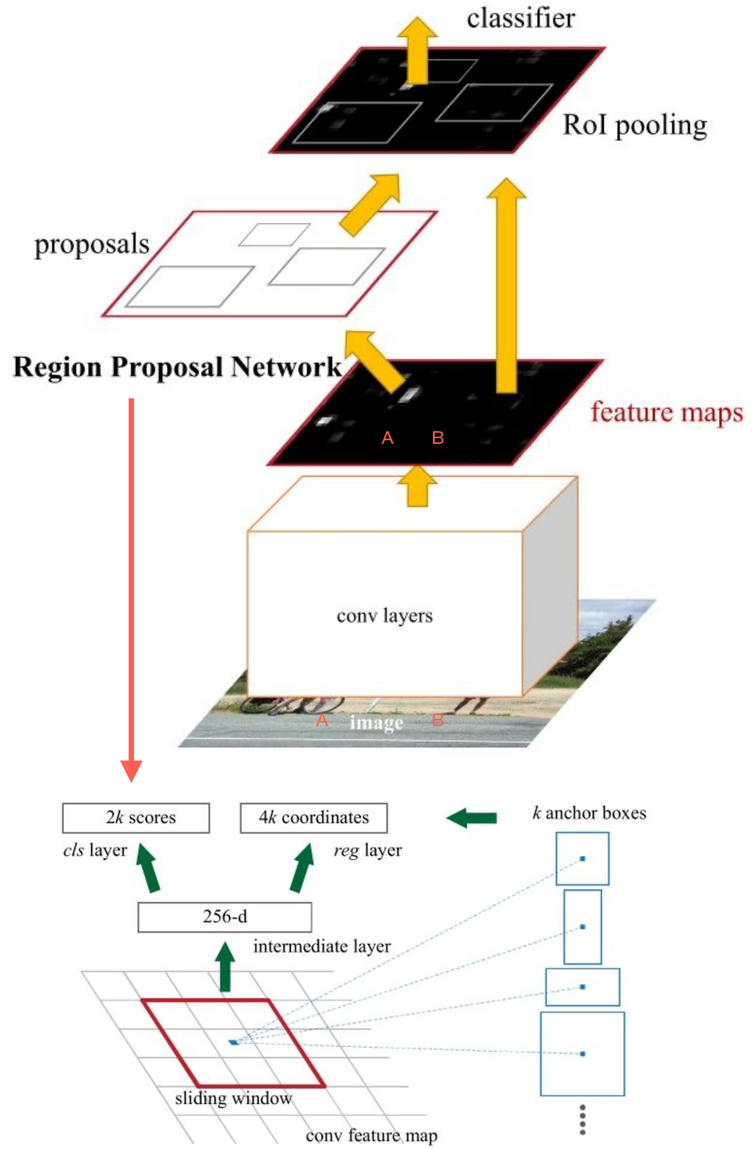
Regional CNN

- Ross Girshick et al. (ArXiv: 1311.2524)
- Proposing part of the image as region proposal
- Run CNN classifier in the anchor to classify
- Label each anchor with classification decisions
- This model requires explicit region proposal, which is a highly repetitive and redundant process



Faster R-CNN

- Shaoqing Ren et al.(ArXiv: 1703.06870)
- CNN feature maps have a useful property: the relative location of images is preserved
  - Resultant of equivariance of translation
- Proposing anchors based on the output feature maps to the original image
- Training the network for two objectives:
  - Classification for the types of objects
  - Regression for the accurate location of anchor boxes
- Read the original paper for more technical details

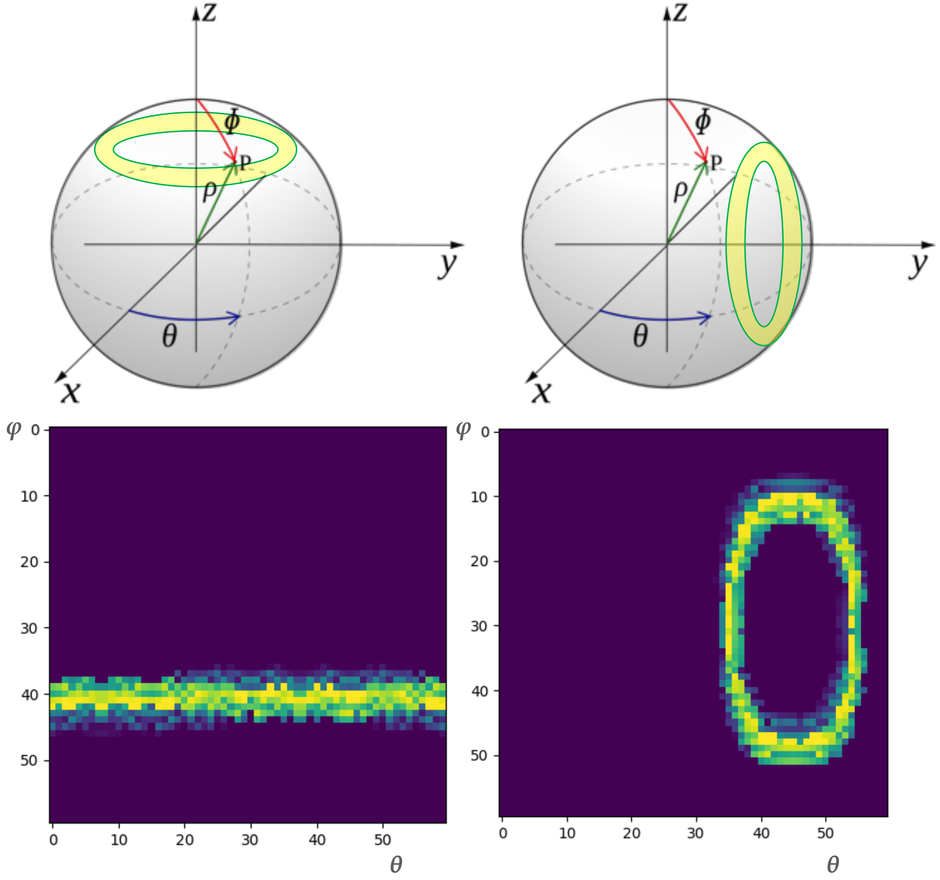


The Achille's Heel of Vanilla CNN

Spherical distortion of image

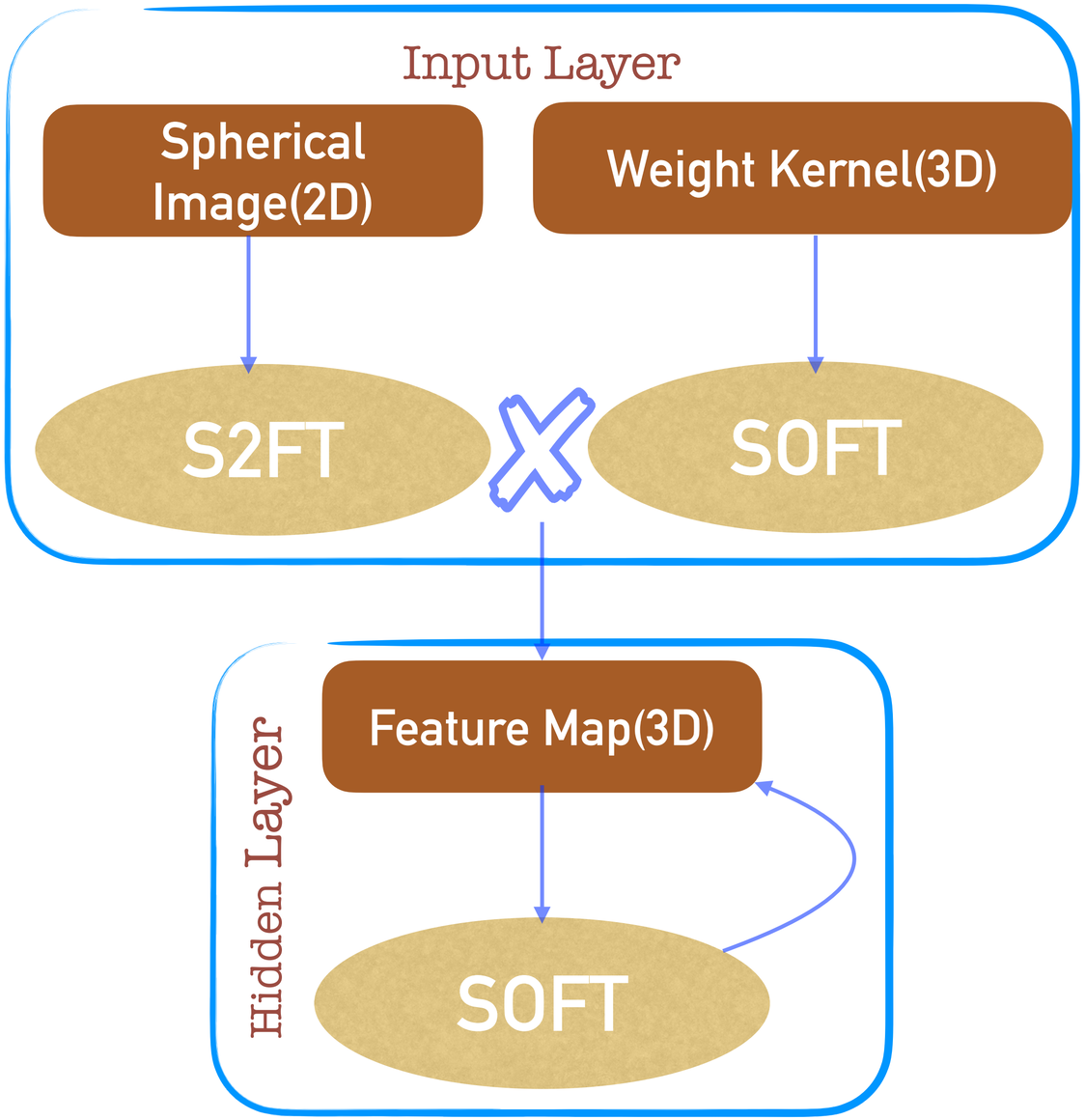
LATIS Lab

CNN is not rotational invariant



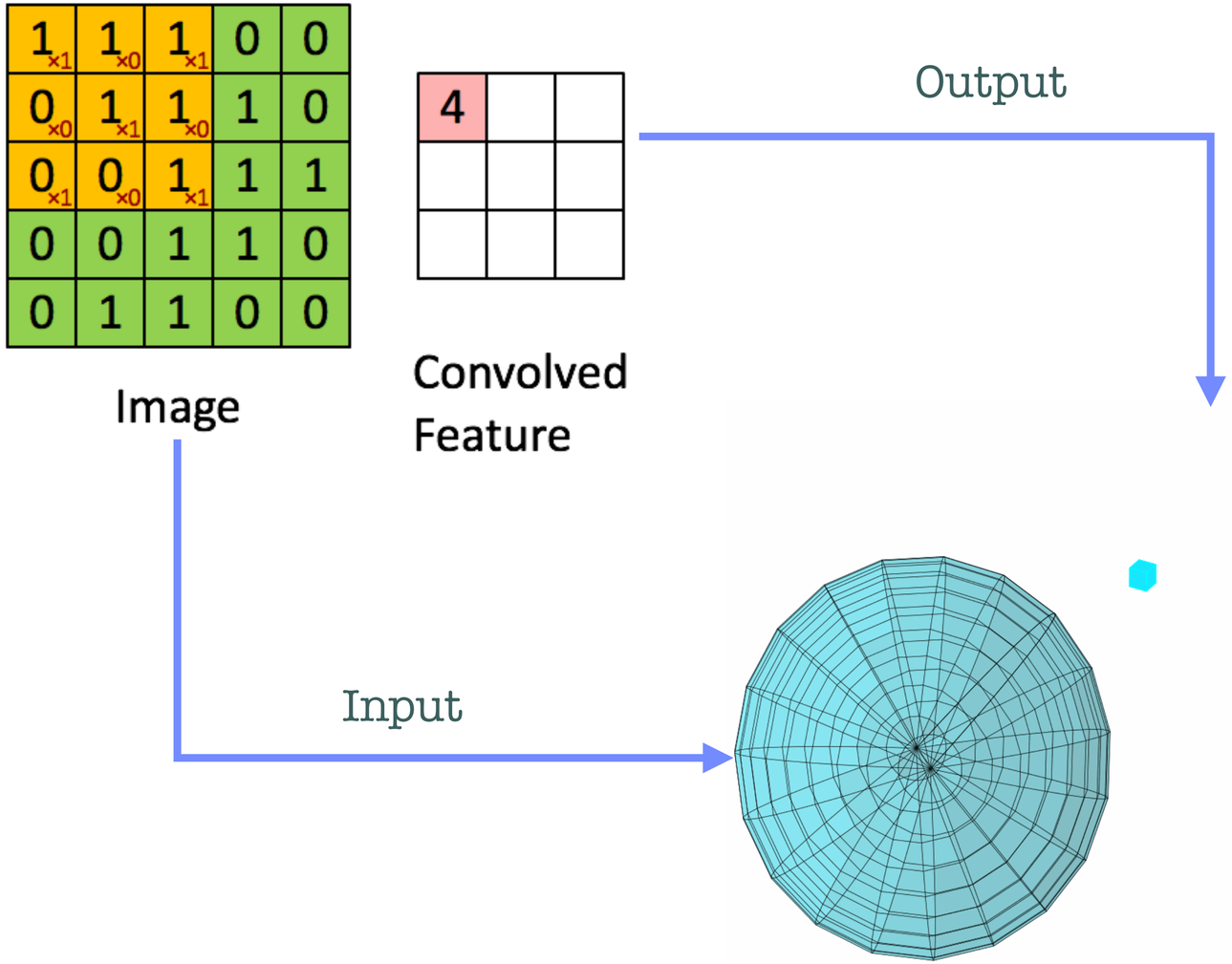
Spherical CNN

- Cohen et al. (ArXiv:1801.10130)
- Convolution is equivalent to production in Fourier space
- If we can calculate the Fourier transform of both spherical image and rotation, then we can perform convolution on spherical images
  - This is achieved by SO(3) Fourier transform (SOFT)
- Performing FT on spherical image (2D) and weight kernel (3D)
- Multiply them in Fourier space
  - Multiplication in Fourier space is convolution in real space
- Output feature map in 3D
- Stacking more layers by introducing new weight kernel



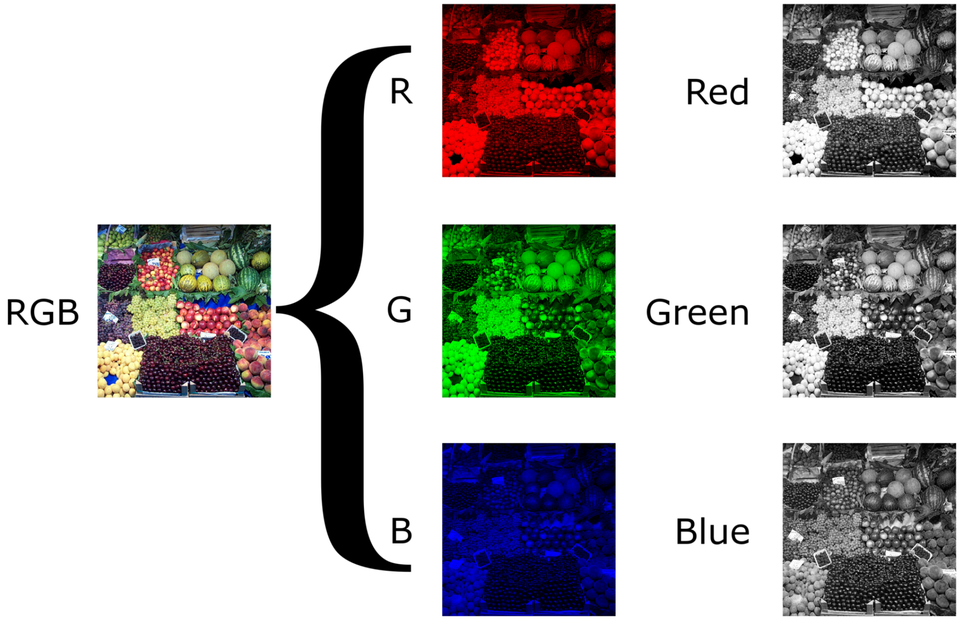
Spherical CNN

- CNN views image by scanning a rectangular filter through the image
- CNN outputs 2D feature map in Euclidean space
- Spherical CNN views image by rotating it into different angles
  - The “filter” covers the entire sphere
  - Exhibit rotational invariance
- Spherical CNN outputs 3D feature map in Euler angle space



Spatiotemporal Data

- Spatiotemporal data means that the dataset contains both spatial and temporal information
  - Videos → a series of frames, labeled by time
- Canonical approach: using Conv3D
  - 3D convolution is usually very expensive
  - There is a lot of repetitive information in the 3D tensors
- Alternative Approach: using Conv2D
  - Treating each frame as a channel to the input
  - Less expensive than Conv3D, but still very large kernel size due to overwhelming number of channels
  - Channel is not ordered, thus we lose order information by doing this

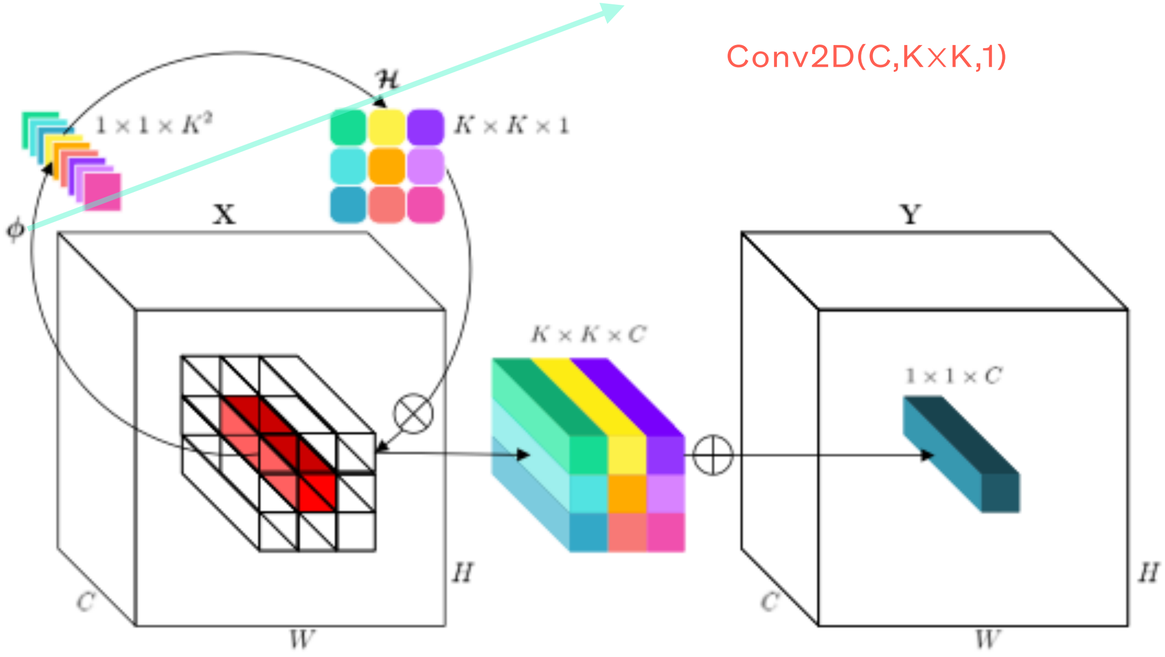


Involution for Spatiotemporal Data

- Duo Li et al. (ArXiv 2103.06255)
- Proposed a novel basic structure called Involution instead of Convolution
- The spirit is to perform convolution along the channel direction
- This would allow each channel to “talk” to each other properly
- This model is original proposed to allow weight sharing among different channels
- Since in spatiotemporal data, our channel is the frames of image, involution might help comprehending the temporal dimension

This is achieved by performing 2D convolution:

Conv2D(C,K×K,1)



Summary

- Features of Image Data:
  - Local level information
  - Global awareness
  - Equivariance to transition
  - Channel
- Vanilla CNN
  - ConvLayer and Reception Fields
  - Pooling Layer
  - Batch Normalization
- Convolutional Autoencoder
- Extended CNN Models:
  - Regional CNN and Faster R-CNN
  - Spherical CNN
  - Involution for Spatiotemporal Data