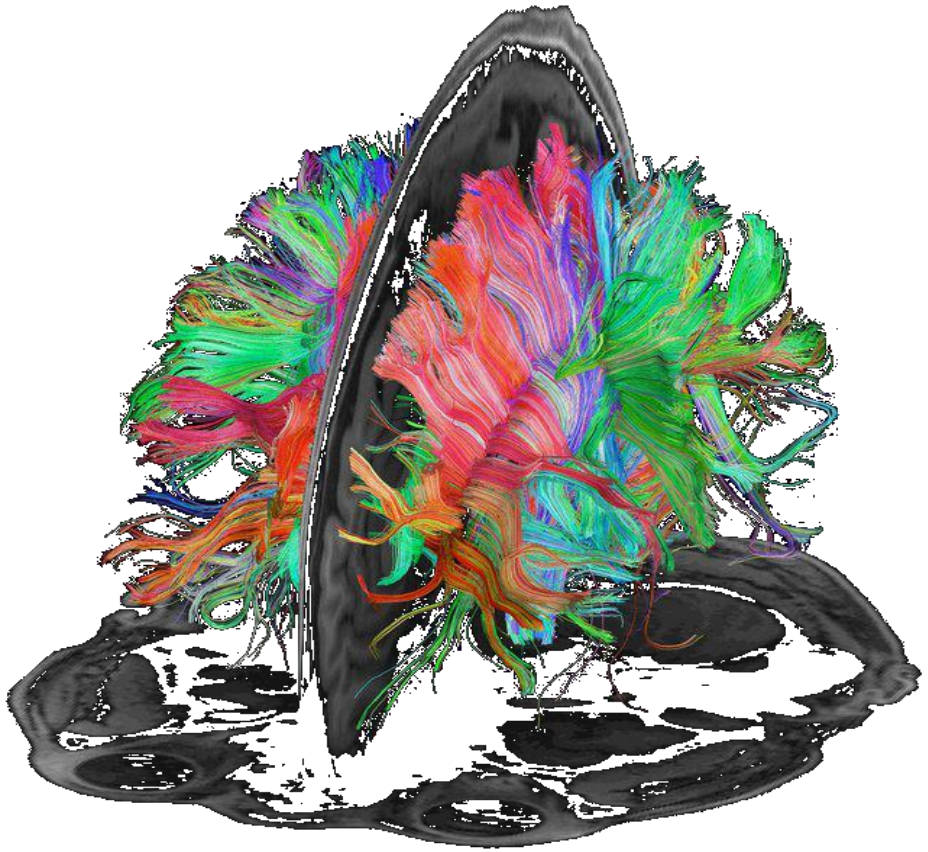
Deep Learning in Medical Image Analysis



Dr. Hichem Felouat
hichemfel@gmail.com
https://www.researchgate.net/profile/Hichem_Felouat
https://www.linkedin.com/in/hichemfelouat

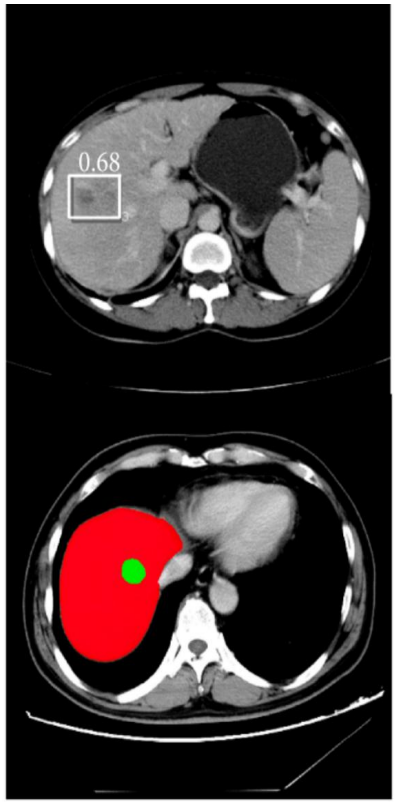
Medical Images

- Medical imaging is the technique and process of creating visual representations of the interior of a body for clinical analysis and medical intervention, as well as visual representation of the function of some organs or tissues.

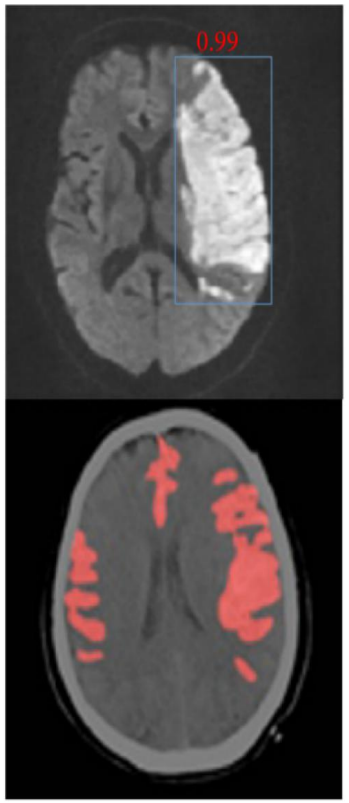
- Medical imaging seeks to reveal internal structures hidden by the skin and bones, as well as to diagnose and treat diseases.

Medical Image Modalities

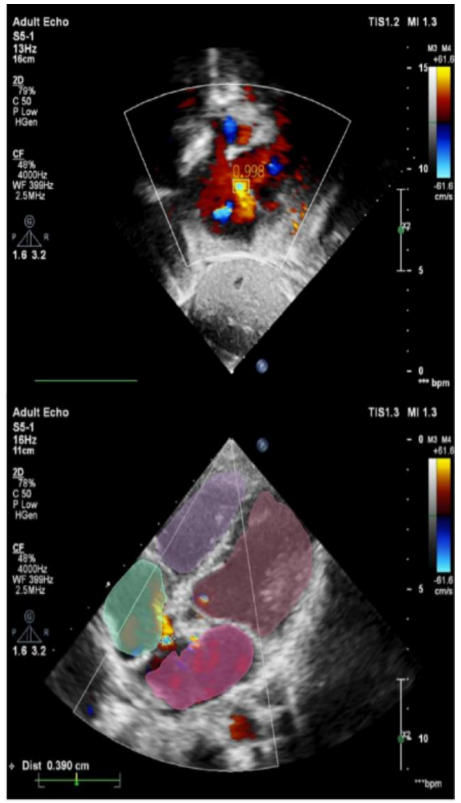
Bone X-ray



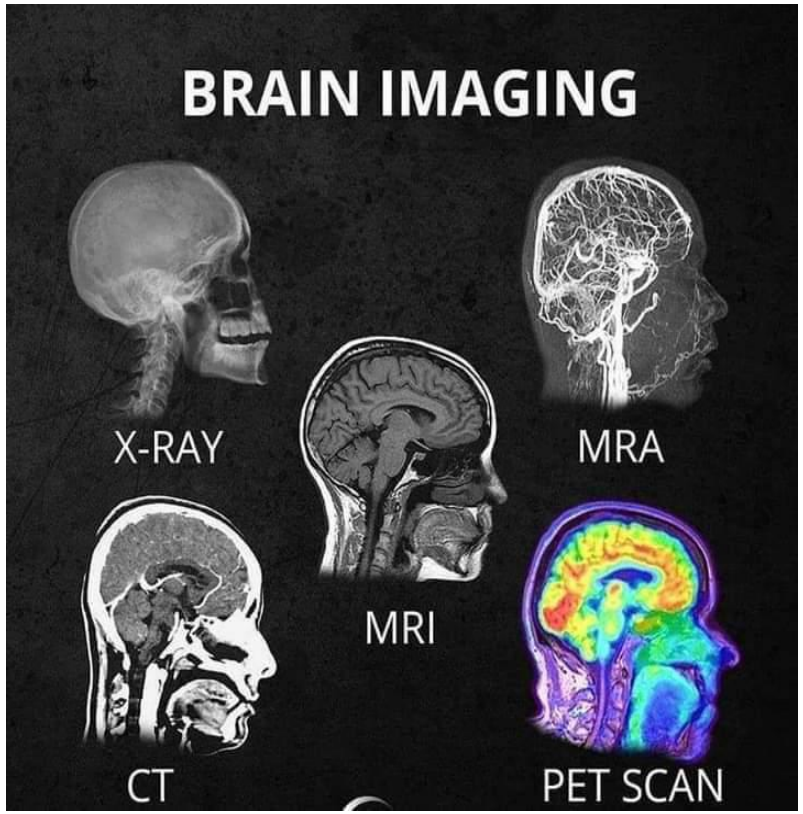
Liver CT



Brain MRI



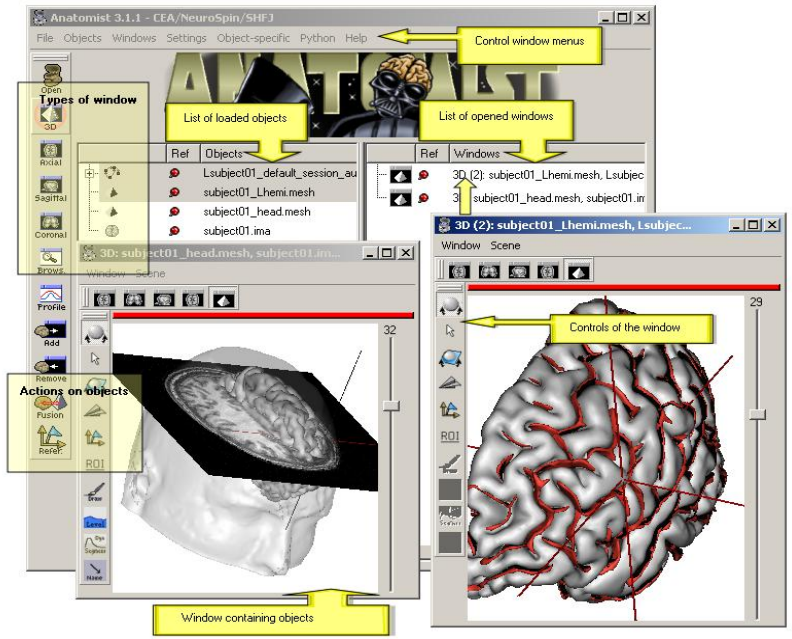
Cardiac ultrasound

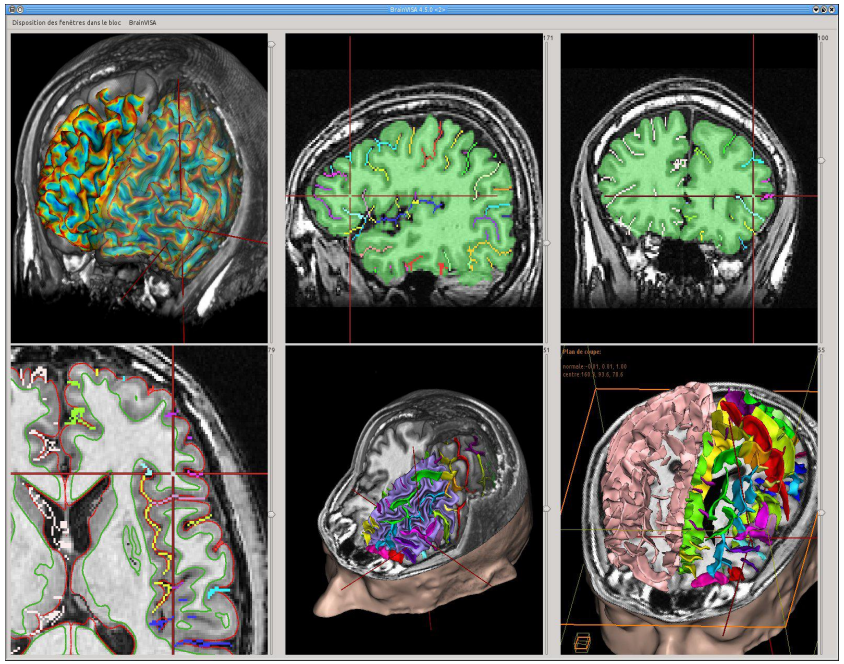


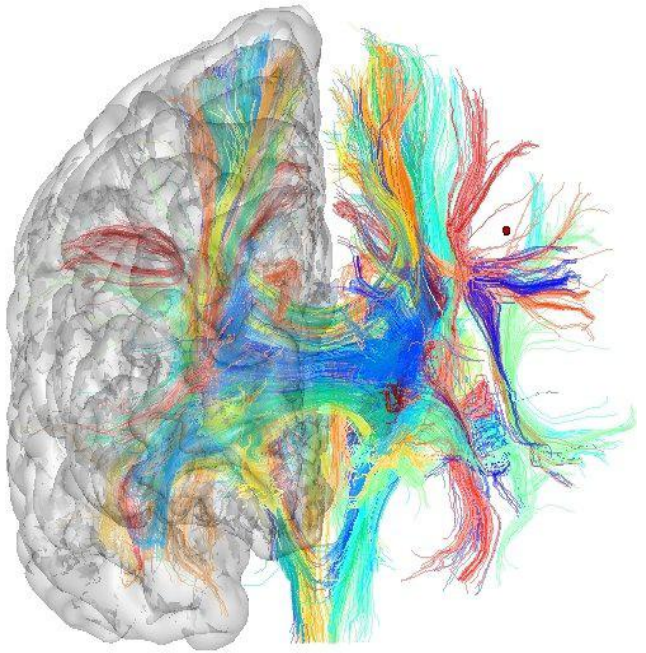
- **X-ray** radiography - **US:** Ultrasound - **MR/MRI/DMRI:** Magnetic Resonance Imaging - **PET:** Positron Emission Tomography - **MG:** Mammography - **CT:** Computed Tomography - **RGB:** Optical Images.

Medical Image Visualization

- Visualization is the process of **exploring**, **transforming**, and **viewing data** as images to gain understanding and insight into the data, which requires fast interactive speed and high image quality.

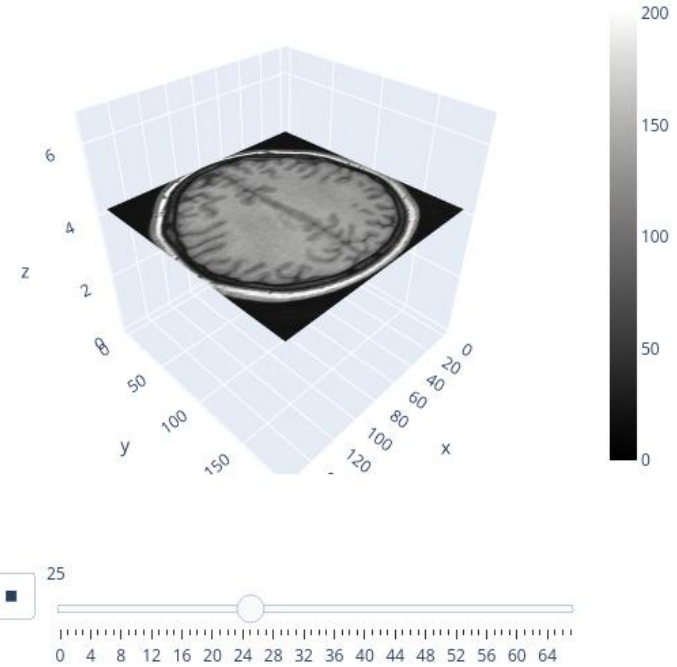




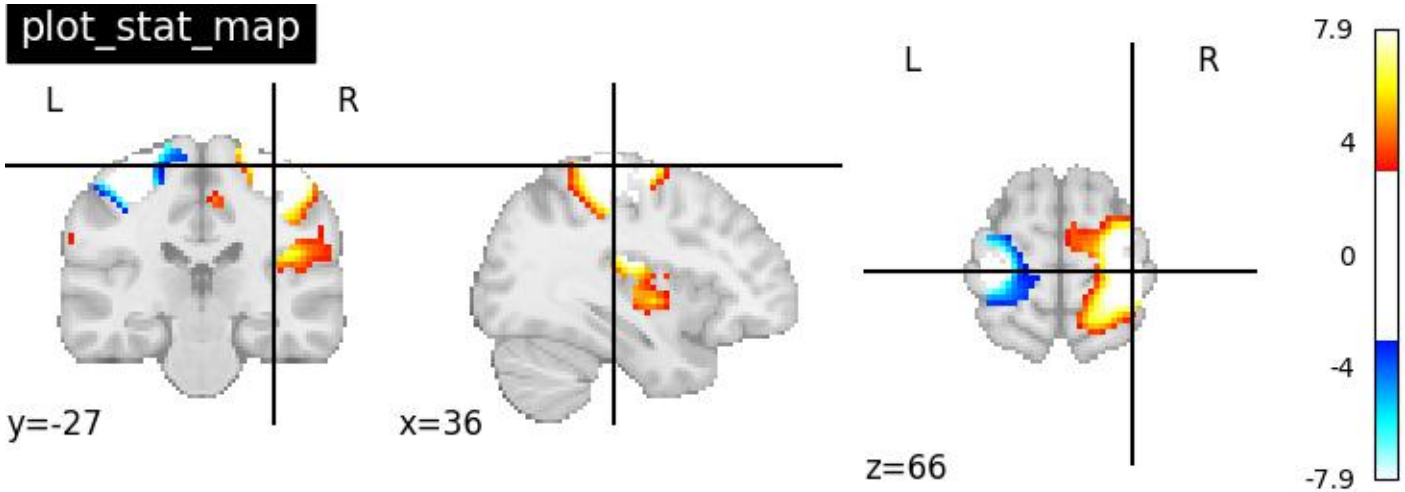


Anatomist :
https://brainvisa.info/web/

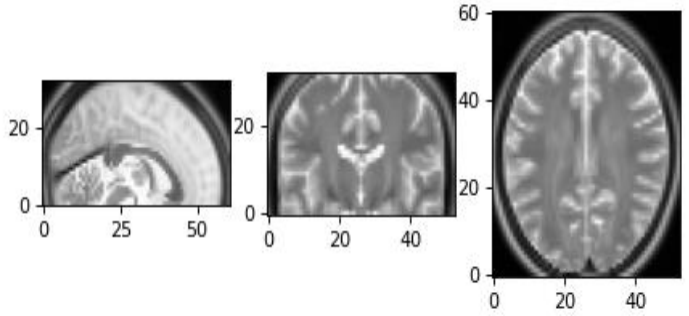
Medical Image Visualization



Plotly [1]



Nilearn [2]



NiBabel [3]

1) *https://plotly.com/python/visualizing-mri-volume-slices/*
2) *https://nilearn.github.io/stable/index.html*
3) *https://nipy.org/nibabel/coordinate_systems.html*

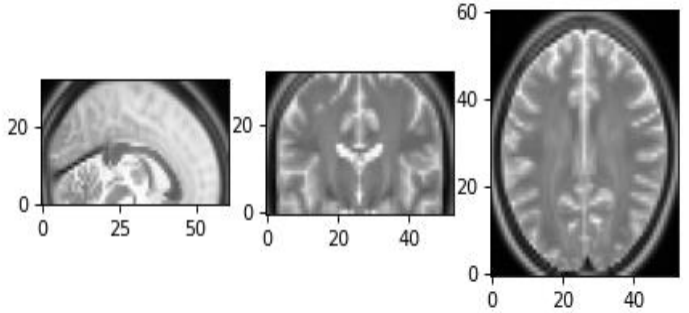
Medical Image Data I/O

We can load up the EPI image to get the image data array:

```
>>> import nibabel as nib
>>> epi_img = nib.load('downloads/someones_epi.nii.gz')
>>> epi_img_data = epi_img.get_fdata()
>>> epi_img_data.shape
(53, 61, 33)
```

Then we have a look at slices over the first, second and third dimensions of the array.

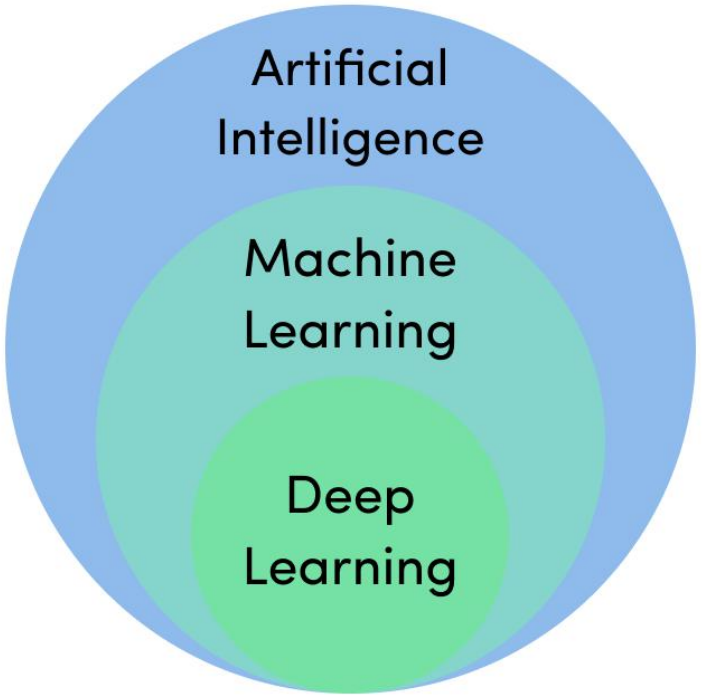
```
>>> import matplotlib.pyplot as plt
>>> def show_slices(slices):
...     """ Function to display row of image slices """
...     fig, axes = plt.subplots(1, len(slices))
...     for i, slice in enumerate(slices):
...         axes[i].imshow(slice.T, cmap="gray", origin="lower")
>>>
>>> slice_0 = epi_img_data[26, :, :]
>>> slice_1 = epi_img_data[:, 30, :]
>>> slice_2 = epi_img_data[:, :, 16]
>>> show_slices([slice_0, slice_1, slice_2])
>>> plt.suptitle("Center slices for EPI image")
```



https://nipy.org/nibabel/coordinate_systems.html

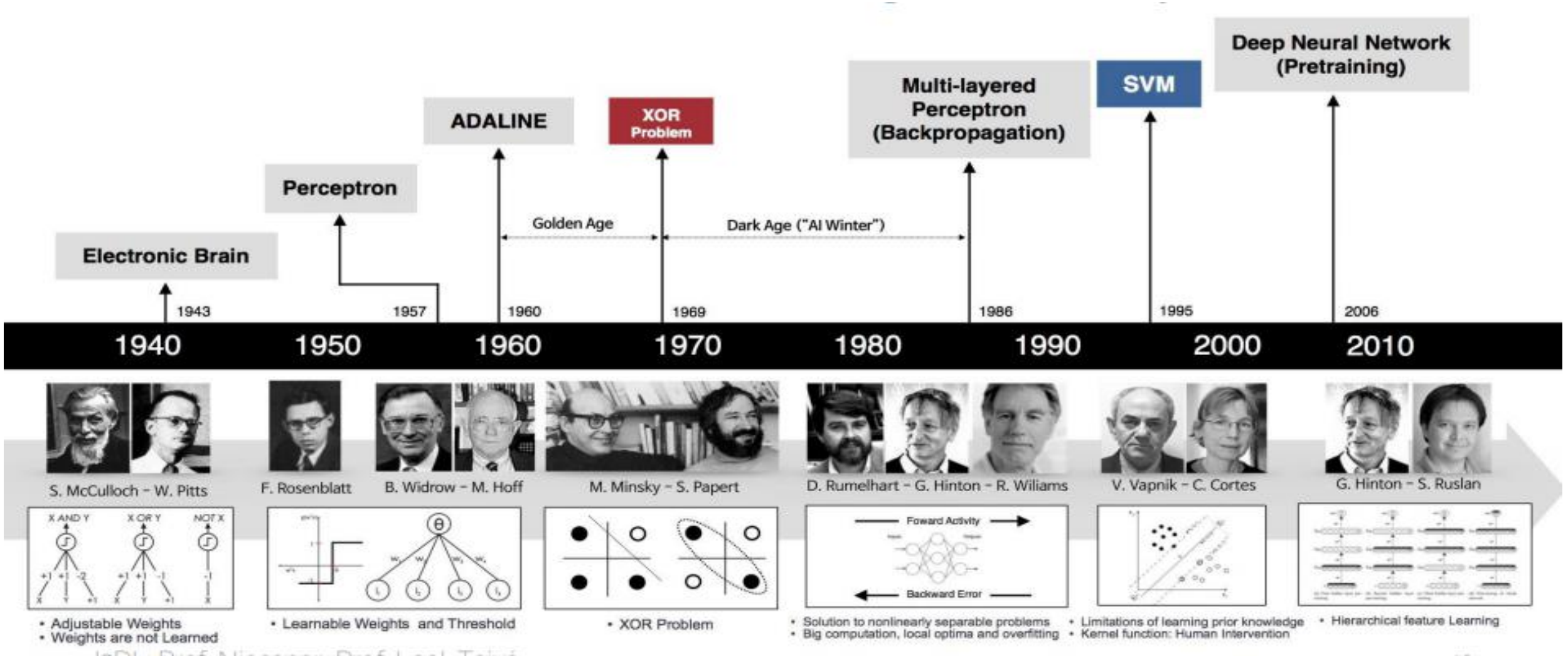
Deep Learning DL

DL is a subfield of ML, developed by several researchers.

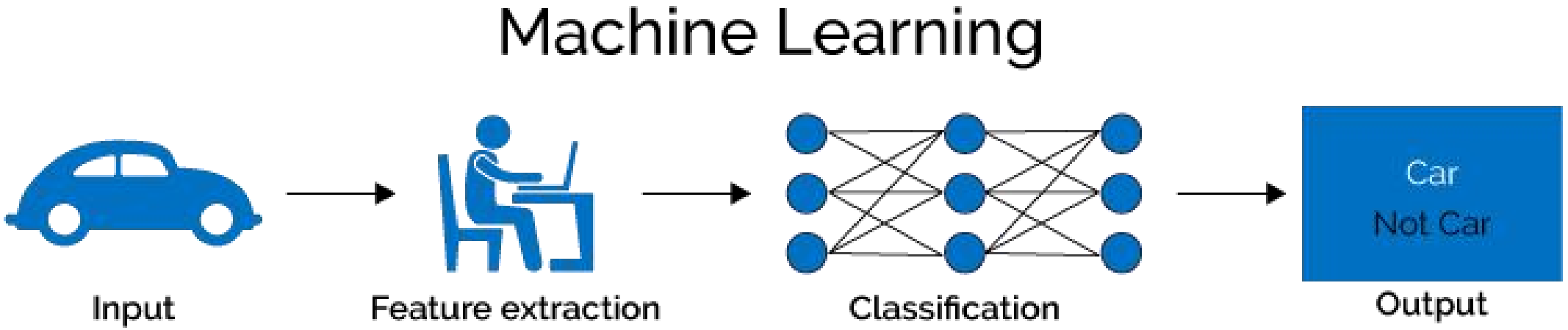


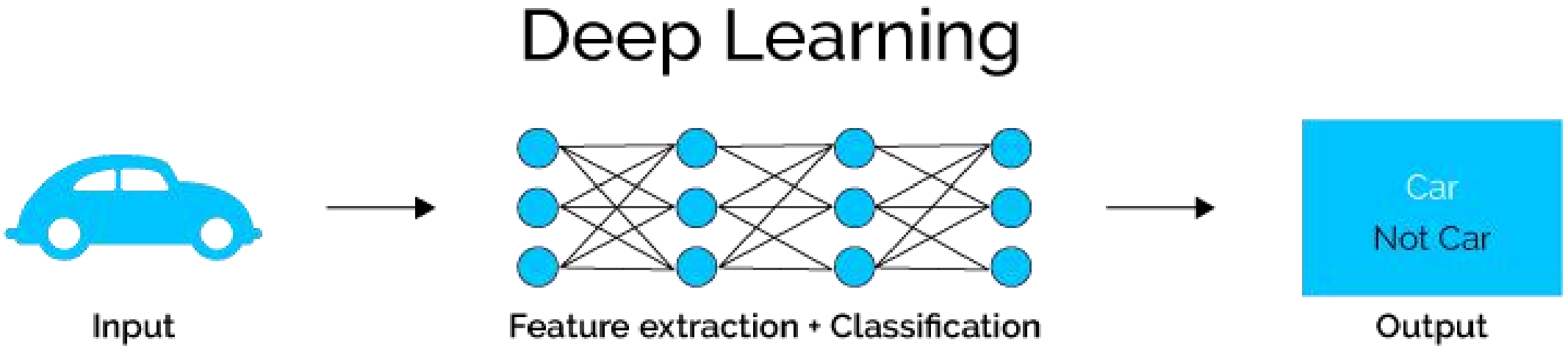
ACM Turing Award 2019 (Nobel Prize of Computing)
Yann LeCun, Geoffrey Hinton, and Yoshua Bengio

Deep Learning DL



Deep Learning DL



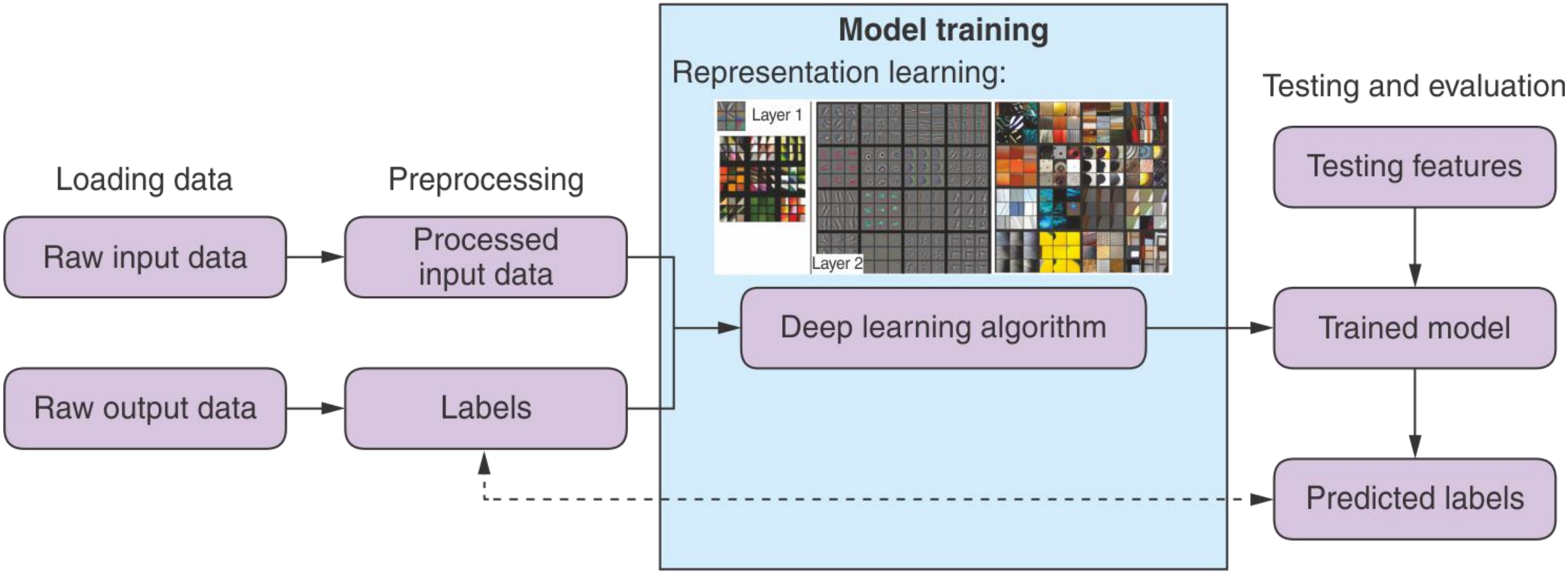


Deep Learning DL

Several DL models have been proposed :

- Convolutional neural networks (CNNs)
- Autoencoders (Aes)
- Recurrent neural networks (RNNs)
- Generative adversarial networks (GANs)
- Faster RCNN and Mask RCNN
- U-Net
- Vision Transformer (ViT)
- Graph Neural Networks (GNNs)

Deep Learning DL



Deep Learning DL

- In DL area, there are many different tasks: Image Classification, Regression, Object Localization, Object Detection, Instance Segmentation, Image captioning, etc..

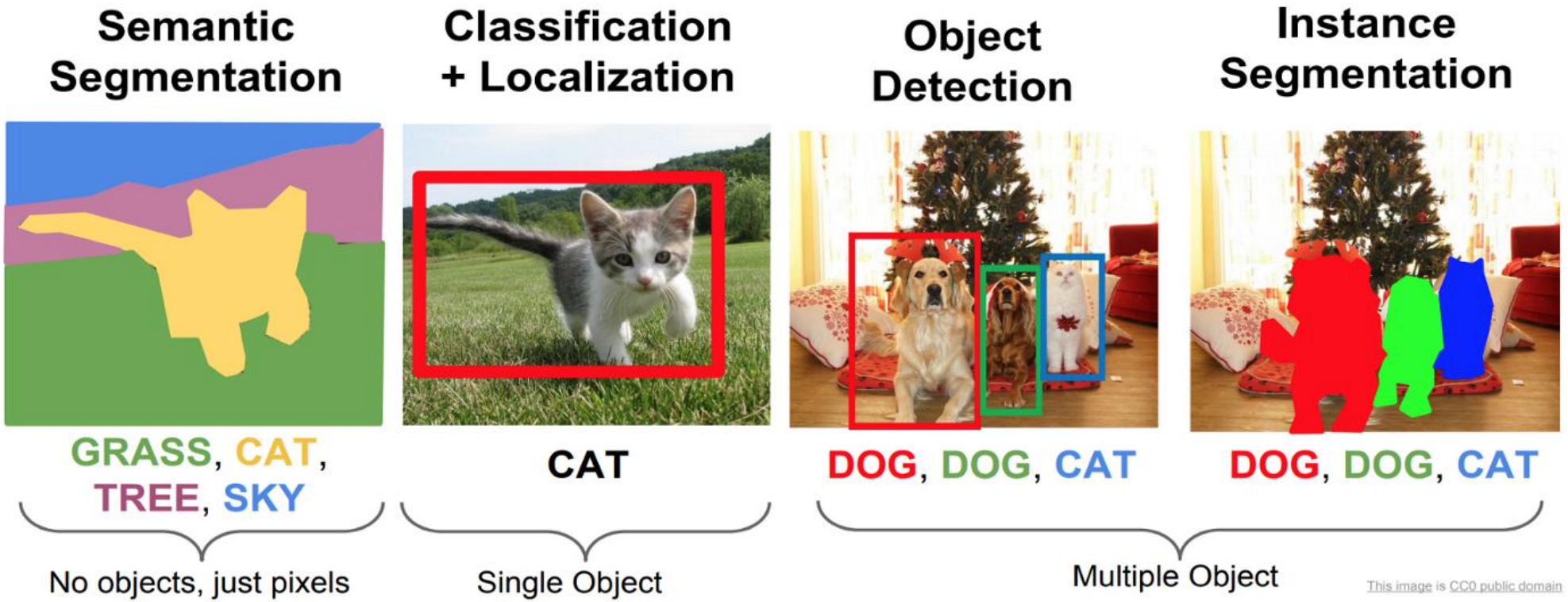


Image Classification for Medical Image Analysis

- Convolutional neural network (**CNN**) is the dominant classification framework for image analysis.

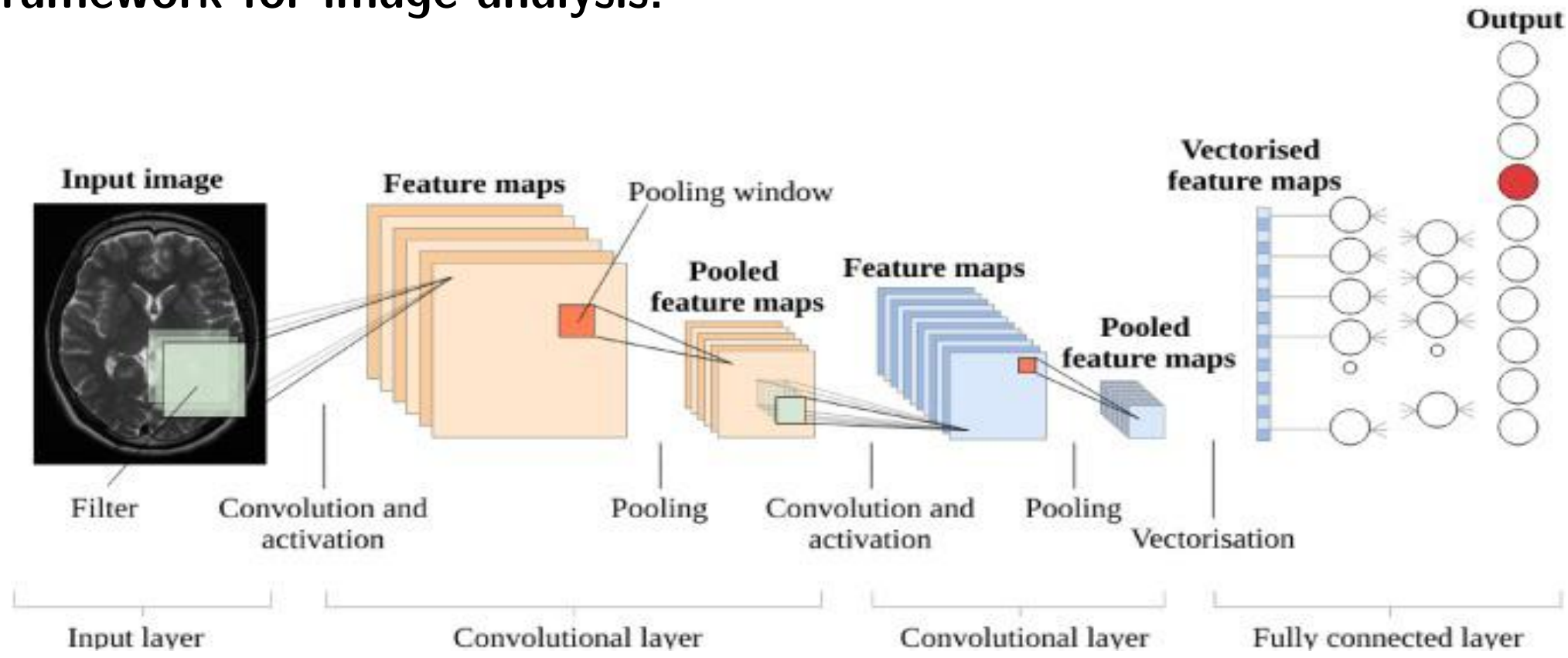
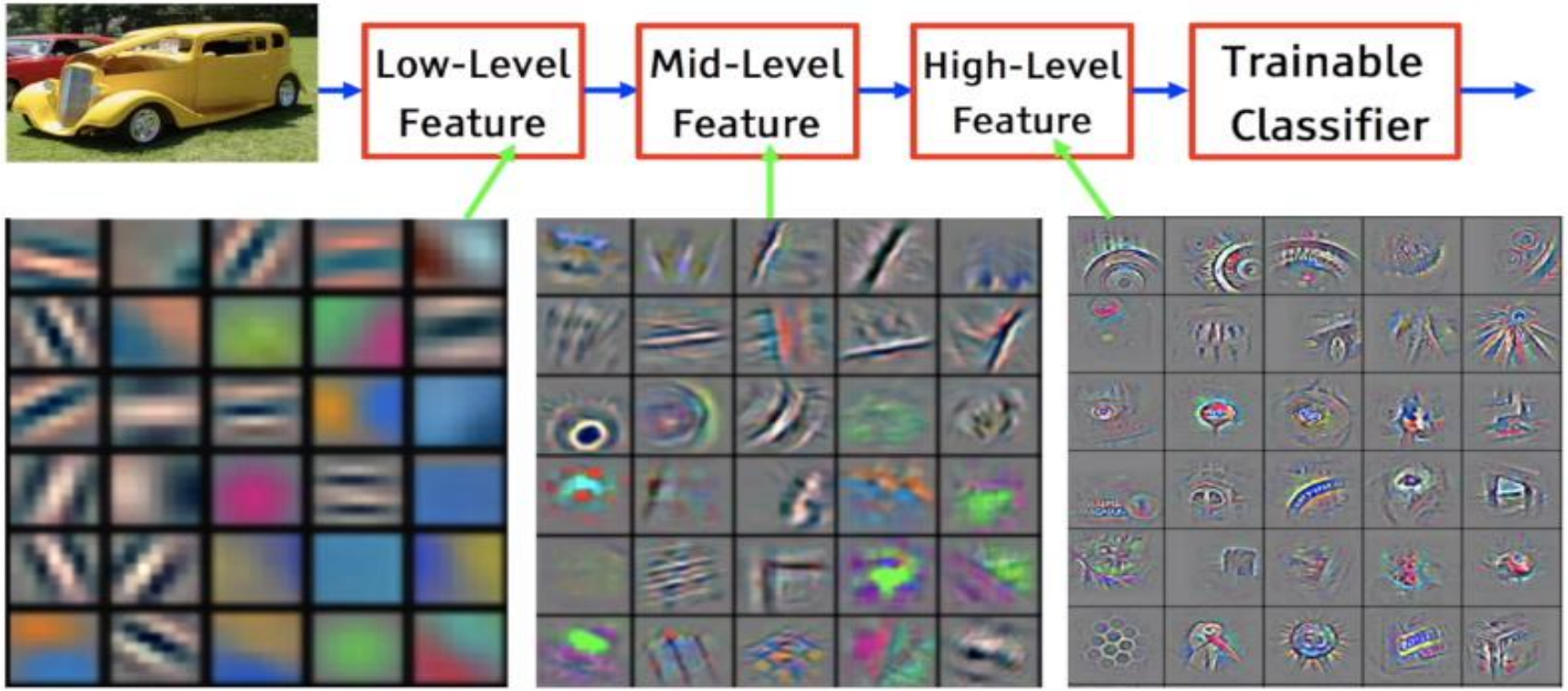


Image Classification for Medical Image Analysis - The eyes of CNN

- CNN is designed for working with **two-dimensional** image data, also they can be used with **one-dimensional** and **three-dimensional** data.



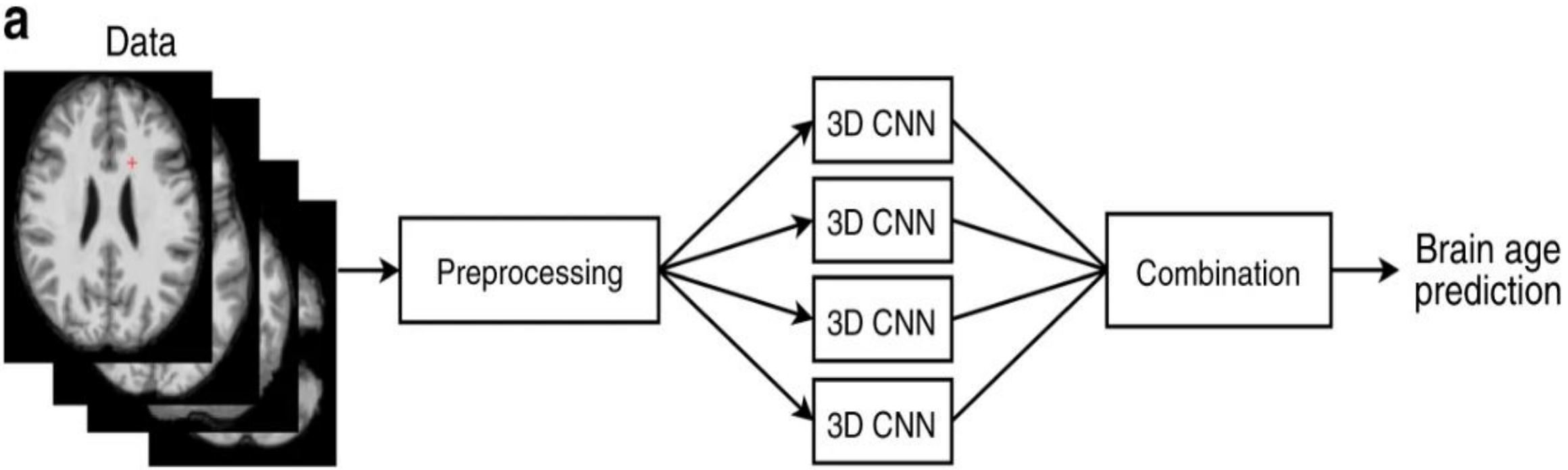
A simple 2D CNN

```
# Creating the model using the Sequential API
model = keras.models.Sequential()

model.add(keras.layers.Conv2D(filters=64, kernel_size=7, strides=1,
                              padding="same", activation="relu", name="Conv1", input_shape= (256,256,1)))
model.add(keras.layers.MaxPool2D(pool_size=2, name="Pool1"))
model.add(keras.layers.Conv2D(filters=128, kernel_size=5, strides=1,
                                          padding="same", activation="relu", name="Conv2"))
model.add(keras.layers.MaxPool2D(pool_size=2, name="Pool2"))
model.add(keras.layers.Conv2D(filters=256, kernel_size=3, strides=1,
                                          padding="same", activation="relu", name="Conv3"))
model.add(keras.layers.MaxPool2D(pool_size=2, name="Pool3"))

model.add(keras.layers.Flatten(name="Flatten1"))
model.add(keras.layers.Dense(128, activation="relu", name="Dense1"))
model.add(keras.layers.Dense(64, activation="relu", name="Dense2"))
model.add(keras.layers.Dense(1, activation="sigmoid", name="Output"))

# The model's summary() method displays all the model's layers
print(model.summary())

```

A simple 3D CNN

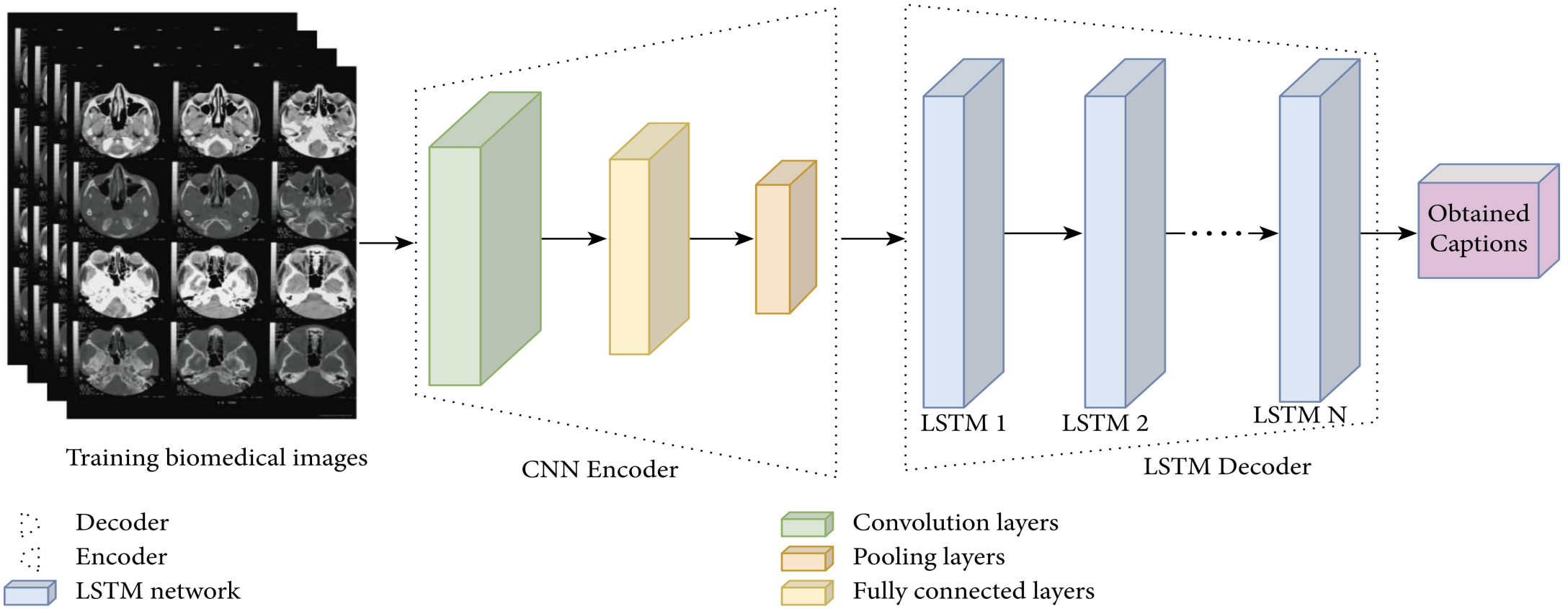
```
# Creating the model using the Sequential API
model = keras.models.Sequential()
model.add(keras.layers.Conv3D(filters=64, kernel_size=7, strides=1,
                              padding="same", activation="relu", input_shape= (128,128,64,3)))
model.add(keras.layers.MaxPool3D(pool_size=2))
model.add(keras.layers.Conv3D(filters=128, kernel_size=5, strides=1,
                              padding="same", activation="relu"))
model.add(keras.layers.MaxPool3D(pool_size=2))
model.add(keras.layers.Conv3D(filters=256, kernel_size=3, strides=1,
                              padding="same", activation="relu"))
model.add(keras.layers.MaxPool3D(pool_size=2))

model.add(keras.layers.Flatten())
model.add(keras.layers.Dense(512, activation="relu"))
model.add(keras.layers.Dense(128, activation="relu"))
model.add(keras.layers.Dense(1,   activation="sigmoid"))

# The model's summary() method displays all the model's layers
print(model.summary())

```

Image Regression for Medical Image Analysis



Brain age prediction using deep learning :
https://www.nature.com/articles/s41467-019-13163-9

Medical Image Captioning Using DL



Medical Image Captioning Using Optimized Deep Learning Model :
https://www.hindawi.com/journals/cin/2022/9638438/

Transfer Learning TL

- Transfer learning is a machine learning method where a model developed for a task is reused as the starting point for a model on a second task.

- The intuition behind transfer learning for image classification is that if a model is trained on a large and general enough dataset, this model will effectively serve as a generic model of the visual world. You can then take advantage of these learned feature maps without having to start from scratch by training a large model on a large dataset.

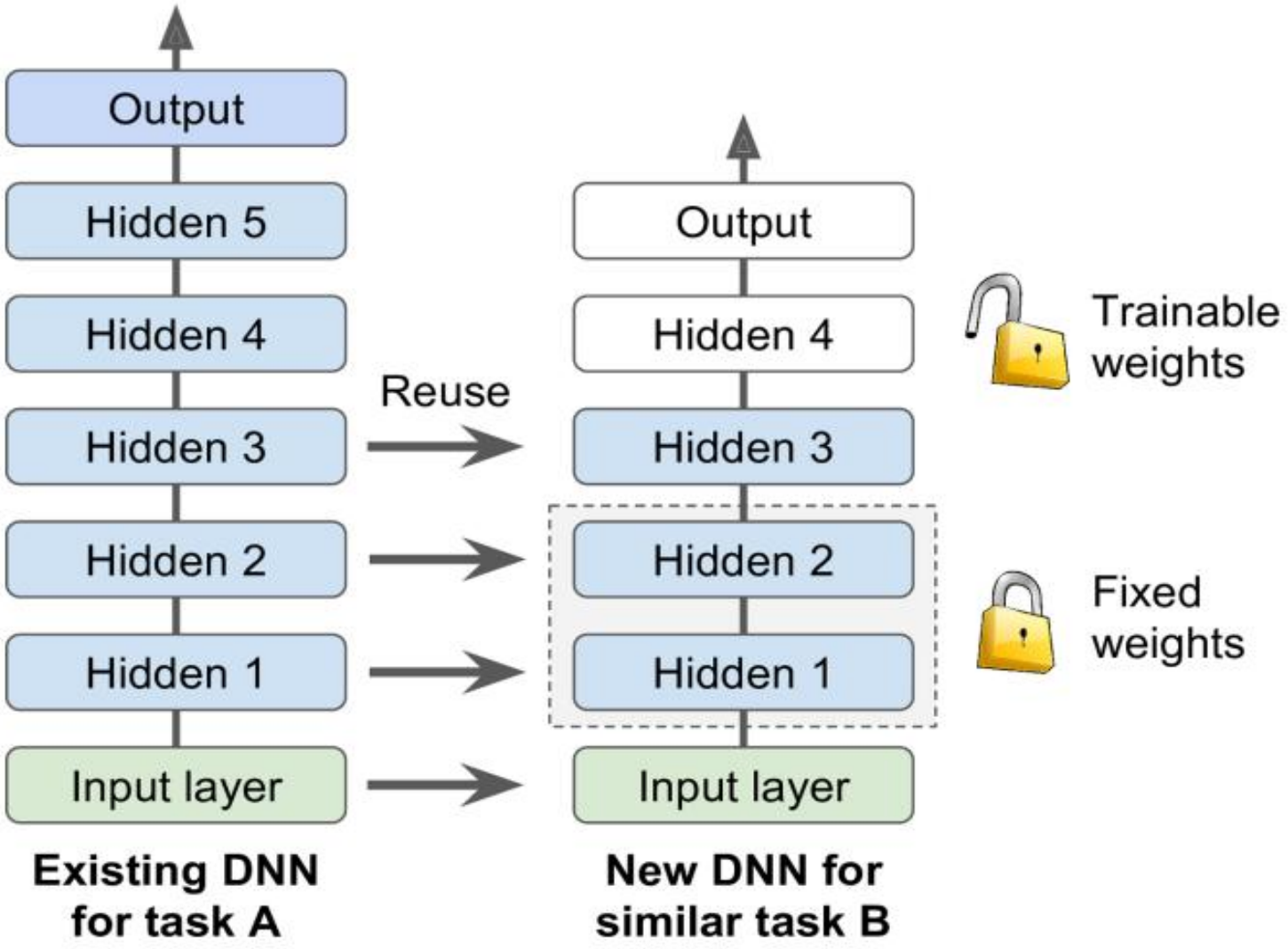
Transfer Learning TL

- It is generally **not a good idea** to train a **very large DNN from scratch**: instead, you should always try to find an existing neural network that accomplishes a similar task to the one you are trying to tackle then **reuse the lower layers** of this network.

- It will not only **speed up training** considerably but also require significantly **less training data**.

- The output layer of the original model **should usually be replaced** because it is most likely not useful at all for the new task, and it may not even have the right number of outputs for the new task.

Transfer Learning TL



Transfer Learning TL

Available models in Keras:

Models for image classification with weights trained on ImageNet:

Available models

Model	Size	Top-1 Accuracy	Top-5 Accuracy	Parameters	Depth
Xception	88 MB	0.790	0.945	22,910,480	126
VGG16	528 MB	0.713	0.901	138,357,544	23
VGG19	549 MB	0.713	0.900	143,667,240	26
ResNet50	98 MB	0.749	0.921	25,636,712	-
ResNet101	171 MB	0.764	0.928	44,707,176	-
ResNet152	232 MB	0.766	0.931	60,419,944	-
ResNet50V2	98 MB	0.760	0.930	25,613,800	-
ResNet101V2	171 MB	0.772	0.938	44,675,560	-
ResNet152V2	232 MB	0.780	0.942	60,380,648	-
InceptionV3	92 MB	0.779	0.937	23,851,784	159
InceptionResNetV2	215 MB	0.803	0.953	55,873,736	572
MobileNet	16 MB	0.704	0.895	4,253,864	88
MobileNetV2	14 MB	0.713	0.901	3,538,984	88

https://keras.io/applications/

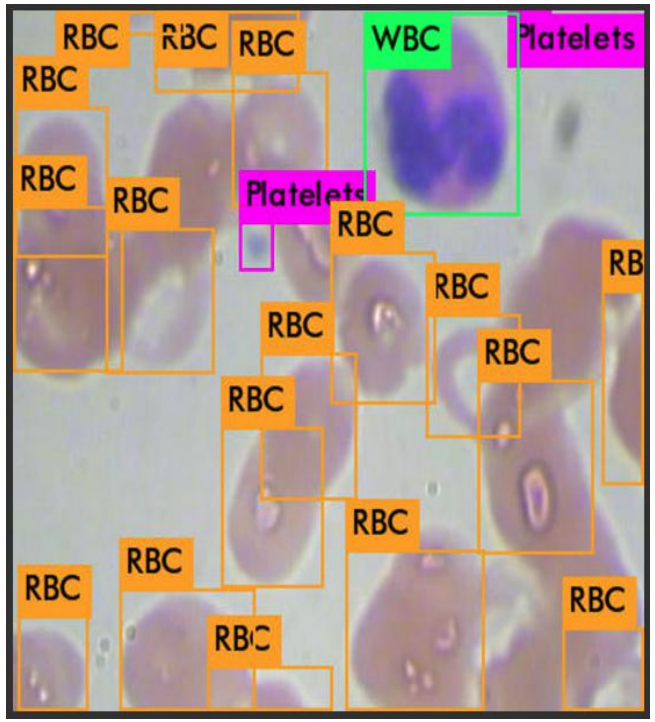
Transfer Learning TL

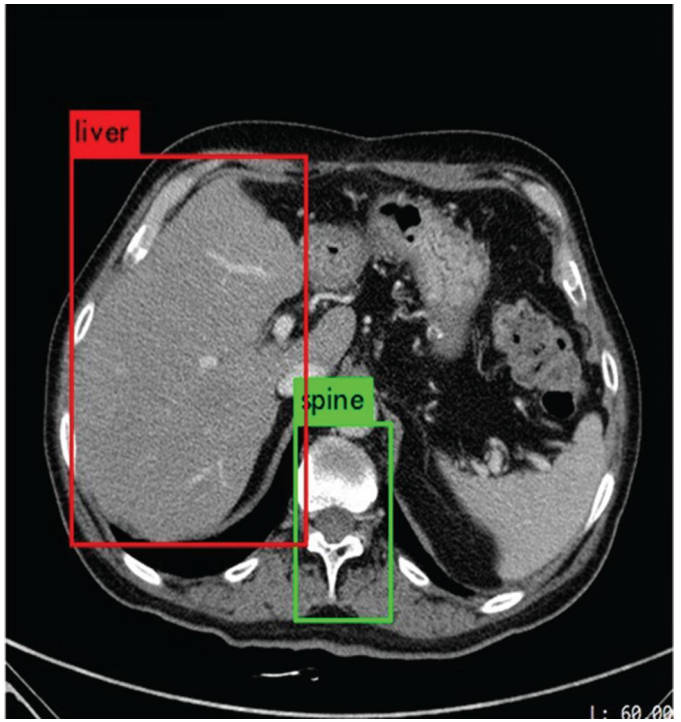
```
base_model = keras.applications.xception.Xception(weights="imagenet", include_top=False)

avg = keras.layers.GlobalAveragePooling2D()(base_model.output)
output = keras.layers.Dense(n_classes, activation="softmax")(avg)
model = keras.Model(inputs=base_model.input, outputs=output)

for layer in model.layers[:-2]:
    layer.trainable = False

optimizer = keras.optimizers.SGD(lr=0.2, momentum=0.9, decay=0.01)
model.compile(loss="sparse_categorical_crossentropy", optimizer=optimizer, metrics=["accuracy"])
history = model.fit(train_set, epochs=5, validation_data=valid_set)

for layer in model.layers[-5:]:
    layer.trainable = True

optimizer = keras.optimizers.SGD(lr=0.01, momentum=0.9, decay=0.001)
model.compile(...)
history = model.fit(...)
```

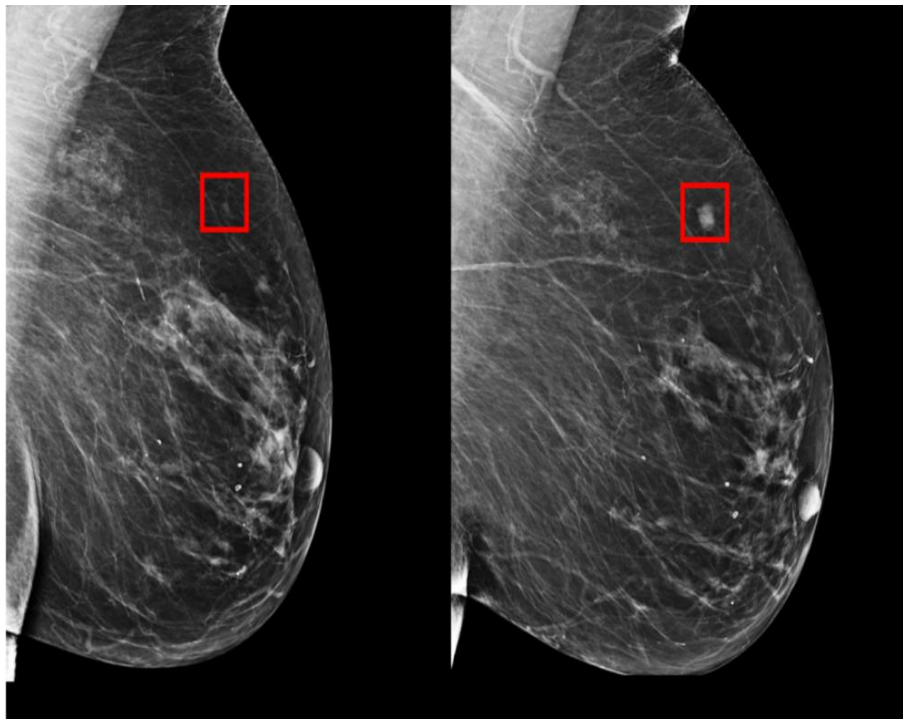
The two new layers

Object Detection

- Localizing an object in a picture means predicting a **bounding box** around the object and can be expressed as a **regression** task.







Object Detection

Problem: the dataset does not have bounding boxes around the objects, how can we train our model?

- We need to **add them ourselves**. This is often one of the **hardest and most costly parts** of a Machine Learning project: **getting the labels**.

- It is a good idea to spend time looking for the **right tools**.

Object Detection

- An **image labeling** or **annotation tool** is used to label the images for bounding box object detection and segmentation.

Open-source image labeling tool like :

- VGG Image
- Annotator
- LabelImg
- OpenLabeler
- ImgLab

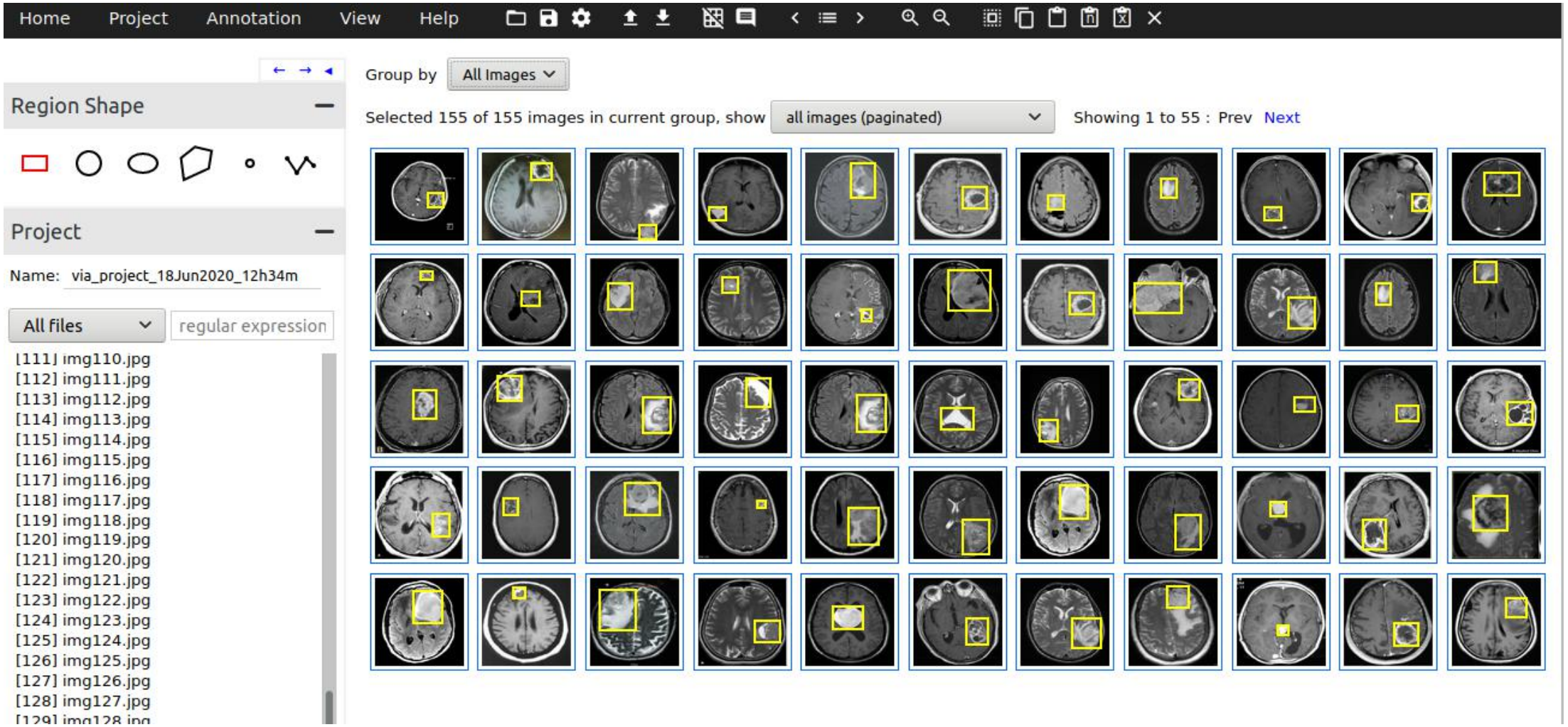
Commercial tool like :

- LabelBox
- Supervisely

Crowdsourcing Platform like :

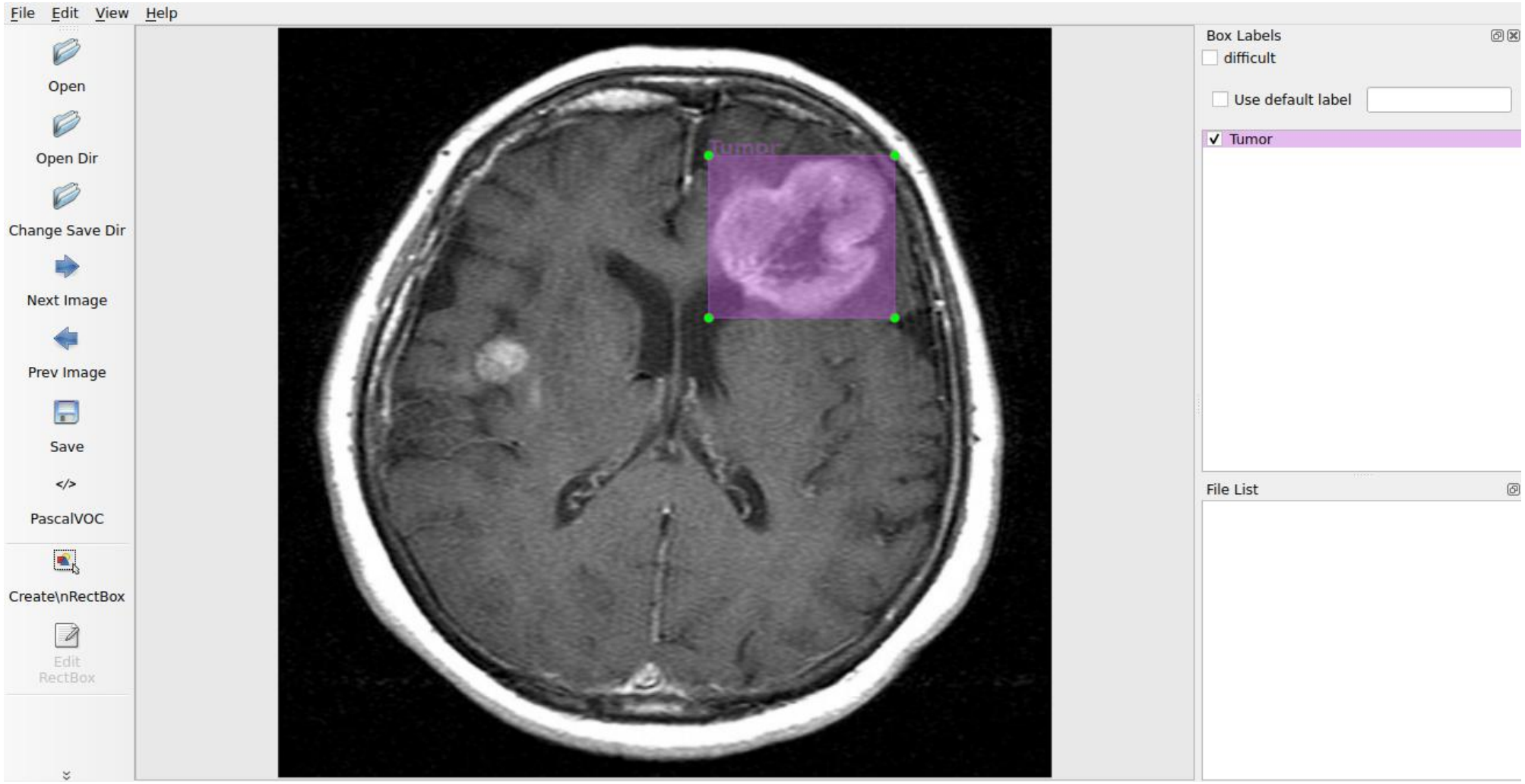
- Amazon Mechanical Turk

Object Detection - VGG



VGG Image: *http://www.robots.ox.ac.uk/~vgg/software/via/*

Object Detection - labelImg

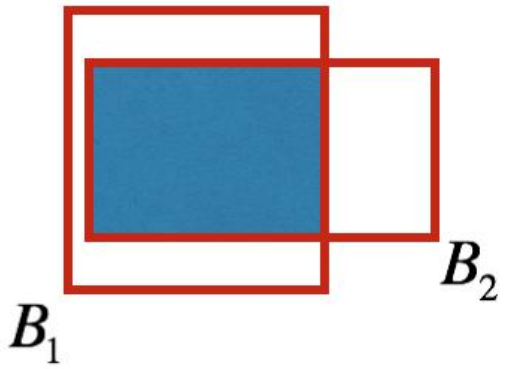


Object Detection

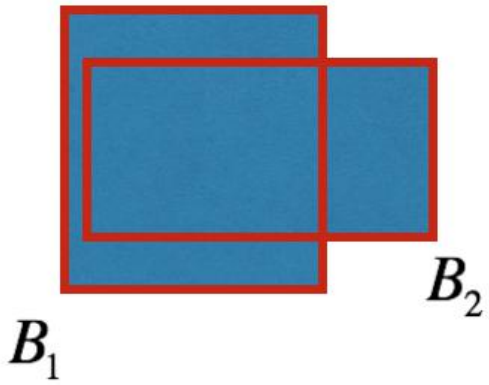
- The **MSE** often works fairly well as a **cost function** to train the model, but it is not a great metric to evaluate how well the model can predict bounding boxes.

- The most common metric for this is **the Intersection over Union (IoU)**.

Intersection

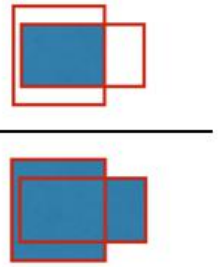


Union



Intersection over Union

$$IoU = \frac{B_1 \cap B_2}{B_1 \cup B_2} =$$



Object Detection

mean Average Precision :

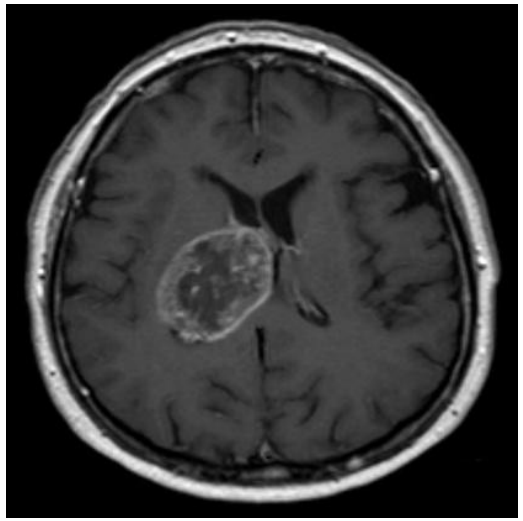
In order to calculate **mAP**, we draw a series of **precision-recall curves** with the **IoU threshold** set at varying levels of difficulty. In COCO evaluation, the IoU threshold ranges from **0.5 to 0.95** with a step size of **0.05** represented as **AP@[.5:.05:.95]**

$$\text{AP[class]} = \frac{1}{\#\text{thresolds}} \sum_{\text{iou} \in \text{thresholds}} AP[class, iou]$$

- **Calculate the final AP by averaging the AP over different classes.**

$$\text{AP} = \frac{1}{\#\text{classes}} \sum_{\text{class} \in \text{classes}} AP[class] \quad = \quad \textbf{mAP-IoU}_{\textbf{thresholds}}$$

Object Detection



raw image
must have the same
size

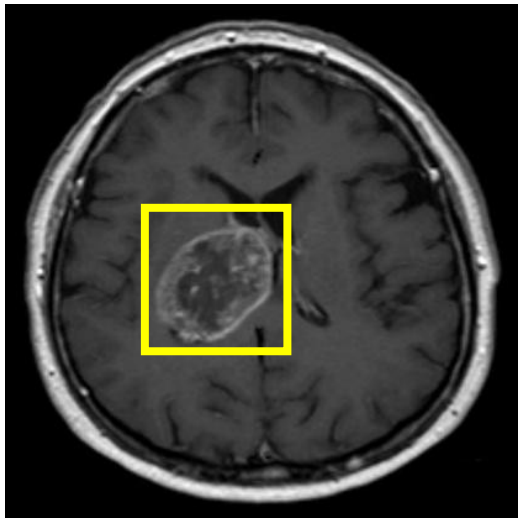
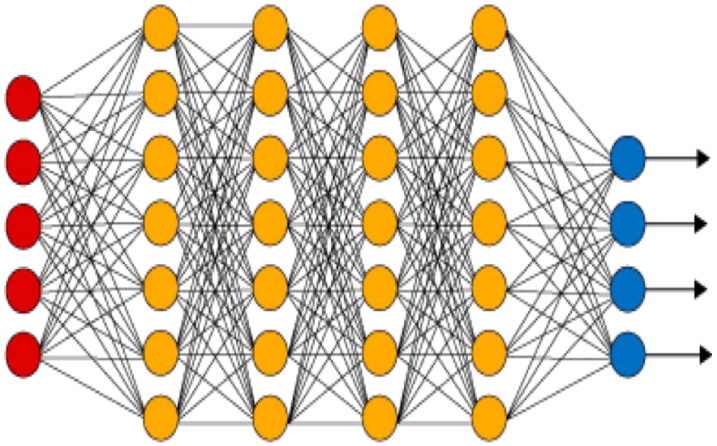
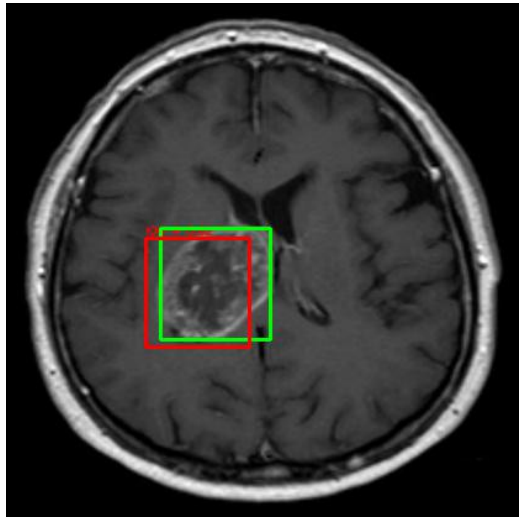


image labeling
(C, X,Y, W, H)



model

(C, X,Y, W, H)



- Each item should be a tuple of the form :

(images,
(class_labels, bounding_boxes))

Object Detection

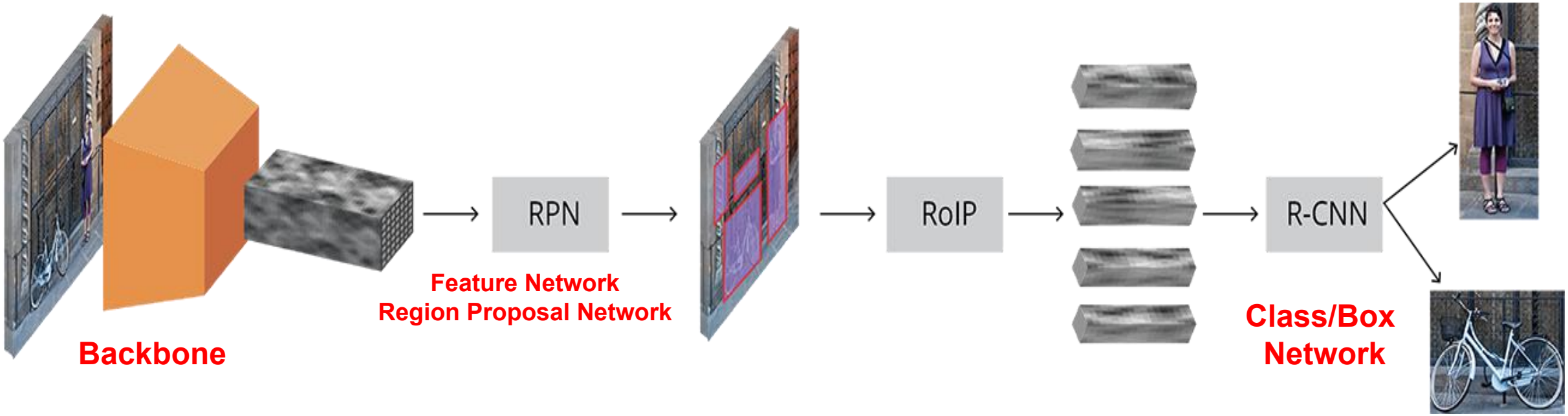
In general, object detectors have three (3) main components:

1) The **backbone** that **extracts features** from the given image.

2) The **feature network** that takes multiple levels of features from the backbone as input and **outputs a list of fused features** that represent salient characteristics of the image.

3) The final **class/box network** that uses the fused features to **predict the class and location** of each object.

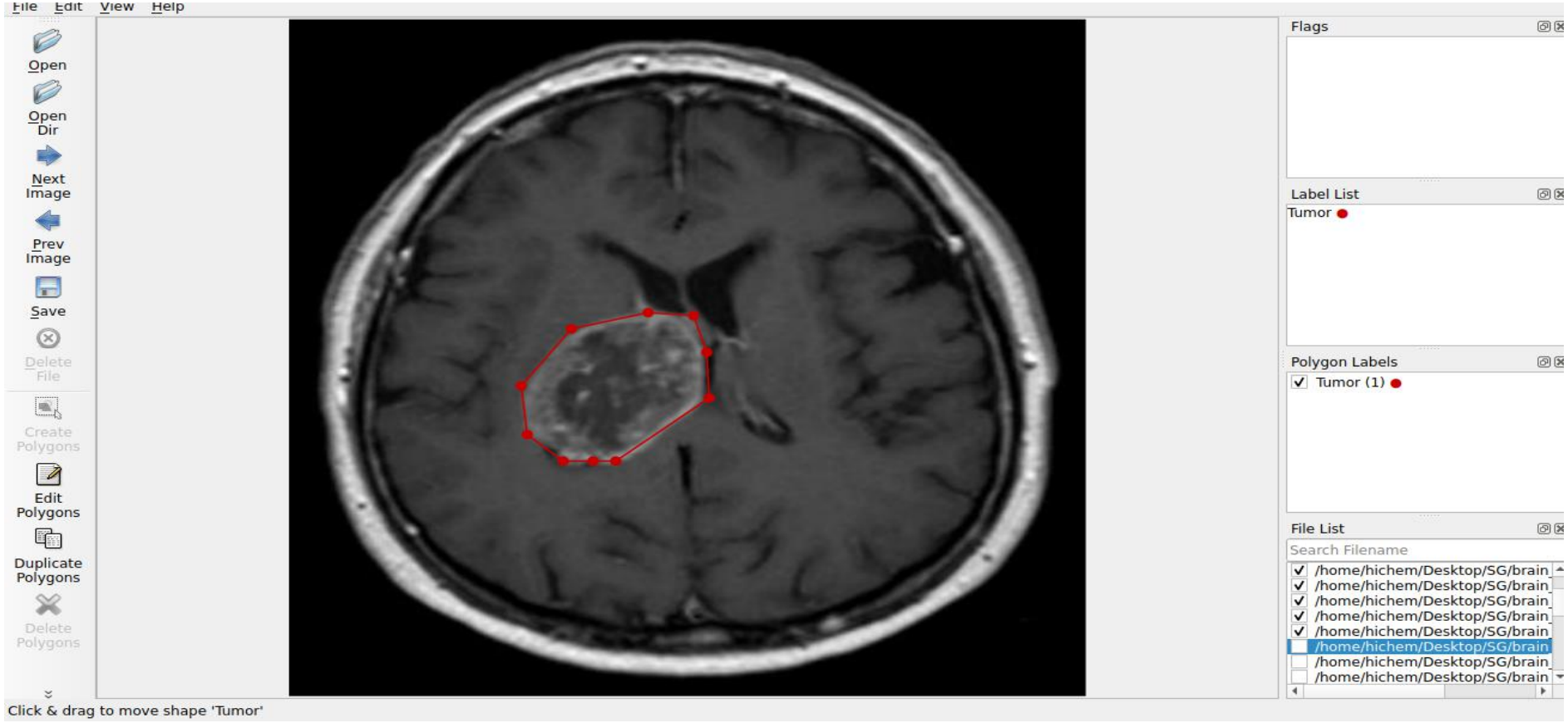
Object Detection - Faster RCNN



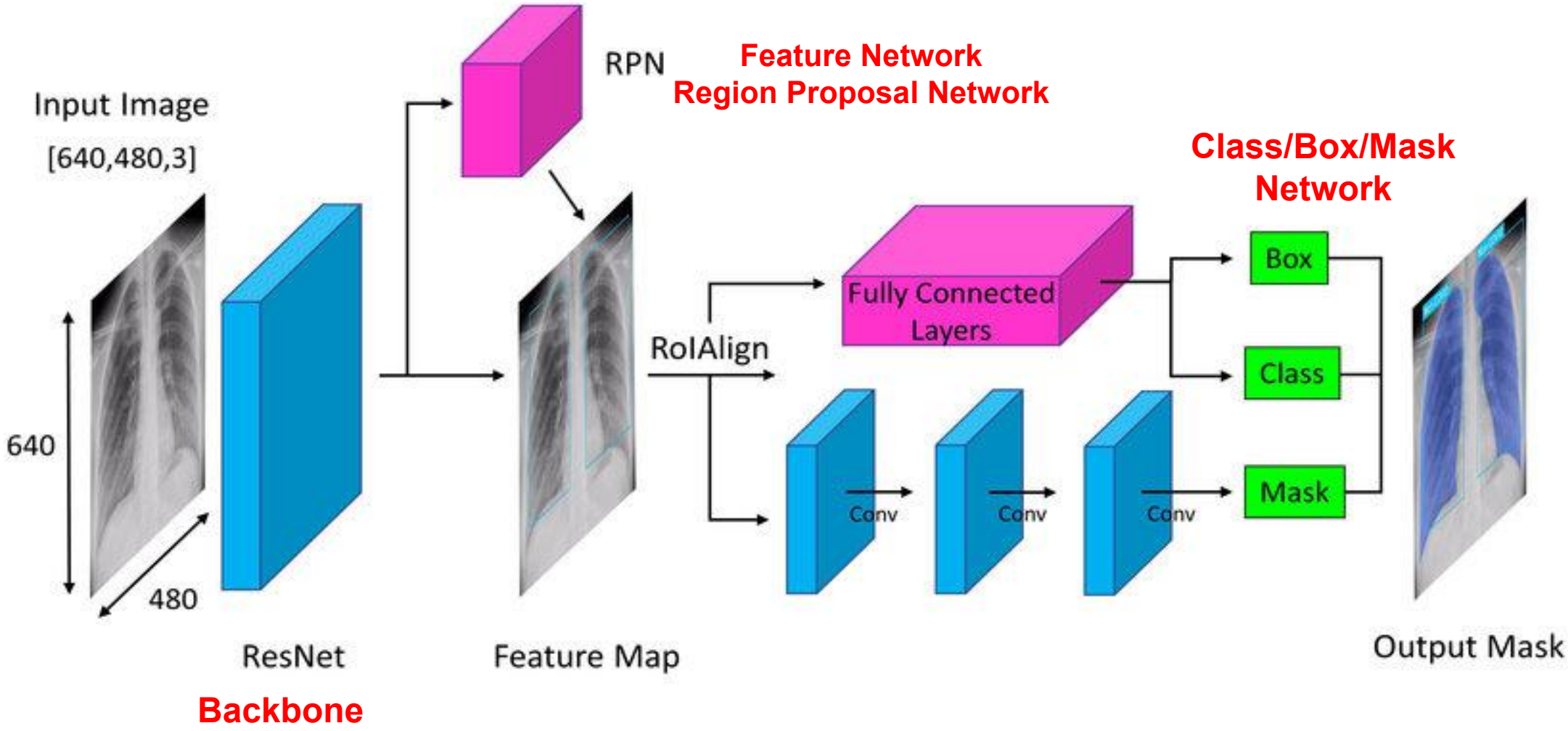
Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks
https://arxiv.org/abs/1506.01497

Instance Segmentation - labelme

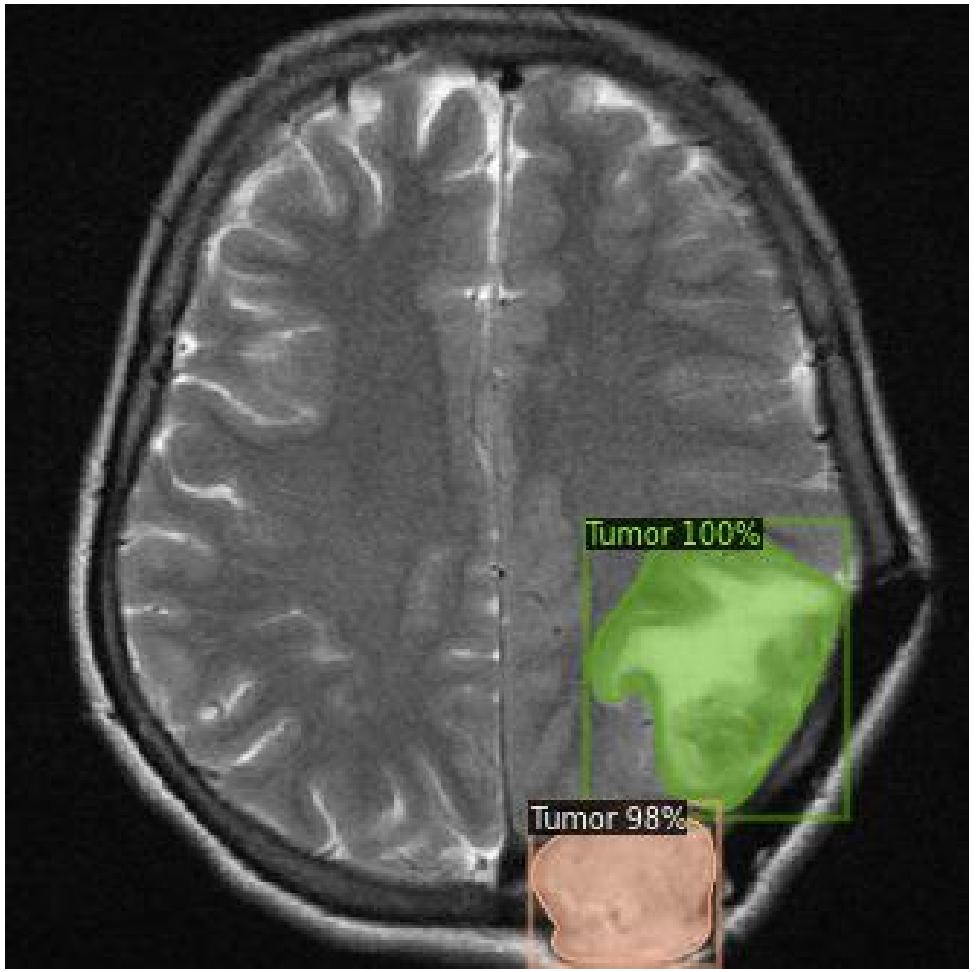
- **Instance Segmentation** aims to predicting the object class-label and the pixel-specific object instance-mask.



Instance Segmentation - Mask R-CNN



Instance Segmentation - Mask R-CNN



https://github.com/hichemfelouat/my-codes-of-machine-learning/blob/master/Mask_RCNN_TF2OD_Custom_dataset.ipynb

YOLO

- **You Only Look Once (YOLO)** is an algorithm that uses convolutional neural networks for object detection.

- It is one of **the faster object detection** algorithms out there.

- It is a very good choice when we need **real-time detection**, without loss of too much accuracy.

YOLOV5: https://github.com/ultralytics/yolov5
How to Train YOLOv5 On a Custom Dataset :
https://blog.roboflow.com/how-to-train-yolov5-on-a-custom-dataset/

Detectron2

- Detectron2 was built by Facebook AI Research (FAIR) to support rapid implementation and evaluation of novel computer vision research.
- Detectron2 is now implemented in PyTorch.
- Detectron2 is flexible and extensible, and able to provide fast training on single or multiple GPU servers.
- Detectron2 can be used as a library to support different projects on top of it.

Detectron2: A PyTorch-based modular object detection library
https://ai.facebook.com/blog/-detectron2-a-pytorch-based-modular-object-detection-library-/
Detectron2 :
https://github.com/facebookresearch/detectron2?fbclid=IwAR2CdXQoTU9i-ebKPZIc7BQw8R6NKgp0B-yUkGr1BF3w1VKWzNhxFHi6Zbw
detectron2's documentation: https://detectron2.readthedocs.io/

Detectron2

COCO Object Detection Baselines

Faster R-CNN:

Name	lr sched	train time (s/iter)	inference time (s/im)	train mem (GB)	box AP	model id	download
R50-C4	1x	0.551	0.102	4.8	35.7	137257644	model | metrics
R50-DC5	1x	0.380	0.068	5.0	37.3	137847829	model | metrics
R50-FPN	1x	0.210	0.038	3.0	37.9	137257794	model | metrics
R50-C4	3x	0.543	0.104	4.8	38.4	137849393	model | metrics
R50-DC5	3x	0.378	0.070	5.0	39.0	137849425	model | metrics
R50-FPN	3x	0.209	0.038	3.0	40.2	137849458	model | metrics
R101-C4	3x	0.619	0.139	5.9	41.1	138204752	model | metrics
R101-DC5	3x	0.452	0.086	6.1	40.6	138204841	model | metrics
R101-FPN	3x	0.286	0.051	4.1	42.0	137851257	model | metrics
X101-FPN	3x	0.638	0.098	6.7	43.0	139173657	model | metrics

COCO Instance Segmentation Baselines with Mask R-CNN

Name	lr sched	train time (s/iter)	inference time (s/im)	train mem (GB)	box AP	mask AP	model id	download
R50-C4	1x	0.584	0.110	5.2	36.8	32.2	137259246	model | metrics
R50-DC5	1x	0.471	0.076	6.5	38.3	34.2	137260150	model | metrics
R50-FPN	1x	0.261	0.043	3.4	38.6	35.2	137260431	model | metrics
R50-C4	3x	0.575	0.111	5.2	39.8	34.4	137849525	model | metrics
R50-DC5	3x	0.470	0.076	6.5	40.0	35.9	137849551	model | metrics
R50-FPN	3x	0.261	0.043	3.4	41.0	37.2	137849600	model | metrics
R101-C4	3x	0.652	0.145	6.3	42.6	36.7	138363239	model | metrics
R101-DC5	3x	0.545	0.092	7.6	41.9	37.3	138363294	model | metrics
R101-FPN	3x	0.340	0.056	4.6	42.9	38.6	138205316	model | metrics
X101-FPN	3x	0.690	0.103	7.2	44.3	39.5	139653917	model | metrics

Detectron2 Model Zoo : https://github.com/facebookresearch/detectron2/blob/master/MODEL_ZOO.md

TensorFlow 2 Object Detection API

- The TensorFlow Object Detection API is an open-source framework built on top of TensorFlow that makes it easy to construct, train, and deploy object detection models.

- The TensorFlow Object Detection API allows you to train a collection state of the art object detection models under a unified framework.

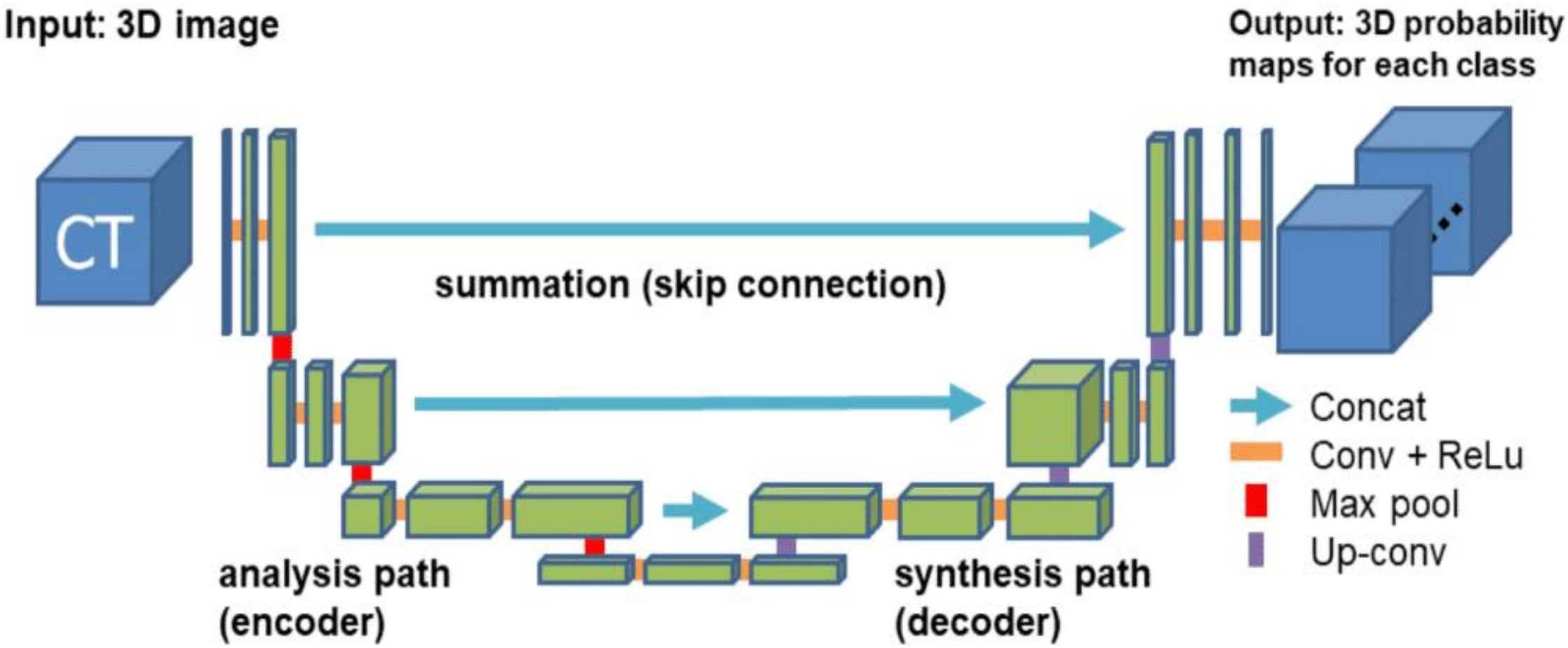
TensorFlow Object Detection API : https://github.com/tensorflow/models/tree/master/research/object_detection

TensorFlow 2 Object Detection API

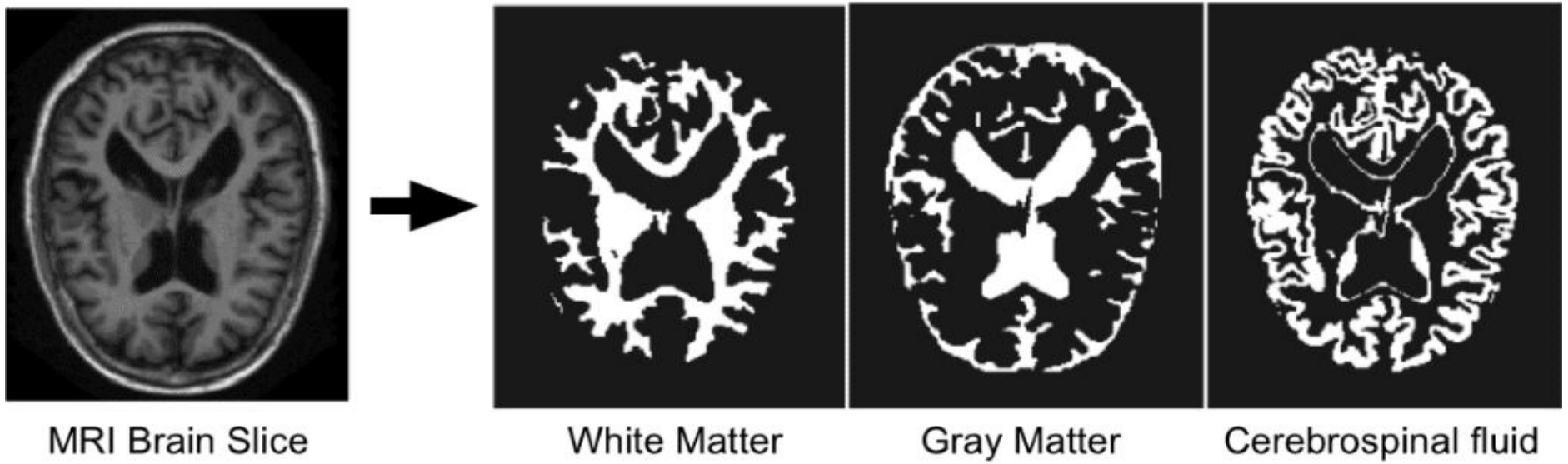
Model name	Speed (ms)	COCO mAP	Outputs
CenterNet HourGlass104 512x512	70	41.9	Boxes
CenterNet HourGlass104 Keypoints 512x512	76	40.0/61.4	Boxes/Keypoints
CenterNet HourGlass104 1024x1024	197	44.5	Boxes
CenterNet HourGlass104 Keypoints 1024x1024	211	42.8/64.5	Boxes/Keypoints
CenterNet Resnet50 V1 FPN 512x512	27	31.2	Boxes
CenterNet Resnet50 V1 FPN Keypoints 512x512	30	29.3/50.7	Boxes/Keypoints
CenterNet Resnet101 V1 FPN 512x512	34	34.2	Boxes
CenterNet Resnet50 V2 512x512	27	29.5	Boxes
CenterNet Resnet50 V2 Keypoints 512x512	30	27.6/48.2	Boxes/Keypoints
EfficientDet D0 512x512	39	33.6	Boxes
EfficientDet D1 640x640	54	38.4	Boxes
EfficientDet D2 768x768	67	41.8	Boxes
EfficientDet D3 896x896	95	45.4	Boxes
EfficientDet D4 1024x1024	133	48.5	Boxes
EfficientDet D5 1280x1280	222	49.7	Boxes
EfficientDet D6 1280x1280	268	50.5	Boxes
EfficientDet D7 1536x1536	325	51.2	Boxes
SSD ResNet50 V1 FPN 640x640 (RetinaNet50)	46	34.3	Boxes
SSD ResNet50 V1 FPN 1024x1024 (RetinaNet50)	87	38.3	Boxes
SSD ResNet101 V1 FPN 640x640 (RetinaNet101)	57	35.6	Boxes
SSD ResNet101 V1 FPN 1024x1024 (RetinaNet101)	104	39.5	Boxes
SSD ResNet152 V1 FPN 640x640 (RetinaNet152)	80	35.4	Boxes
SSD ResNet152 V1 FPN 1024x1024 (RetinaNet152)	111	39.6	Boxes
Faster R-CNN ResNet50 V1 640x640	53	29.3	Boxes
Faster R-CNN ResNet50 V1 1024x1024	65	31.0	Boxes
Faster R-CNN ResNet50 V1 800x1333	65	31.6	Boxes
Faster R-CNN ResNet101 V1 640x640	55	31.8	Boxes
Faster R-CNN ResNet101 V1 1024x1024	72	37.1	Boxes
Faster R-CNN ResNet101 V1 800x1333	77	36.6	Boxes
Faster R-CNN ResNet152 V1 640x640	64	32.4	Boxes
Faster R-CNN ResNet152 V1 1024x1024	85	37.6	Boxes
Faster R-CNN ResNet152 V1 800x1333	101	37.4	Boxes
Faster R-CNN Inception ResNet V2 640x640	206	37.7	Boxes
Faster R-CNN Inception ResNet V2 1024x1024	236	38.7	Boxes
Mask R-CNN Inception ResNet V2 1024x1024	301	39.0/34.6	Boxes/Masks

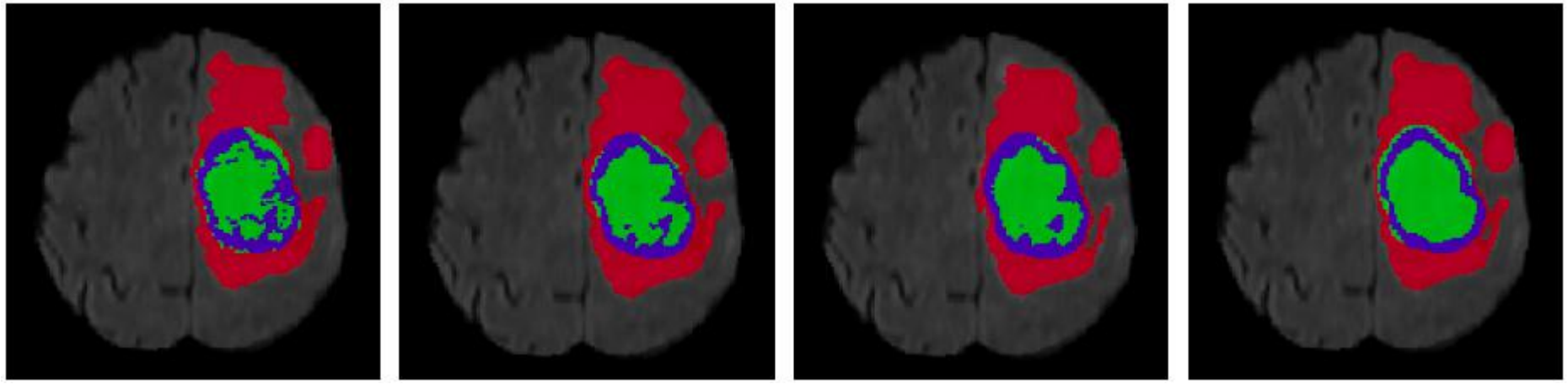
Model Zoo : https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/tf2_detection_zoo.md

3D-Unet

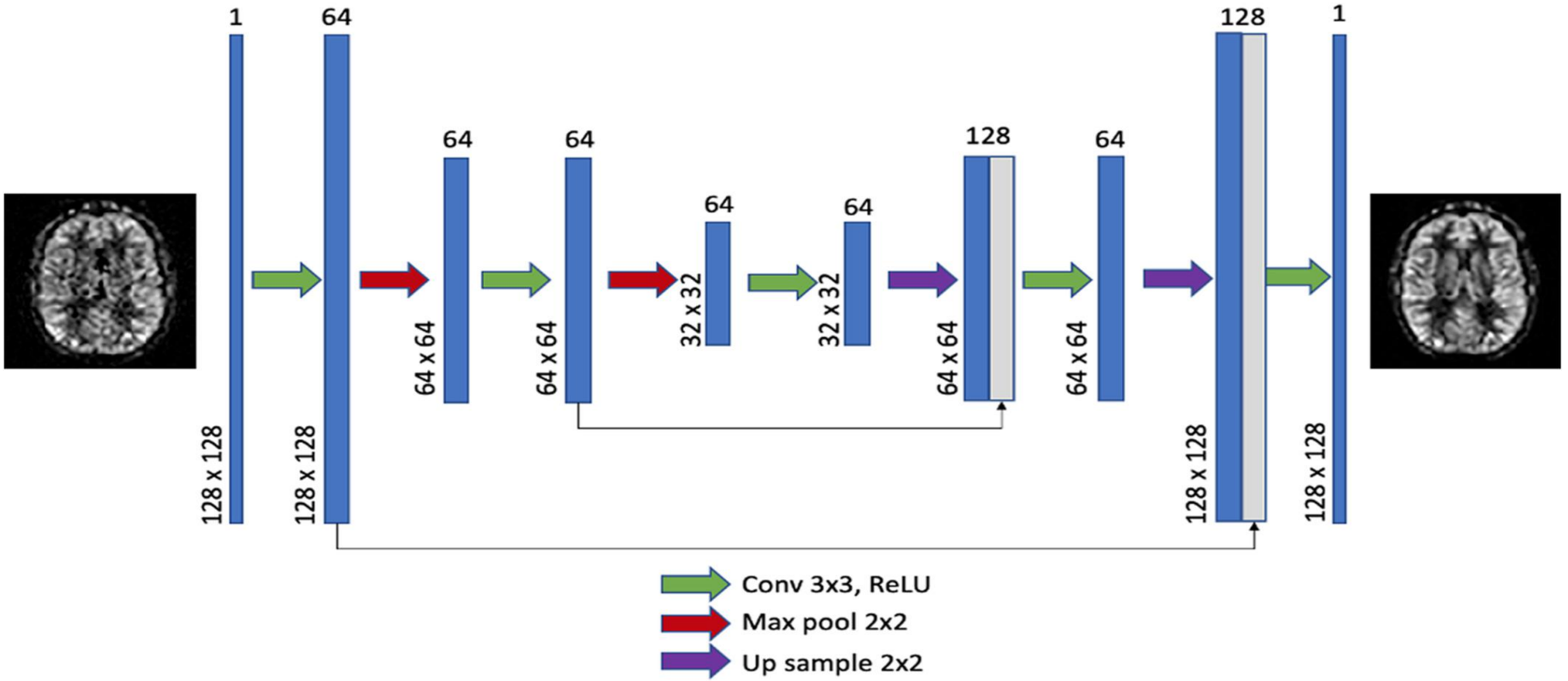

Input: 3D image
Output: 3D probability maps for each class
CT
summation (skip connection)
analysis path (encoder)
synthesis path (decoder)
Concat
Conv + ReLu
Max pool
Up-conv

3D-Unet


MRI Brain Slice
White Matter
Gray Matter
Cerebrospinal fluid



Autoencoders in Medical Imaging



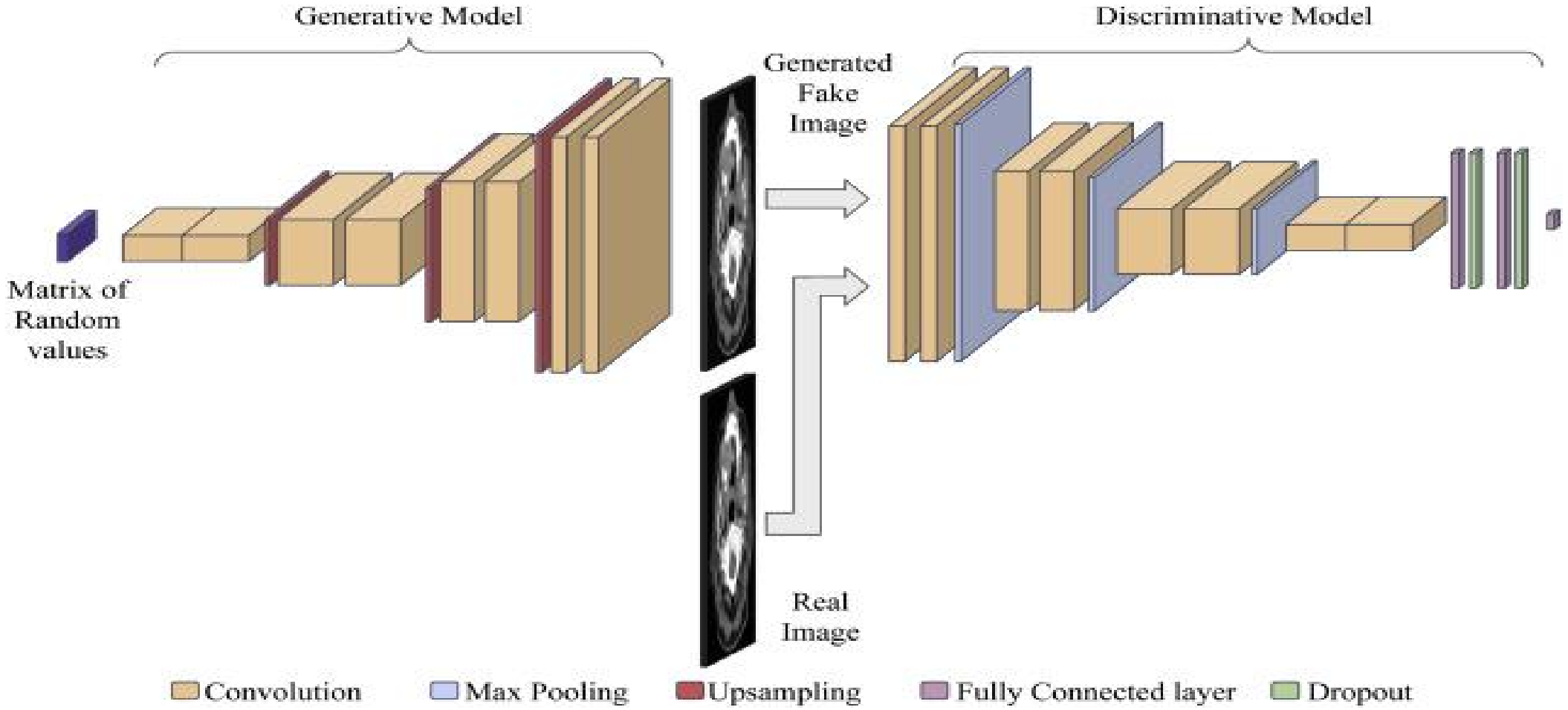
Architecture of the denoising autoencoder model, with an example low-SNR, single-repetition dM raw image (left), and the corresponding high-SNR dM mean image.

Combined Denoising and Suppression of Transient Artifacts in Arterial Spin Labeling MRI Using Deep Learning :
https://onlinelibrary.wiley.com/doi/10.1002/jmri.27255

Autoencoders in Medical Imaging

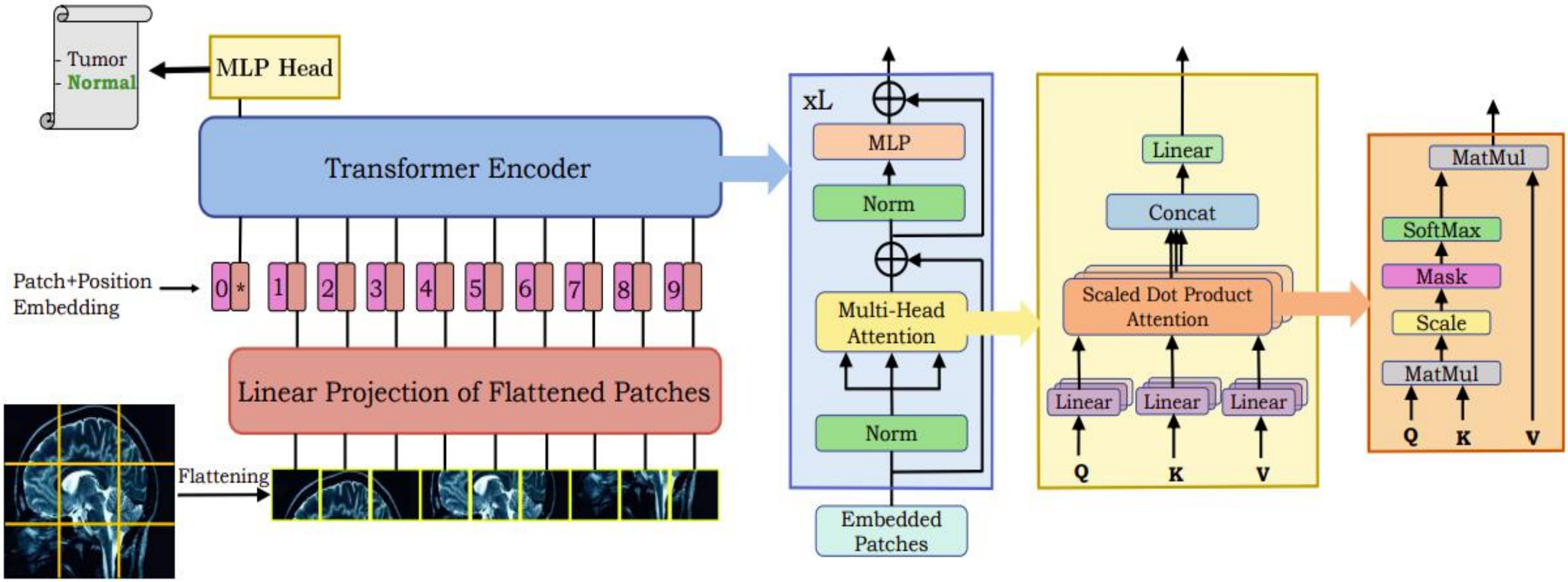
```
# Build Autoencoder
input = layers.Input(shape=(28, 28, 1))

# Encoder
Encoder = layers.Conv2D(32, (3, 3), activation="relu", padding="same")(input)
Encoder = layers.MaxPooling2D((2, 2), padding="same")(Encoder)
Encoder = layers.Conv2D(32, (3, 3), activation="relu", padding="same")(Encoder)
Encoder = layers.MaxPooling2D((2, 2), padding="same")(Encoder)

# Decoder
Decoder = layers.Conv2DTranspose(32, (3, 3), strides=2, activation="relu", padding="same")(Encoder)
Decoder = layers.Conv2DTranspose(32, (3, 3), strides=2, activation="relu", padding="same")(Decoder)
Decoder = layers.Conv2D(1, (3, 3), activation="sigmoid", padding="same")(Decoder)

# Autoencoder
autoencoder = Model(input, Decoder)
autoencoder.summary()

from tensorflow.keras.utils import plot_model
plot_model(autoencoder, show_shapes=True)
```

Generative Adversarial Networks GANs in Medical Imaging



GANs for medical image analysis :
https://doi.org/10.1016/j.artmed.2020.101938

Transformers in Medical Imaging



Vision Transformer (ViT) for Image Classification (cifar10 dataset) :

https://github.com/hichemfelouat/my-codes-of-machine-learning/blob/master/Vision_Transformer_(ViT)_for_Image_Classification_(cifar10_dataset).ipynb

Transformers in Medical Imaging

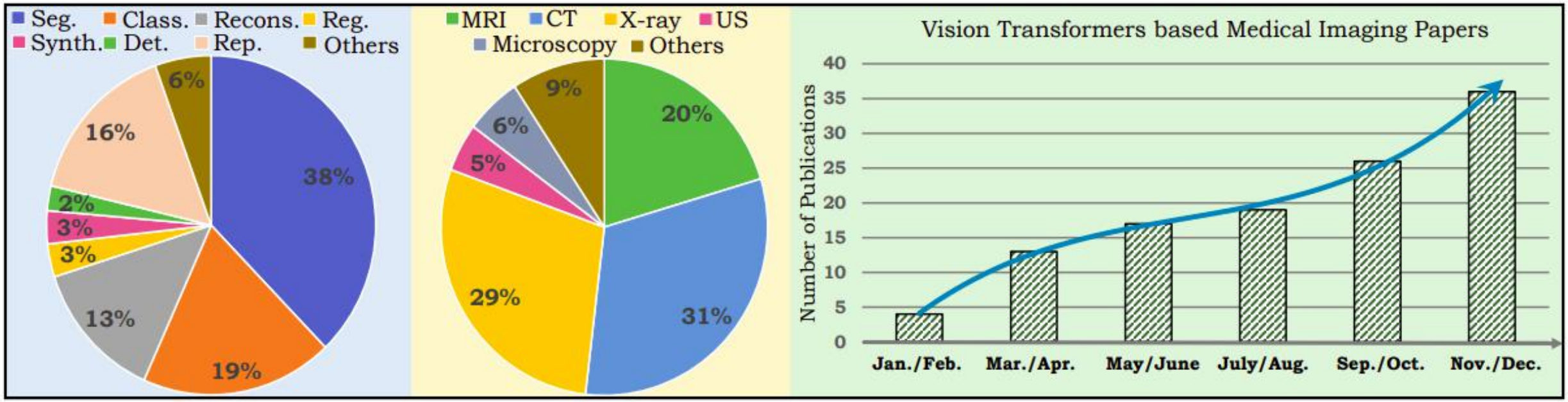


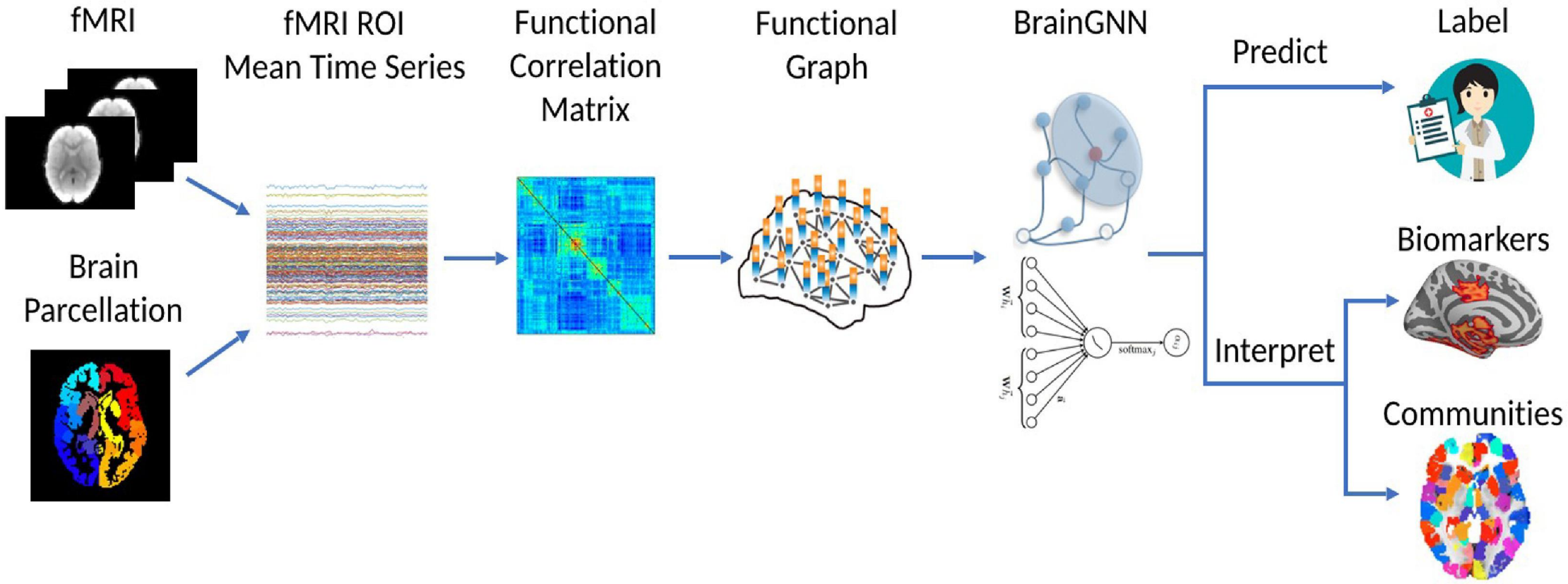
Figure 1: (Left) The pie-charts show statistics of the papers included in this survey according to medical imaging problem settings and data modalities. The rightmost figure shows consistent growth in the recent literature (for year 2021). Seg: segmentation, Class: classification, Recons: reconstruction, Reg: registration, Synth: synthesis, Det: detection, Rep: report generation, US: ultrasound.

Transformers in Medical Imaging: A Survey
https://arxiv.org/abs/2201.09873v1

Transformers in Medical Imaging

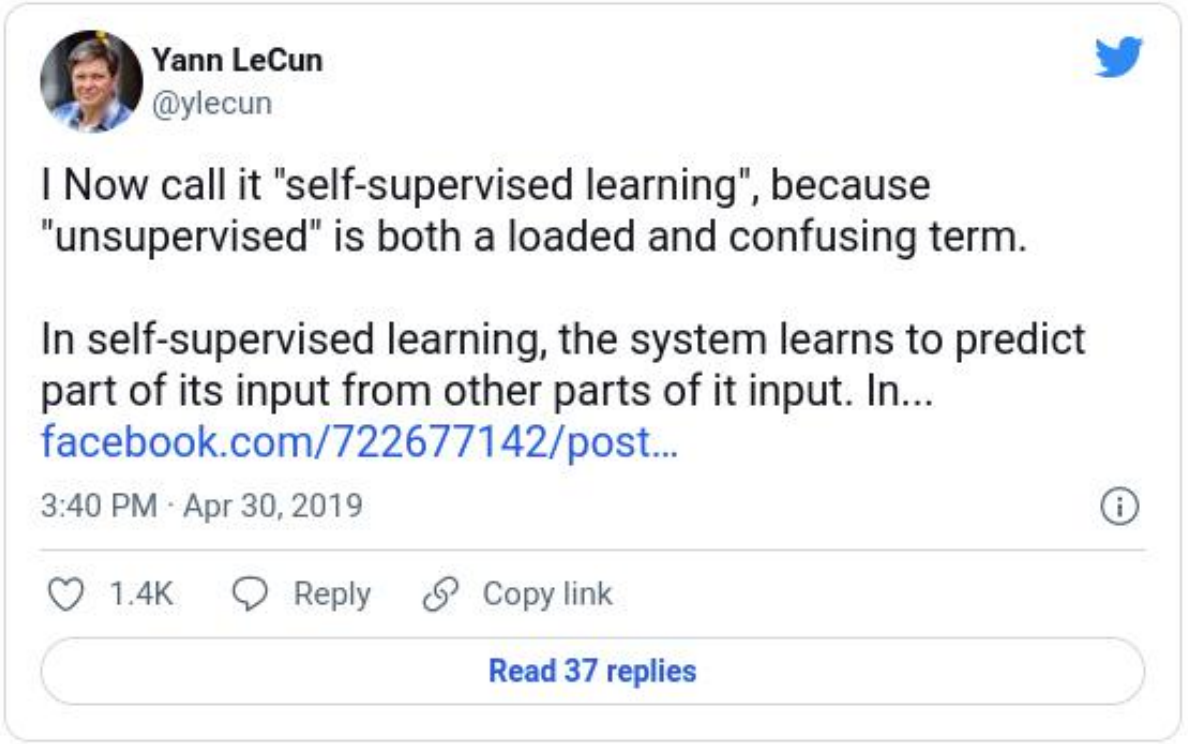
```python
# !pip install transformers
from transformers import TFViTModel
import tensorflow as tf
from tensorflow import keras

# google/vit-base-patch32-384
model_id = "google/vit-base-patch16-224-in21k"
base_model = TFViTModel.from_pretrained(model_id)

# Inputs
pixel_values = keras.layers.Input(shape=(3,224,224), name="pixel_values", dtype="float32")

# Model layer
vit = base_model.vit(pixel_values)[0]
classifier = keras.layers.Dense(1, activation="softmax", name="outputs")(vit[:, 0, :])

# Model
vit_model = keras.Model(inputs=pixel_values, outputs=classifier)

# The model's summary() method displays all the model's layers
print(vit_model.summary())

```

Graph Neural Network in Medical Image Analysis



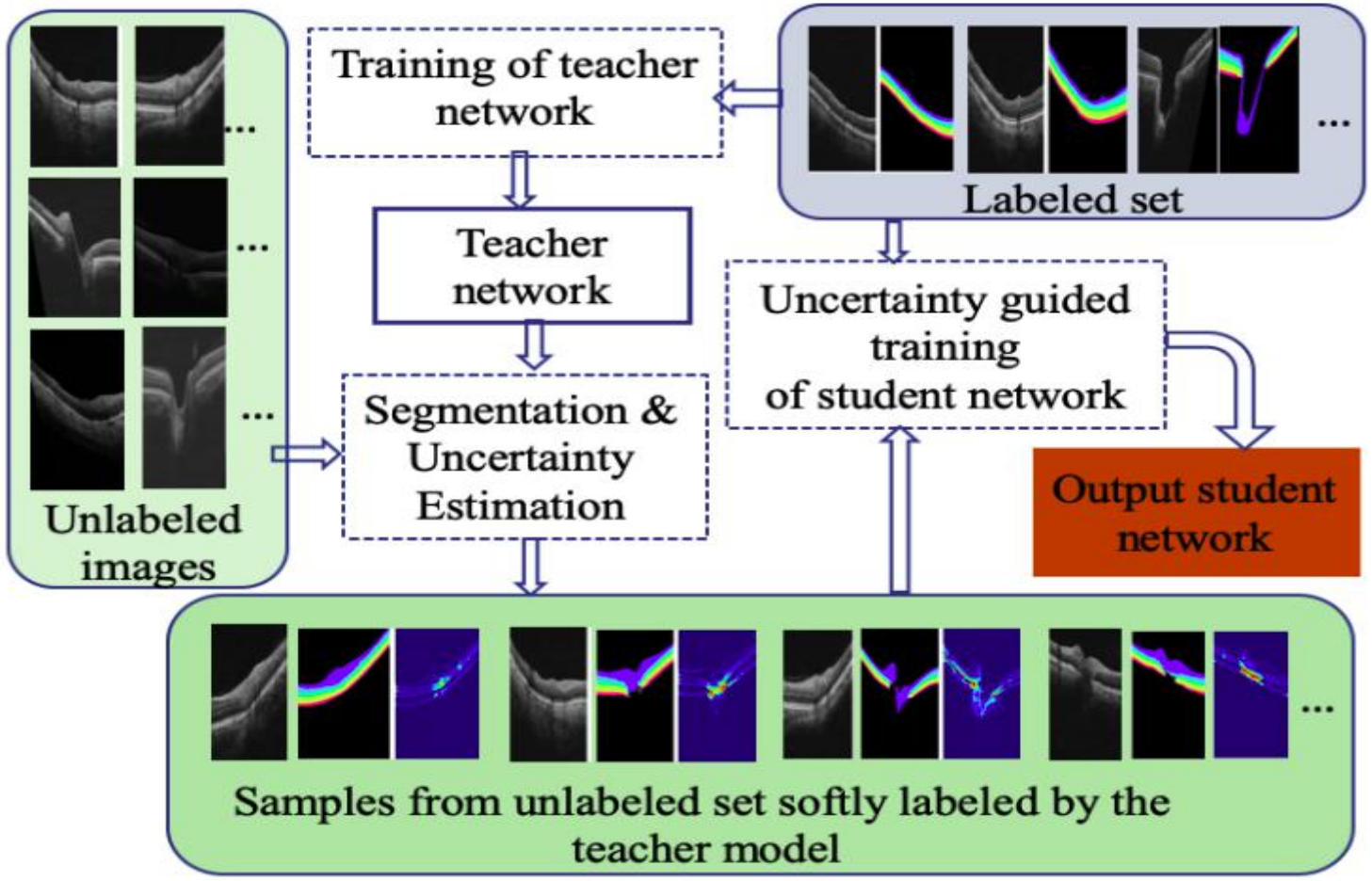
BrainGNN: Interpretable Brain Graph Neural Network for fMRI Analysis :
https://doi.org/10.1016/j.media.2021.102233
Graph-Based Deep Learning for Medical Diagnosis and Analysis: Past, Present and Future :
https://arxiv.org/abs/2105.13137

Self-Supervised and Semi-Supervised Learning



In the self-supervised learning technique, the model depends on the underlying structure of data to predict outcomes. It involves no labelled data. However, in semi-supervised learning, we still provide a small amount of labelled data.

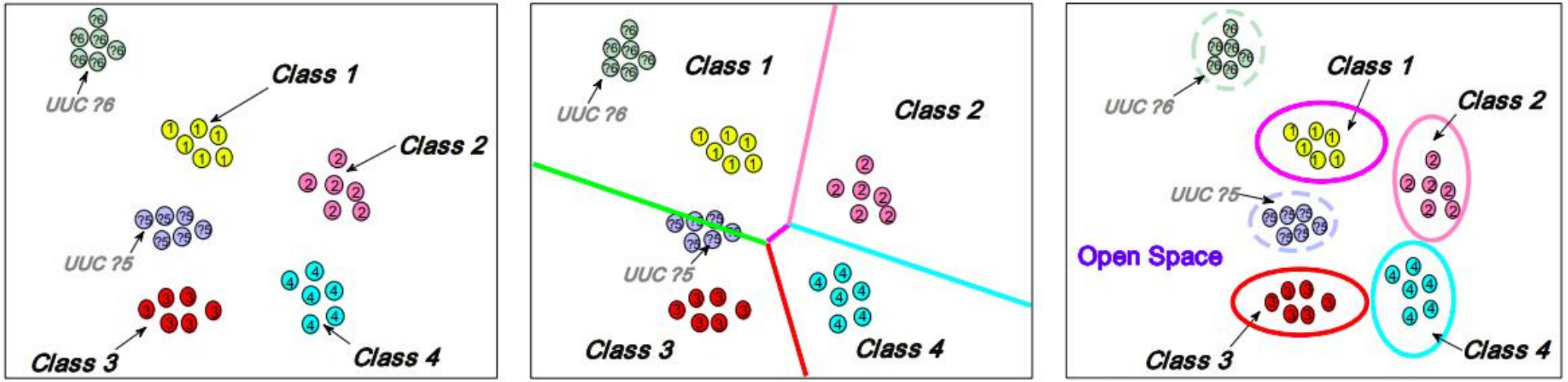
Self-Supervised and Semi-Supervised Learning



Uncertainty Guided Semi-supervised Segmentation of Retinal Layers in OCT Images
https://link.springer.com/chapter/10.1007/978-3-030-32239-7_32
Semi-Supervised Learning in Computer Vision
https://amitness.com/2020/07/semi-supervised-learning/
A Survey of Self-Supervised and Few-Shot Object Detection
https://arxiv.org/abs/2110.14711

Open Set Learning OSL

Traditional supervised learning aims to train a classifier in the closed-set world, where training and test samples share the same label space. Open set learning (OSL) is a more challenging and realistic setting, where there exist test samples from the classes that are unseen during training.



(a) Distribution of the original data set.

(b) Traditional recognition/classification problem.

(c) Open set recognition/classification problem.

Fig (c): describes open set recognition, where the decision boundaries limit the scope of KKCs 1,2,3,4, reserving space for UUCs ?5,?6. Via these decision boundaries, the samples from some UUCs are labeled as "unknown" or rejected rather than misclassified as KKCs.

Recent Advances in Open Set Recognition: A Survey
https://arxiv.org/abs/1811.08581

Thanks For Your Attention

Hichem Felouat ...