MIT GSL
GLOBAL STARTUP LABS

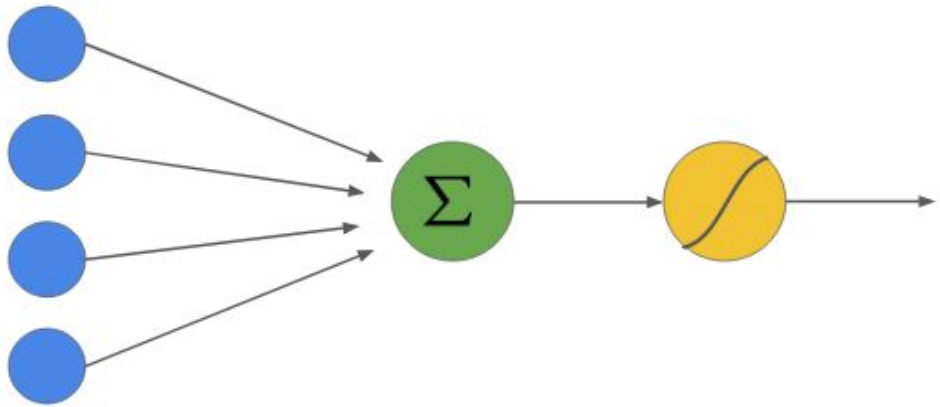
Intro to Deep Learning and Computer Vision

MIT GSL-PRO
Ryan Sander

How Does Deep Learning Relate to Previous Models We've Covered?

MIT GSL
GLOBAL STARTUP LABS

- Let's revisit our **logistic regression** model:

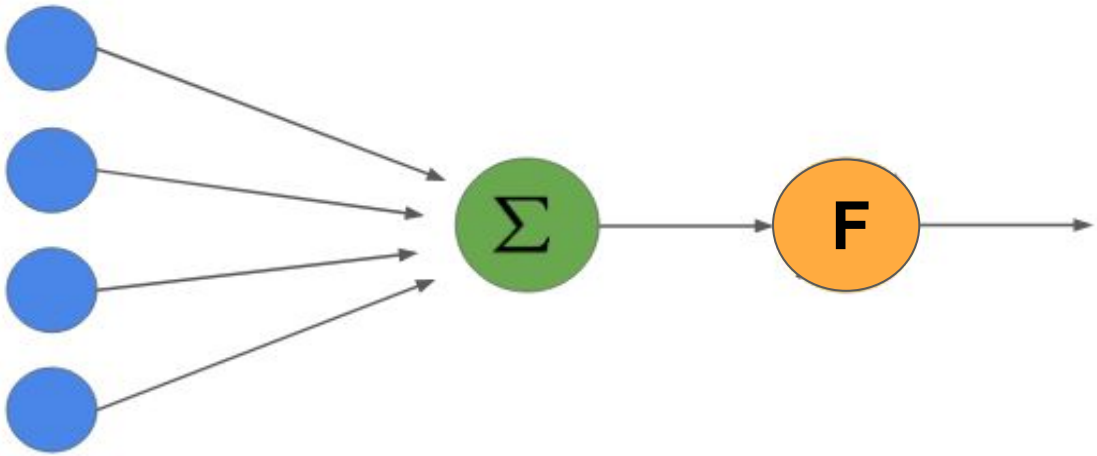


- We can think of this as a **single unit neural network** with a **sigmoid activation**!

Neural Networks As Cascaded Groupings of Units

MIT GSL
GLOBAL STARTUP LABS

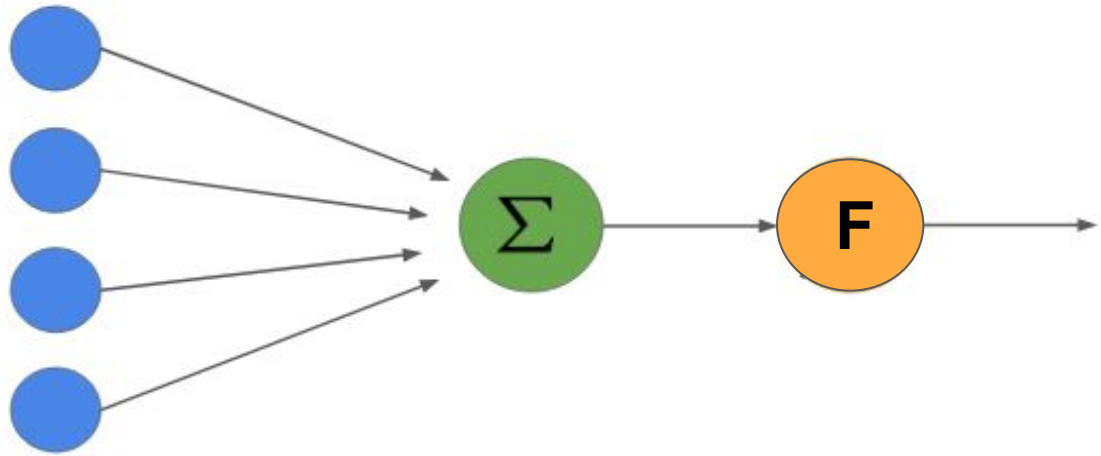
- When **F** is a **sigmoid**, this becomes logistic regression.
- When **F** is **linear**, this becomes linear regression.
- When the **F** is **cross-entropy**, this becomes binary/multiclass classification.



Neural Networks As Cascaded Groupings of Units

MIT GSL
GLOBAL STARTUP LABS

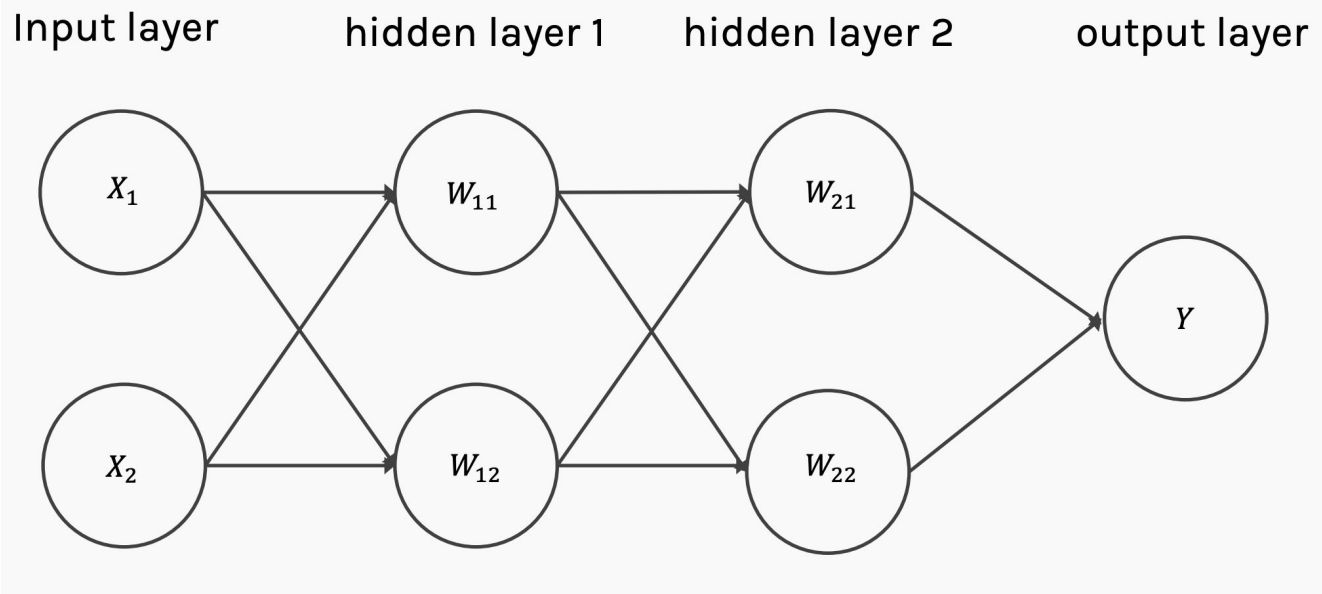
- This intuition will be helpful as we discuss how networks **make predictions**, and how they are **trained.**



Review from Last Week – Neural Networks

MIT GSL
GLOBAL STARTUP LABS

- Last week, we discussed **neural networks**, and how they can be used for **supervised learning**.
- In addition to **NLP** and **e-commerce**, neural networks can also be used to solve problems in **computer vision** and **healthcare**.



Applications of Neural Networks

MIT GSL
GLOBAL STARTUP LABS

Autonomous Vehicles

Google

Insurance Claims

Chatbots

Different Classes of Neural Networks

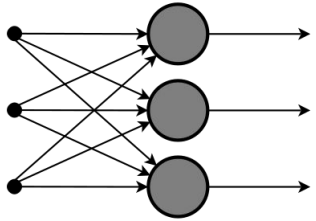
MIT GSL
GLOBAL STARTUP LABS

Many different kinds of neural networks:

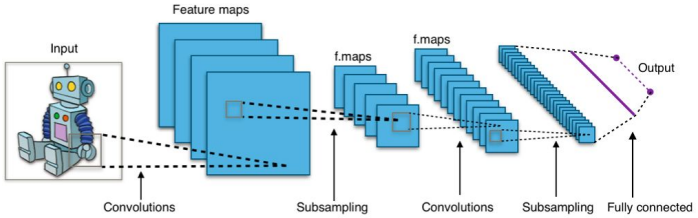
- **Feedforward**
- **Convolutional (CNNs)**
- Recurrent (RNNs)
- Graph
- Variational Autoencoders* (VAEs)
- Generative Adversarial Networks (GANs)

This course only analyzes the first two kinds of networks.

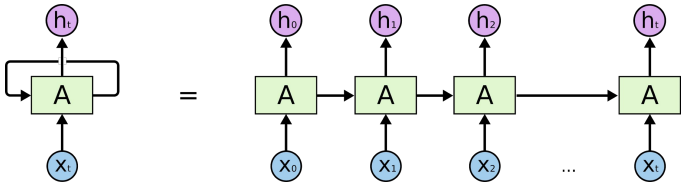
[1] **Feedforward** network class



[2] **Convolutional** network class



[3] **Recurrent** network class



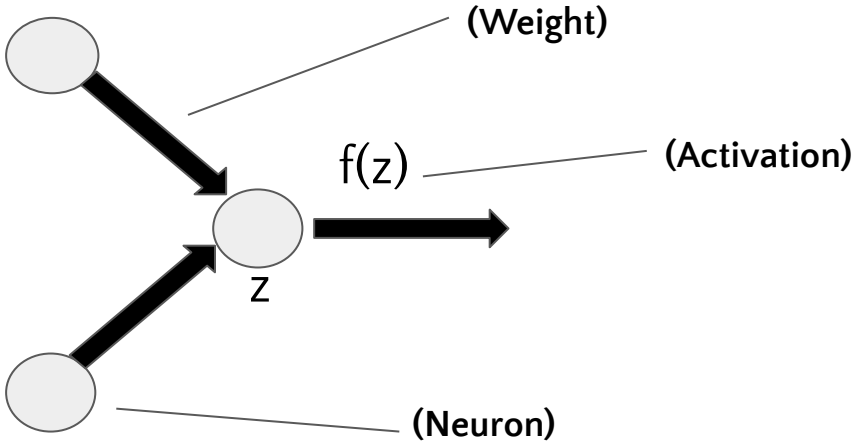
* VAEs are actually unsupervised! They are part of a class of algorithms known as **deep generative models.**

Anatomy of a Feedforward Neural Network

MIT GSL
GLOBAL STARTUP LABS

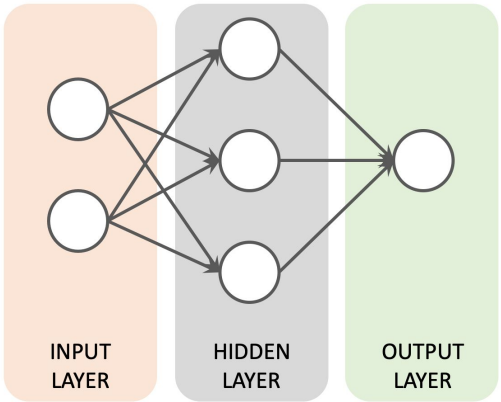
Feedforward neural networks have 3 basic elements:

- **Neurons**
- **Weights**
- **Activations**



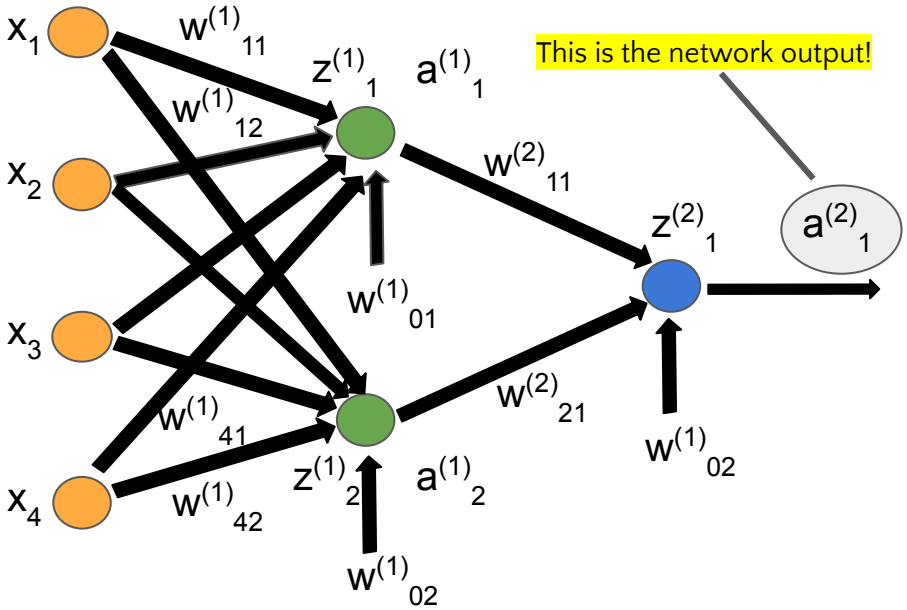
Feedforward networks are composed of **cascaded layers** of neurons, weights, and activations

- **Input** layer
- **Hidden** layer(s)
- **Output** layer



How Neural Networks Make Predictions

- Input is propagated through the network with the **forward** algorithm
- At each layer:
  - Neuron **activations** from previous layer are multiplied by layer **weights**, and **summed** together.
  - Activation function is applied to the **sum**.



[1] Weighted sum of activations:

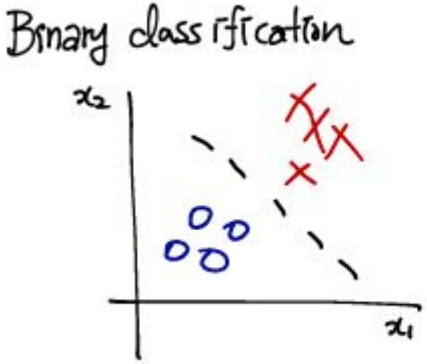
$$z^{(L)}_i = \sum x_j w^{(L-1)}_{ji} + w^{(L-1)}_{0i}$$

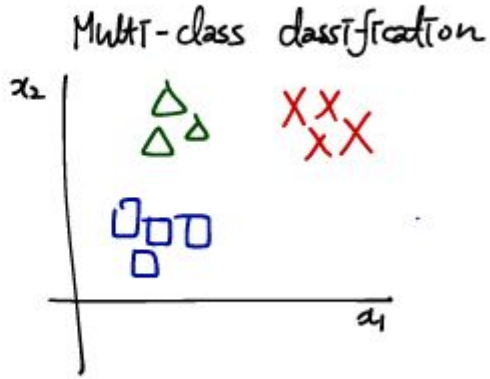
[2] Activation of sum:

$$a^{(L)}_i = f(z^{(L)}_i)$$

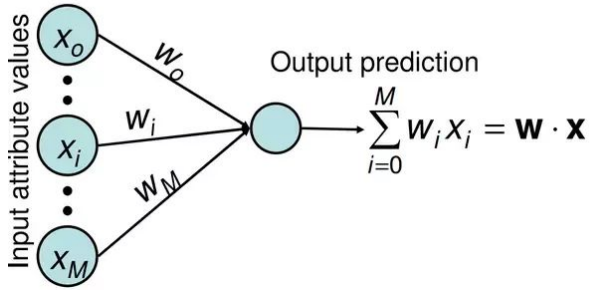
Intuition for Neural Network Predictions

- Trained neural networks learn “when activations should be **high** or **low**”
- Output layers typically reflect what the network is used for:
  - Any real number output → **Regression**
  - {1, 0} output → **Binary Classification**
  - {1, 2, 3, ..., 10} output → **Multiclass Classification**





Linear Regression with Neural Networks!

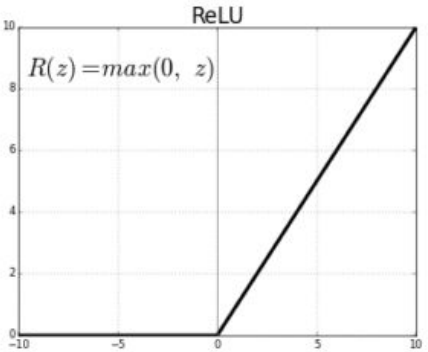


Output Activation Functions and their Applications

MIT GSL
GLOBAL STARTUP LABS

[1] Regression:

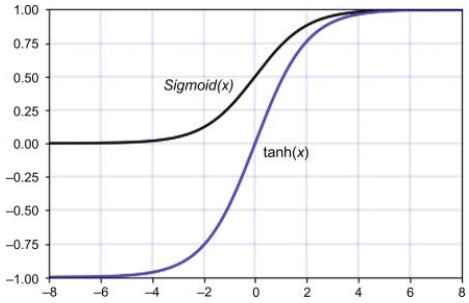
- Linear: **f(x) = x,** varies from **(-∞,∞)**
- Rectified Linear Unit (ReLU): **ReLU(x),** varies from **(0,∞)**



[1]

[2] Logistic Regression/Binary Classification:

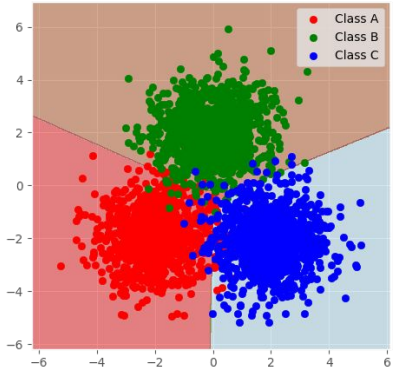
- Sigmoid: **σ(x),** varies from **(0,1)**
- Hyperbolic Tangent: **tanh(x),** varies from **(-1,1)**



[2]

[3] Multiclass Classification

- SoftMax: **SM(x_i),** varies from **(0,1)**

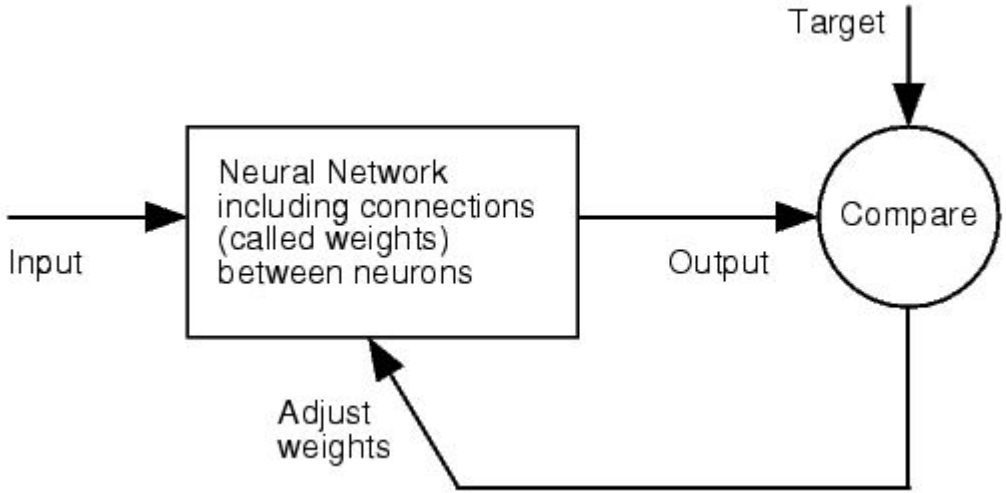


[3]

Training Neural Networks – Overview

MIT GSL
GLOBAL STARTUP LABS

- For neural networks to be useful, we need to **train** them
- Neural networks can be trained by **optimizing** their **weights**.
  - This is how neural networks "**learn**"!
  - Different **weights** lead to different **predictions**
- These weights are "learned" through the **backpropagation** algorithm.

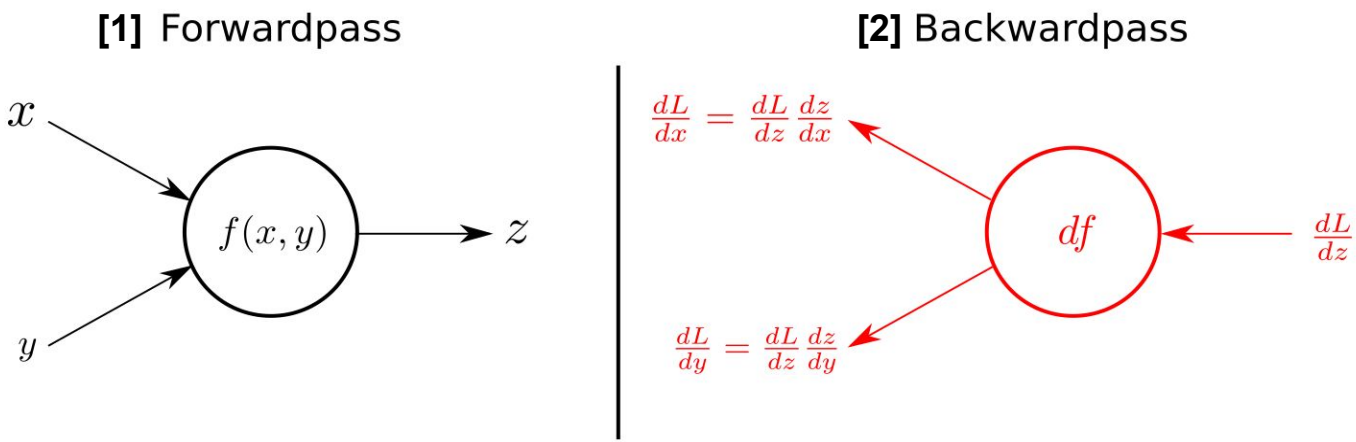


The Backpropagation Algorithm

MIT GSL
GLOBAL STARTUP LABS

Algorithm composed of two stages:

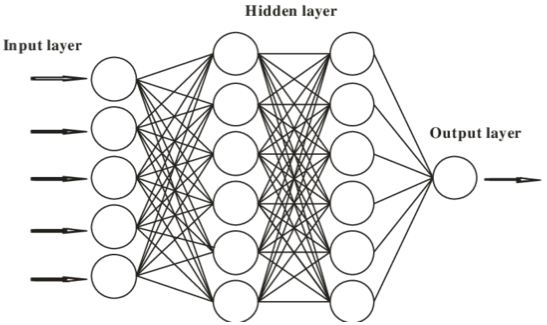
1. **Forward pass:** Network makes prediction with labeled training data.
2. **Backward pass:** Network weights are updated according to how "wrong" the prediction is.

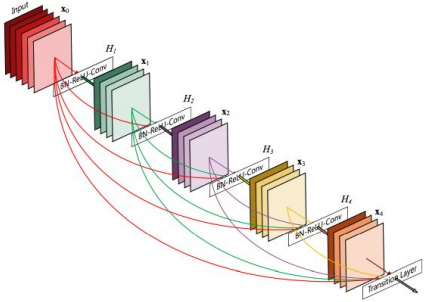


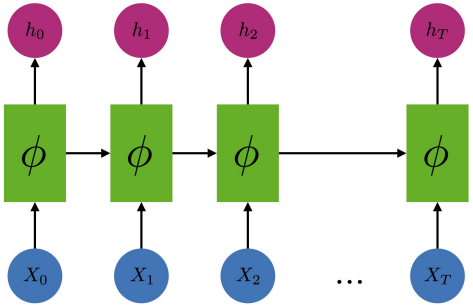
Overview - Deep Learning

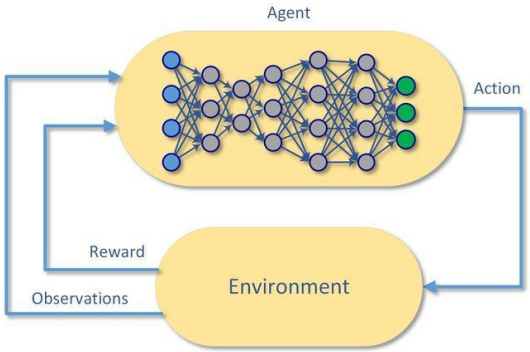
Field of **Deep Learning** uses **Deep Neural Networks (DNNs),** which can learn remarkably complicated tasks, given sufficient data and training. Some examples include:

- Object **detection** and **classification**
- Speech and image **generation**
- High-dimensional **predictions**
- **Recommendation** systems
- **Decision-making** for agents









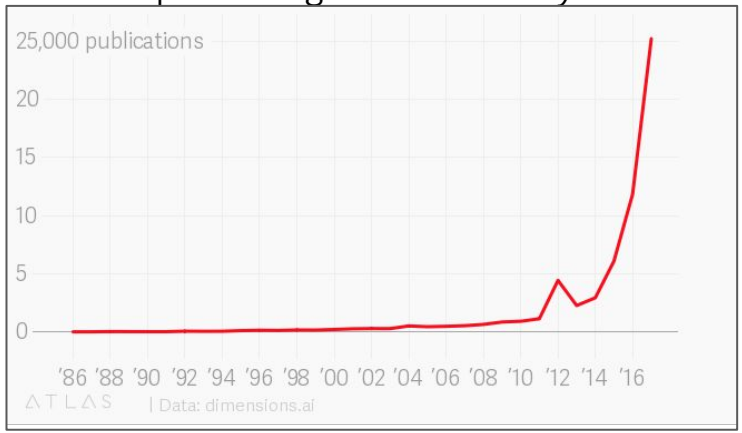
Deep Learning Is A Very New Field

MIT GSL
GLOBAL STARTUP LABS

Field has seen a lot of advances in recent years from development of better **CPUs** and graphical/tensor processing units (**GPUs/TPUs**).

Advances are quite literally happening every day!

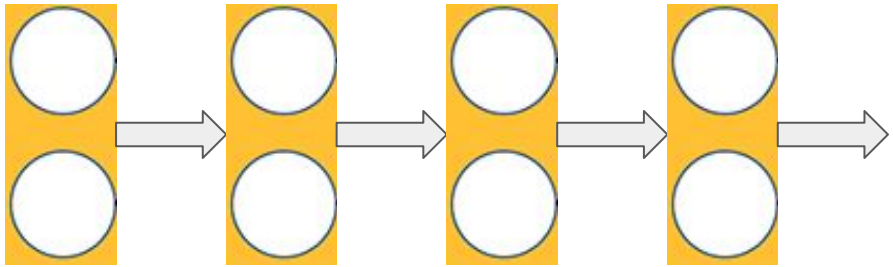
Deep Learning Publications by Year

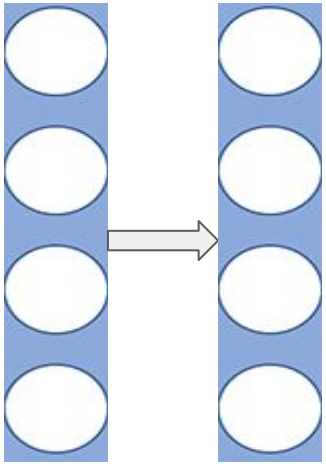


Deeper, Not Wider, Neural Networks*

MIT GSL
GLOBAL STARTUP LABS

more layers with **less weights** > **less layers** with **more weights**



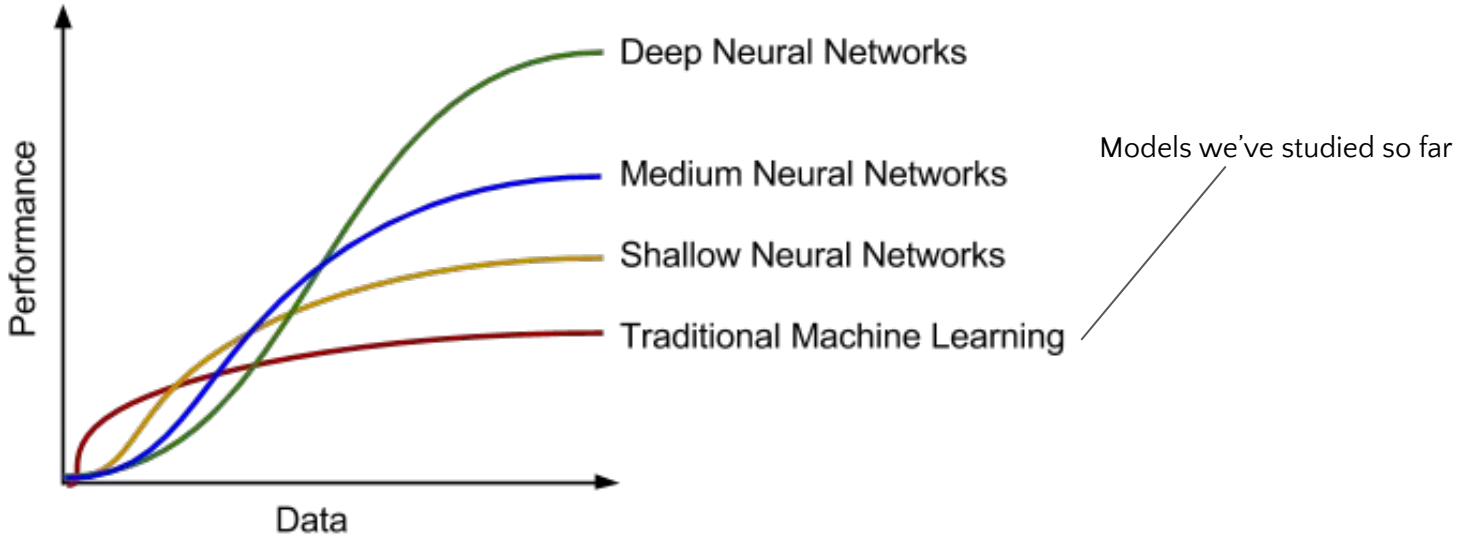


*For More Information on this: Stack Overflow Post.

Why Deep Learning?

MIT GSL
GLOBAL STARTUP LABS

- Why should we even consider deep learning when we already have machine learning models that can solve our problems?
- Deep learning can often achieve **higher performance** than other models
- Caveat: **Need more data** in order to achieve **higher performance**



Deep Learning Packages in Python

MIT GSL
GLOBAL STARTUP LABS

These packages allow us to implement, train, and evaluate million-parameter neural networks in **less than 15 lines of code!**

Main deep learning package we will use in this class: **Keras.**

Other packages to explore:

- **TensorFlow**
- **PyTorch**
- **Caffe**
- **Chainer**
- **FastAI**

MIT GSL
GLOBAL STARTUP LABS

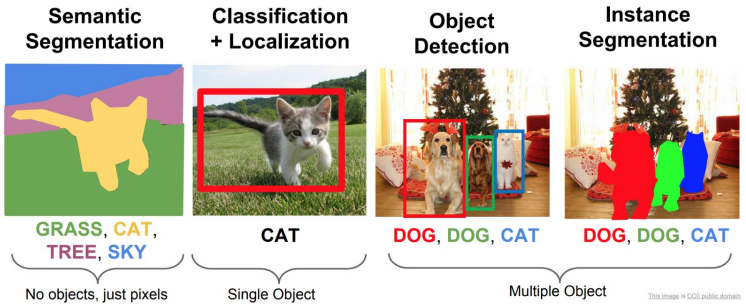
Intro to Computer Vision

MIT GSL-PRO
Ryan Sander

Overview - Computer Vision

MIT GSL
GLOBAL STARTUP LABS

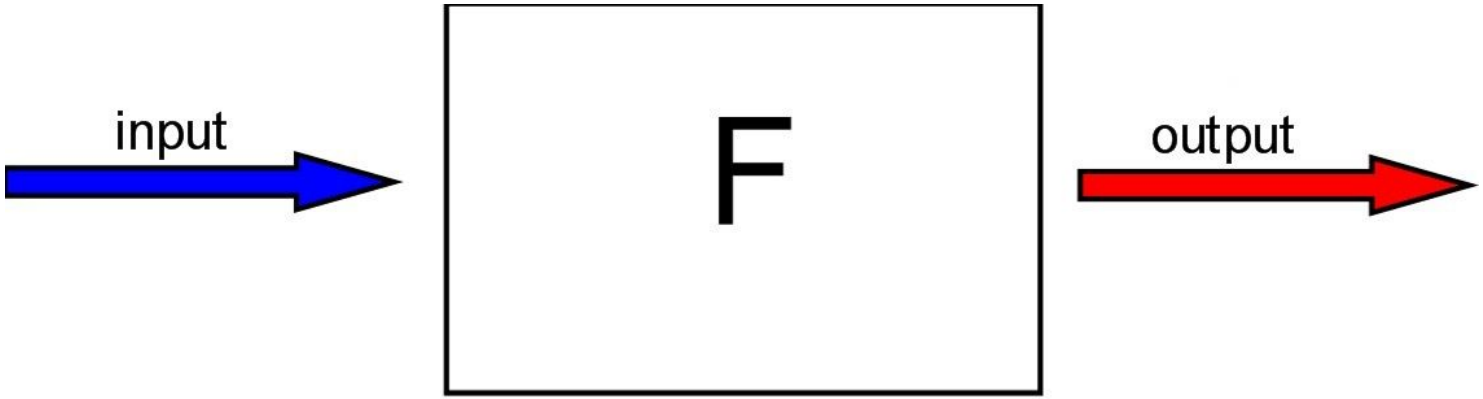
- **Computer Vision** is a subset of artificial intelligence primarily concerned with understanding **spatial and imagery** data. Typical applications include:
  - Image blurring, sharpening, and edge detection
  - Capturing motion
  - Image classification/object detection and classification/scene segmentation
  - Image and video generation
- This field developed independently of deep neural networks, but has improved substantially over the last decade through integration of these deep network frameworks.



Computer Vision as Filtering

- **Filtering** is the most central idea in **computer vision.**

- Intuitive idea behind filtering: A **system F transforms** an **input** into an **output.**



Filtering – Convolutions

MIT GSL
GLOBAL STARTUP LABS

- Filtering is done through **convolution** and **kernels**
- **Convolution** is a mathematical operation that computes a **sum of linear combinations of the input**

$$x[n] * h[n] = \sum_{\mathbf{K}=-\infty}^{\infty} x[n - \mathbf{K}] \cdot h[\mathbf{K}]$$

Input

Filter/Kernel

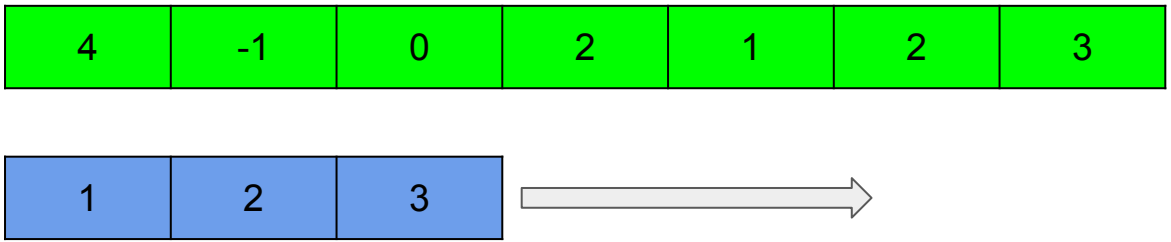
- **Kernels** are matrices* which we **convolve** our **input** with to **filter!**

* For discrete problems, kernels can be matrices. For continuous problems, they take the more general form of an inner product of a function applied to the two inputs. Link HERE.

Convolution* – Building Intuition

MIT GSL
GLOBAL STARTUP LABS

- Think of convolution as **sliding a filter** across the **input**, and computing the product between the **filter/kernel** and **input** for each input value
- This concept is illustrated on the next slides

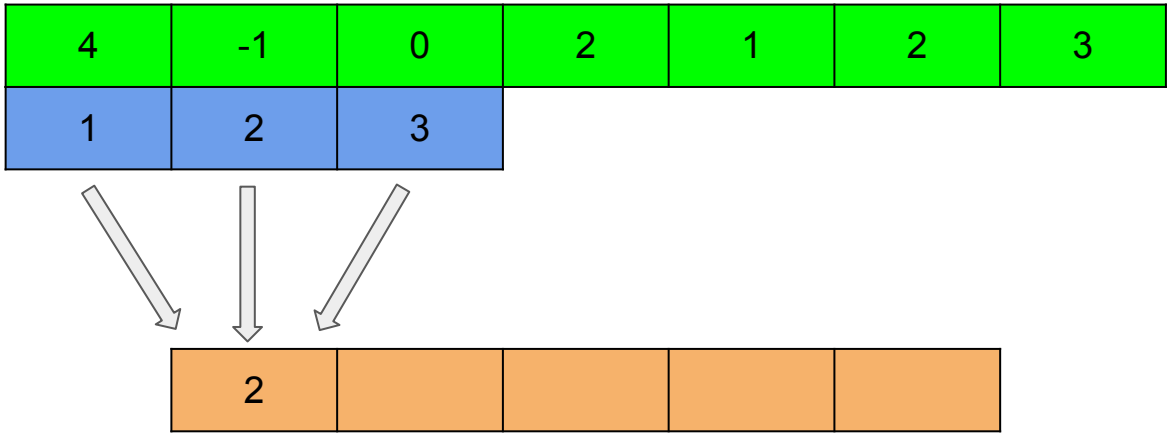


x[n] (Input)

h[n] (Filter)

* Technically, this is **correlation**, not **convolution.** They are both closely related, with just a simple change of sign. If interested, see the link HERE.

Convolution – 1D Example

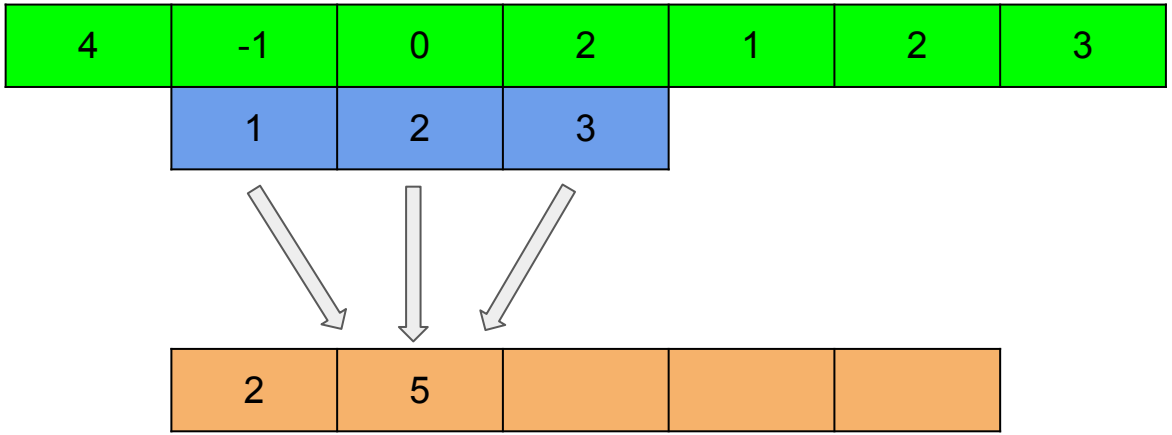


x[n] (Input)

h[n] (Filter)

$y[n] = x[n] * h[-n]$

Convolution – 1D Example

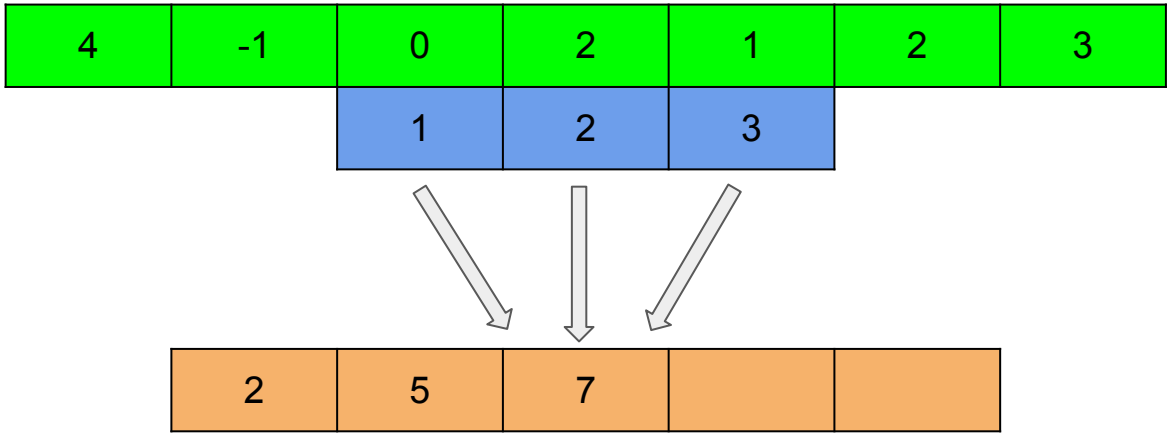


x[n] (Input)

h[n] (Filter)

$y[n] = x[n] * h[-n]$

Convolution – 1D Example

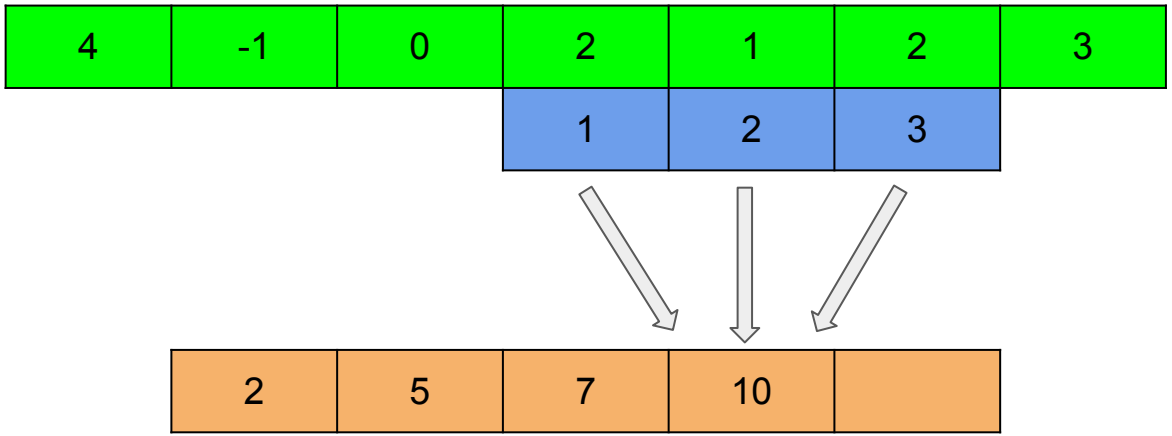


x[n] (Input)

h[n] (Filter)

$y[n] = x[n] * h[-n]$

Convolution – 1D Example

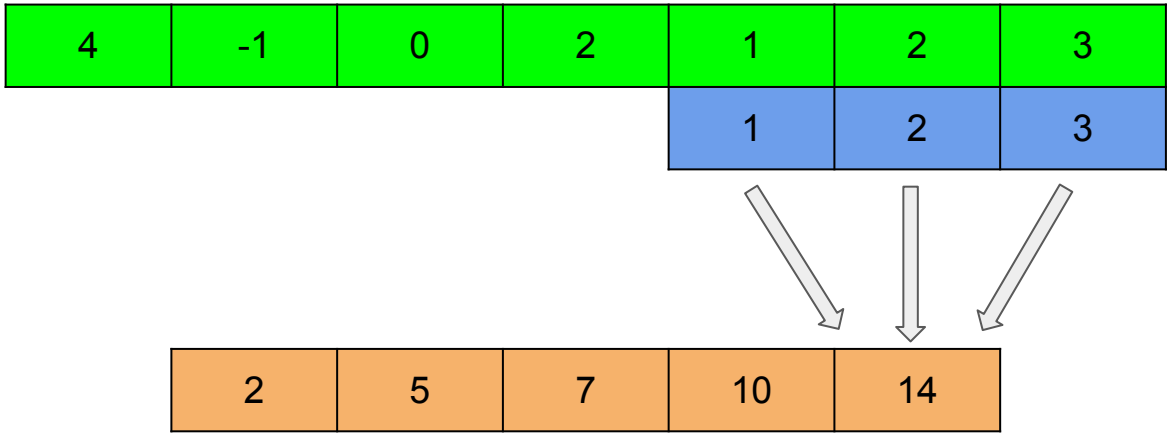


x[n] (Input)

h[n] (Filter)

$y[n] = x[n] * h[-n]$

Convolution – 1D Example



x[n] (Input)

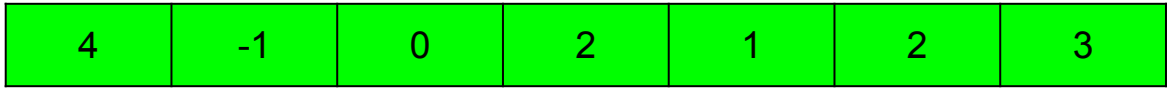
h[n] (Filter)

$y[n] = x[n] * h[-n]$

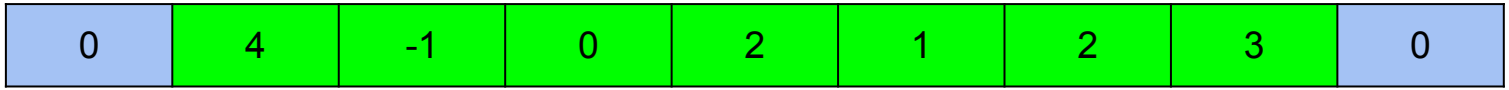
Filtering – “Padding” Techniques

MIT GSL
GLOBAL STARTUP LABS

- What do we do if we want the output to be the same size as input?
  - We can use **padding**!
- Different Types of Padding:
  - **Valid padding** – Do nothing (does not fix size problem)

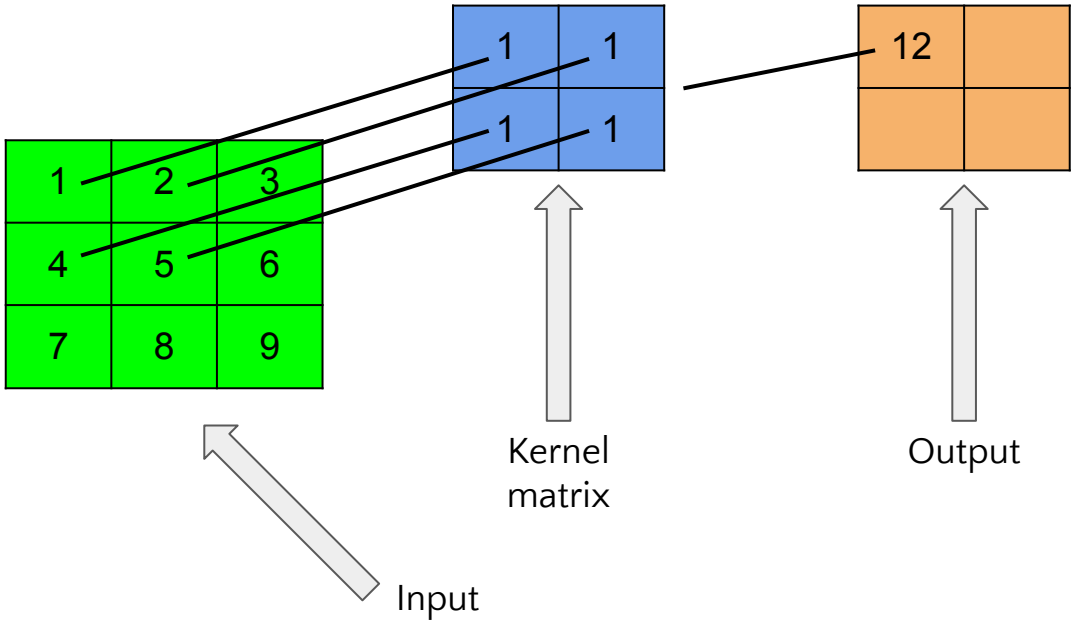


  - **Zero padding** – Add **0**’s to the edges of the input



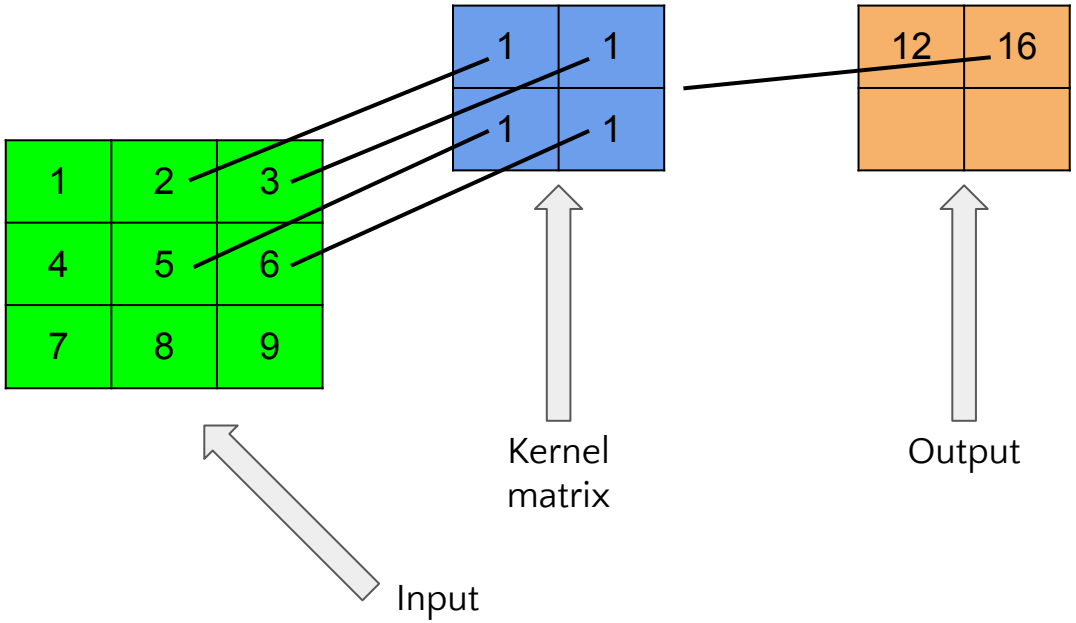
Convolution – 2D Example

- In 2D, we use **matrices** for our **kernels**



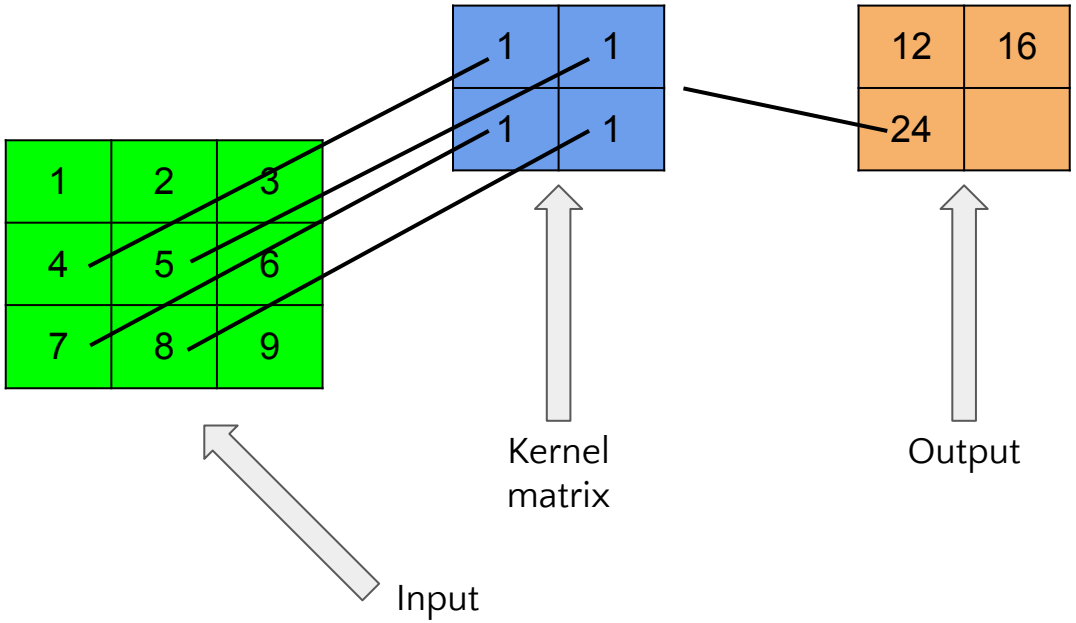
Convolution – 2D Example

- In 2D, we use **matrices** for our **kernels**



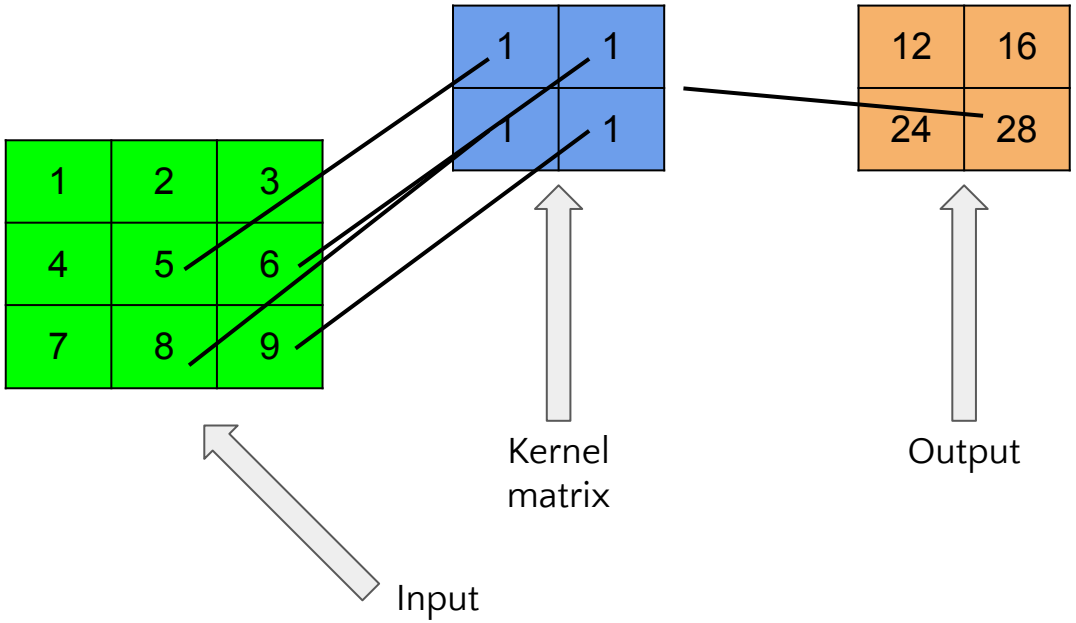
Convolution – 2D Example

- In 2D, we use **matrices** for our **kernels**



Convolution – 2D Example

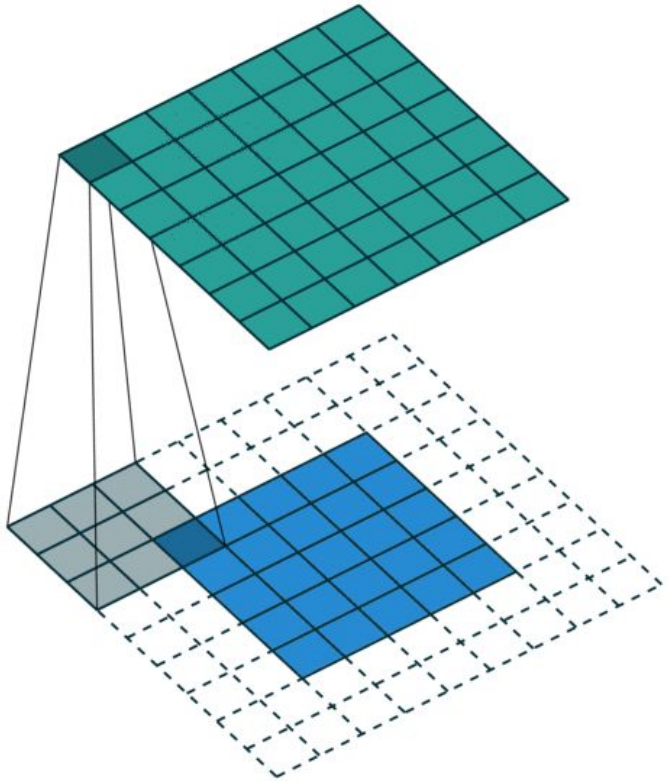
- In 2D, we use **matrices** for our **kernels**



2D Convolution with Padding

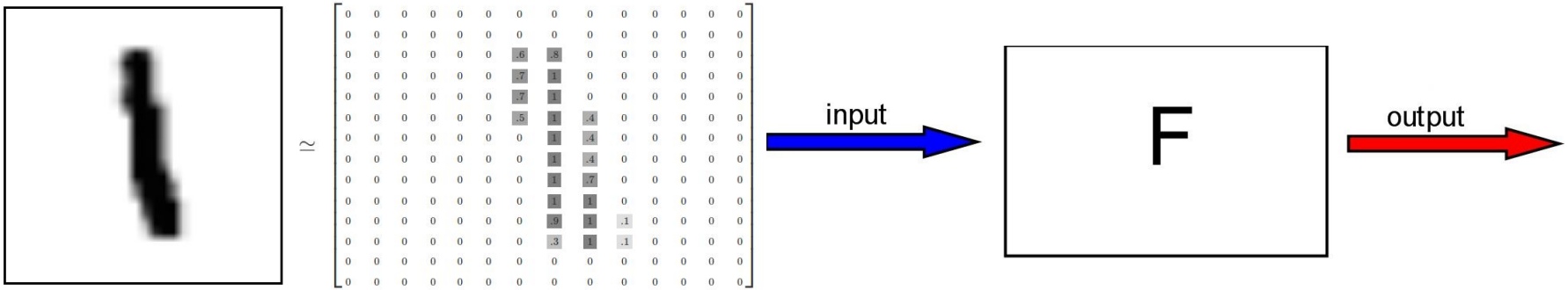
MIT GSL
GLOBAL STARTUP LABS

- In 2D, we use **matrices** for our **kernels**
- **Padding with zeros** ensures the output is the same size as the input
- Notice how we do the same sum of products between elements of the kernel and input as we saw before!



Filtering for Images

- Filtering primarily used in the image domain
- Can use filters for a variety of tasks, such as:
  - **Blurring**
  - **Sharpening**
  - **Edge detection**
  - **Corner Detection**



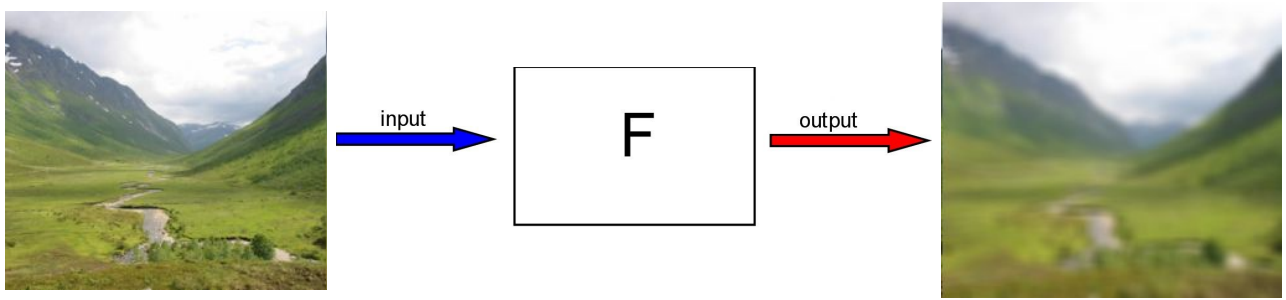
Filtering – Types of Kernels

- Blurring

$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

- Gaussian Blurring

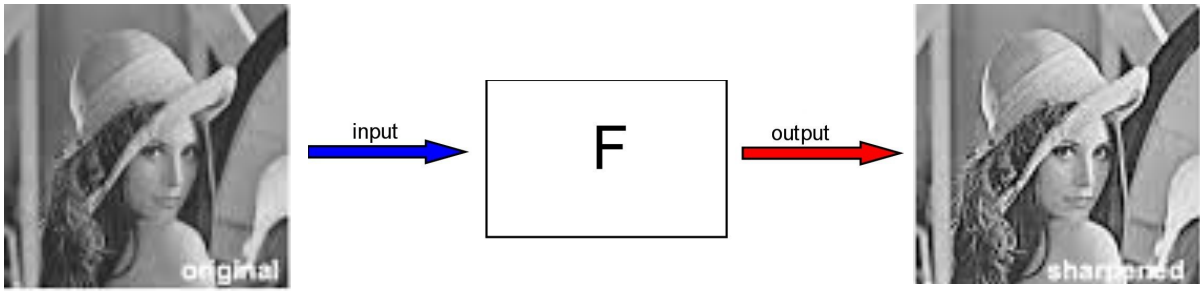
$$\frac{1}{209}\begin{bmatrix} 16 & 26 & 16 \\ 26 & 41 & 26 \\ 16 & 26 & 16 \end{bmatrix}$$



Filtering – Types of Kernels

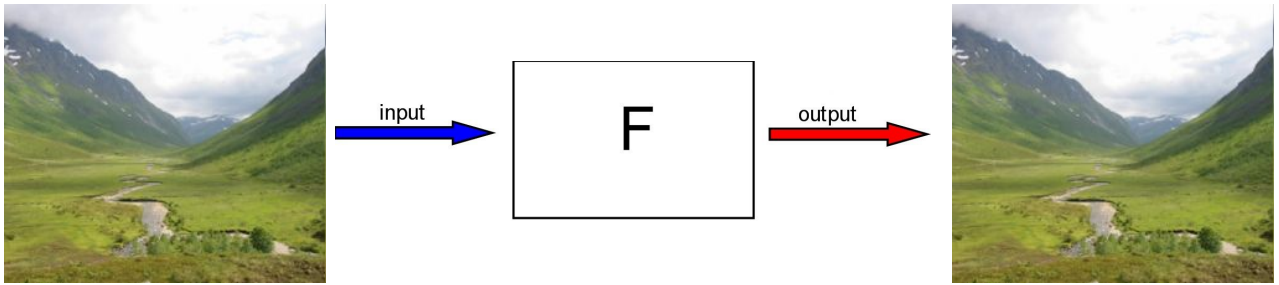
- Sharpening

-1	0	1
-2	0	2
-1	0	1



- Identity

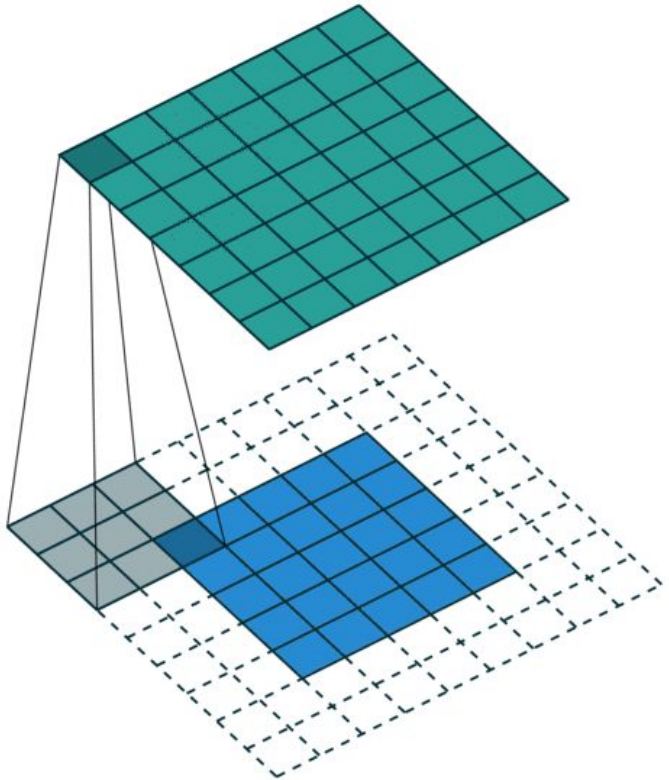
0	0	0
0	1	0
0	0	0



Computer Vision with Deep Learning – CNNs

MIT GSL
GLOBAL STARTUP LABS

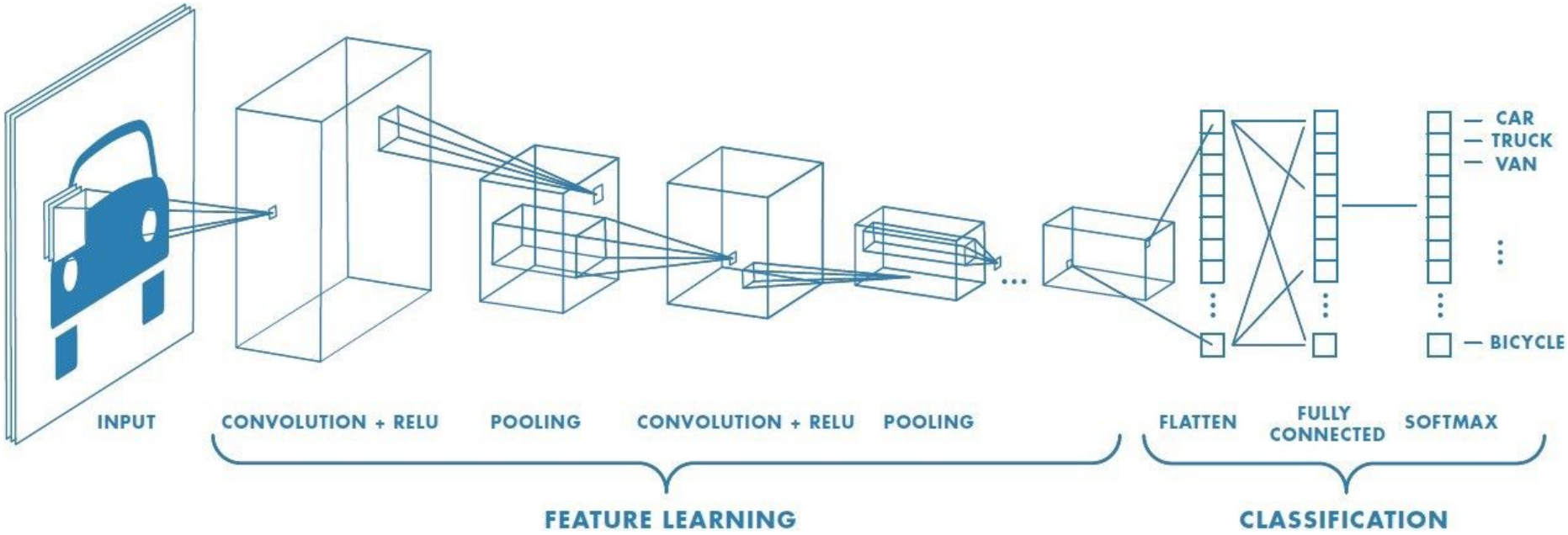
- Computer Vision connected to Deep Learning primarily through **Convolutional Neural Networks (CNNs)**.
- These networks use **convolutional layers**
  - Finds **local features** in an image
  - But can find features **anywhere** in an image (**translation invariance**)
- Convolutional layers use the same **convolution*** operation that we saw for filtering!



* Technically, this is **correlation**, not **convolution.** They are both closely related, with just a simple change of sign. If interested, see the link HERE.

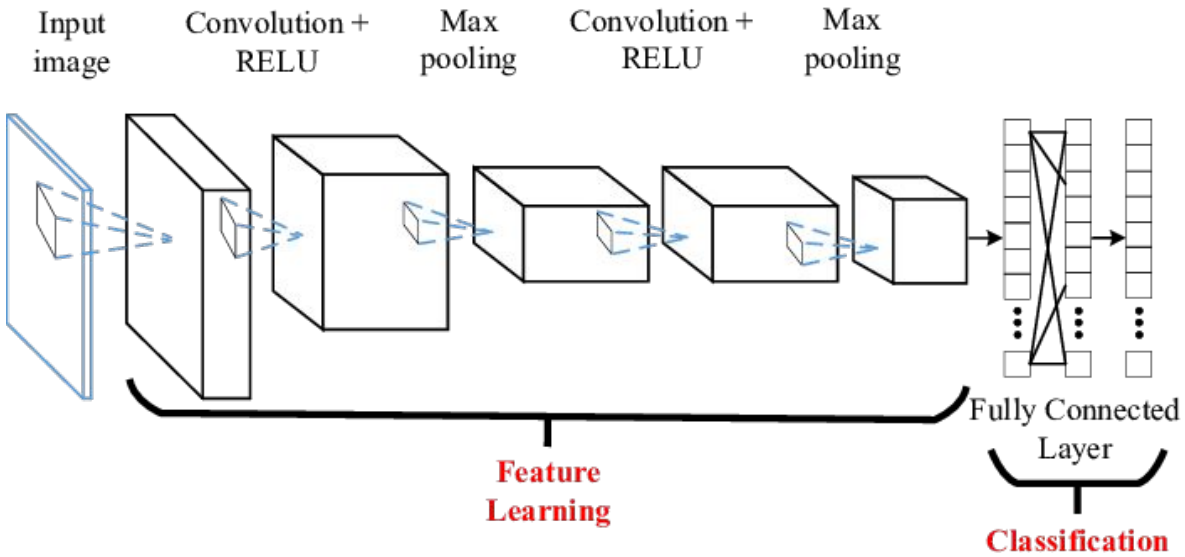
Anatomy of a Convolutional Neural Network (CNN)

MIT GSL
GLOBAL STARTUP LABS



Core Elements of CNNs

- Input “layer”
- Convolutional layers
- Max pooling layers
- Flattening layer*
- Fully Connected layers*
- Output layers

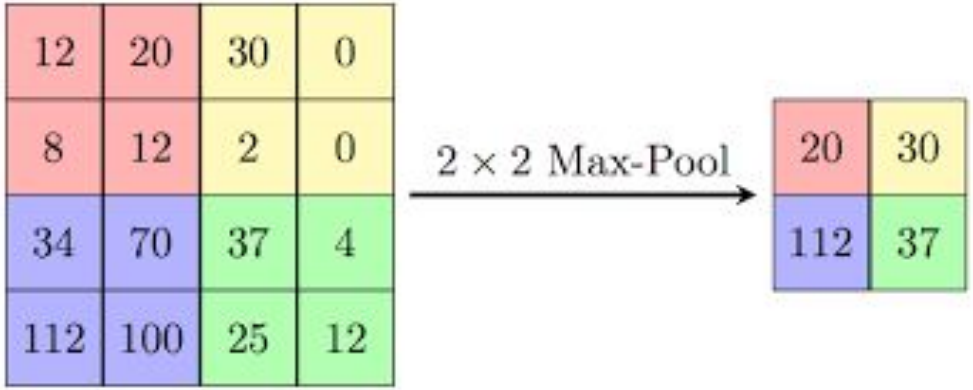


*Some neural networks, such as Fully Convolutional Neural Networks, don’t use these elements.

Introducing Non-Linearities in CNNs: Max Pooling

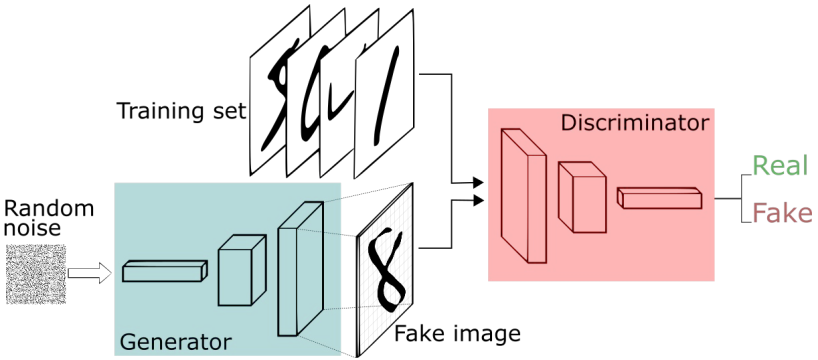
MIT GSL
GLOBAL STARTUP LABS

- Recall with feedforward networks that we primarily used **ReLU** as a means to introduce non-linearities into our neural networks.
- In CNNs, we will introduce non-linearity as well - through **max pooling** functions.



Tasks in CV + DL

- Recent advances in automated tasks have been achieved through recent advances in CV and DL:
  - Object detection
  - Semantic segmentation
  - Image classification
  - Image and video generation



Applications: Using CNNs for Vision in Healthcare

MIT GSL
GLOBAL STARTUP LABS

[1] Pneumonia Prediction:

This week, we'll be using Convolutional Neural Networks to predict whether a patient has pneumonia using chest x-ray imagery.

[2] Breast Cancer Prediction:

A hybrid Deep Learning and Logistic Regression model yielded substantial improvement in breast cancer diagnosis over only the Logistic Regression model. Paper HERE.

[3] Automated Detection of Diabetic Retinopathy and Diabetic Macular Edema:

Uses CNNs; named by JAMA as one of the best papers of the decade. Paper HERE.

MIT GSL
GLOBAL STARTUP LABS

ADDITIONAL SLIDES

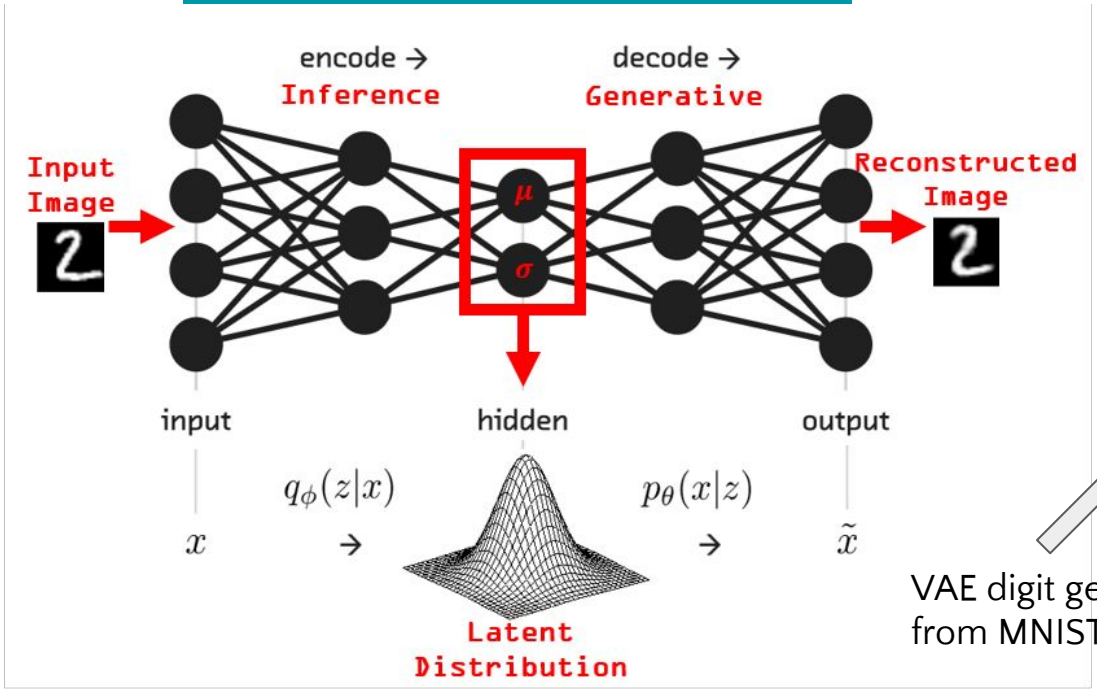
(for students' reference)

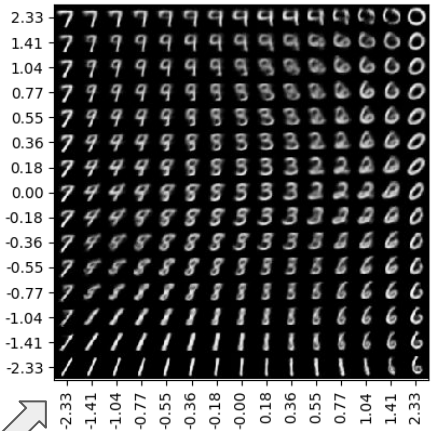
Variational Autoencoders (VAEs) – NNs for Unsupervised Learning

MIT GSL
GLOBAL STARTUP LABS

- Variants of neural networks that don't require **labeled data:**

Variational Autoencoders (VAEs)



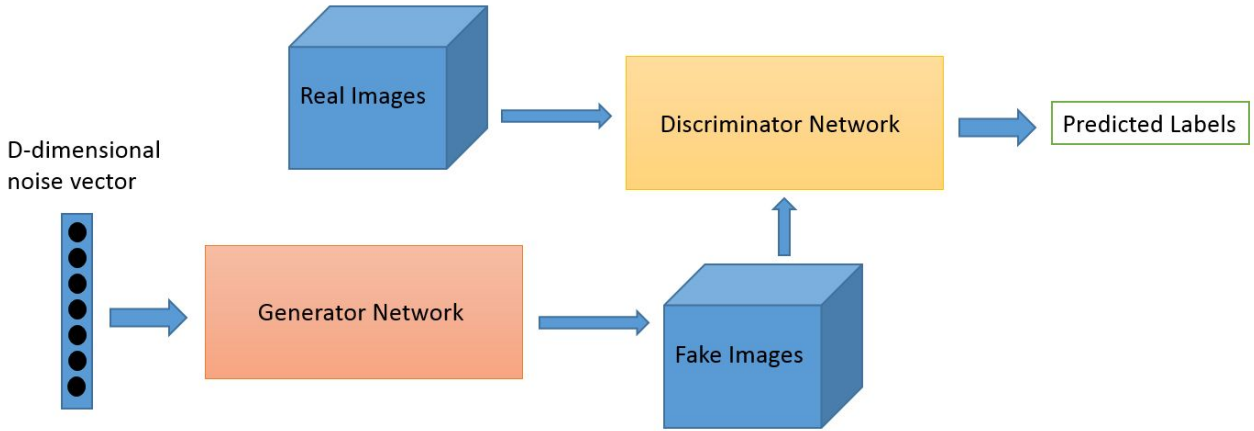


VAE digit generation from MNIST data!

Generative Adversarial Neural Networks (GANs)

MIT GSL
GLOBAL STARTUP LABS

- Composed of **generator** and **discriminator** networks.
- **Generator** is given random noise, and tries to output fake images to fool the discriminator.
- **Discriminator** is trained on real and fake images, and tries to learn how to spot fake images.
- Practical note: Proper training of these networks is quite difficult, and requires careful tuning.

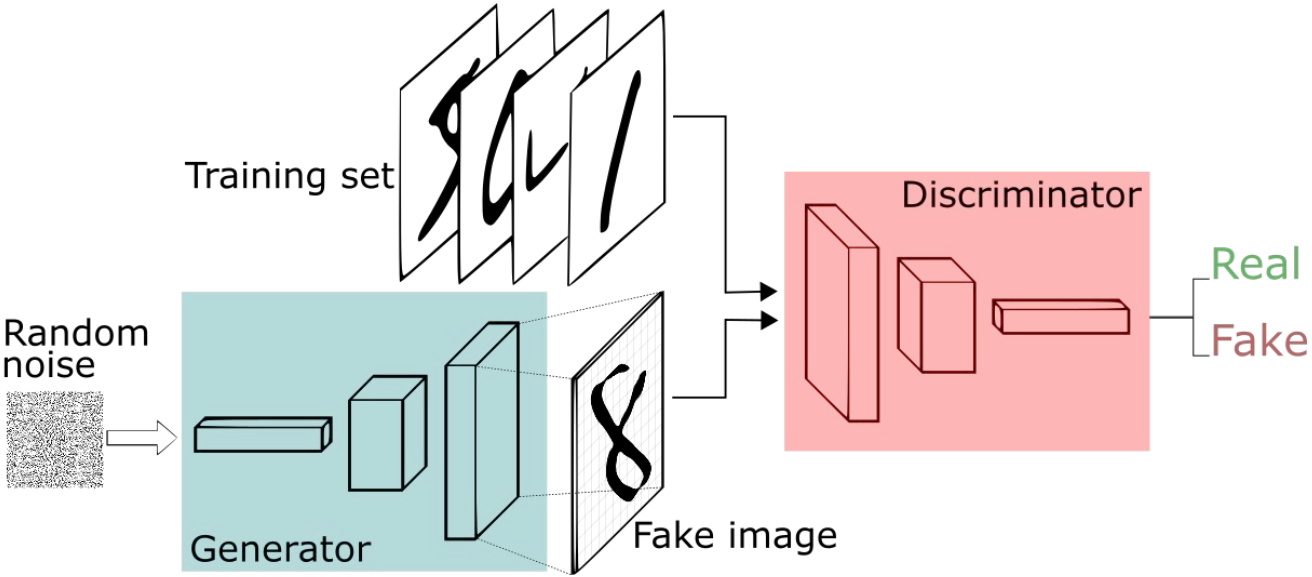


GANs + CV: Deep Convolutional GANs (DCGAN)

MIT GSL
GLOBAL STARTUP LABS

- Useful link for tutorial here: TensorFlow

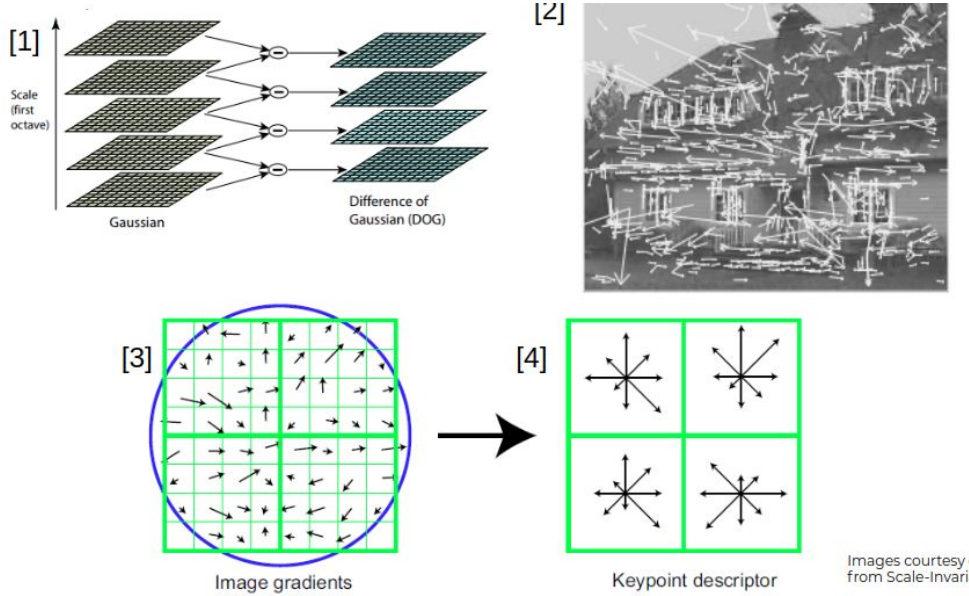
- This architecture can be used for real-world image generation!



Classical CV Object Detection: Scale-Invariant Feature Transform (SIFT)

MIT GSL
GLOBAL STARTUP LABS

- CV technique for object detection that doesn't use deep learning
- Original paper can be found HERE, code can be found HERE



SIFT (Scale-Invariant Feature Transform)

[1] Find extremes

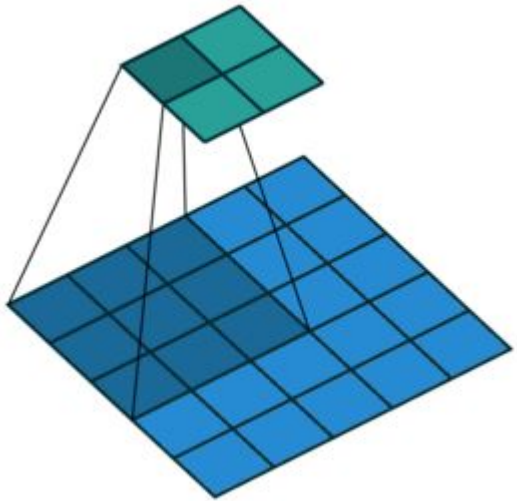
[2] Detect keypoints

[3] Assign orientations

[4] Assign keypoint descriptors

Images courtesy of David G. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints."

Template Matching: When Invariance Is Not Important



GIF courtesy of Towards Data Science

Template Matching

[1] Slide template over image using correlation or least square metric.

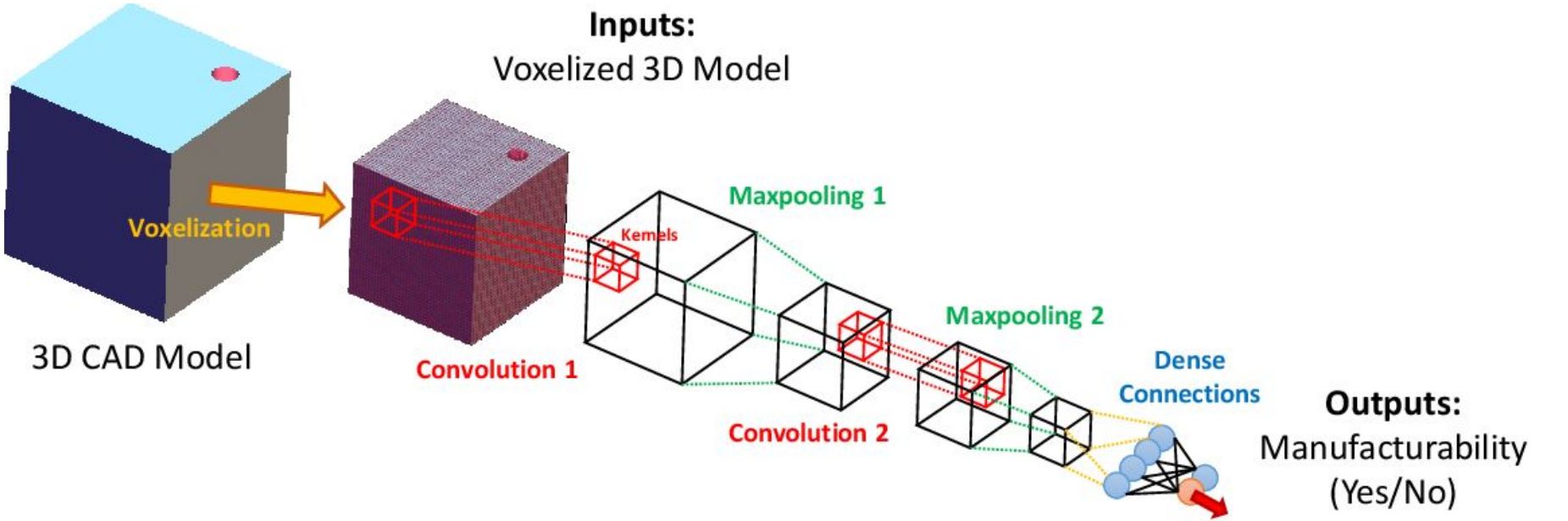
[2] Return peak by finding min or max point of metric.

[3] Draw bounding box using min/max point as top left corner.

CNNs in 3D: Volumetric CNNs

MIT GSL
GLOBAL STARTUP LABS

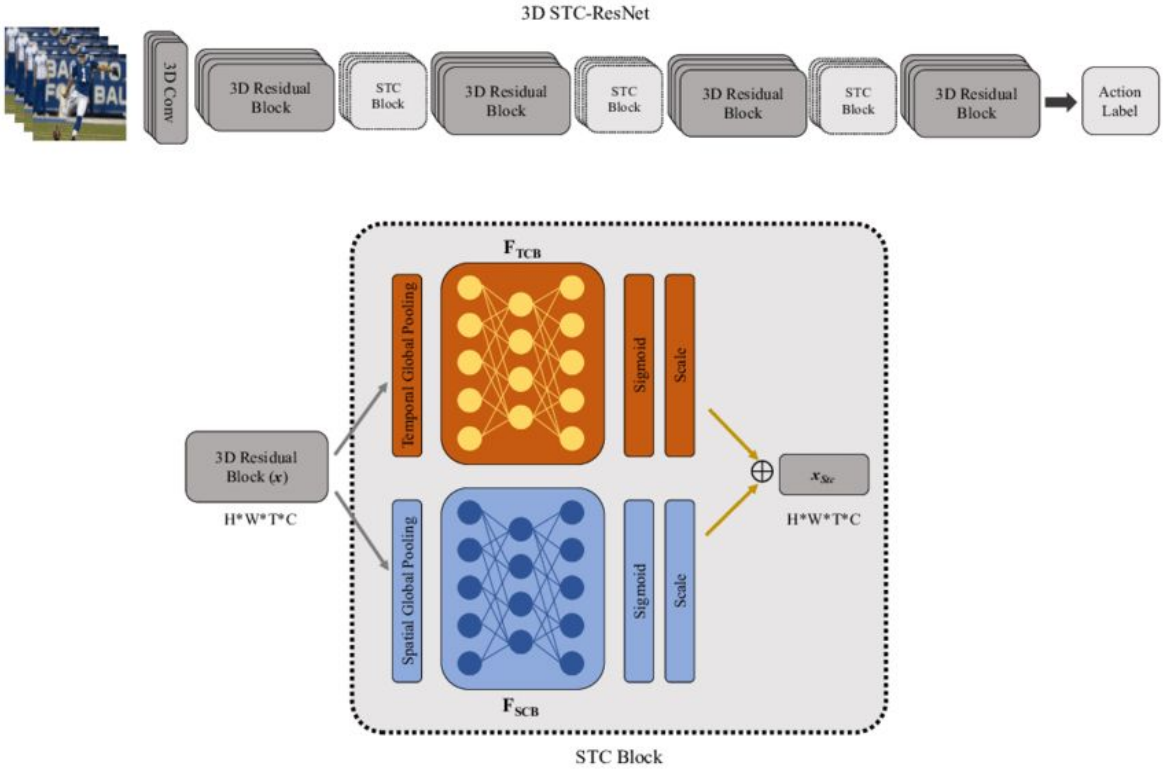
Link to GitHub repository HERE



CNNs over Time: Video Classification CNNs

MIT GSL
GLOBAL STARTUP LABS

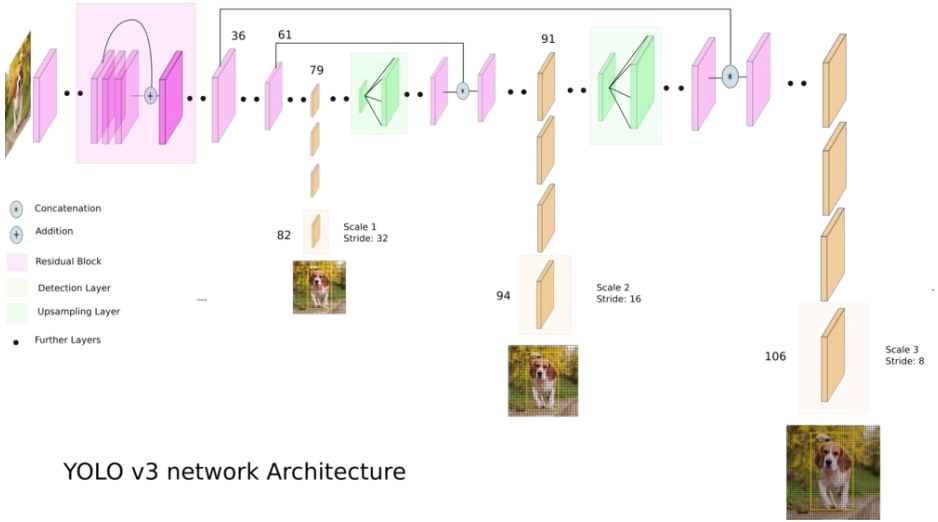
Link to GitHub repository HERE



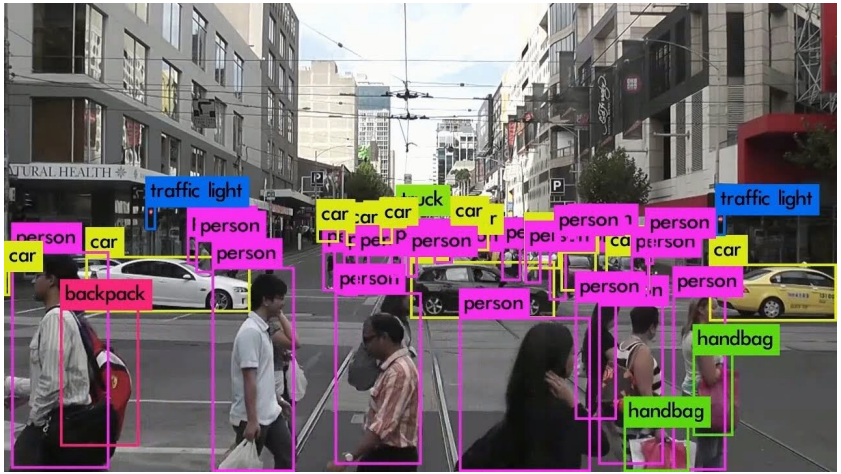
CNNs for Self-Driving Cars: YOLO-v3

MIT GSL
GLOBAL STARTUP LABS

Link to GitHub repository HERE



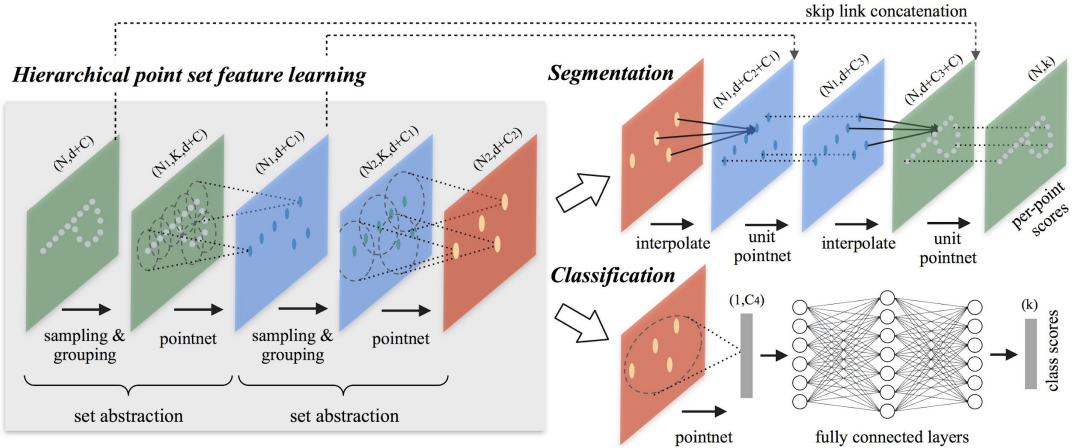
YOLO v3 network Architecture



CNNs for Self-Driving Cars: PointNet++

MIT GSL
GLOBAL STARTUP LABS

Link to GitHub repository HERE



(Some) Neural Network Architectures for Computer Vision

MIT GSL
GLOBAL STARTUP LABS

ResNet – Residual neural network for image classification

VGG-16/SegNet – Fully convolutional encoder-decoder networks (FCNNs)

Inception – Image classification

AlexNet – Original network for MNIST digit classification

YOLOv3 – Real-time object detection and classification

CycleGAN – Image translation using GANs

MeshCNN – 3D convolutional neural networks

PointNet++ – Lidar Point Cloud Object Segmentation

MIT GSL
GLOBAL STARTUP LABS

Review and Advice for Deep Learning and Computer Vision

MIT GSL-PRO
Ryan Sander

* Credits to Evan Shelhamer from Adobe for these slides (source)

There's a lot to remember about deep learning. Here are some tips we have for it.

Machine Learning is Great, But Does it Work For Every Application?

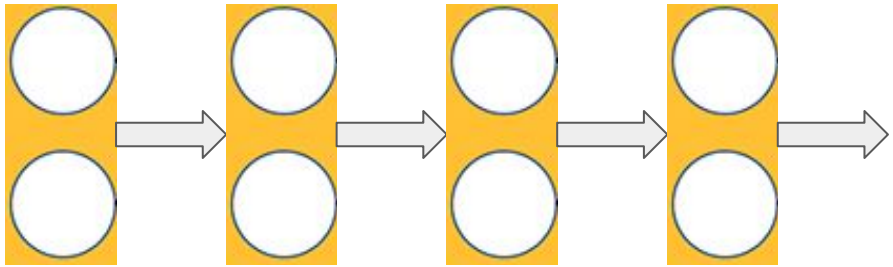
MIT GSL
GLOBAL STARTUP LABS

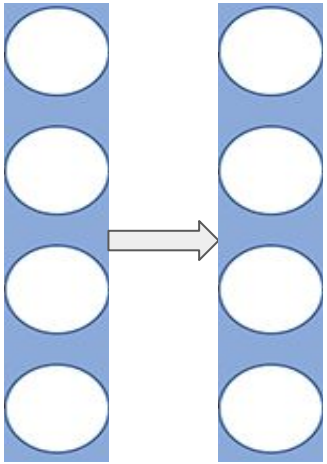
- Machine learning is not the solution to every problem involving data.
- Machine learning cannot solve every problem with data.
- Common modes of failure:
  - **Limited Information** – Not enough features/data
  - **Learning the wrong behavior** – Test data differs significantly from training data
  - **Task is too complicated** – (At least for now)
- Great video explaining this (in an entertaining way)

Deeper, Not Wider, Neural Networks*

MIT GSL
GLOBAL STARTUP LABS

more layers with **less weights** > **less layers** with **more weights**





*For More Information on this: Stack Overflow Post.

Practical Tips for Deep Learning – Datasets

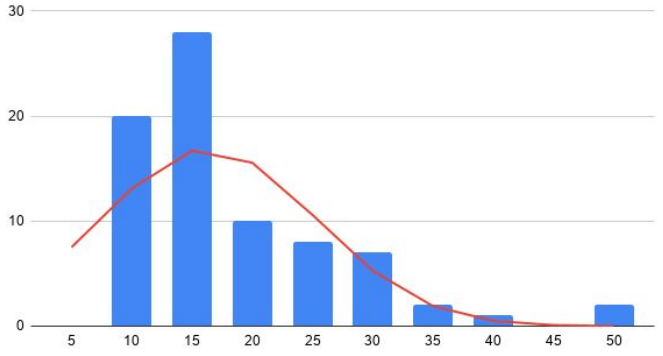
MIT GSL
GLOBAL STARTUP LABS

Always check your datasets

- Check for incomplete data
  - If you have enough observations, throw out incomplete examples
- Check for correct labels, if possible
  - You cannot train a neural net correctly with an incorrect dataset!
- Check the dimensions of your data

Check summary statistics

- Get histogram, mean, variance, etc. of **input features** and **target labels**

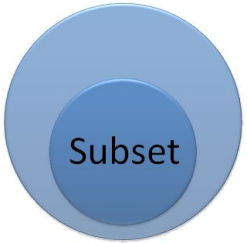


Practical Tips for Deep Learning – Datasets/Training

MIT GSL
GLOBAL STARTUP LABS

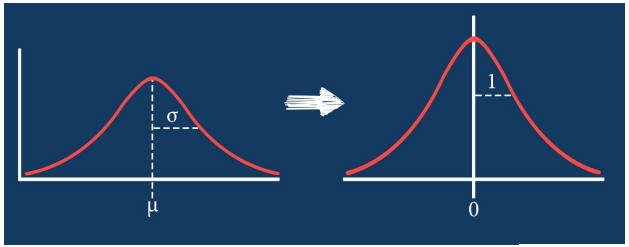
Start small

- Start by training on a **subset** of the data



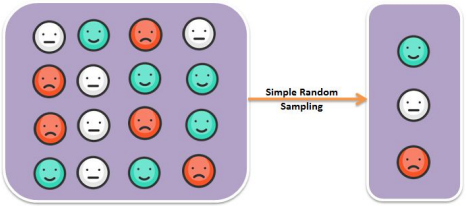
Standardize

- Normalize **inputs** and **targets**



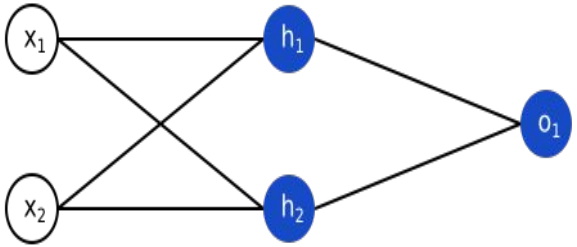
Decorrelate through random sampling

- Sample from the dataset **randomly** to avoid correlations



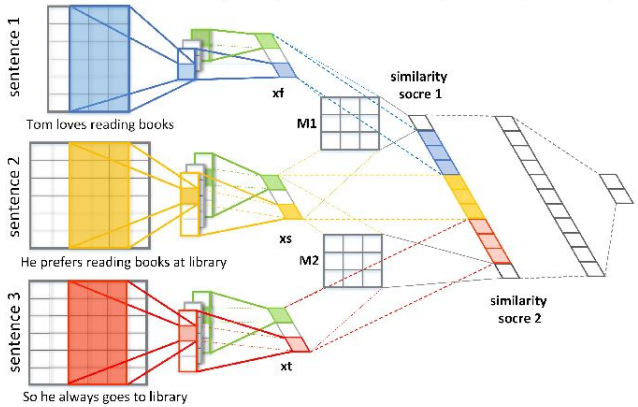
Practical Tips for Deep Learning – Models

Start with the **simplest model possible**, validate, then build complexity toward **state-of-the-art**



[1] Start with a simple model

[2] Slowly build complexity

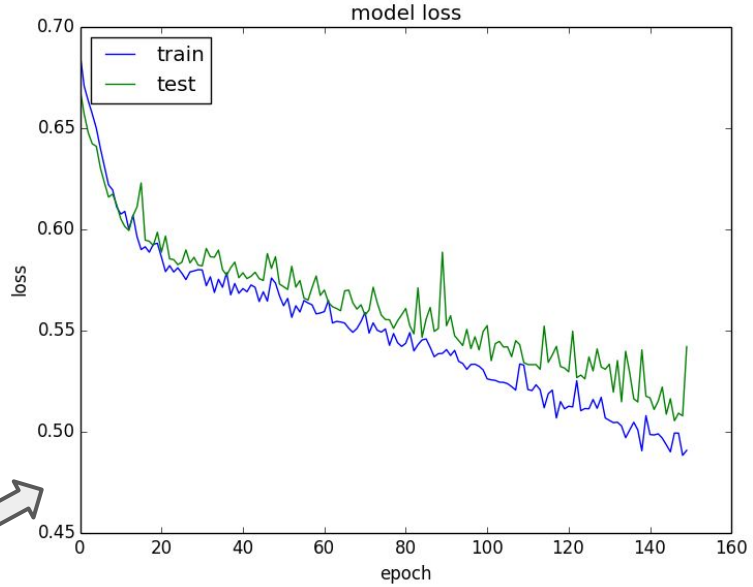


[3] State-of-the-art

Practical Tips for Deep Learning – Optimization

MIT GSL
GLOBAL STARTUP LABS

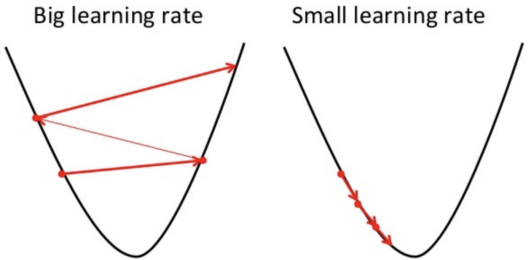
- We are still learning to optimize – no "silver bullet"
- Balance **exploration** with **exploitation**
- Check your losses:
  - If loss is constant, check for weights that have been initialized to zero
  - Check loss against suitable reference value
  - Make sure sign of loss makes sense

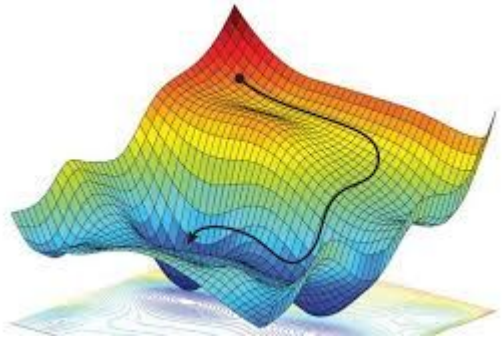


Goal: Have your loss vs. epochs look like this!

Practical Tips for Deep Learning – Optimization

- Most important hyperparameters: **batch size** and **learning rate**
  - If loss unstable, try **decreasing** learning rate or **increasing batch size**
  - Make **batch size** as **large** as possible **without** crashing GPU
  - Start with a constant learning rate
- Clear **gradients** – otherwise they may accumulate (PyTorch)
- Check for **sign** and **magnitude** of gradients
- Save checkpoints (your model)!



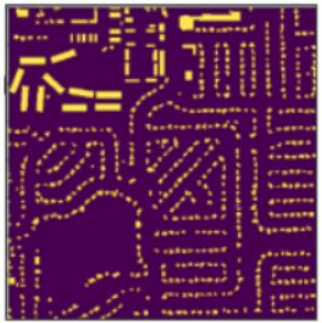


Practical Tips for Deep Learning – Evaluation

MIT GSL
GLOBAL STARTUP LABS

- Make sure you switch from **optimization** to **evaluation** (automatically done in Keras)
- Know what your output is; metrics do not tell the complete picture
- Test your metrics before training/evaluation
- Make sure to name your evaluation results! You don't want to misinterpret which models give which results

Predicted Building Footprint **(m,n,1)**



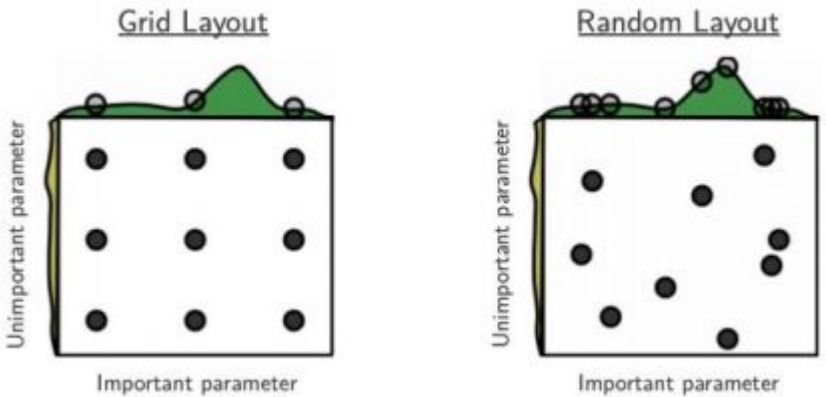
Ground Truth Building Footprint **(m,n,1)**

Practical Tips for Deep Learning – Parameter Tuning

MIT GSL
GLOBAL STARTUP LABS

- Change **one** parameter at a time → It's just the scientific method!

- Use **random search** rather than **grid search***



Instead of:

[0.1, 1, 10, 100]

Use:

np.random.random(min=0.1, max=100)

*Paper discussing this

Practical Tips for Deep Learning – Execution

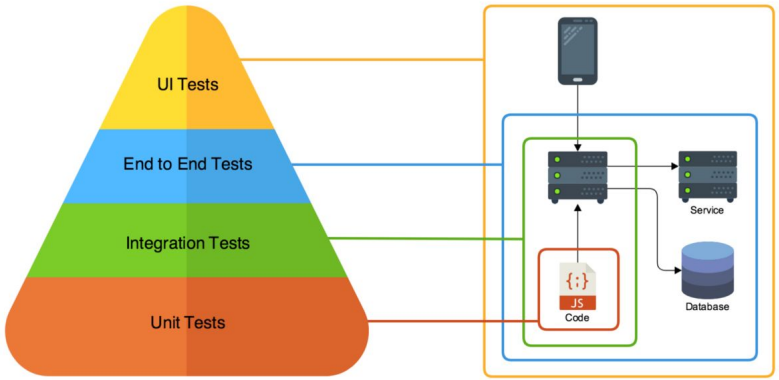
MIT GSL
GLOBAL STARTUP LABS

Create **log files** of everything – create a directory of these with appropriate names

- Example name: **lr_0.01_epochs_10_batch_32_opt_ADAM.h5**

Assume everything is **incorrect**

- Test to prove it is correct
- Bugs do not always give errors

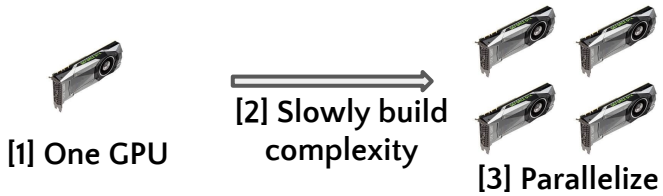


Practical Tips for Deep Learning – Execution

MIT GSL
GLOBAL STARTUP LABS

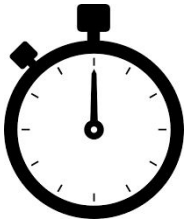
Multiple devices

- More hardware → more problems
- Start with a single device first, get a model working, then focus on **parallelization**



Check if more devices actually speeds things up

- Check if **iterations/time** actually improves with more devices



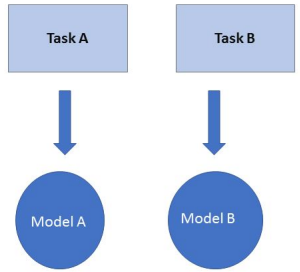
May (A Lack of) Data Never Stop You

MIT GSL
GLOBAL STARTUP LABS

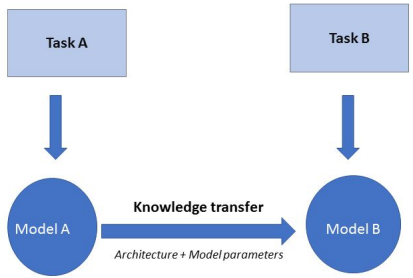
We can overcome limited data!

- Transfer/Few-shot Learning
- Many proxy datasets exist
- Organizations can (usually)

  support/sponsor the collection of data



Traditional Machine Learning



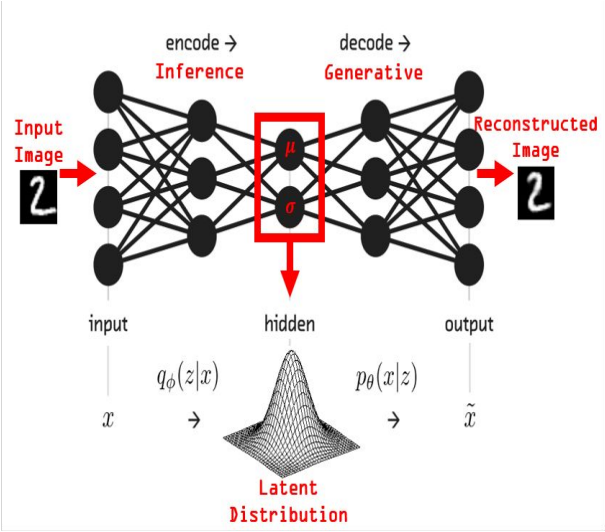
Transfer Learning

May (A Lack of) Labels Never Stop You

MIT GSL
GLOBAL STARTUP LABS

We can overcome limited labels!

- Encoders/Decoders can learn latent features in an unsupervised manner
- Many proxy datasets exist
- Transfer/Few-shot Learning
- Amazon Mechanical Turk can label your data for you



amazon mechanical turk

There's a lot to remember about computer vision. Here are some tips we have for it*.

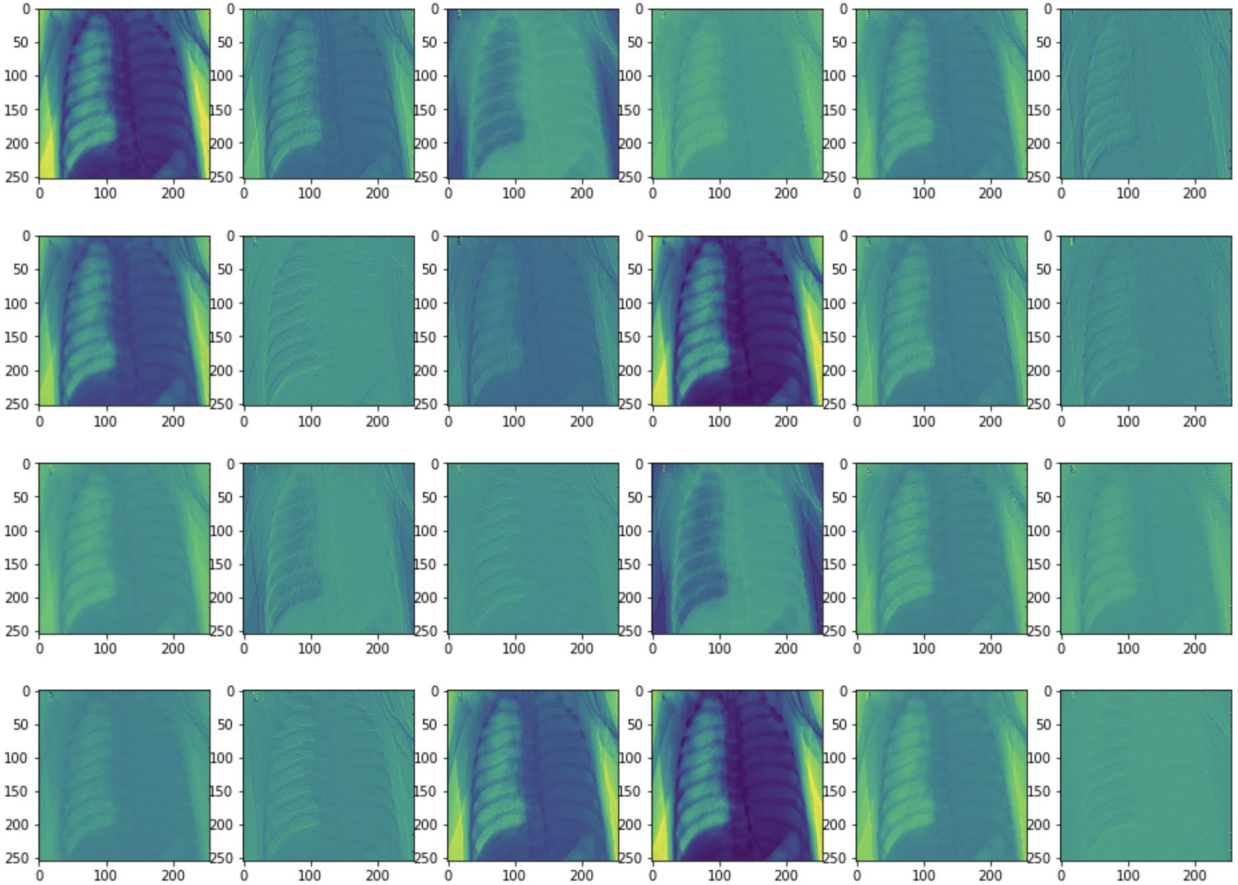
*__NOTE:__ A lot of the tips from above are also applicable here.

Practical Tips for Computer Vision Training

MIT GSL
GLOBAL STARTUP LABS

- GPUs are especially helpful
- Be wary of large batch sizes when training - they can flood GPUs!
- Image datasets are very large
- Data augmentation for images
  - Image rotations
  - Downsampled images
  - Random croppings

Visualize Your Activations/Filters!



Class Activation Maps – Another Great Way to Understand What Your Network Learns

MIT GSL
GLOBAL STARTUP LABS

Figure 20: Class Activation Map (CAM) for the "village" class. Notice how areas in the image corresponding to buildings yield the highest degree of activation.

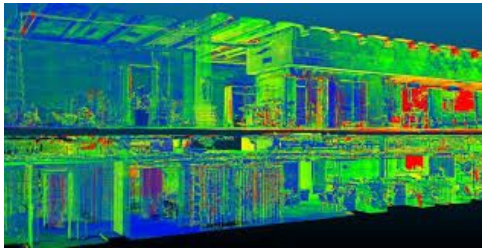
Figure 21: Class Activation Map (CAM) for the "islet" class. Notice how areas in the image corresponding to mountains and islands yield the highest degree of activation.

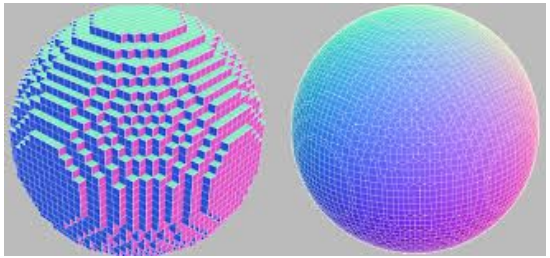
Vision Extends Well Beyond RGB and Grey-scale

MIT GSL
GLOBAL STARTUP LABS

3D Vision:

- Laser scans (Lidar, Radar)
- Voxels/meshgrids





Data Fusion:

- RGB/stereo
- Imagery + text
- Imagery + sound
- Imagery + time (video!)