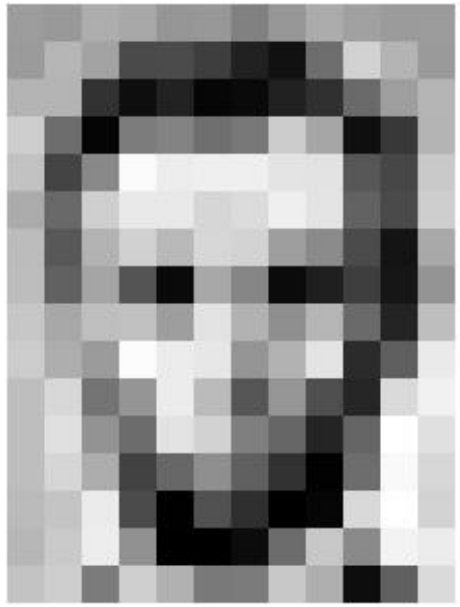
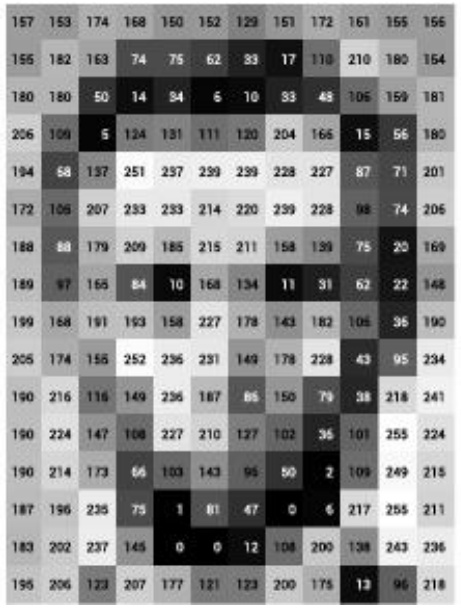
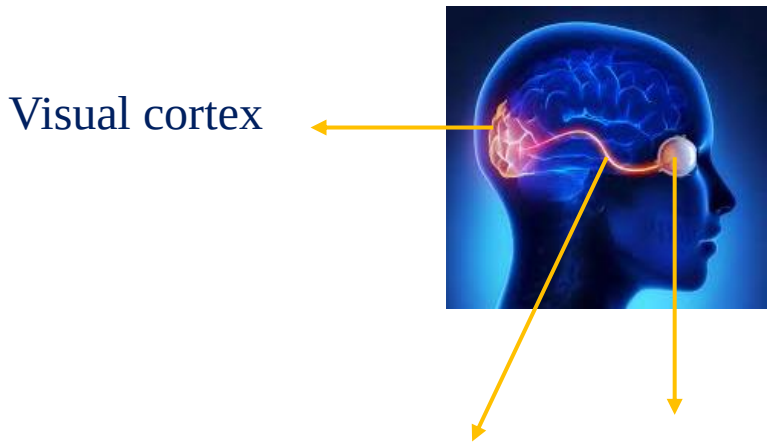
Module 5 – Part I
Deep Computer Vision
Basics





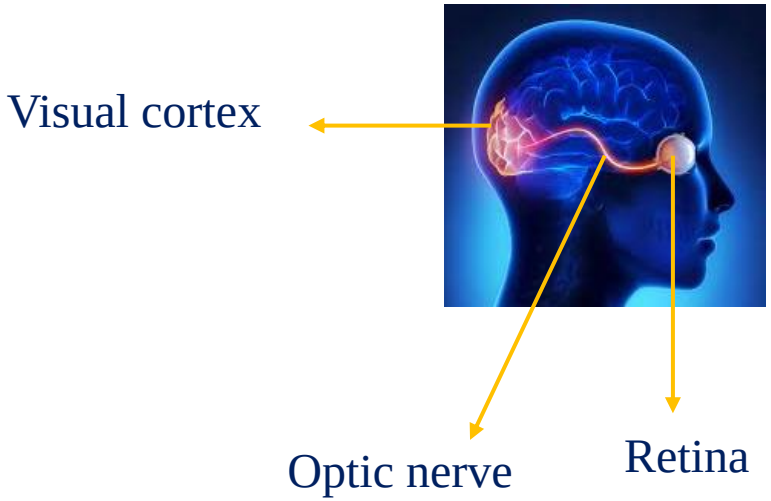


Road map!

- Module 1- Introduction to Deep Learning
- Module 2- Setting up Deep Learning Environment
- Module 3- Machine Learning review (ML fundamentals + models)
- Module 4- Deep Neural Networks (NN and DNN)
- Module 5- Deep Computer Vision (CNN, R-CNN, YOLO, FCN)
- Module 6- Deep Sequence Modeling (RNN, LSTM)
- Module 7- Transformers (Attention is all you need!)
- Module 8- Deep Generative Modeling (AE, VAE, GAN)
- Module 9- Deep Reinforcement Learning (DQN, PG)

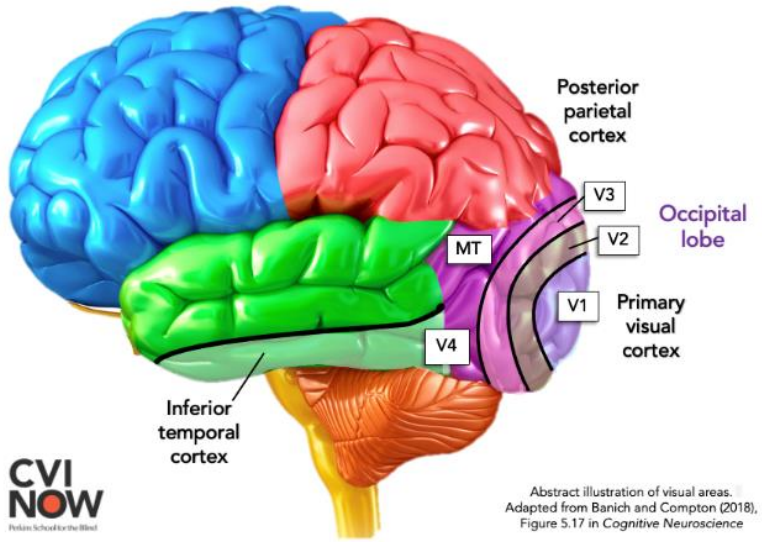
Human vision: How do we see?

- The visual cortex is the part of the brain that processes visual information from the eyes. When we see an image, light from the image enters the eye and is focused onto the retina.
- The retina converts the light into electrical signals, which are then transmitted through the optic nerve to the brain.
- Visual Cortex is made up of several different areas, each with a specific role in processing visual information.



Human vision: How do we see?

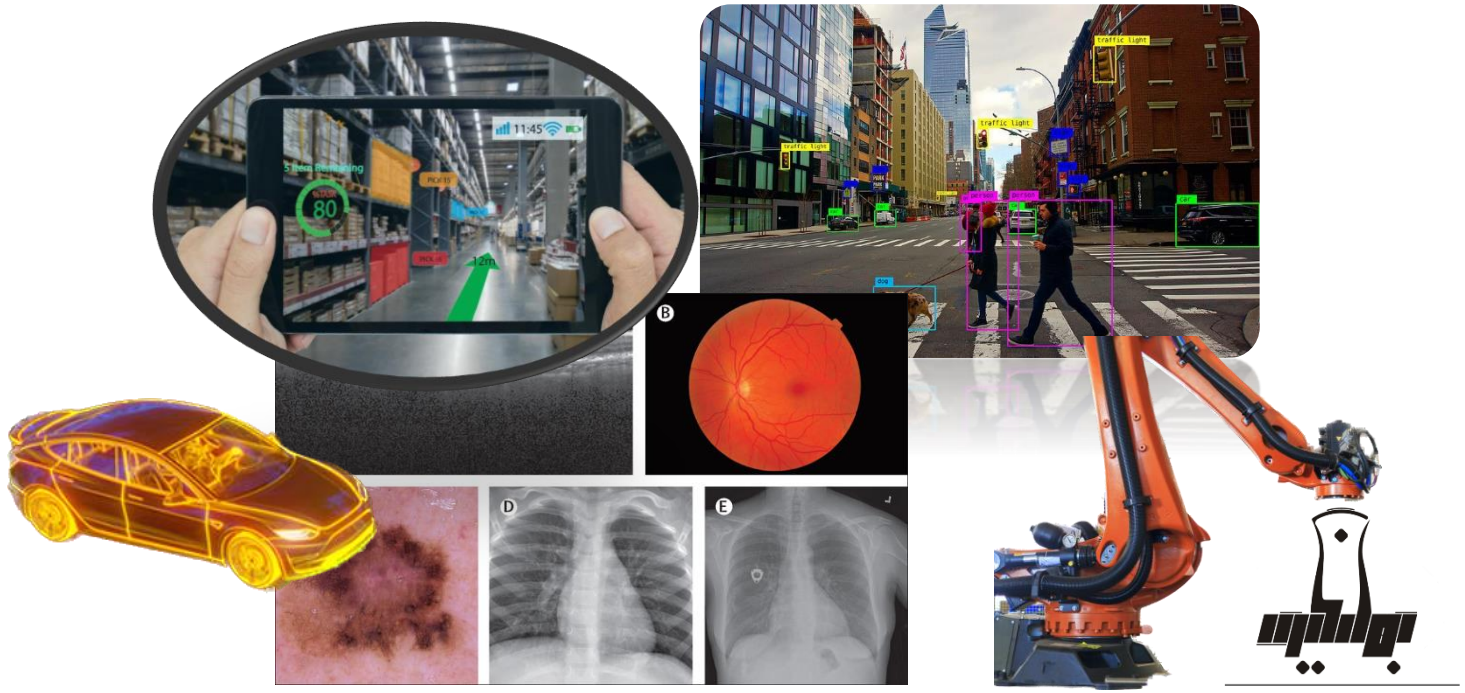
- V1 receives input from the retina and is responsible for early processing of visual information, such as edge detection and basic object recognition
- V2, receives input from V1 and is responsible for more advanced processing, such as color and texture processing.
- V3, receives input from V2 and is responsible for even more advanced processing, such as spatial relationships and depth perception.



Computer Vision: What is it?

- Computer vision involves the development of algorithms and systems, to enable computers to see, understand, and interpret the visual world.
- History: The study of computer vision can be traced back to the 1950s, but it was not until the development of digital cameras and the proliferation of cheap computing power in the 1990s that the field truly took off

- Applications:
  - Robotics
  - Medical imaging
  - Surveillance
  - Augmented reality
  - Image and video analysis

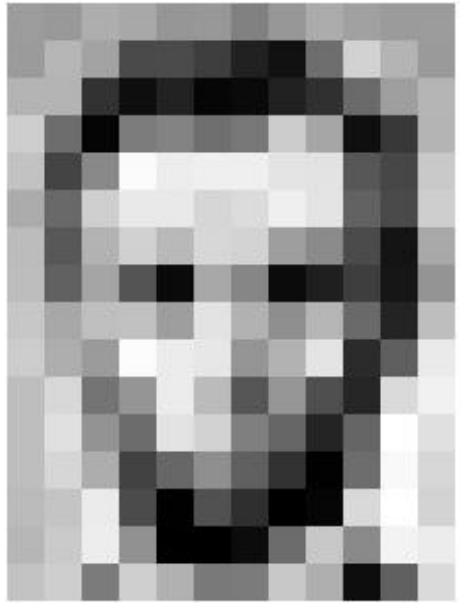


Jon M. Huntsman School of Business
UtahStateUniversity

Pedram Jahangiry

Computer Vision: What do they see?

- Computer converts the image into a digital representation!
- This digital representation consists of a grid of pixels, each of which has a numerical value that represents the color and intensity of the light at that point in the image.
- By analyzing these pixels, computers can recognize, detect, and categorize images.



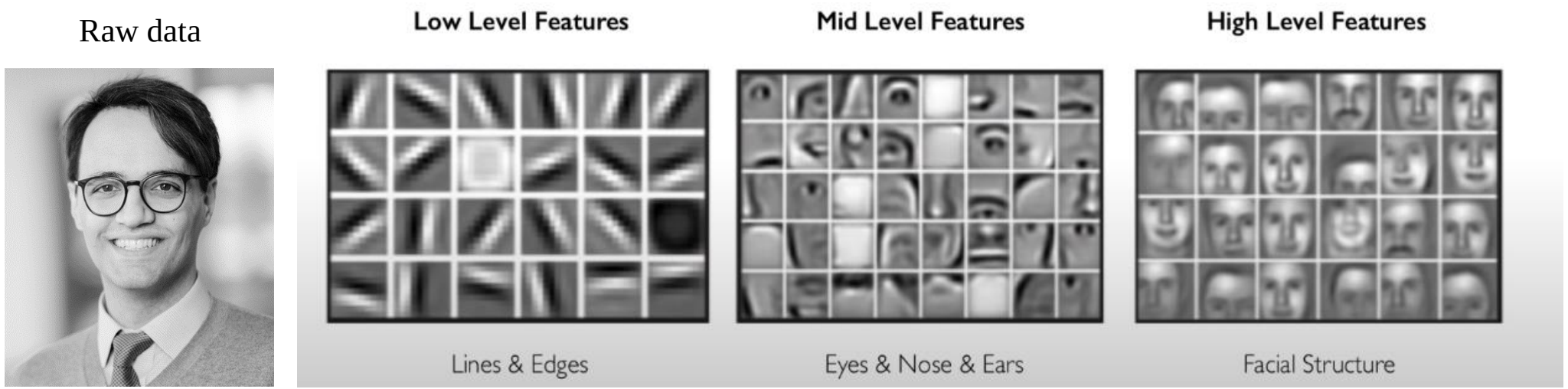
157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218

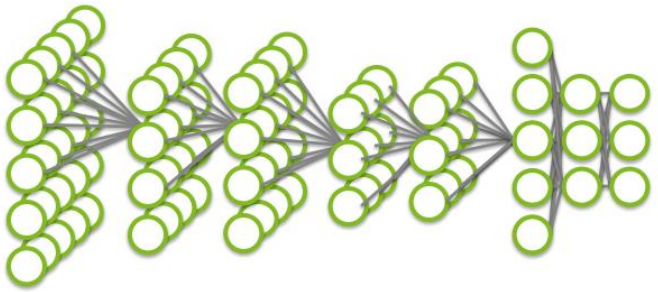
157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218

Photo by Nguyen Dang Hoang Nhu on Unsplash

Computer Vision: How do they see?

- Computers learn features in a hierarchical manner, like what happens in the visual cortex.
- No hand-engineered features!
- High-level feature detection.





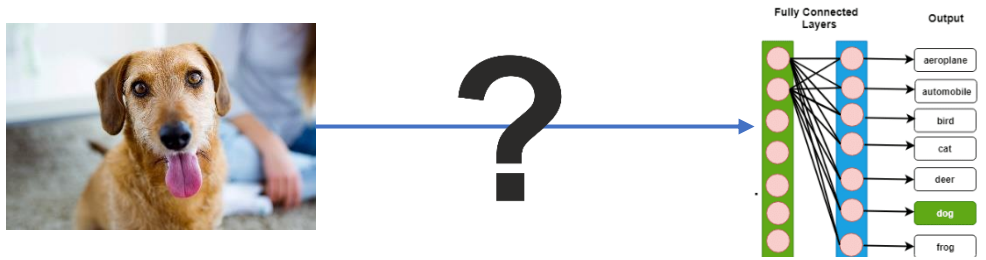
Fully connected networks?

- Could we use fully connected DNN to process image data?
- Theoretically yes, practically no! Why?

1. Lose spatial information
2. Many many parameters!!

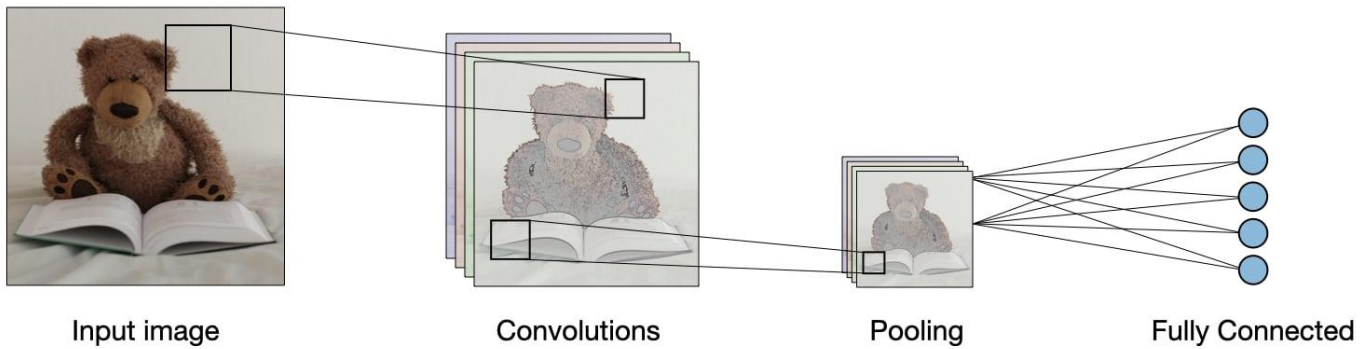
Introduction to Convolutional Neural Networks (CNNs)

- A new NN Architecture is required which can:
  - preserve spatial structure
  - requires fewer parameters! Each neuron should see a patch of pixels as input instead of the entire vectorized image! local feature extraction.
- Convolutional neural networks (CNNs) are artificial neural networks designed specifically for image recognition and processing. They are composed of multiple layers of nodes that each extract specific features from the input image.
- CNNs were first introduced in the 1980s, but only became widely used in the early 2010s, when large amounts of labeled data and powerful hardware became available.



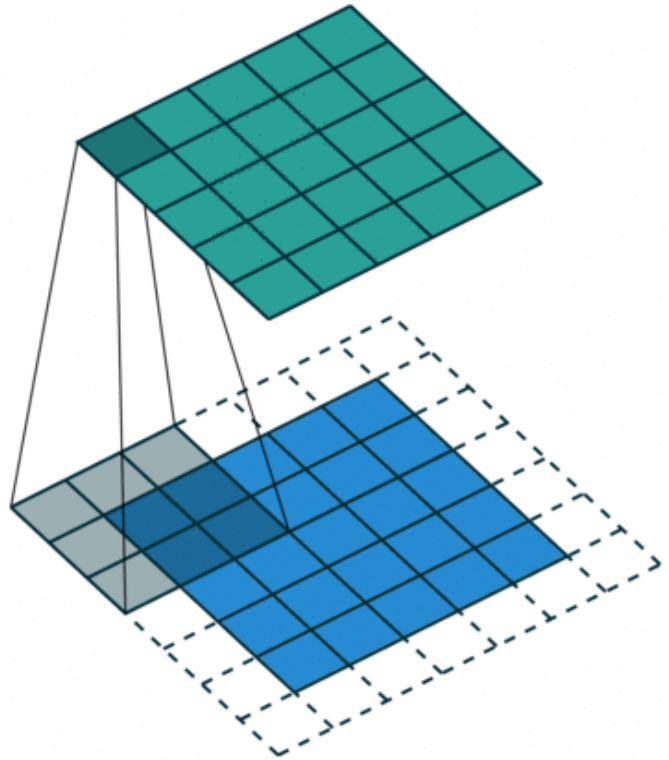
CNN Architecture

- A CNN typically consists of an input layer, one or more convolutional layers, one or more pooling layers, and one or more fully connected layers.
- The convolutional layers are responsible for extracting features from the input data using a set of learnable filters.
- The pooling layers are responsible for down-sampling the feature maps produced by the convolutional layers.
- The fully connected layers are responsible for classifying the features extracted by the CNN



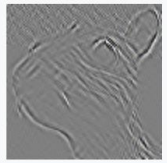
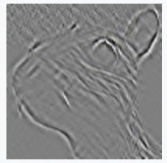
What is a Convolution?

- In the context of image processing and computer vision, convolution is often used to extract features from images.
- Convolution involves taking a small matrix of numbers, called a kernel or filter, and sliding it over the input image, performing an element-wise multiplication.
- The result of this multiplication is then summed up and forms the output of the convolution at that position. This process is repeated for every position of the kernel over the input image.
- The output of this process is a feature map, which is a modified version of the input image that has been transformed by the kernel.

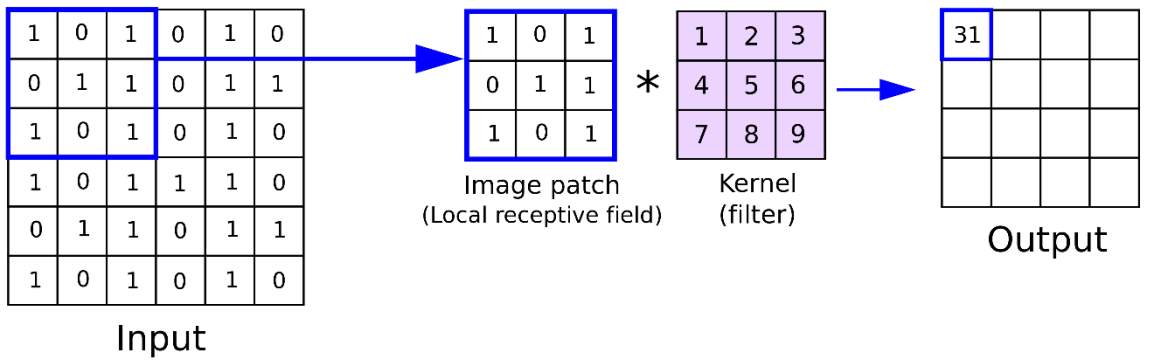


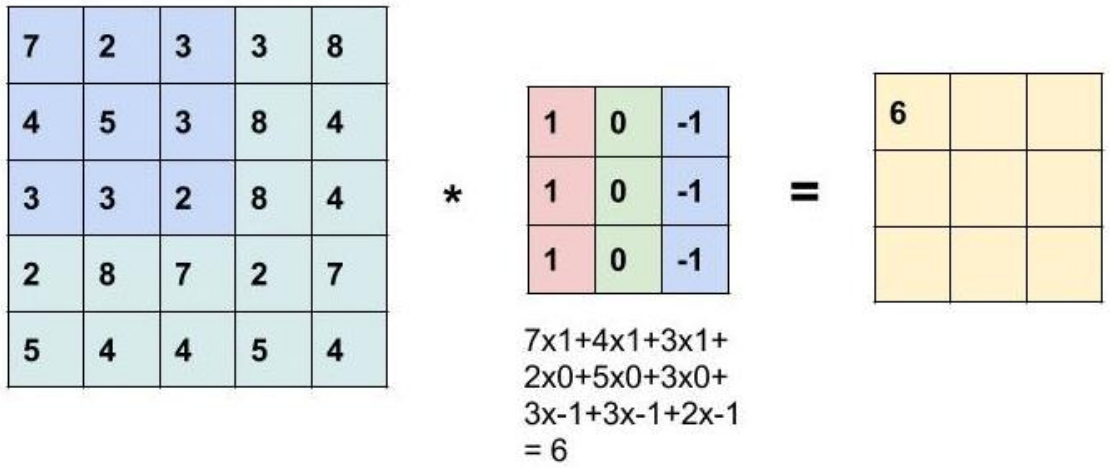
Filter/Kernel details

- A filter or kernel is a small matrix of weights that is used in CNNs to extract features from images, such as edges, corners, or patterns.
- Different filters can be used to detect different types of features, and multiple filters can be applied to the same input image to extract a variety of features.

Operation	Kernel ω	Image result g(x,y)
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Ridge or edge detection	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 4 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	

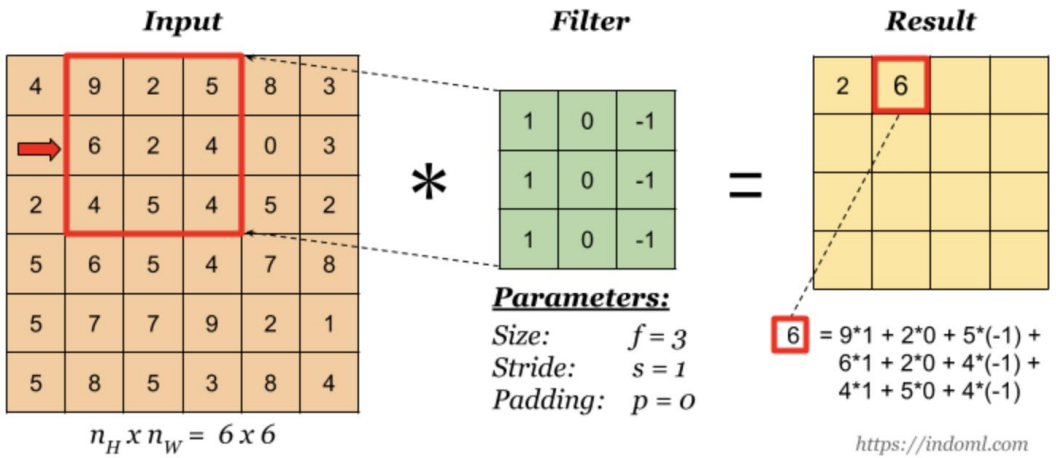
Convolution numerical example

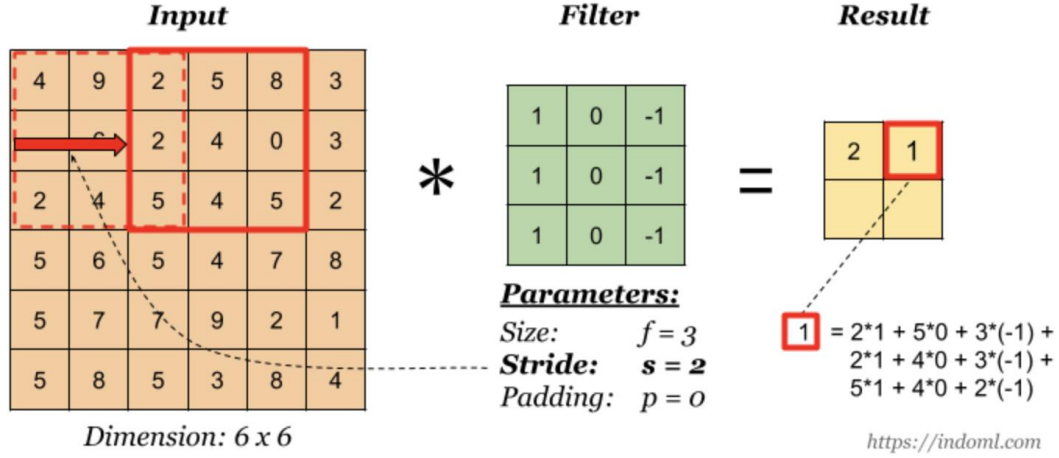




Filter Size, Stride, Padding

- The size, stride, and padding of a filter are three parameters that control the way in which the filter is applied to the input image during the convolution operation.
- Size: Refers to the dimensions of the kernel matrix. ex, 3*3
- Stride: Refers to the number of pixels that the kernel is moved at each step as it is slid over the input image. A larger stride results in a smaller output image.





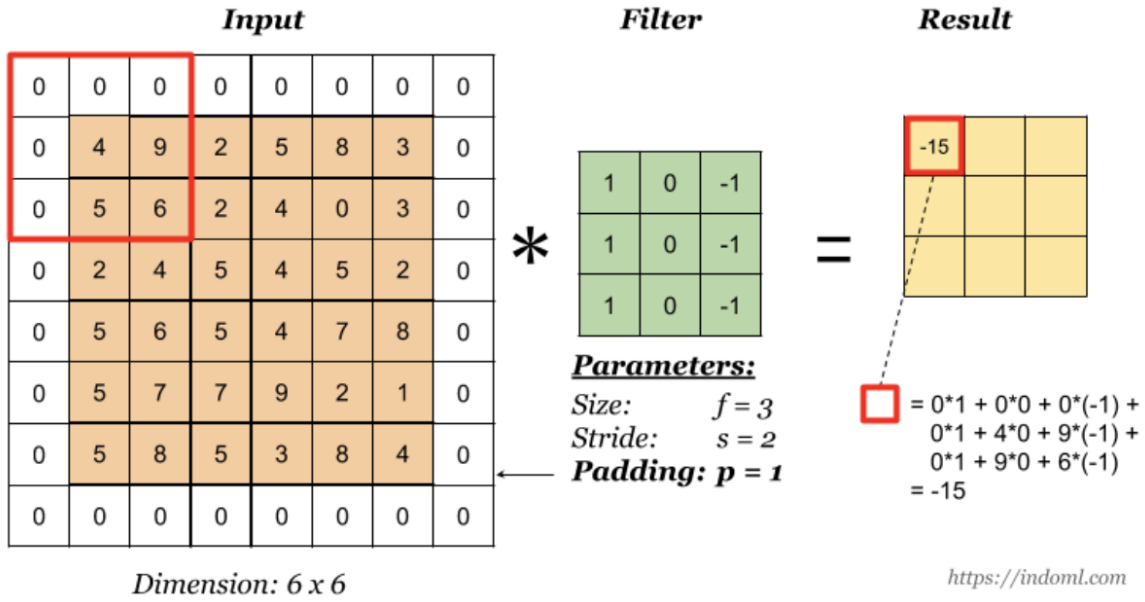
Jon M. Huntsman School of Business
UtahStateUniversity

Pedram Jahangiry

Filter Size, Stride, Padding

- Padding: refers to the addition of extra pixels around the border of the input image.
- Padding is often used in CNNs to preserve the spatial dimensions of the input image and to control the size of the output image.
- This is important for building deeper networks, since otherwise the height/width would shrink as we go to deeper layers.
- Padding helps us keep more of the information at the border of an image.

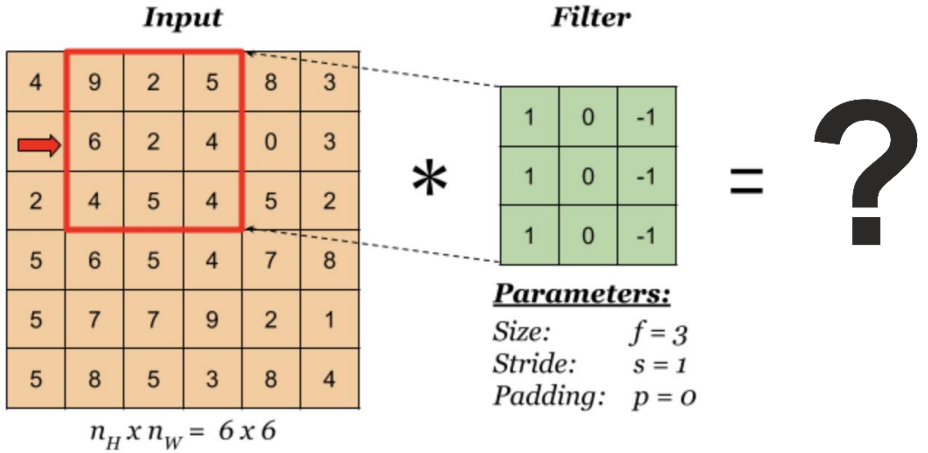
Padding terminology:
- Valid padding: no padding
- Same padding: Preserving the input dimension



Calculating the output dimension

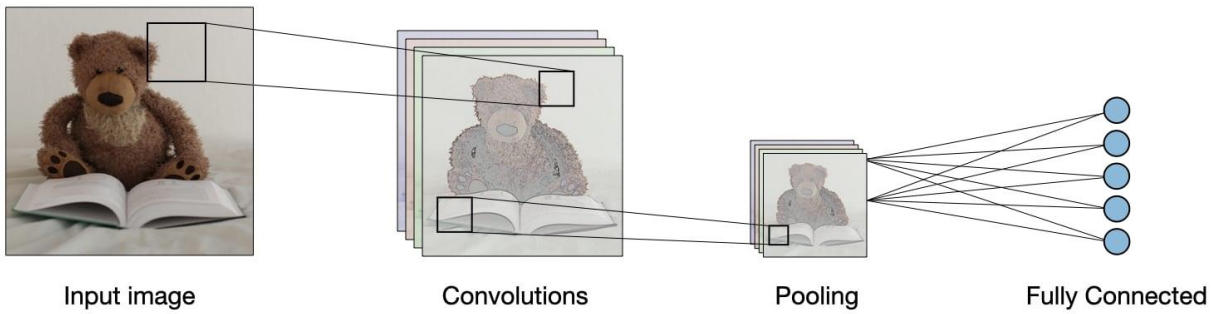
$$n^{[l]} = floor\left(\frac{n^{[l-1]} + 2p^{[l-1]} - f^{[l]}}{s^{[l]}} + 1\right)$$

The output dimension, is the floor of the above formula!

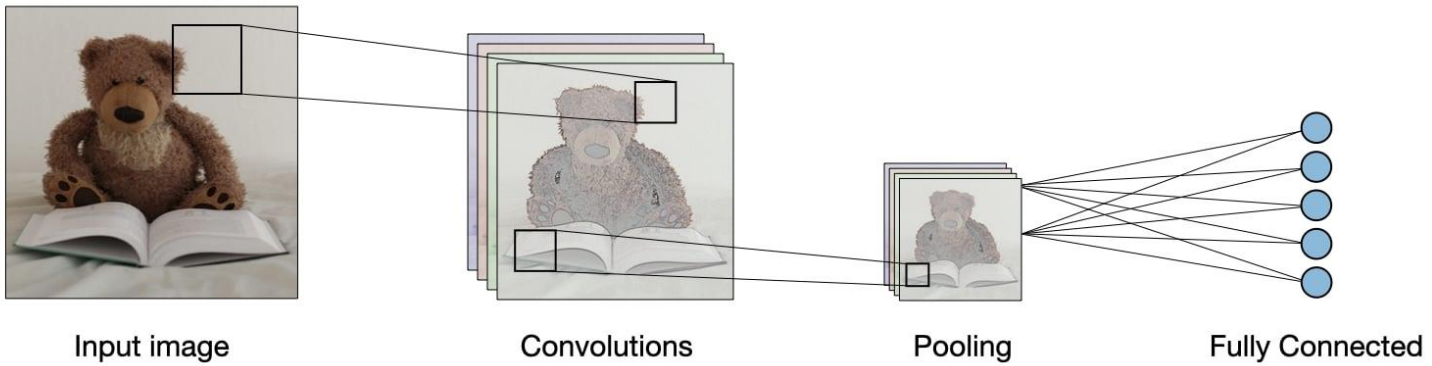


How CNN operates?

- During training, the CNN is presented with a set of labeled training examples. The CNN processes the input image through a series of layers, in which the filters are used to extract features from the input image.
- The filters are updated during the training process in order to learn features that are relevant for a particular task.
- The output of the CNN is then compared to the true label for the input image, and the error is calculated.
- This error is then backpropagated through the network, and the weights of the filters are adjusted in order to reduce the error. This process is repeated for each training example, and the filters are updated accordingly.



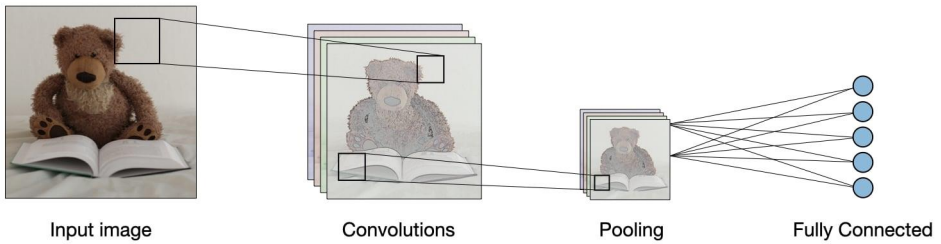
Module 5 – Part II
Deep Computer Vision
Convolutional Neural Network (CNN)



What is a Convolutional layer?

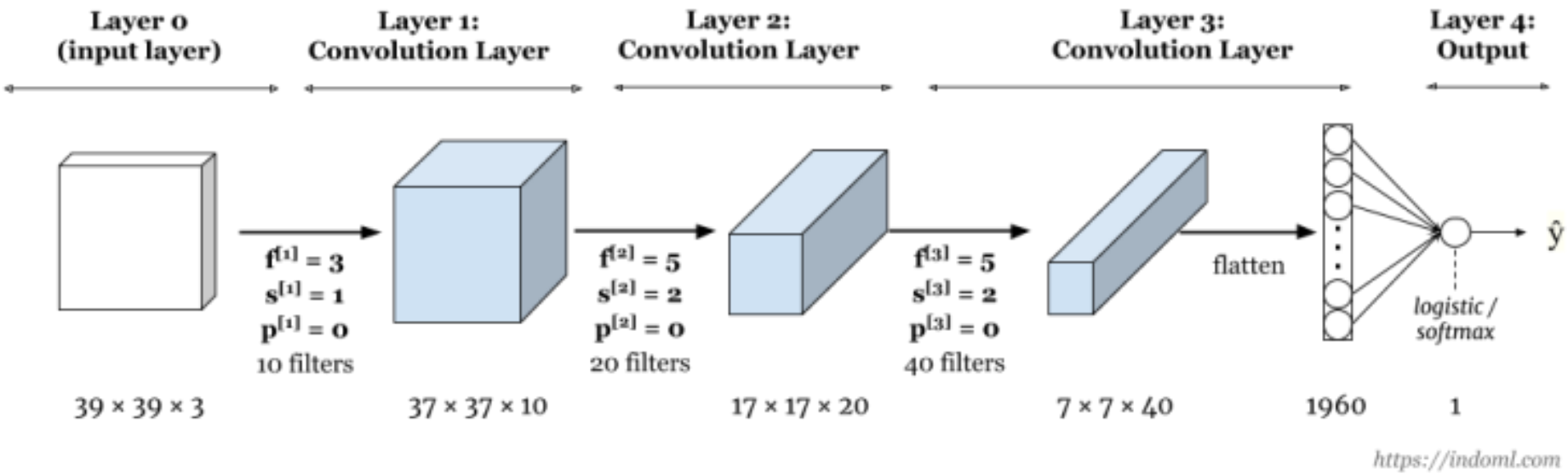
- A convolutional layer consists of a set of filters that are convolved with the input data to produce a set of feature maps.
- Each filter is responsible for extracting a specific feature from the input data, such as edges, corners, or textures.
- The filters are learned during the training process, allowing the CNN to automatically learn relevant features for a given task
- Why Convolutions?

1. Parameter sharing: a filter (such as edge detector) that's useful in one part of the image is probably useful in another part of the image.
2. Sparsity of connections: in each layer, each output value depends only on small number of inputs.



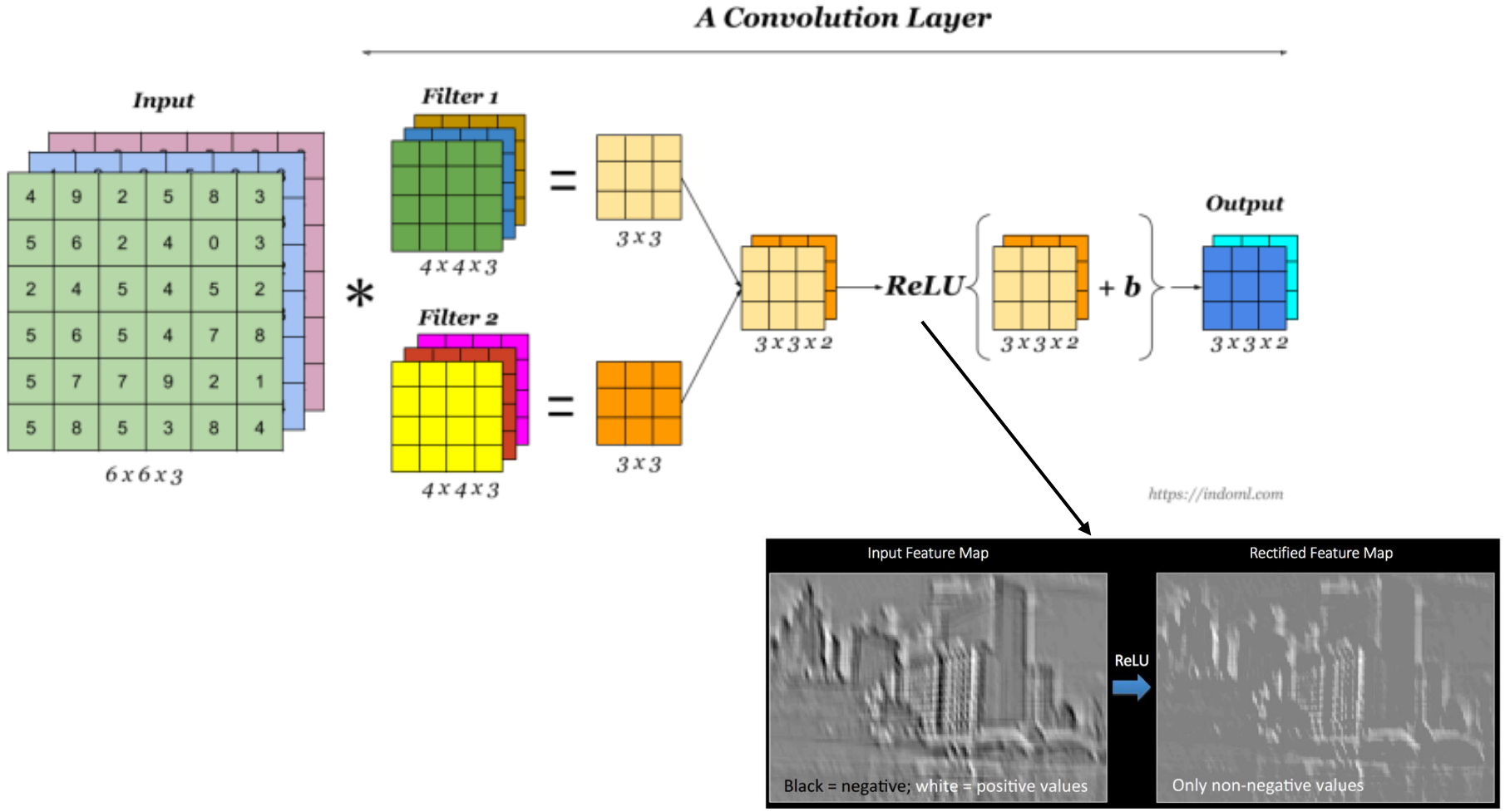
A more complete example!

- Stacking multiple convolution layers!



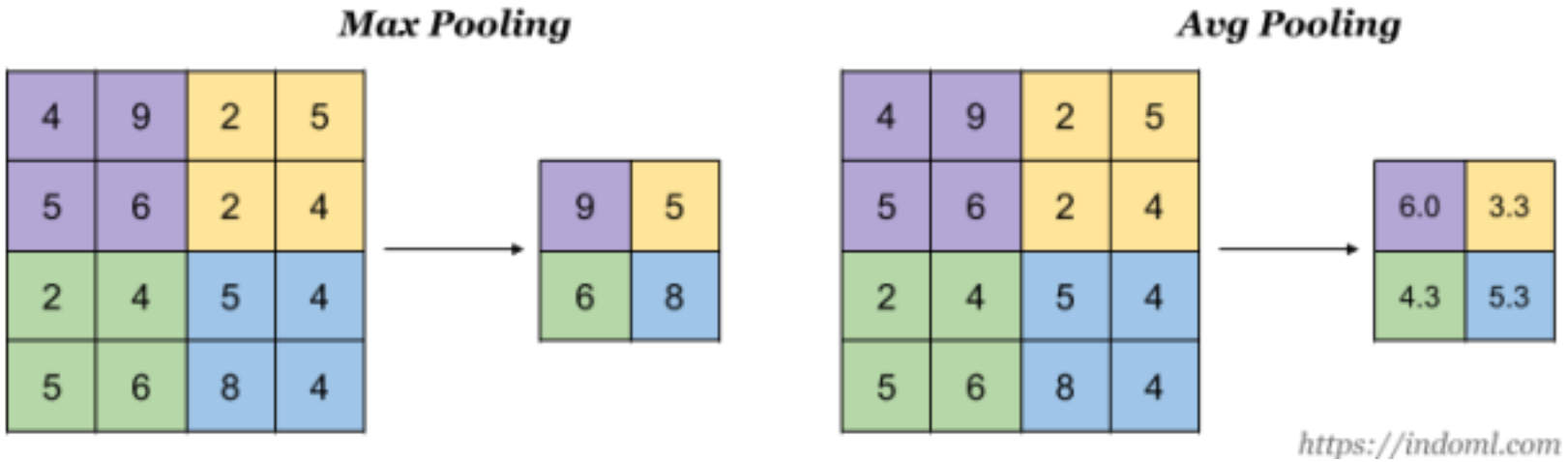
Parameters in a convolution layer!

- At the last step, a bias is added, and an activation function is applied.



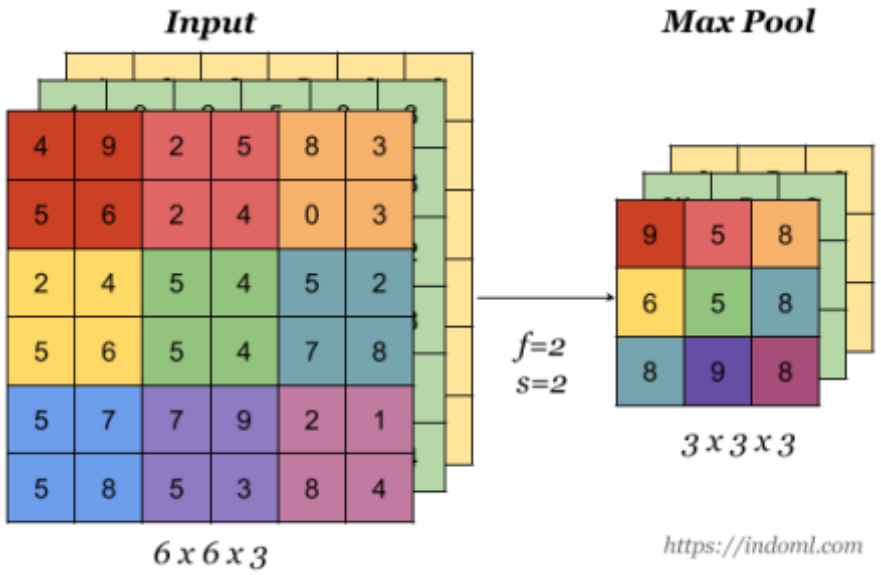
Pooling Layer

- Pooling layer is used to reduce the size of the feature map (generated by the convolutional layer) and to speed up calculations, as well as to make some of the features it detects a bit more robust.
- Pooling reduce the size and makes the model scalable while still preserving spatial structure.

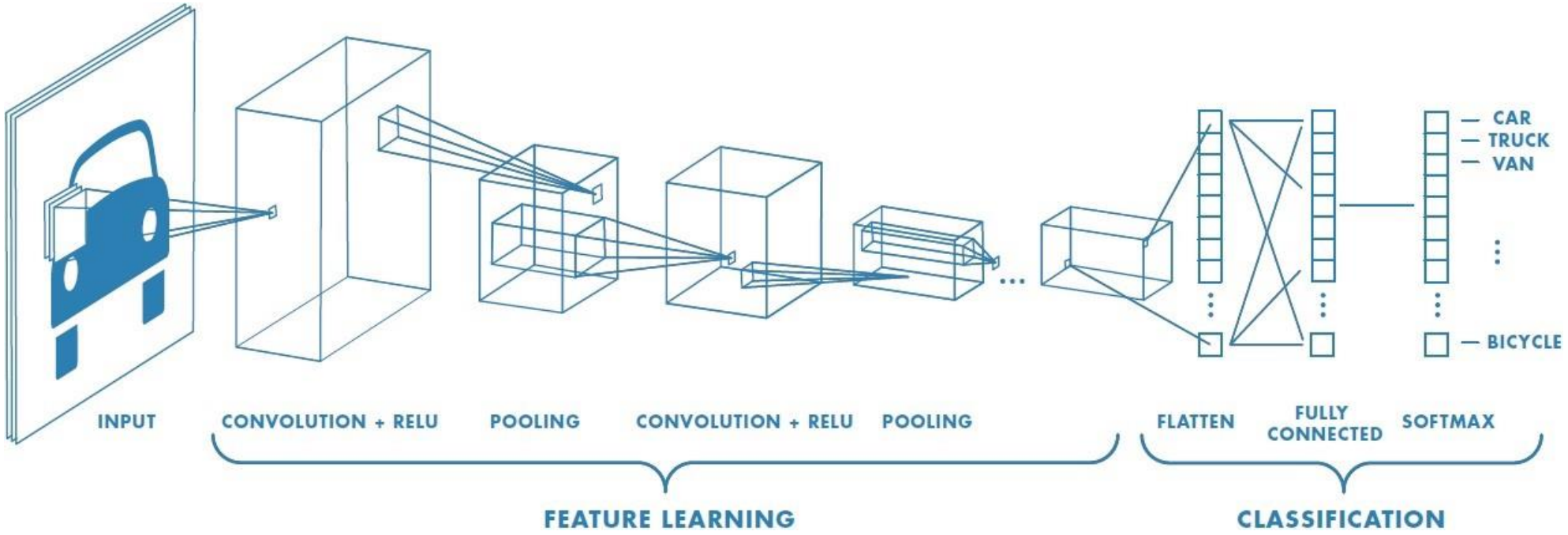


Pooling Layer Properties

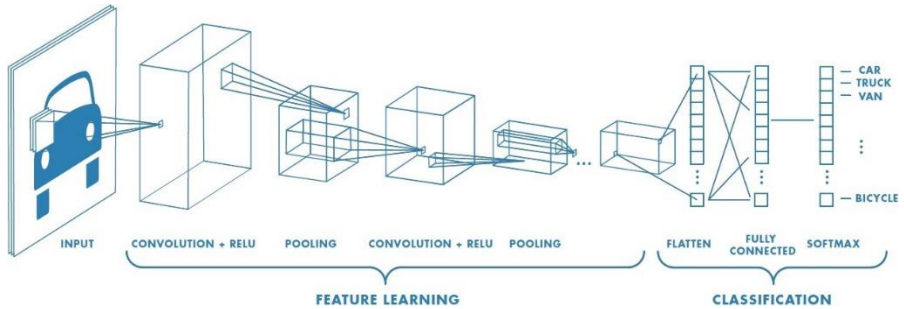
- The hyper parameters of pooling: size, stride, type
- There is no parameters to be learned.
- Unlike Convolutional operation, pooling does not change the number of channels.



Putting it together!



Putting it together!



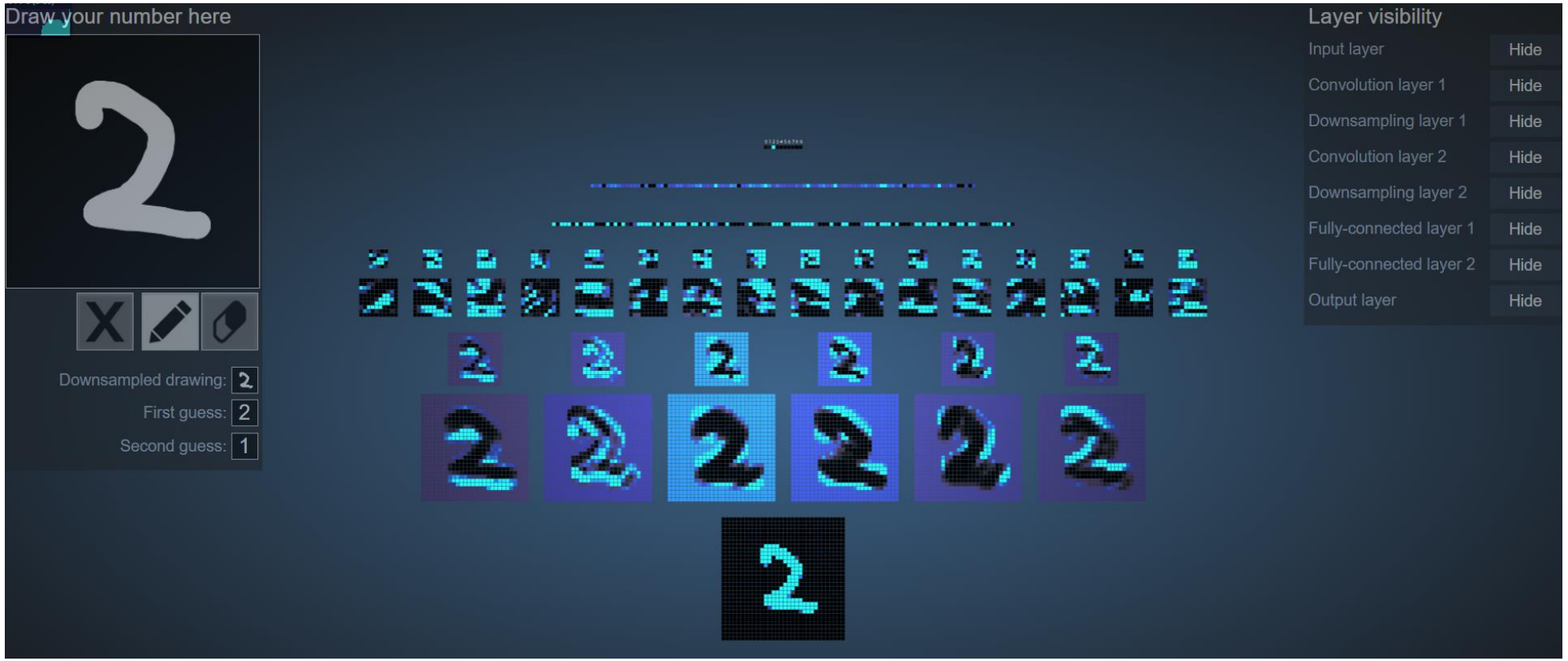

```python
def CNN_builder():
    inputs= keras.Input(shape=(28,28,1), name='Input layer')
    x = layers.Conv2D(filters= 32, kernel_size = 3, strides = (1,1) , padding='valid',  activation='relu' ,name="conv_layer_1")(inputs)
    x = layers.MaxPool2D(pool_size=2, name="pooling_1")(x)
    x = layers.Conv2D(filters= 64, kernel_size = 3, activation='relu', name="conv_layer_2")(x)
    x = layers.MaxPool2D(pool_size=2, name="pooling_2")(x)
    x = layers.Conv2D(filters= 128, kernel_size = 3, activation='relu', name="conv_layer_3")(x)
    x = layers.Flatten(name="flattening_layer")(x)
    x = layers.Dense(units= 64, activation='relu')(x)
    outputs = layers.Dense(units= 10, activation='softmax', name='output_layer')(x)

    model = keras.Model(inputs= inputs , outputs=outputs, name='my_first_CNN_model')

    model.compile(optimizer='rmsprop',
                  loss='sparse_categorical_crossentropy',
                  metrics=['accuracy'])

    return model
```

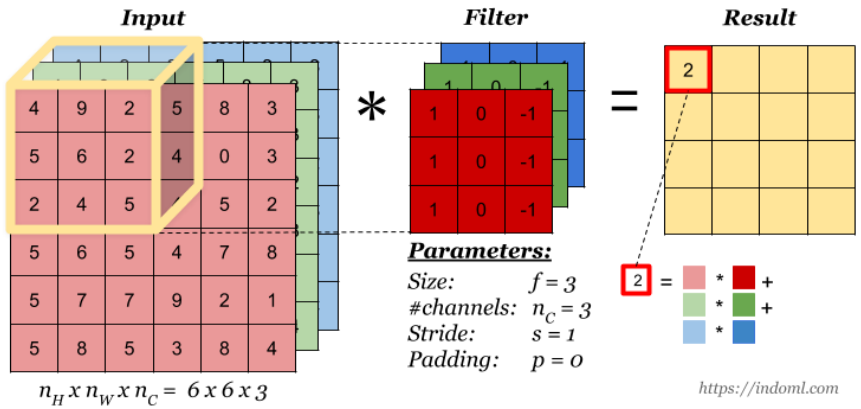
Interactive Node-Link Visualization of Convolutional Neural Networks

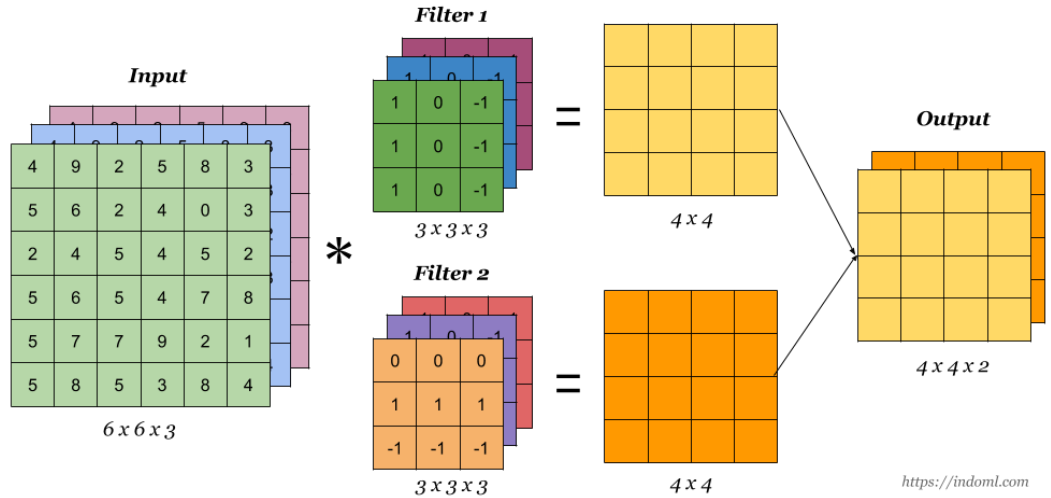


Other Convolution Operations

Convolution operation on Volume

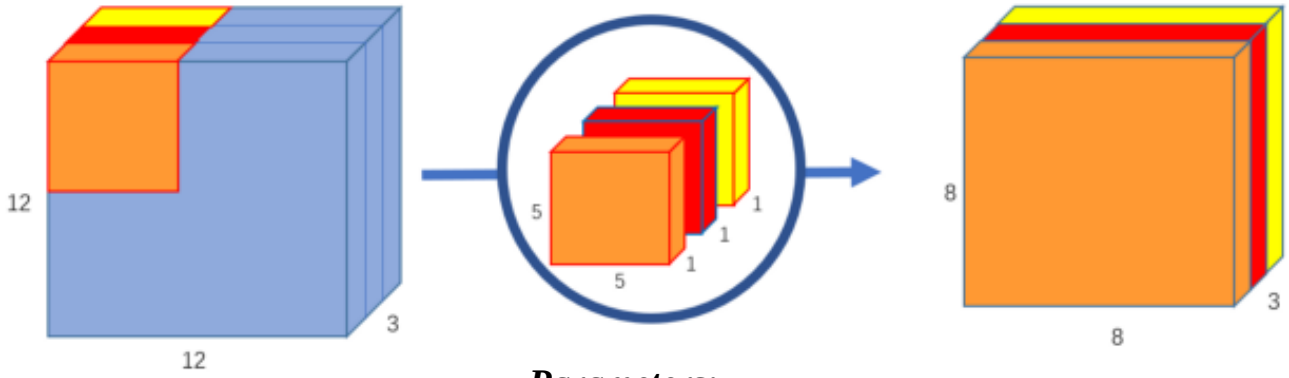
- A filter should have a matching number of channels if the input has more than one channel (e.g., an RGB image).
- In order to calculate one output cell, apply convolution to each matching channel and then add the results together.
- Multiple filters could be applied to the input image!





Depthwise Convolution

- Depthwise convolution is a type of convolution operation that operates on each channel of an input volume independently. In other words, the filter has the same depth as the input volume, and the convolution is applied separately to each channel.
- The output of a Depthwise convolution is a volume with the same number of channels as the input volume
- The Depthwise separable convolution is so named because it deals not just with the spatial dimensions, but with the depth dimension



Parameters:

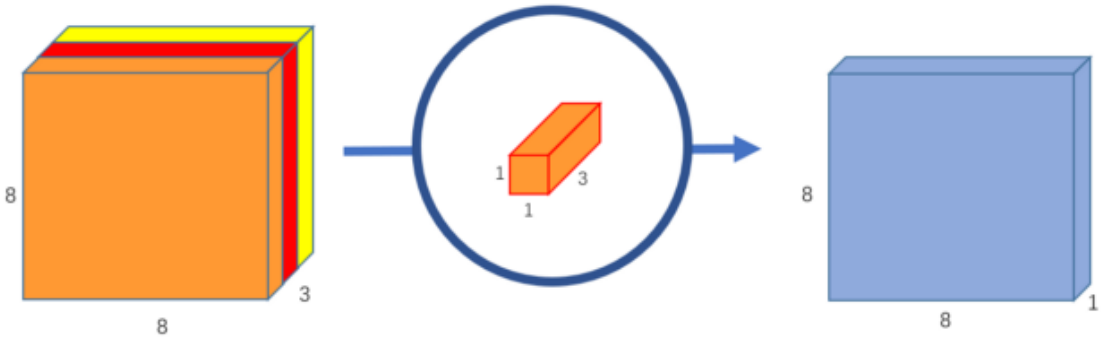
Size: $f = 5$

Stride $s = 1$

Padding $p = 0$

Pointwise Convolution: 1 x 1 Convolution

- A pointwise convolution, is a convolution operation in which the filter has a size of 1x1xC, where C is the number of channels in the input volume
- A 1x1 convolution is used to reduce the number of channels in the input volume, or to combine multiple channels in the input volume into a single channel in the output volume.
- This convolution can be thought of as a linear transformation of the input channels. This makes 1x1 convolutions useful for learning channel-wise relationships in the input data.



Parameters:

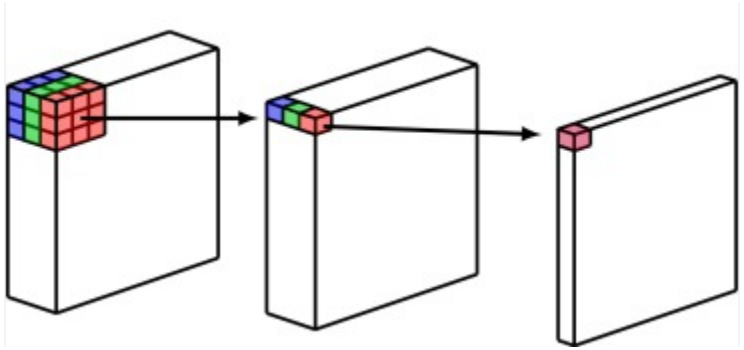
Size: $f = 1$

Stride $s = 1$

Padding $p = 0$

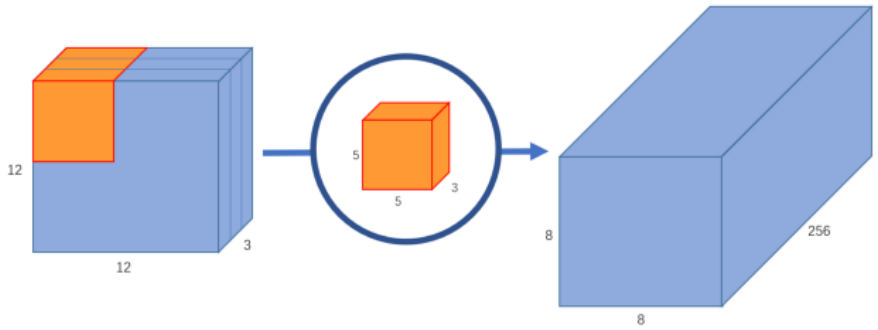
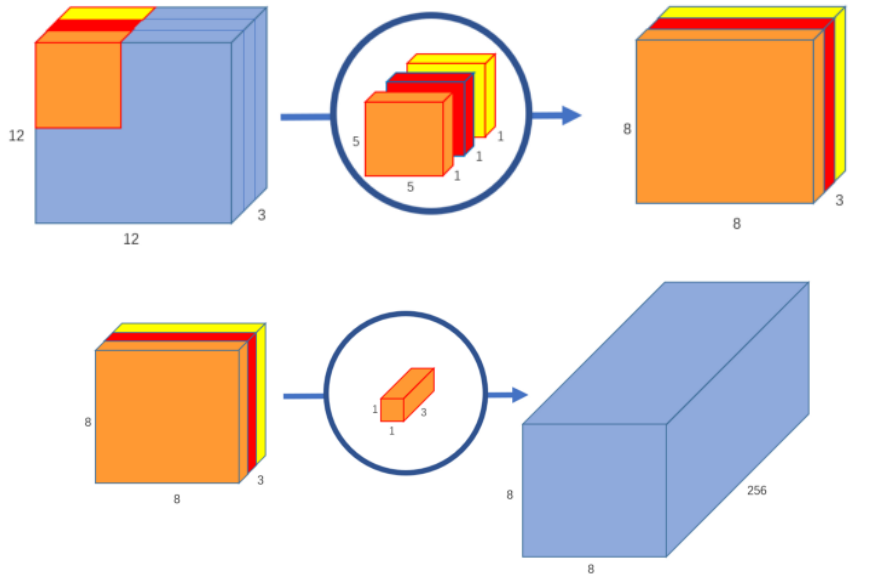
Depthwise Separable Convolution

- Depthwise convolution is often used in conjunction with pointwise convolution (a 1x1 convolution) to build a Depthwise separable convolutional neural network.
- This type of network architecture can be more efficient and faster to train than a traditional convolutional neural network, because it has fewer parameters and requires fewer computations.
- It is often used in mobile-based applications, where computational resources are limited, to build lightweight and efficient models.
- It is also used in models that need to learn spatial features from different channels separately, such as models for image segmentation or object detection.



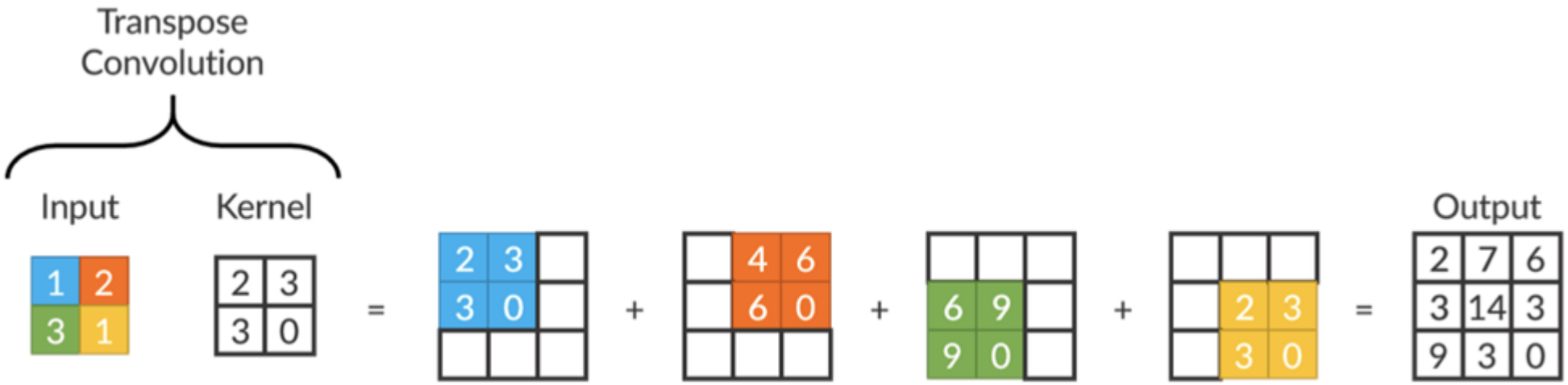
Why Depthwise Separable Convolution? A numerical example

- Imagine we want to use 256 kernels to create 256 of 8x8x1 output images from a 12x12x3 input.

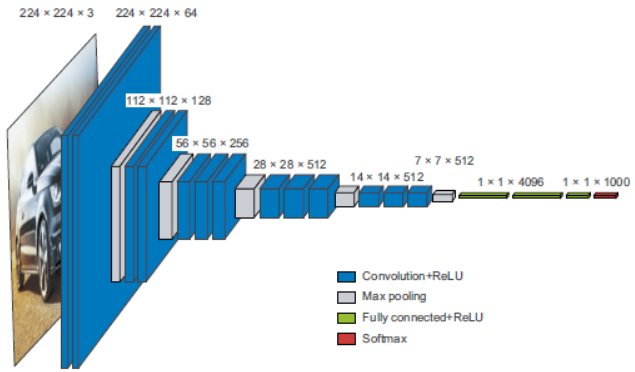
Traditional Convolution operation	Depthwise Separable Convolution
	
There are 256, 5x5x3 kernels that move 8x8 times 256*5*5*3*8*8= **1,228,800** multiplications	There are 3, 5x5x1 kernels that move 8x8 times + There are 256, 1x1x3 kernels that move 8x8 times = 3*5*5*1*8*8 + 256*1*1*3*8*8 = **53,952 multiplications**

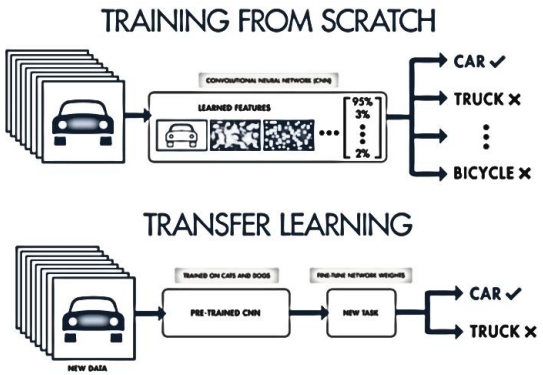
Transposed Convolution

- A transpose convolution, also known as a deconvolution, is a type of convolutional layer that is used to upsample the feature map produced by a convolutional neural network.
- It takes an input feature map and produces an output feature map that is larger in size, by inserting zeros between the elements of the input feature map and convolving with a set of filters



Module 5 – Part III
Deep Computer Vision
Architecture, Interpretable CNN and Transfer Learning



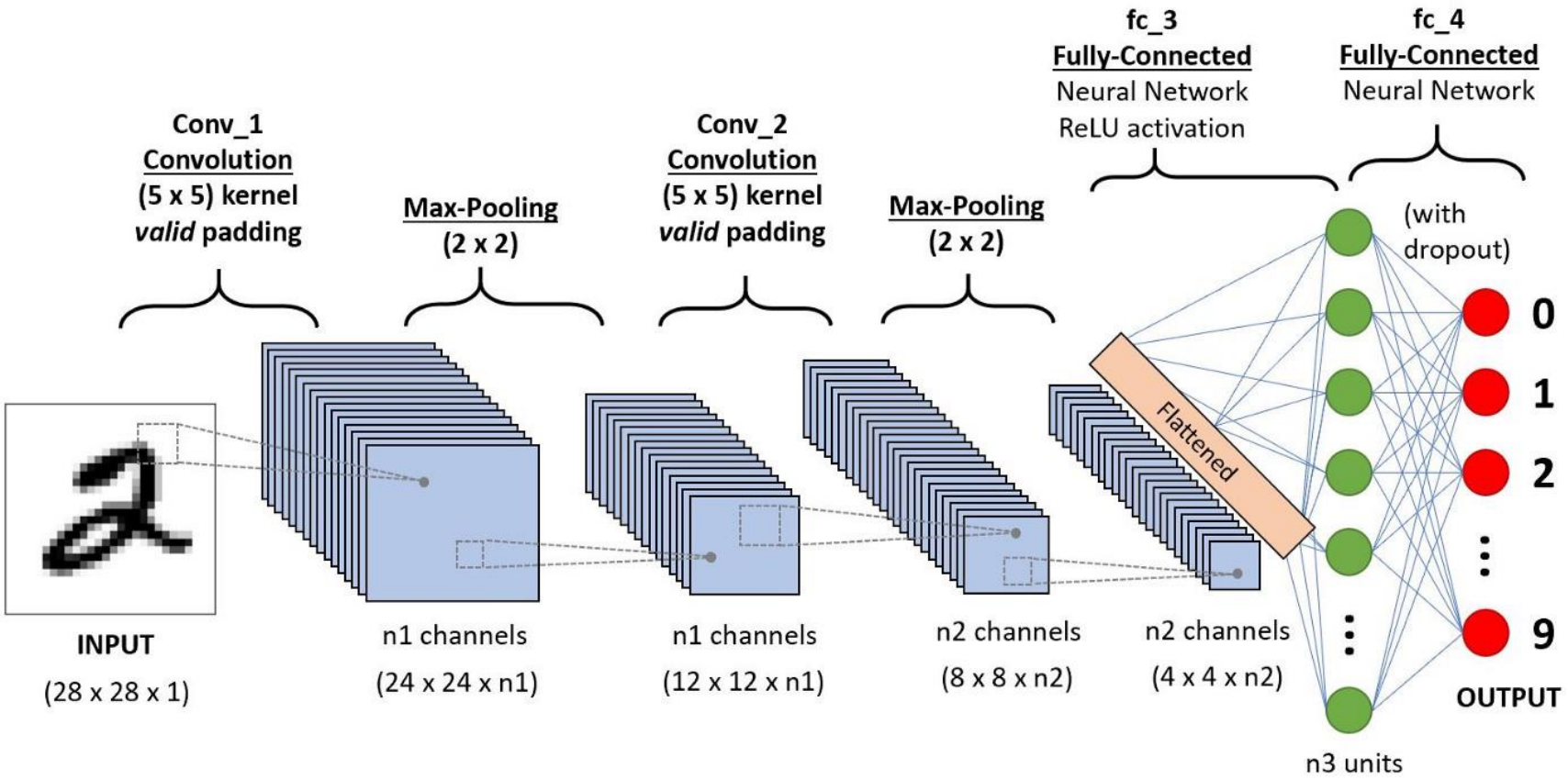


Jon M. Huntsman School of Business
UtahStateUniversity

Pedram Jahangiry

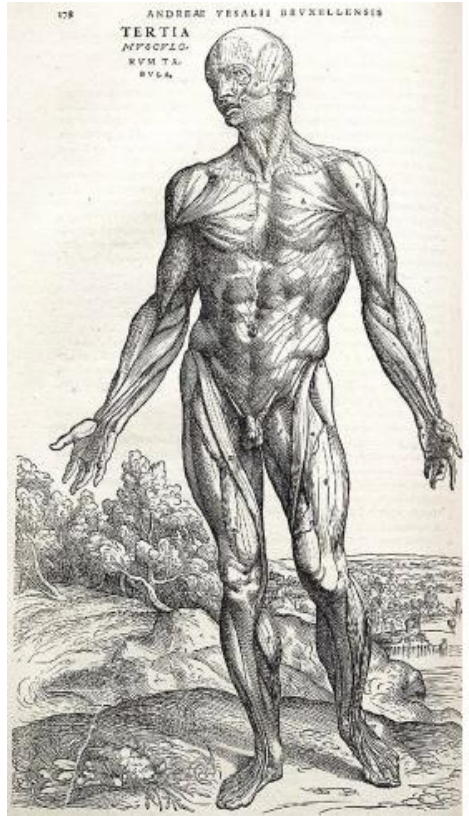
Convnet Architecture
Best Practices

CNN Architecture



The MHR formula

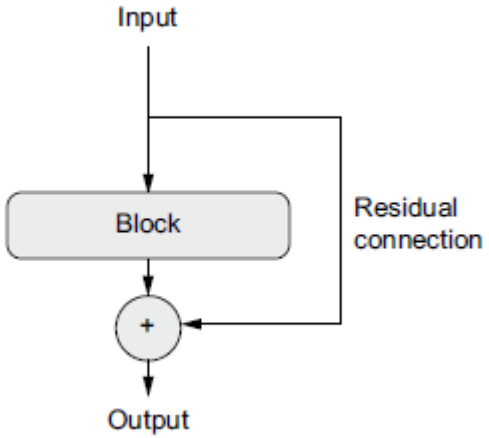
- Universal recipe to make a complex system simpler:
  - Structure your complex model into modules (Modularity)
  - Organize modules into a Hierarchy
  - Start reusing the same modules in multiple places as appropriate (Reuse)
- This is the pattern we observe in the most successful Deep Learning architectures.

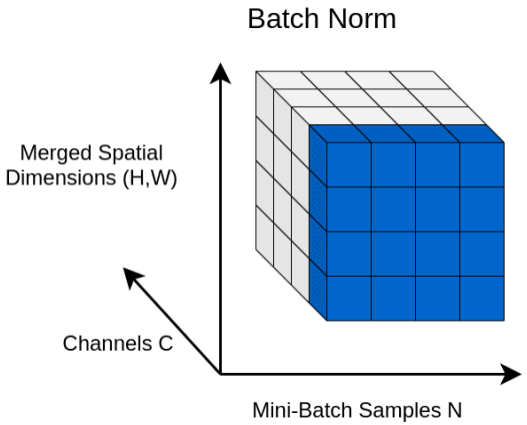


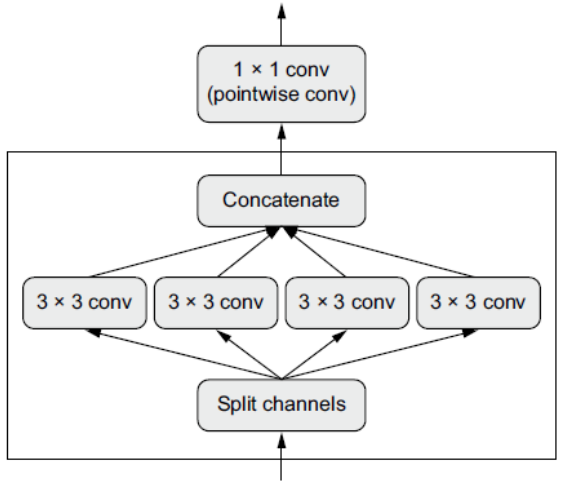
Convnet architecture best practices

- Model architecture is more an art than a science, however, there are some best practices!

1. Residual Connections
2. Batch Normalization
3. Separable Convolutions

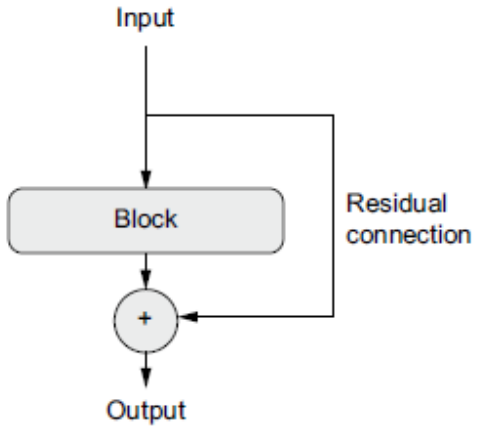






Residual Connections

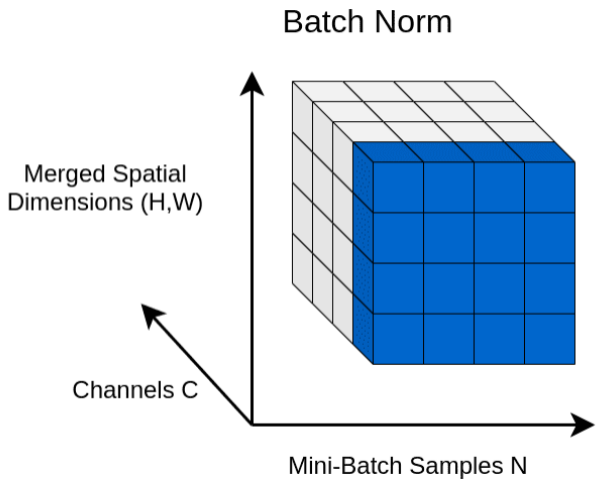
- The game of Telephone (Chinese Whispers)
- If the network is too deep, then successive functions in the chain introduce some noise!
- During training (backpropagation), this noise starts overwhelming the gradient information.



- Solution: add the input of a layer or block of layers back to its output
- Res connections act as *information shortcut* around the noise blocks.
- ResNet family of models was introduced in 2015 (He et al at Microsoft)

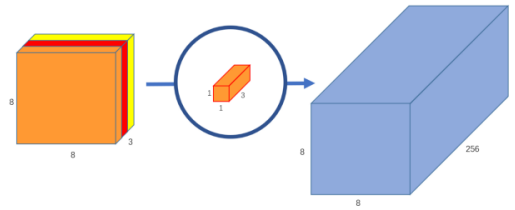
Batch Normalization

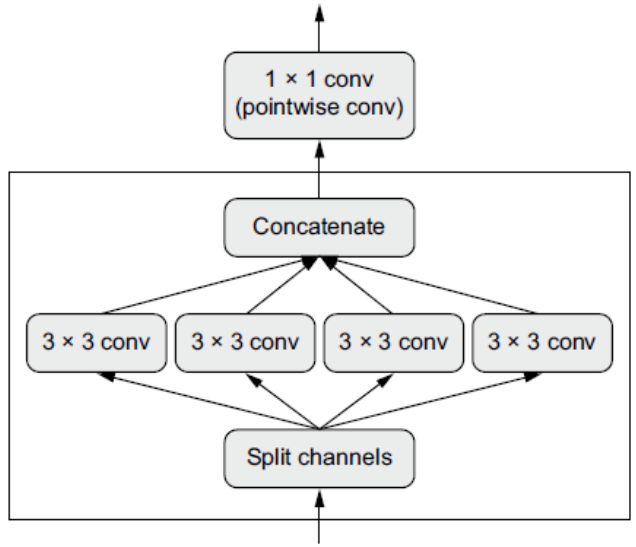
- Normalization: Statistical method to make different samples seen more similar to each other.
- Normalization helps the model learn and generalize better to new data.
- Batch normalization: Normalizes the mean and standard deviation for each individual channel /feature map (intermediate activations) using the current batch of data.
- Batch normalization helps with gradient propagation, allowing for deeper networks.

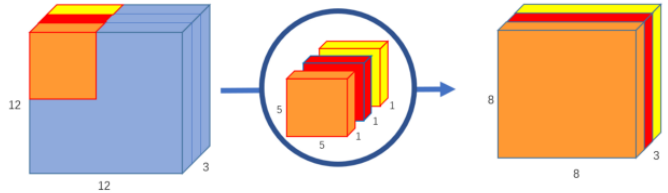


Separable Convolutions

- Depthwise separable convolution layers, perform spatial convolution on each channel of the input, independently and then mixing output channels using a pointwise convolution.
- Separable convolutions will:
  - Make the model smaller (fewer trainable parameters)
  - Require fewer float point operations (seen this before)
  - Improve model performance







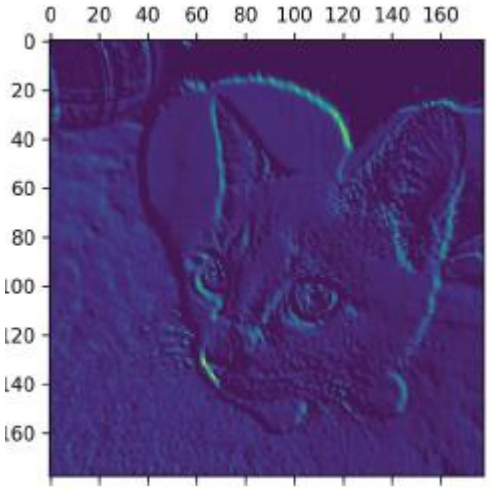
Best practices summary

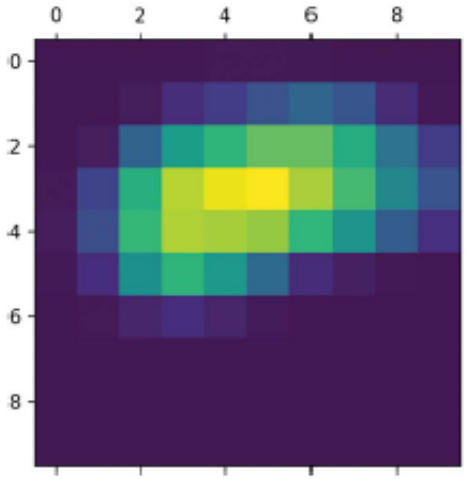
- Your model should be organized into repeated blocks of layers, usually made of multiple convolution layers and a max pooling layer.
- The number of filters in your layers should increase as the size of the spatial feature maps decreases.
- Deep and narrow is better than broad and shallow.
- Introducing residual connections around blocks of layers helps you train deeper networks.
- It can be beneficial to introduce batch normalization layers after your convolution layers.
- It can be beneficial to replace Conv2D layers with SeparableConv2D layers, which are more parameter-efficient.

Interpreting Convnets

Visualizing what convnets learn

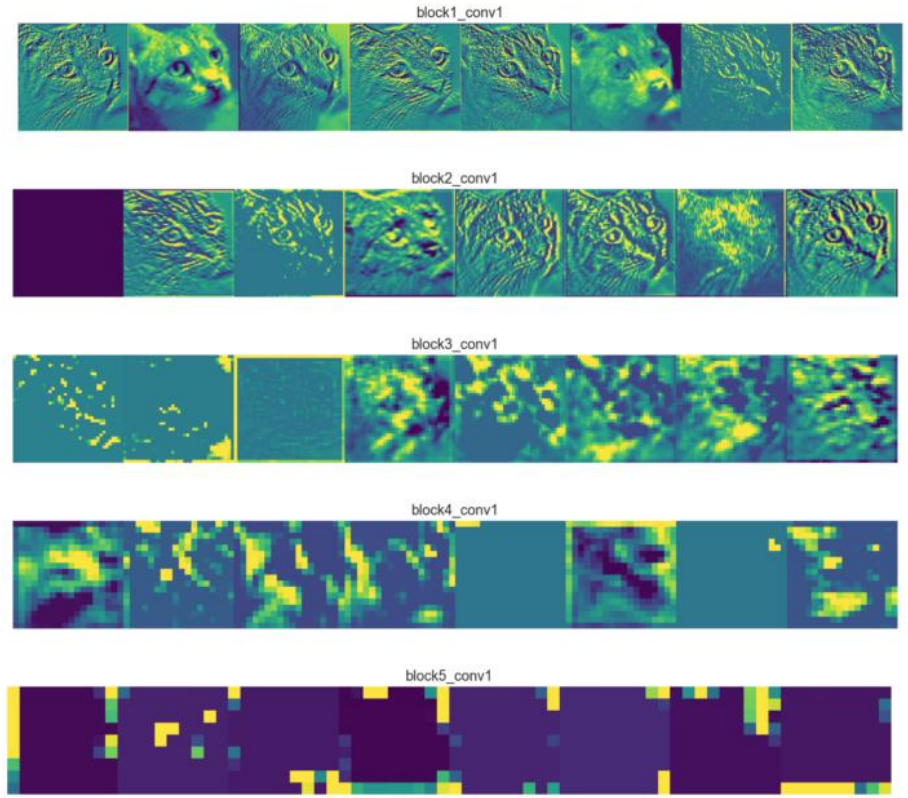
- Visualizing intermediate convnet output
- Visualizing filters
- Visualizing heatmaps (CAM: Class Activation Map)





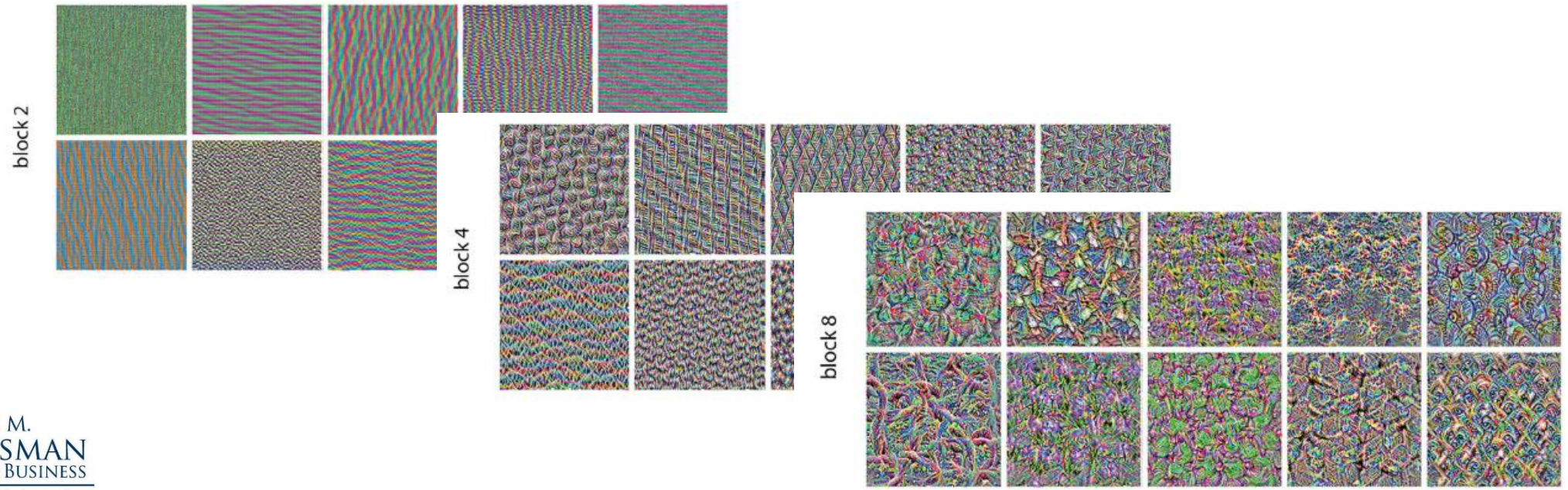
Visualizing conv outputs

- Low level layers act as a collection of various line and edge detectors
- Deeper presentations carry increasingly less information about the visual contents of the image (more abstract) and more information related to the class of the image.



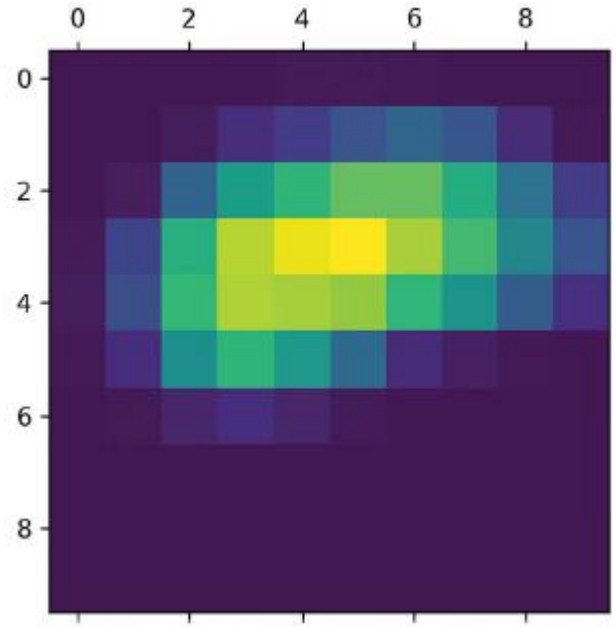
Visualizing Convnet Filters

- The filters from the first layers encode simple directional edges and colors
- The filters deeper in the model encode simple textures made from combinations of edges and colors.
- The filters in higher layers begin to resemble textures found in natural images: eyes, ears, and so on.



Visualizing Heatmaps (CAM)

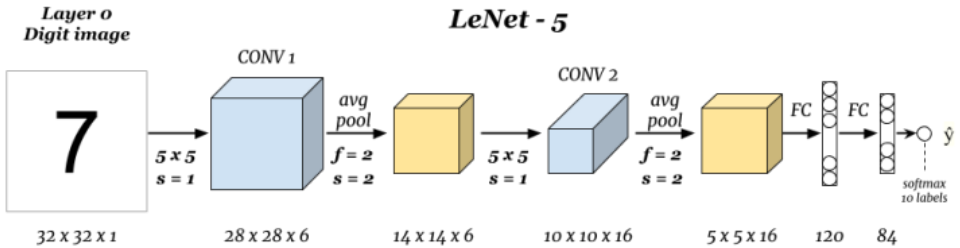
- Class Activation Map (CAM): Producing heatmaps of class activation over input
- CAM indicates how important each location is with respect to the class under consideration.

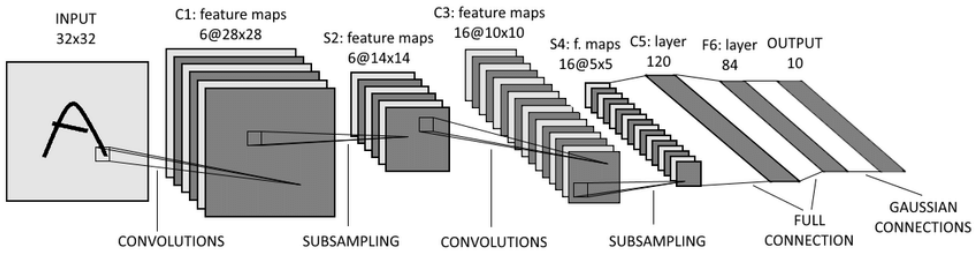


Transfer Learning

LeNet – 5

- LeNet is a deep convolutional neural network architecture that was developed by Yann LeCun et al. in **1998**.
- It was one of the first successful applications of deep learning to the task of handwritten digit recognition and was a breakthrough in the field of computer vision.
- LeNet architecture consists of a series of convolutional and pooling layers (to extract features and reduce the size of feature map) and few fully connected layers.
- The LeNet architecture has been influential in the development of subsequent deep learning models and is still widely used as a baseline model for various tasks in computer vision

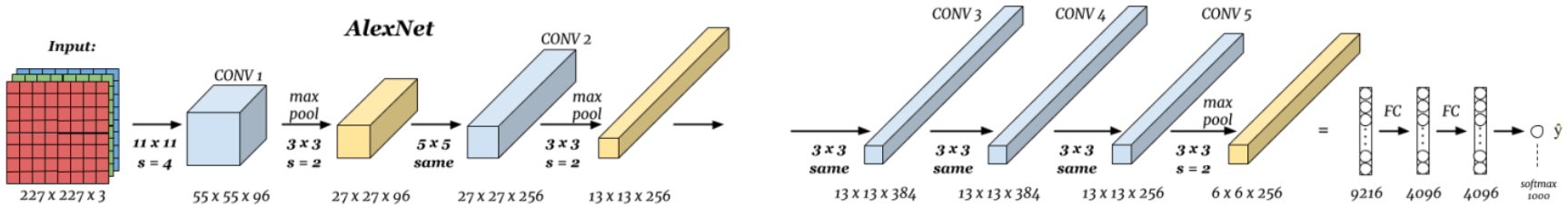




Gradient-Based Learning Applied to Document Recognition paper by Y. Lecun, L. Bottou, Y. Bengio and P. Haffner (1998)

AlexNet

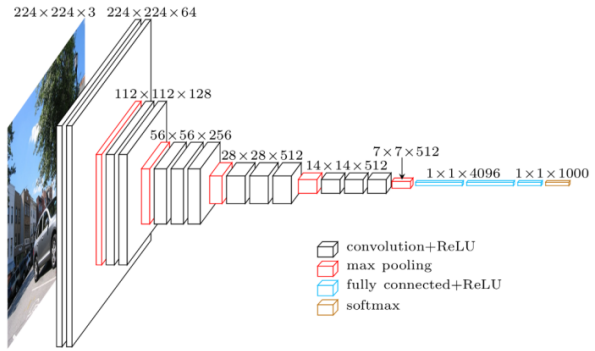
- AlexNet was developed by Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton in **2012**
- AlexNet was one of the first CNNs to achieve significant success in image classification and popularized CNN for this task.
- It was the first CNN to use rectified linear units (ReLUs) as the activation function, which helped to improve the training speed and accuracy of the model.
- AlexNet used a large number of filters in its convolutional layers, which allowed it to learn more complex features from the input data



ImageNet Classification with Deep Convolutional Neural Networks paper by Alex Krizhevsky, Geoffrey Hinton, and Ilya Sutskever (2012).

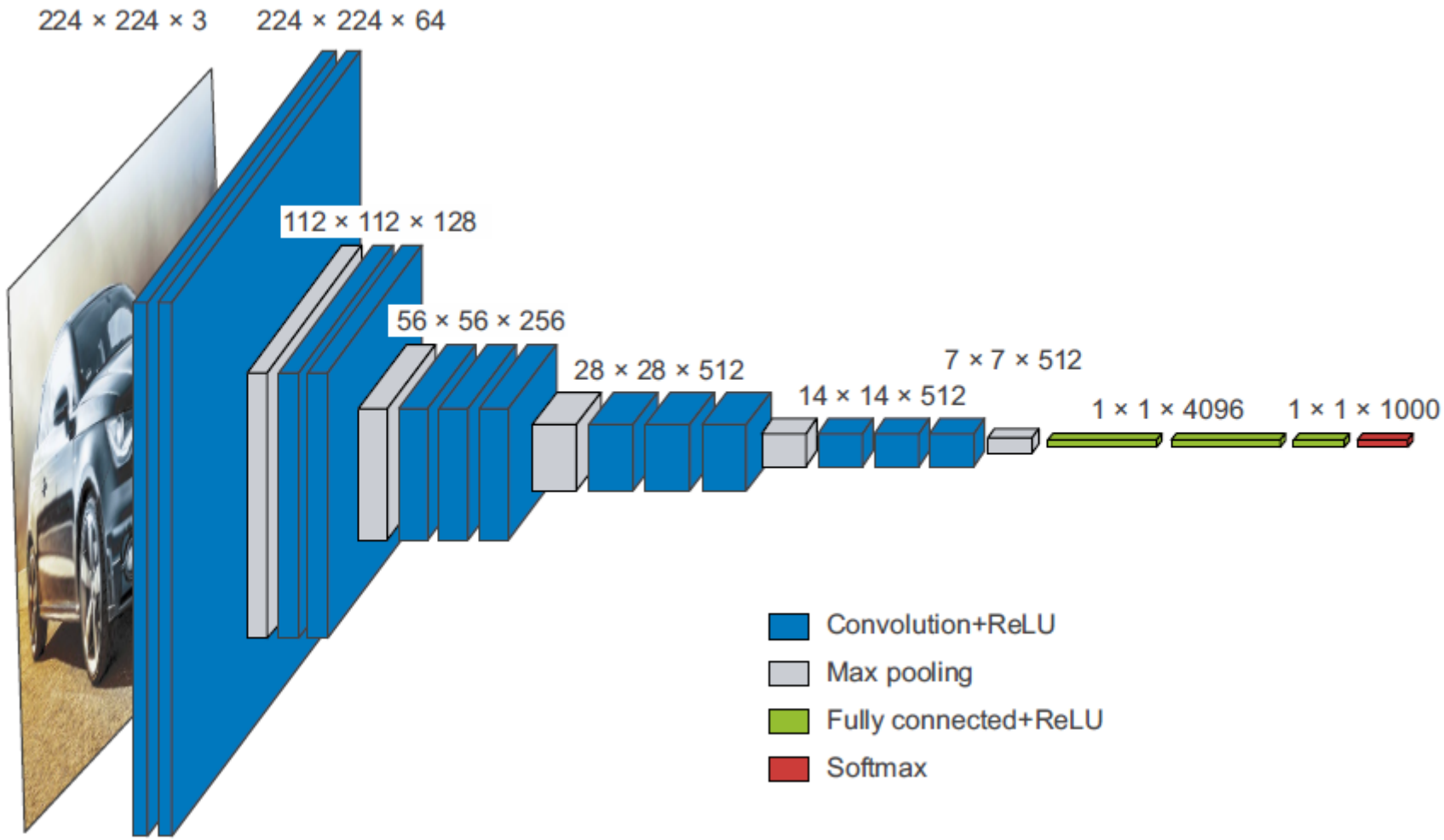
VGG-16

- VGG16 is a CNN developed by Karen Simonyan and Andrew Zisserman in **2014** at Visual Geometry Group (VGG) at the University of Oxford.
- The number 16 refers to the fact that the network has 16 trainable layers (i.e. layers that have weights)
- VGG's strength is in the simplicity: the dimension is halved, and the depth is increased on every step (or stack of layers)
- Combination of a deep architecture, simple design, and relatively small number of parameters has contributed to the success of VGG16 for tasks in computer vision/



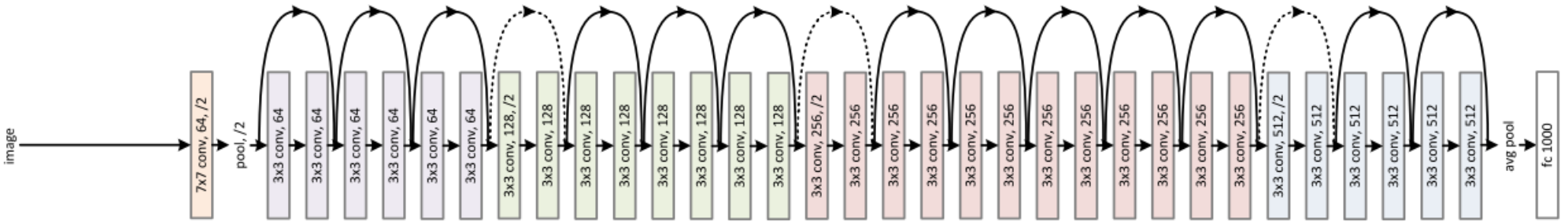
Very Deep Convolutional Networks for Large-Scale Image Recognition paper by Karen Simonyan and Andrew Zisserman (2014).

VGG-16


224 × 224 × 3
224 × 224 × 64
112 × 112 × 128
56 × 56 × 256
28 × 28 × 512
14 × 14 × 512
7 × 7 × 512
1 × 1 × 4096
1 × 1 × 1000
Convolution+ReLU
Max pooling
Fully connected+ReLU
Softmax

ResNet

- ResNet (Residual Network) is a CNN that was developed by Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun in **2015** at Microsoft.
- The key innovation of the ResNet architecture is the use of residual connections
- This allows the model to learn much deeper networks more effectively
- ResNet has been used as a building block for other architectures, such as Mask R-CNN and U-Net.
- There are several variants of the ResNet architecture, including ResNet-50, ResNet-101, and ResNet-152, which differ in the number of layers and the number of parameters.



Deep Residual Learning for Image Recognition. Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun (2015)

Comparing classical models

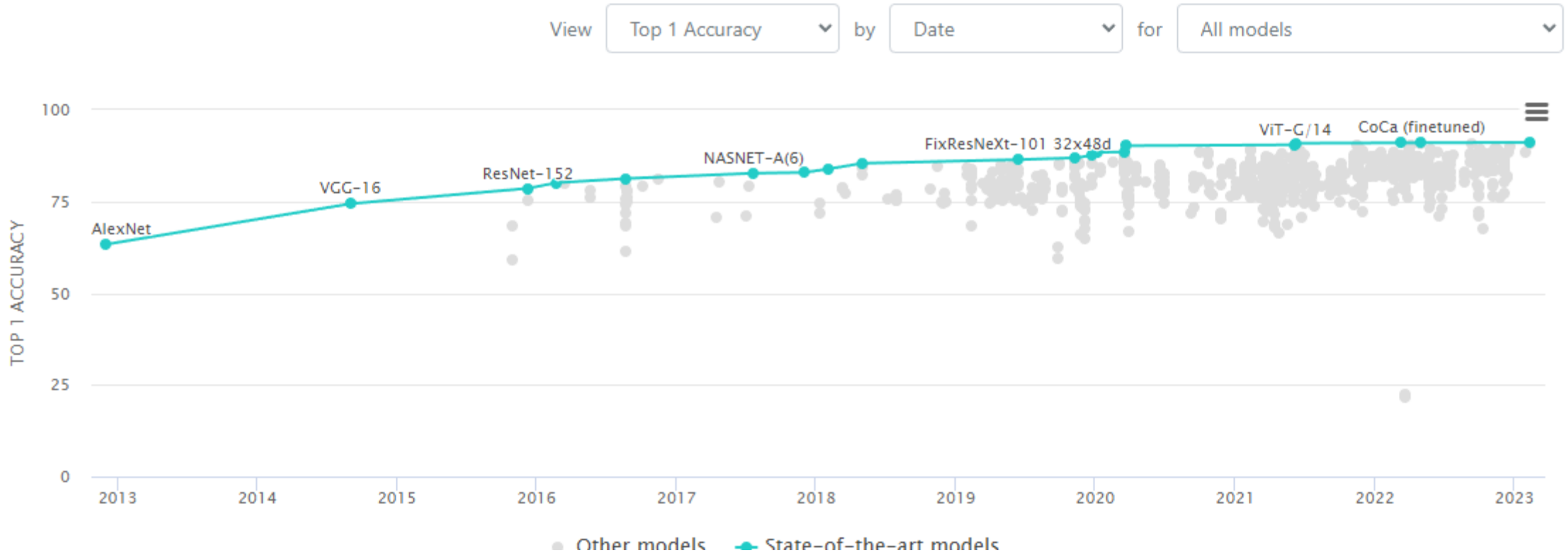
- The ImageNet dataset consists of more than 1 million images and 1000 categories. The categories are organized into a hierarchy, with each category having multiple subcategories.

- ImageNet has played a significant role in the development of deep learning and has contributed to the success of many state-of-the-art models for tasks in computer vision.

Network	Parameters	Top1 accuracy
LeNet 5	~ 60,000	-
AlexNet	~ 60 million	63.3%
VGG16	~138 million	74.4%
ResNet 152	~ 60 million	78.57%

https://paperswithcode.com/sota/image-classification-on-imagenet

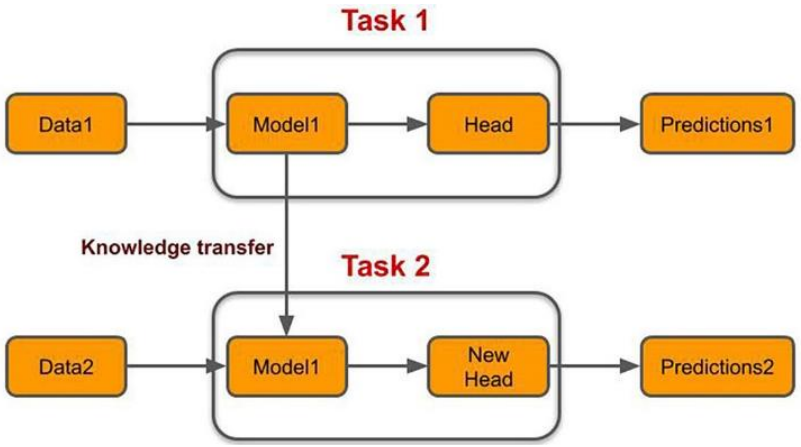
State-of-the-art models



https://paperswithcode.com/sota/image-classification-on-imagenet

Transfer Learning

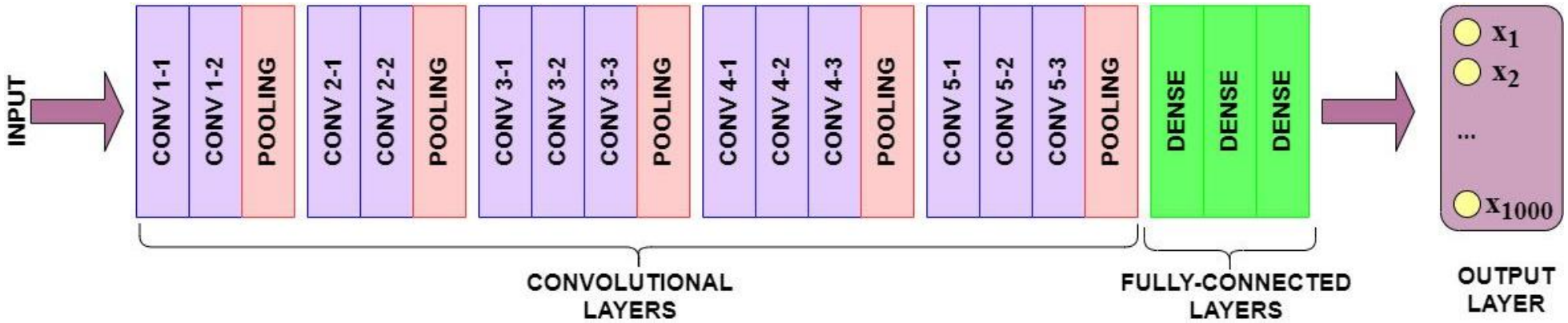
- Transfer learning is a machine learning technique where a model trained on one task is re-purposed on a second related task.
- We use this technique to leverage a model's backbone by popping off its head (the last few layers) and replacing it with untrained layers that are appropriate for our task.
- Transfer learning is useful because it allows the model to leverage the knowledge it has gained from the first task and apply it to the second task, potentially leading to improved performance on the second task.



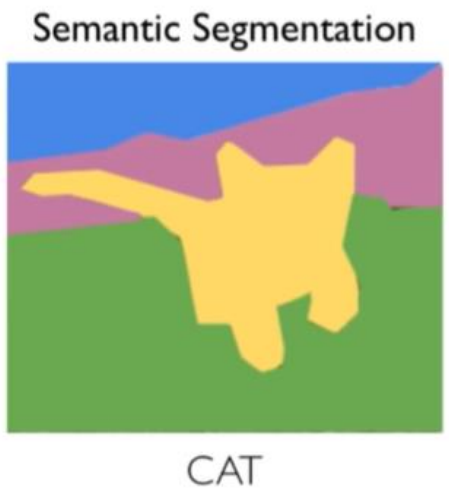
Transfer Learning

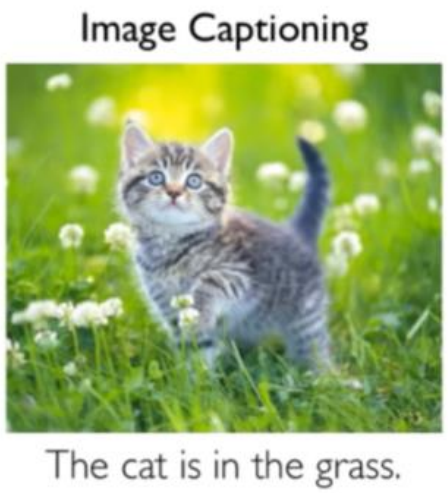
There are two main ways to perform transfer learning:

- Feature Extraction: the pre-trained model is used to extract features, and these features are then fed into a new model that is trained to perform the new task.
- Fine-tuning: unfreezing some of the layers of the pre-trained model + training and fine tuning these layers on the new task



Module 5 – Part IV
Deep Computer Vision
Beyond Classification



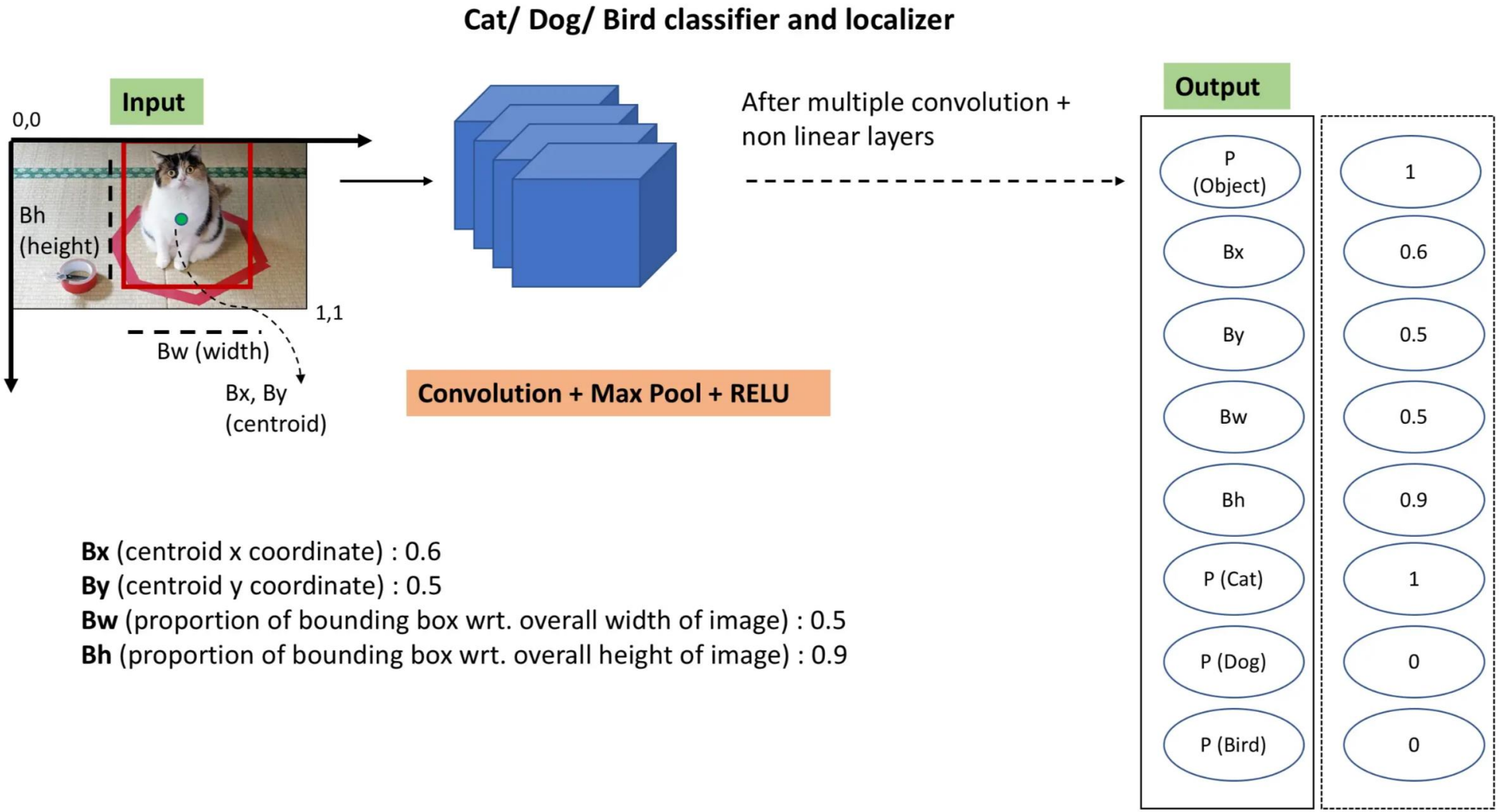


Beyond Classification

Feature	Object Detection	Image Segmentation	Image Captioning
Task	Identify and locate objects in an image	Classify each pixel in an image into a specific category	Generate a textual description of an image
Output	Bounding boxes and labels for objects	A label for each pixel in the image	A written description of the image
Examples	Identifying and localizing cars, pedestrians, and other objects in an image	Assigning labels to each pixel in an image to create a segmentation map, such as "road", “car", “sky"	“The cat is in the grass"

Object Detection

Object Classification and Localization



Bx (centroid x coordinate) : 0.6
By (centroid y coordinate) : 0.5
Bw (proportion of bounding box wrt. overall width of image) : 0.5
Bh (proportion of bounding box wrt. overall height of image) : 0.9

Evolution of Object Detection and Localization Algorithms, Prince Grover, Toward Data Science

Object Detection

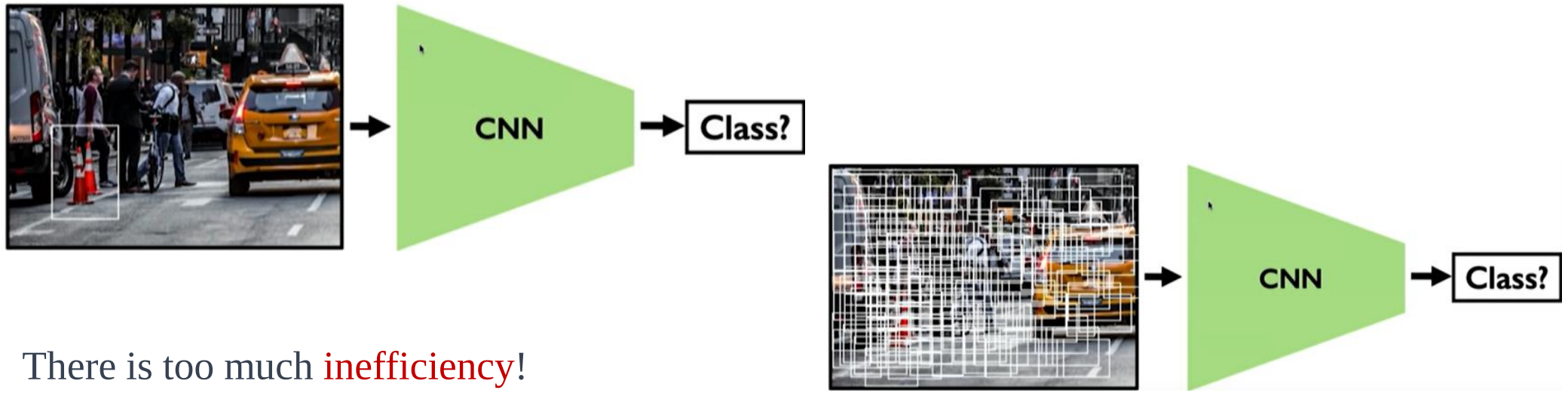
- Object detection is a task that involves identifying and locating objects in images or video.
- The model learns to recognize the objects and predict bounding boxes around them in new images.

Solutions?

- Naïve Approach!
- R-CNN, Fast R-CNN and Faster R-CNN
- YOLO

Object detection: Naïve Approach

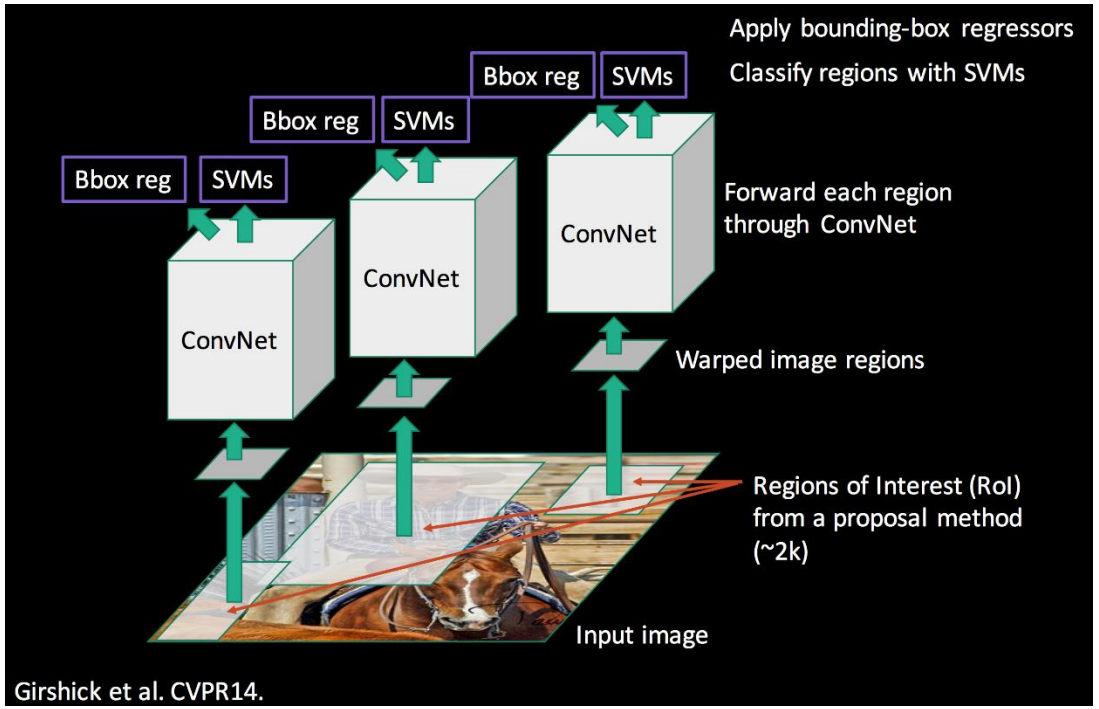
- Scan the image with a sliding window!
- Place a random box (position and size) somewhere in the image!
- Feed this box into a CNN and predict the class.
- Repeat this with different random boxes!!!



- There is too much inefficiency!
- Computationally expensive and too slow for real time uses!

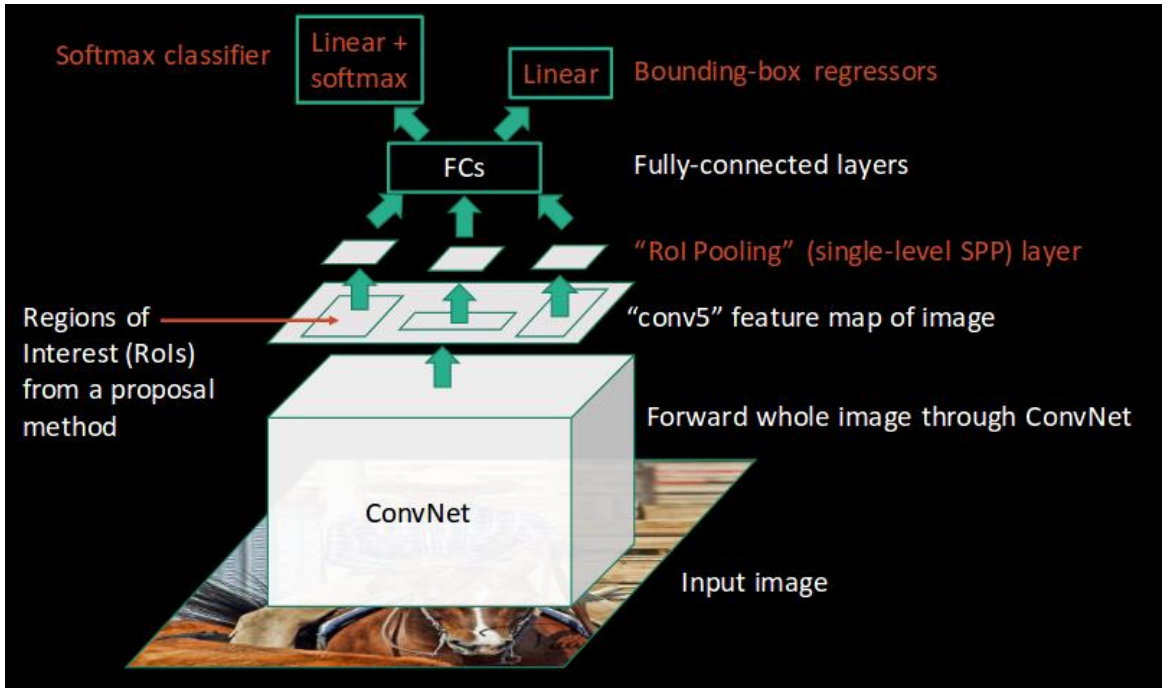
Region-based Convolutional Neural Network (R-CNN)

- Strategically selects interesting regions to run through the classifier.
- Regions of Interest (ROI) are the regions that we think have objects with higher probabilities.
- Problems:
  - Very slow (classify 2k region proposals per image)! No real time use case!
  - How to define region proposals? The selective search algorithm is a fixed algorithm detached from the network.



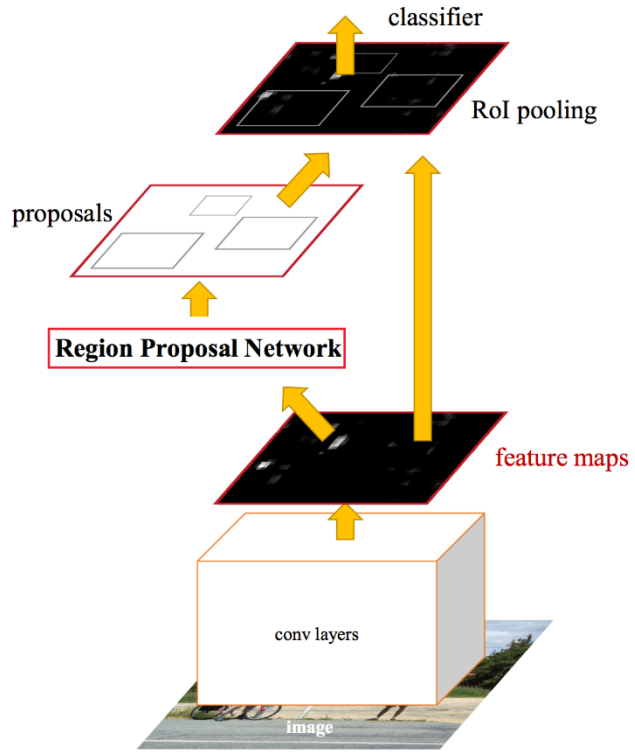
Fast R-CNN

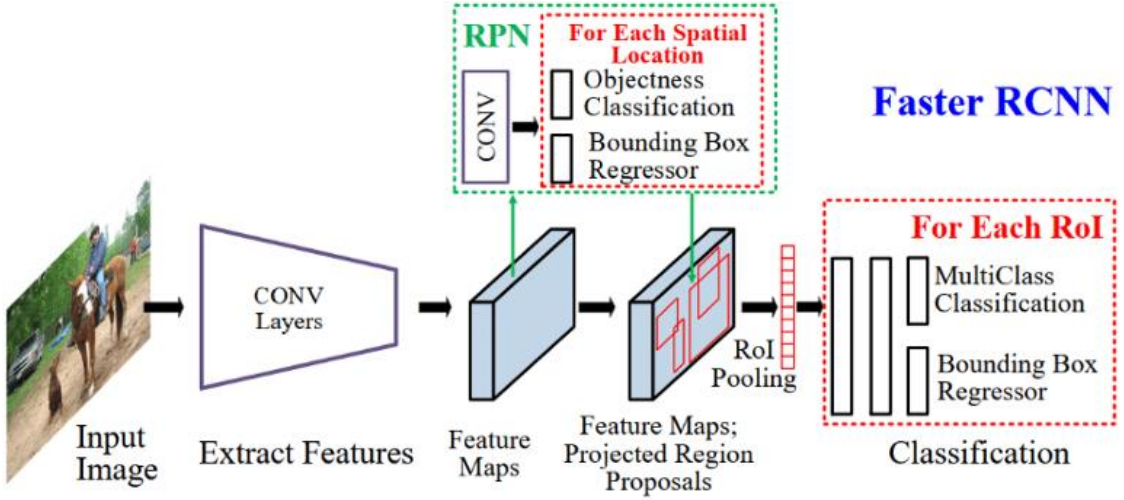
- The approach to Fast R-CNN is similar to the R-CNN algorithm. But, instead of converting 2,000 regions into the corresponding features maps, we convert the whole image once.
- Problems:
  - Still uses selective search to generate the RoIs (detached from the network)



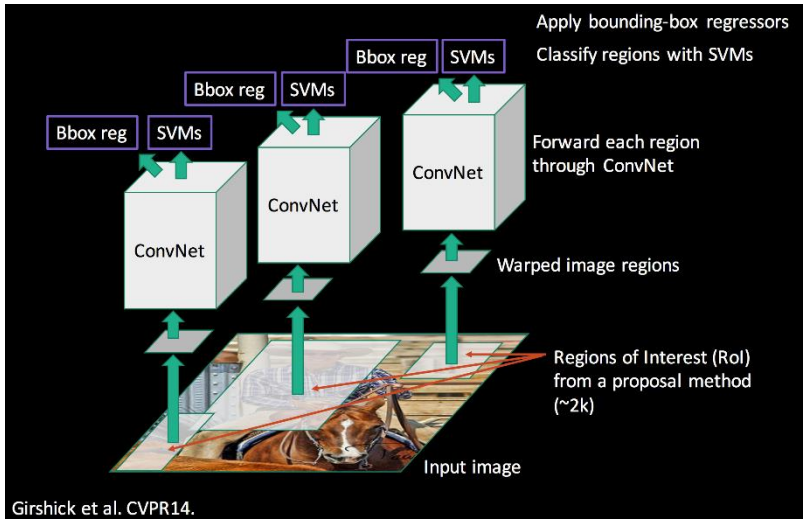
Faster R-CNN

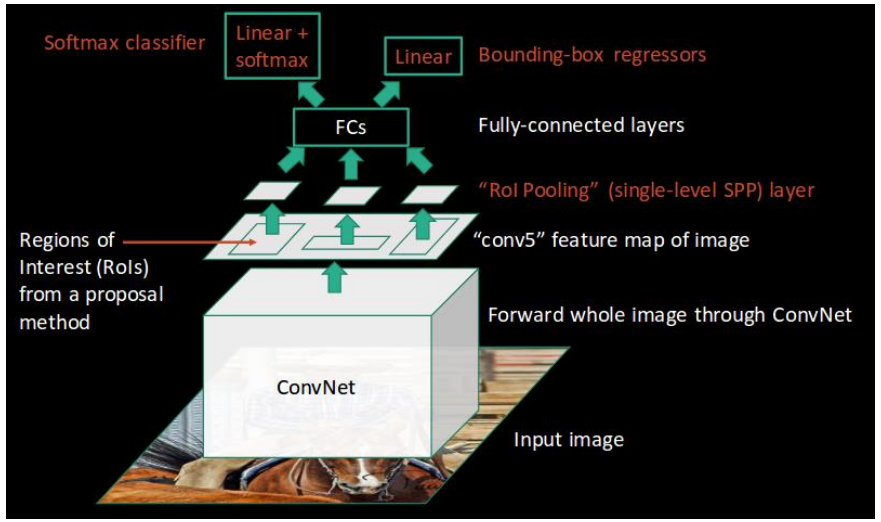
- In Faster R-CNN, instead of using selective search algorithm on the feature map to identify the region proposals, a separate network is used to predict the region proposals.
- The predicted region proposals are then reshaped using a RoI pooling layer which is then used to classify the image within the proposed region and predict the offset values for the bounding boxes.

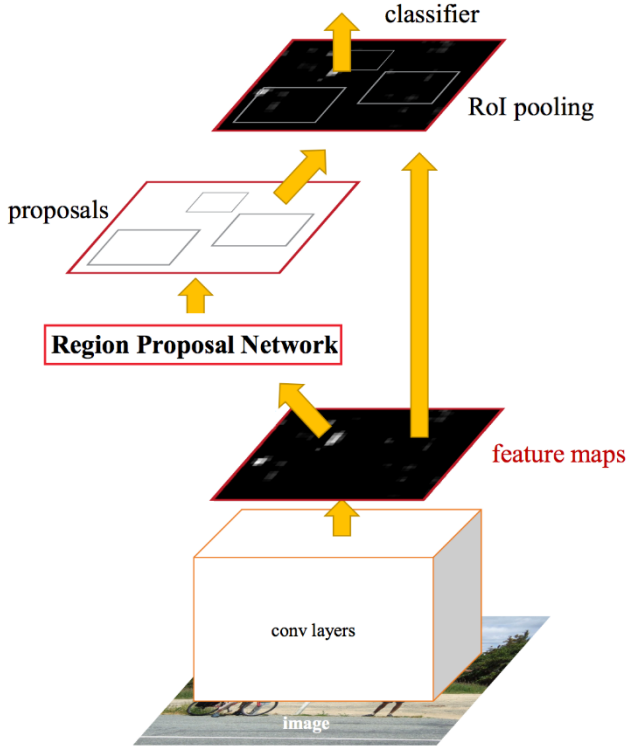




Comparing R-CNN-based models

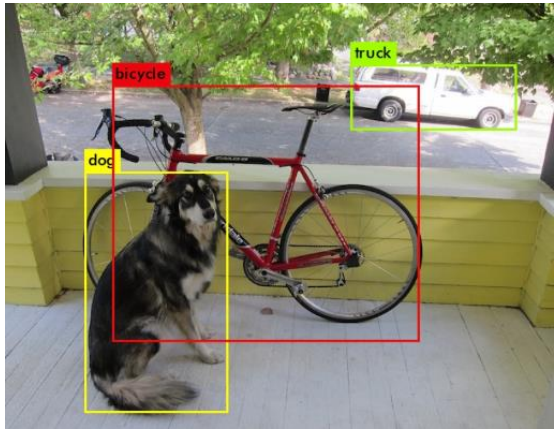






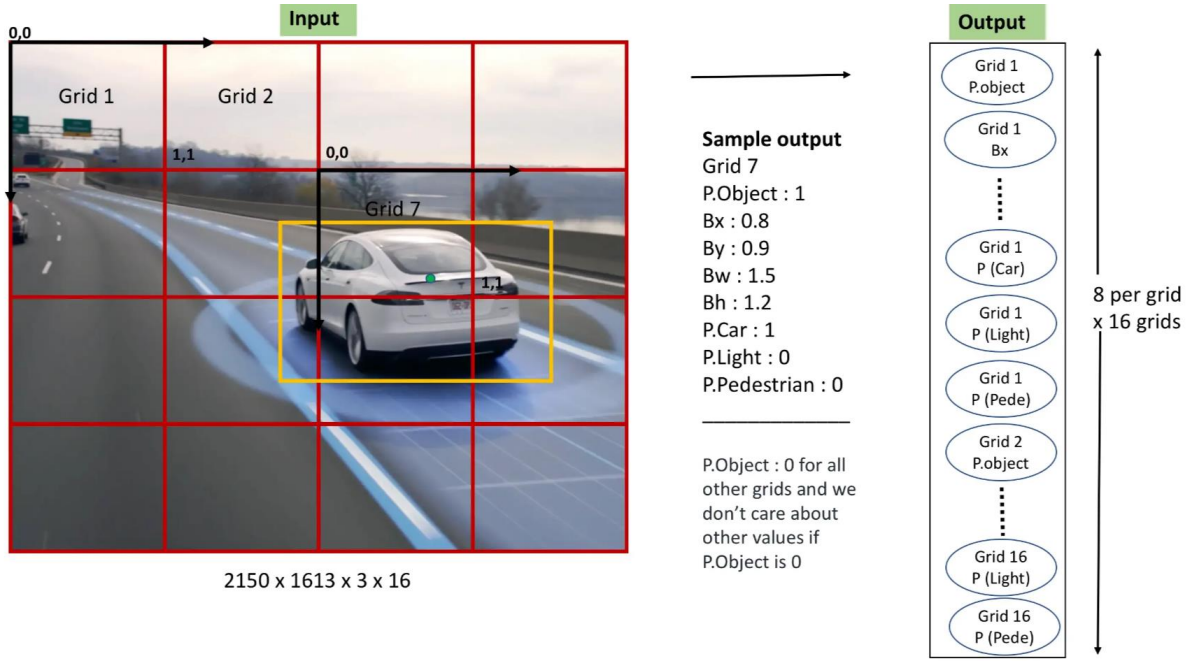
YOLO (You Only Look Once!)

- YOLO is a real-time object detection algorithm. It was developed by Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi at the University of Washington (2015)
- Yolo is extremely fast because it passes the entire image at once into a CNN, rather than making predictions on many individual regions of the image.
- The key idea behind YOLO is to use a single neural network to predict the bounding boxes and class probabilities for objects in an image



YOLO (How it works?)

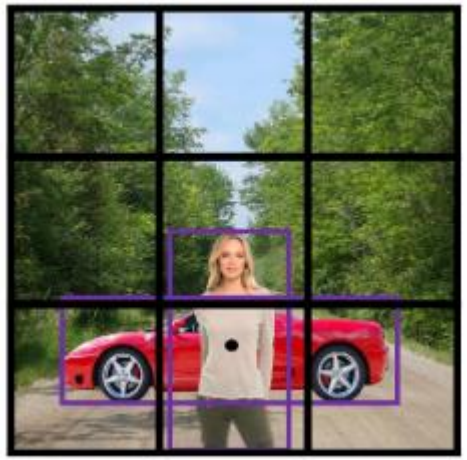
- YOLO divides the input image into a grid of cells and predicts the presence of objects in each cell
- If an object is detected in a cell, the algorithm also predicts the bounding box and the class for the object
- The bounding box coordinates and class probabilities are then used to localize and classify the objects

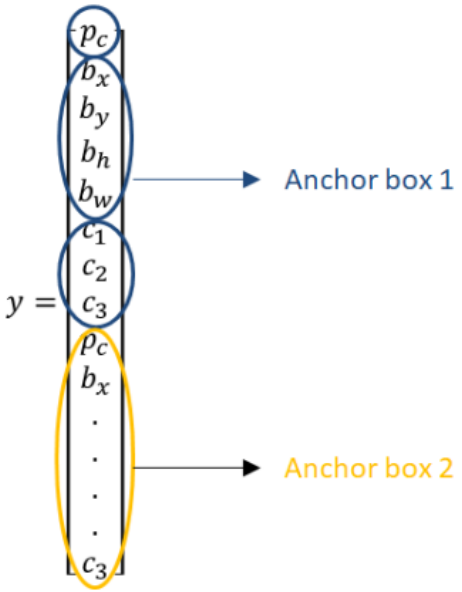


Output size:
(5+C) x S x S

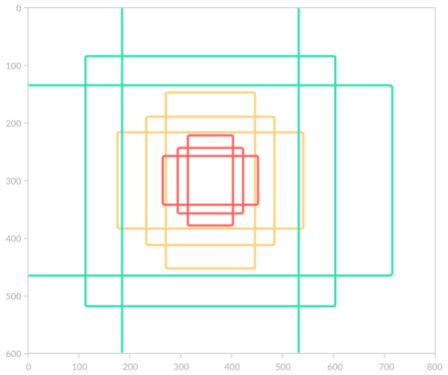
YOLO – Anchor boxes

- One of the Caveats of YOLO is that it can't detect multiple objects in same grid.
- Solution: Anchor boxes. It is a predefined bounding box used in object detection algorithms.
- The anchor box is used to define the size and aspect ratio of the window, and it is defined prior to training the object detection model. The model is then trained to predict the bounding box coordinates and class probabilities for objects relative to the anchor box.





Output size:
(5+C) x B x S x S



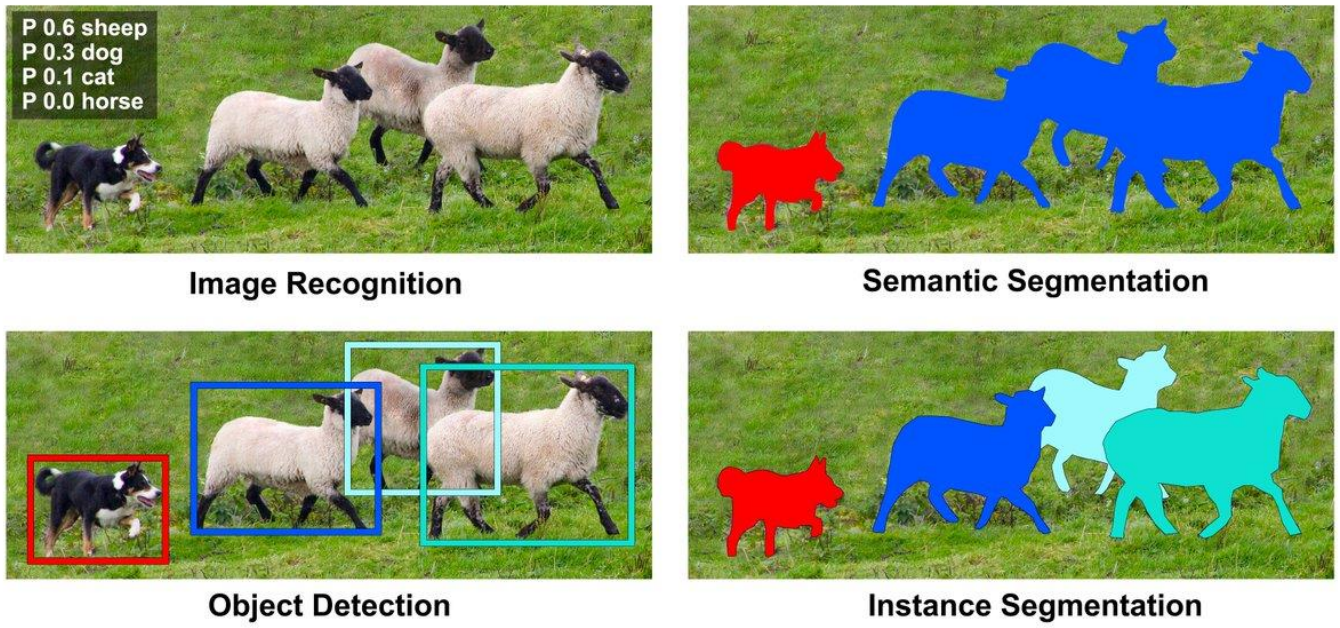
YOLO – Non-max Suppression (NMS)

- Another caveat of YOLO is the possibility of detecting one object multiple times!
- Solution: Non-max Suppression which removes duplicate or overlapping bounding boxes.
- In NMS, the algorithm iterates through all the bounding boxes in the image and selects the box with the highest confidence score.
- It then removes all other boxes that overlap with the selected box by a certain threshold. This process is repeated until all the boxes have been processed.

Image Segmentation

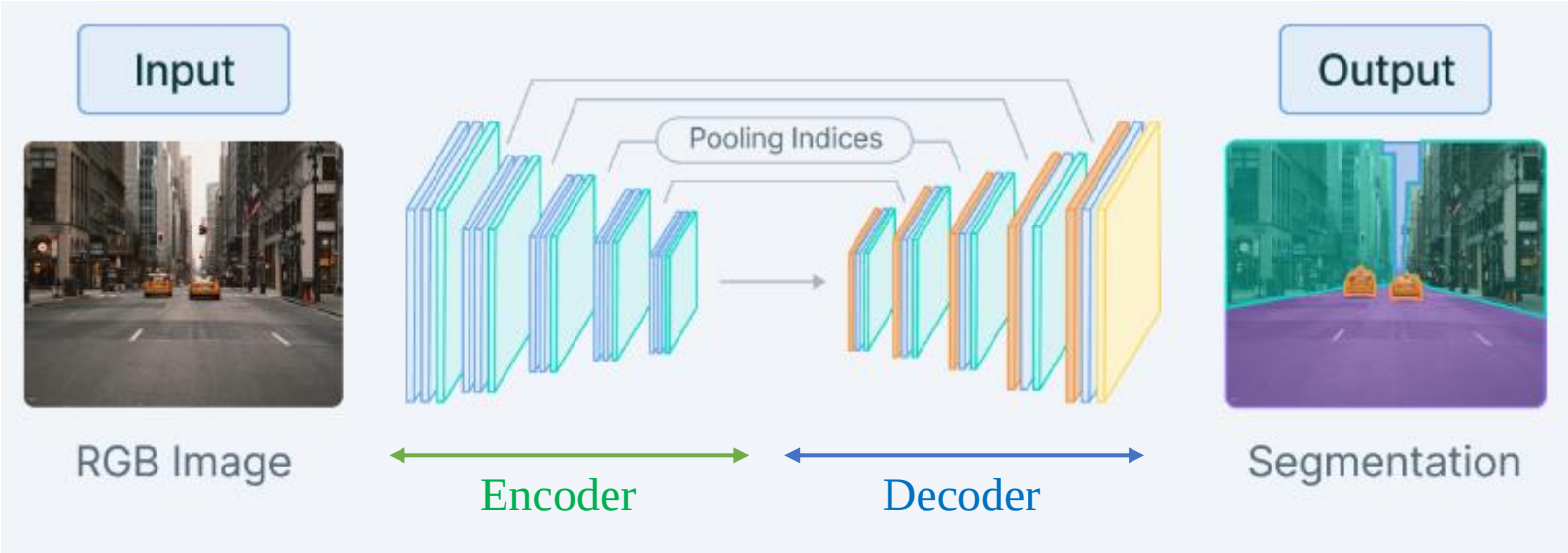
Image Segmentation

- Semantic segmentation is the process of assigning a class label to each pixel in an image, based on the pixel's visual characteristics and its relationship to the surrounding pixels.
- Instance segmentation involves not only classifying each pixel in the image, but also distinguishing between different instances of the same class.



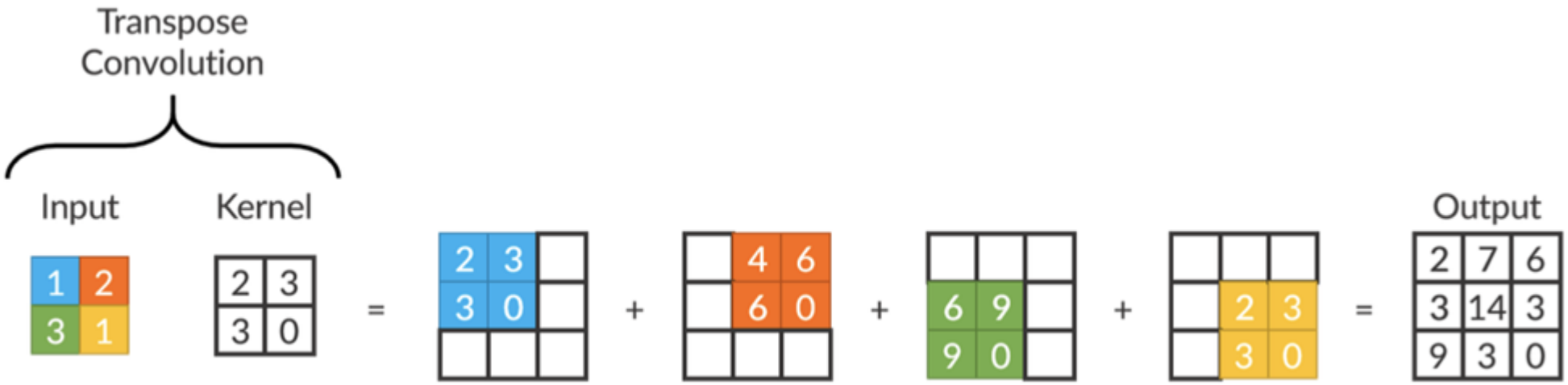
Fully Convolutional Network (FCN), semantic segmentation

- In an FCN, the input image is passed through a series of convolutional layers, which extract features from the image and reduce its spatial resolution (**Encoder**).
- The final layers of an FCN are typically transposed convolutional layers, which upsample the feature map to the original resolution of the input image (**Decoder**), and output a dense prediction for each pixel in the image



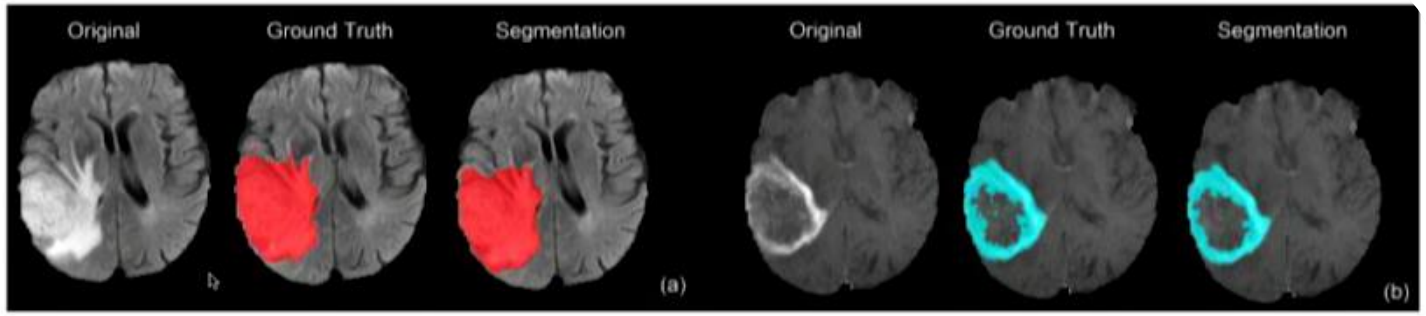
Transposed Convolution

- A transpose convolution, also known as a deconvolution, is a type of convolutional layer that is used to upsample the feature map produced by a convolutional neural network.
- It takes an input feature map and produces an output feature map that is larger in size, by inserting zeros between the elements of the input feature map and convolving with a set of filters

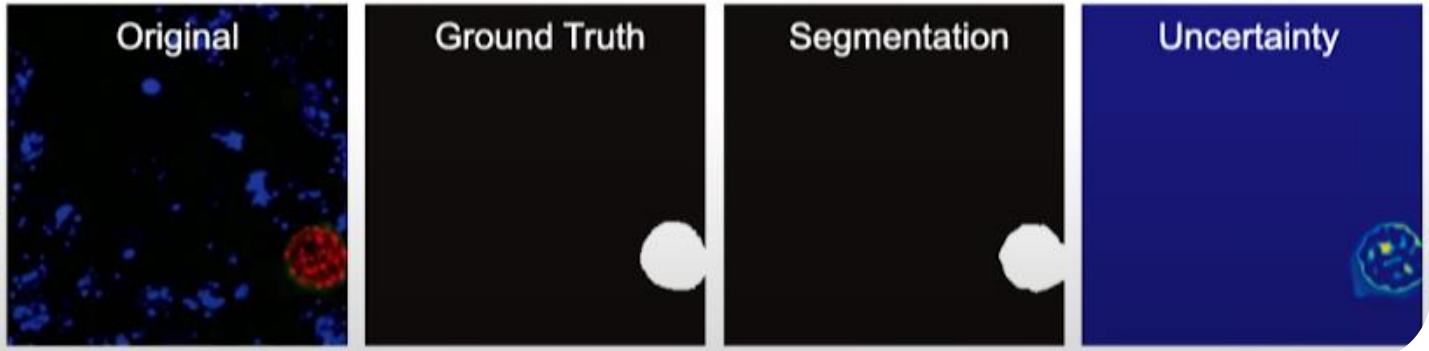


Semantic Segmentation applications

Brain Tumors
Dong+ *MIUA* 2017.



Malaria Infection
Soleimany+ *arXiv* 2019.



Semantic Segmentation applications

Image Captioning

Image Captioning

- Image captioning is the task of generating a textual description of an image, based on the objects, actions, and scenes depicted in the image.
- Image captioning models typically consist of two parts:
  - A vision model that extracts features from the input image (CNN)
  - A language model that process the features and generates the caption (RNN, LSTM)

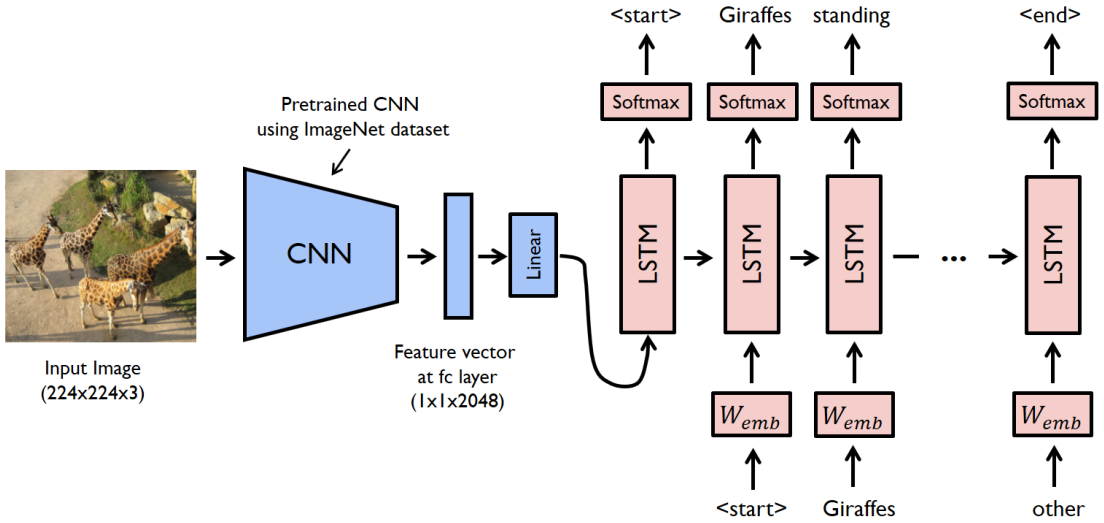
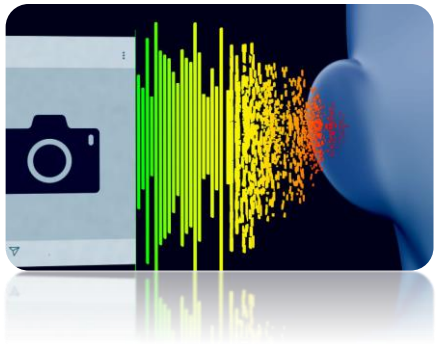


Image Captioning applications

- Generating captions for social media posts and websites
- Creating metadata for image search engines
- Improving the accessibility of images for people with visual impairments



Road map!

- ✓ Module 1- Introduction to Deep Learning
- ✓ Module 2- Setting up Deep Learning Environment
- ✓ Module 3- Machine Learning review (ML fundamentals + models)
- ✓ Module 4- Deep Neural Networks (NN and DNN)
- ✓ Module 5- Deep Computer Vision (CNN, R-CNN, YOLO, FCN)
- Module 6- Deep Sequence Modeling (RNN, LSTM)
- Module 7- Transformers (Attention is all you need!)
- Module 8- Deep Generative Modeling (AE, VAE, GAN)
- Module 9- Deep Reinforcement Learning (DQN, PG)