Medical Image Analysis with CNNs

Matthew Engelhard

Overview

- What can CNNs do, and how is it relevant to clinical medicine?
- How do CNNs work, in brief?
- How do we (re)-train a model to classify medical images?
- Notes on image data pre-processing (time permitting)

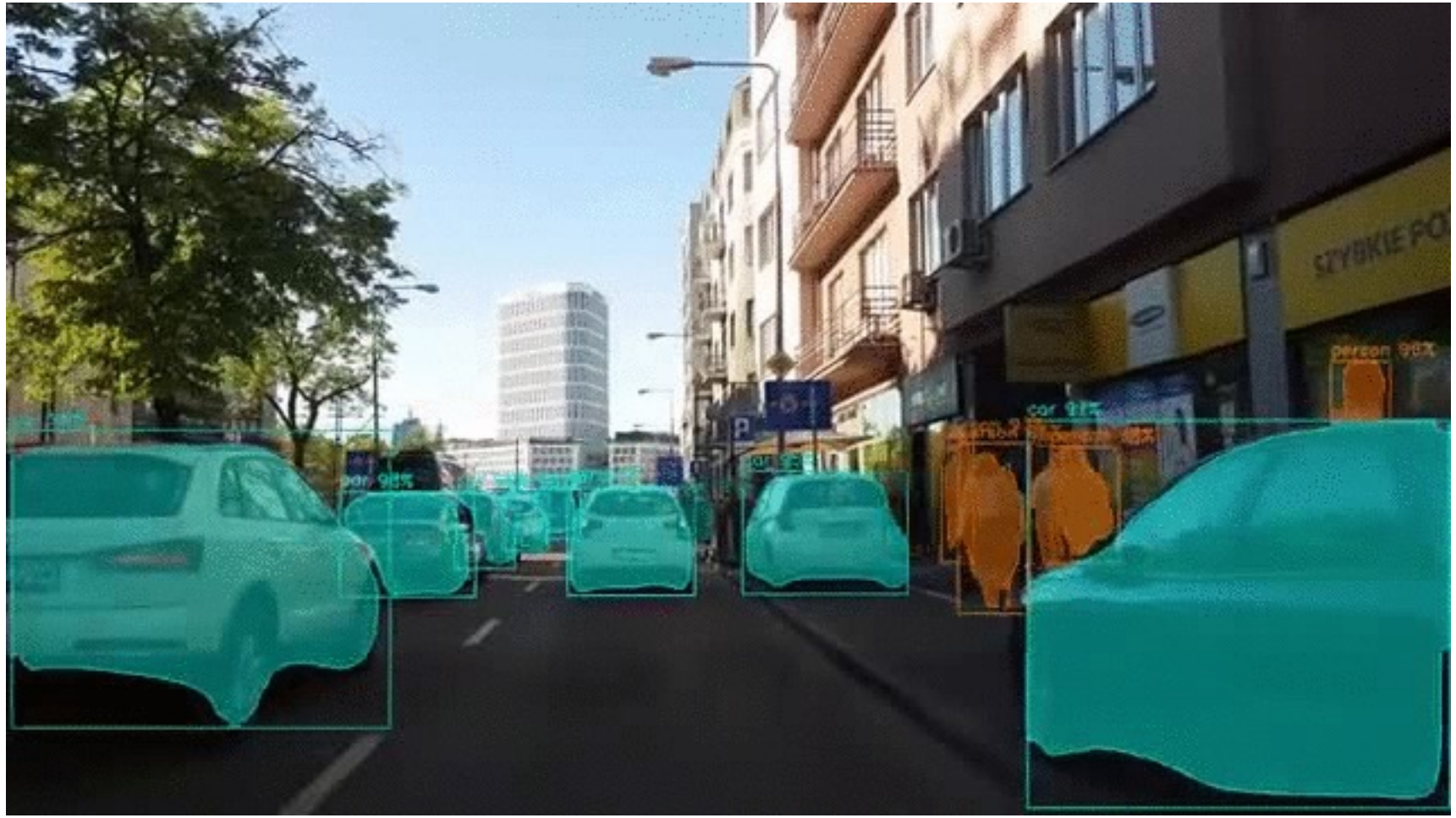
Overview

- What can CNNs do, and how is it relevant to clinical medicine?
- How do CNNs work, in brief?
- How do we (re)-train a model to classify medical images?
  - Take an existing, trained classification model
  - Replace the *final layer* (i.e., *prediction head*)
  - Fine-tune model parameters
- Notes on image data pre-processing (time permitting)

What can CNNs do?

Classification, Detection, Segmentation…

Computer Vision (with a CNN)



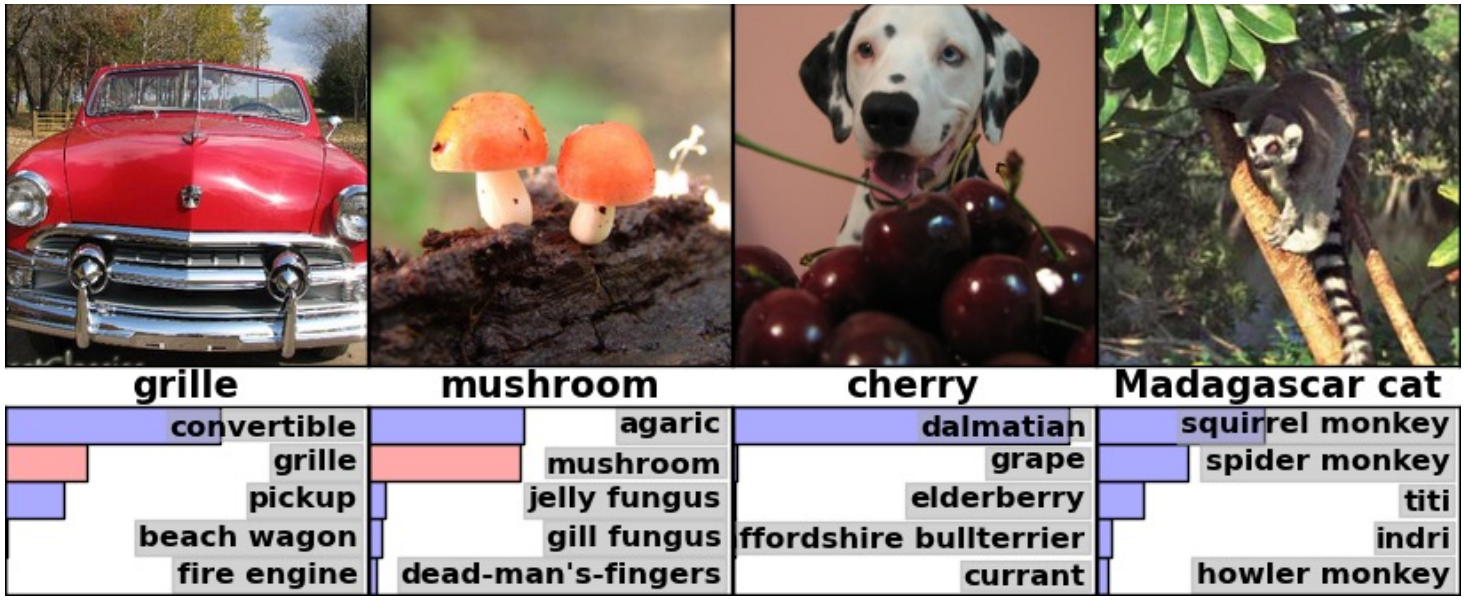
Binary Classification

Diabetic retinopathy? (Yes/No)

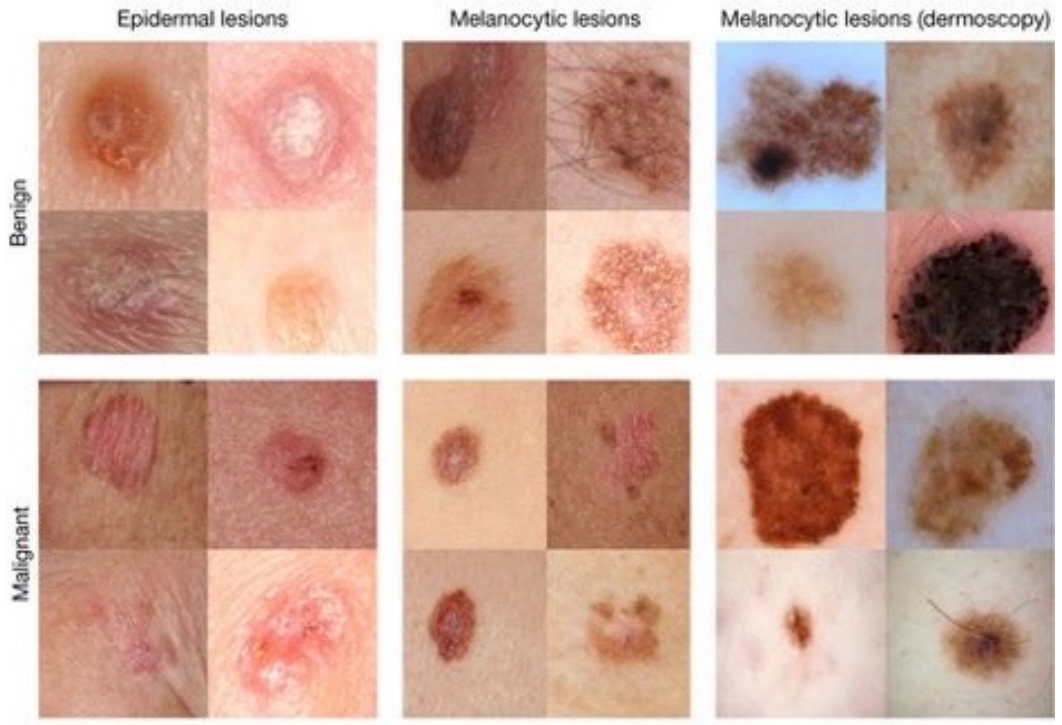
Gulshan et al. *JAMA* (2016)

Multi-Class Classification

Multi-Class Classification



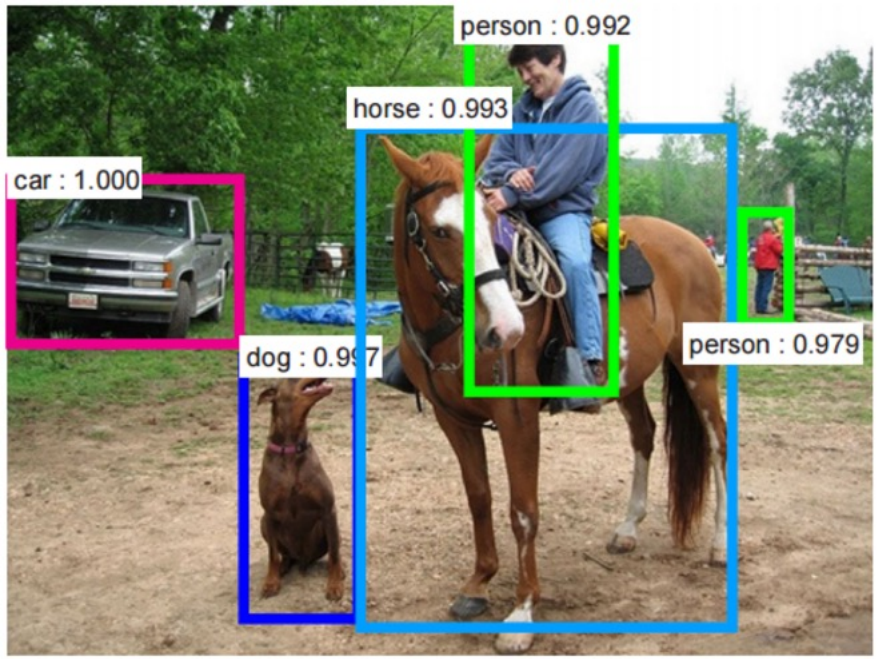
Multi-Class Classification

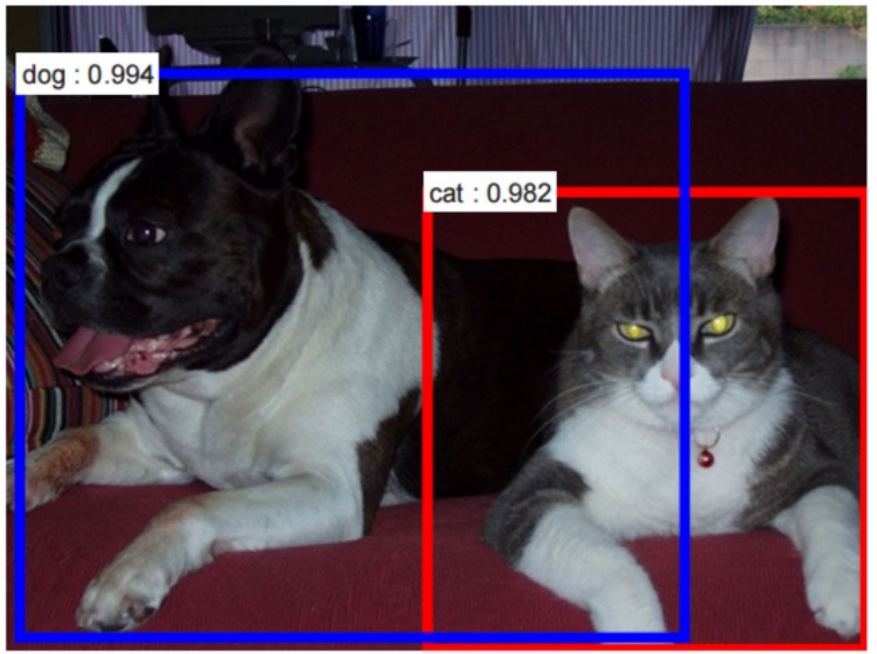


Dermatologist-level classification of skin cancer

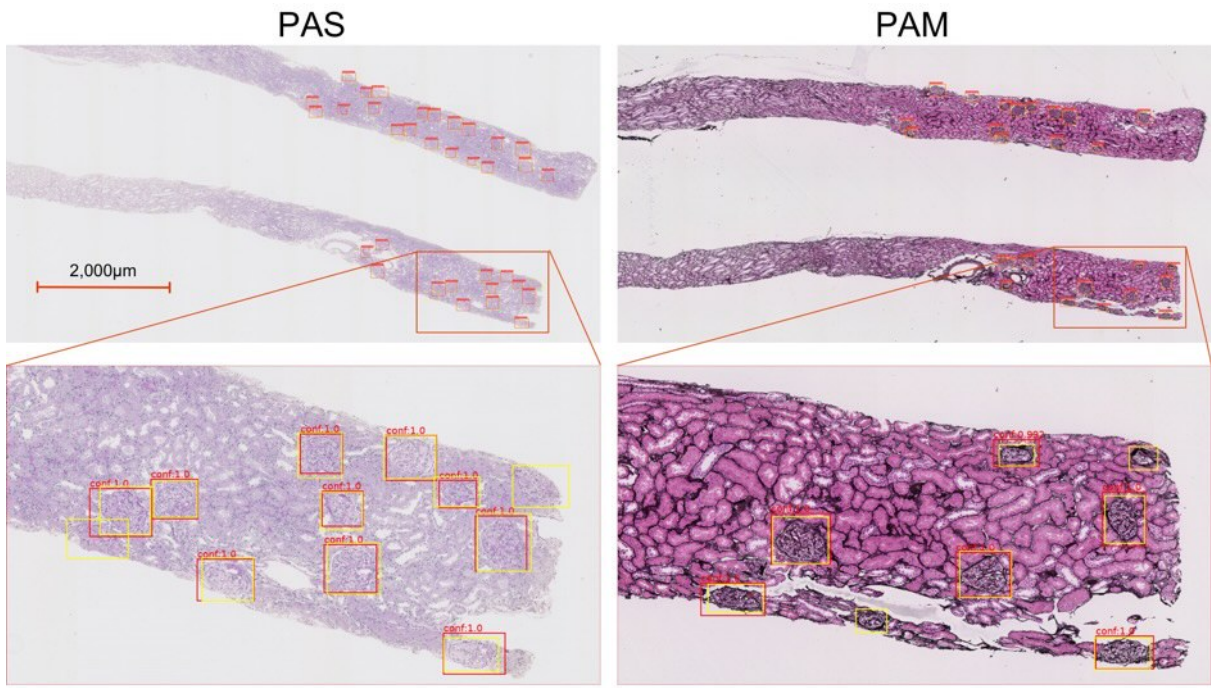
Nature volume 542, pages 115–118 (02 February 2017)

Object Detection



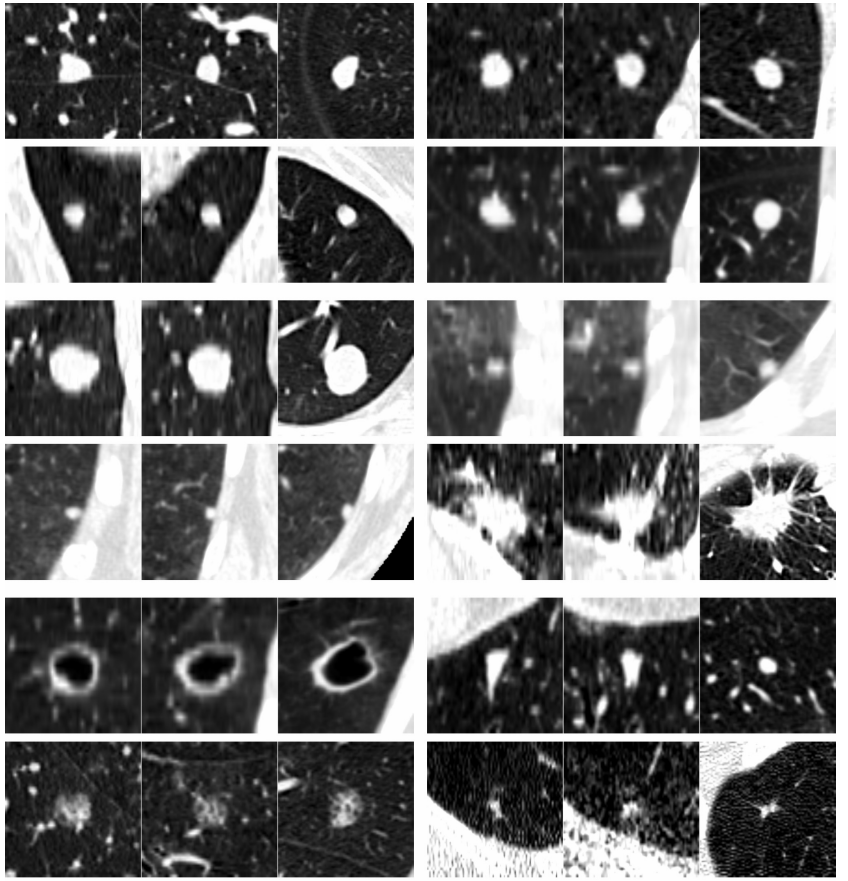


Object Detection in Medicine



Glomerular Detection with Faster-RCNN

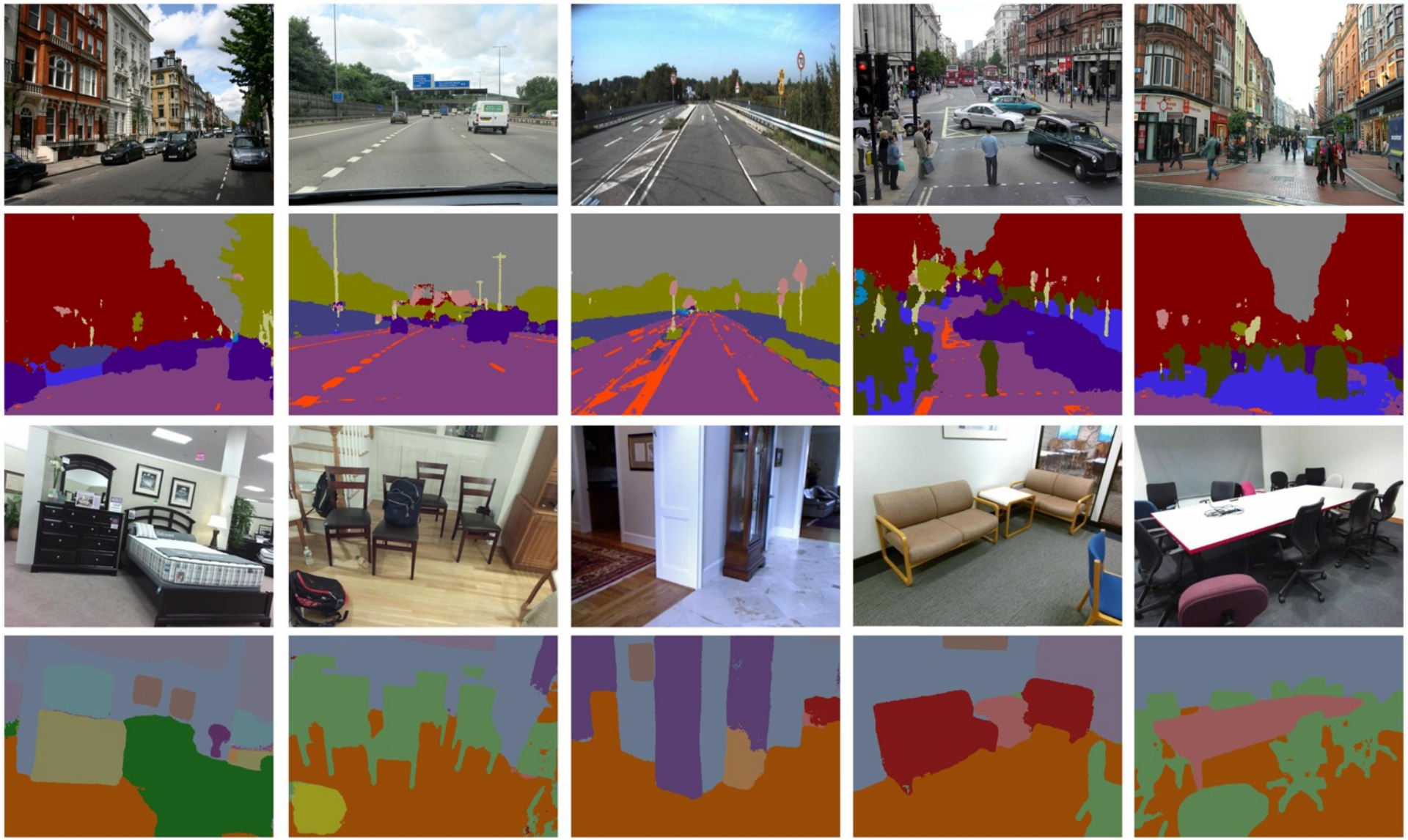
Kawazoe et al., *J. Imaging, 2018*



Pulmonary Nodule detection in CT

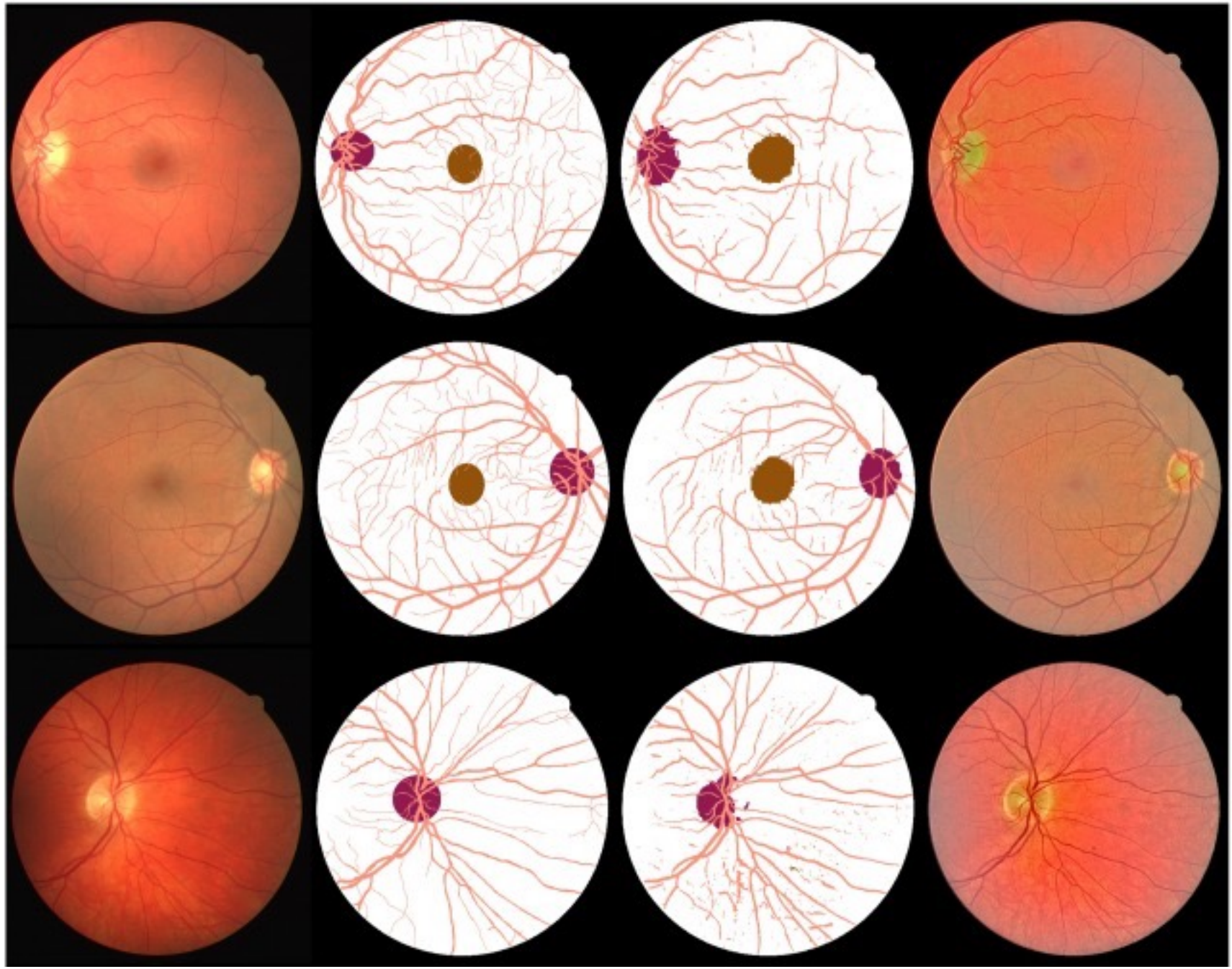
van Ginneken et al., *Biomedical Imaging,* 2015

Semantic Segmentation

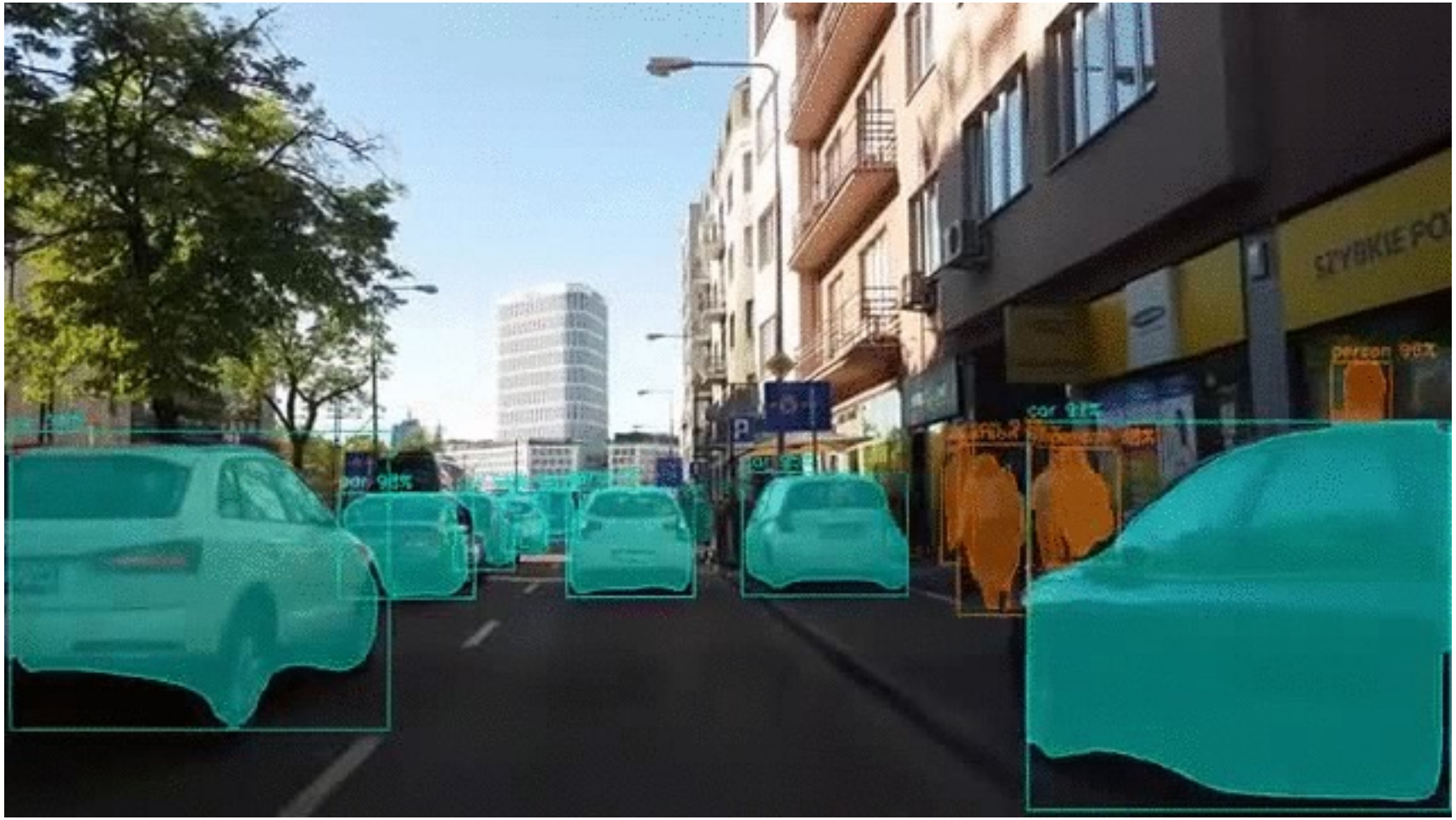


Segmentation of optic disc, fovea and retinal vasculature

Journal of Computational Science, *20*, 70-79 (2017).

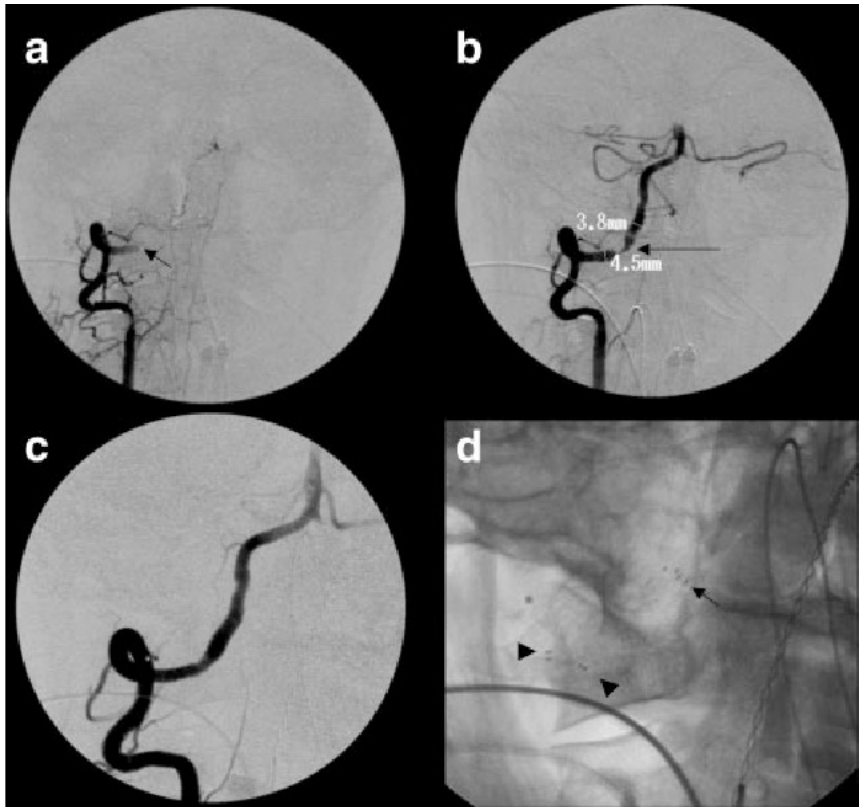


Detect objects, classify them, and define their boundaries (i.e. segment)

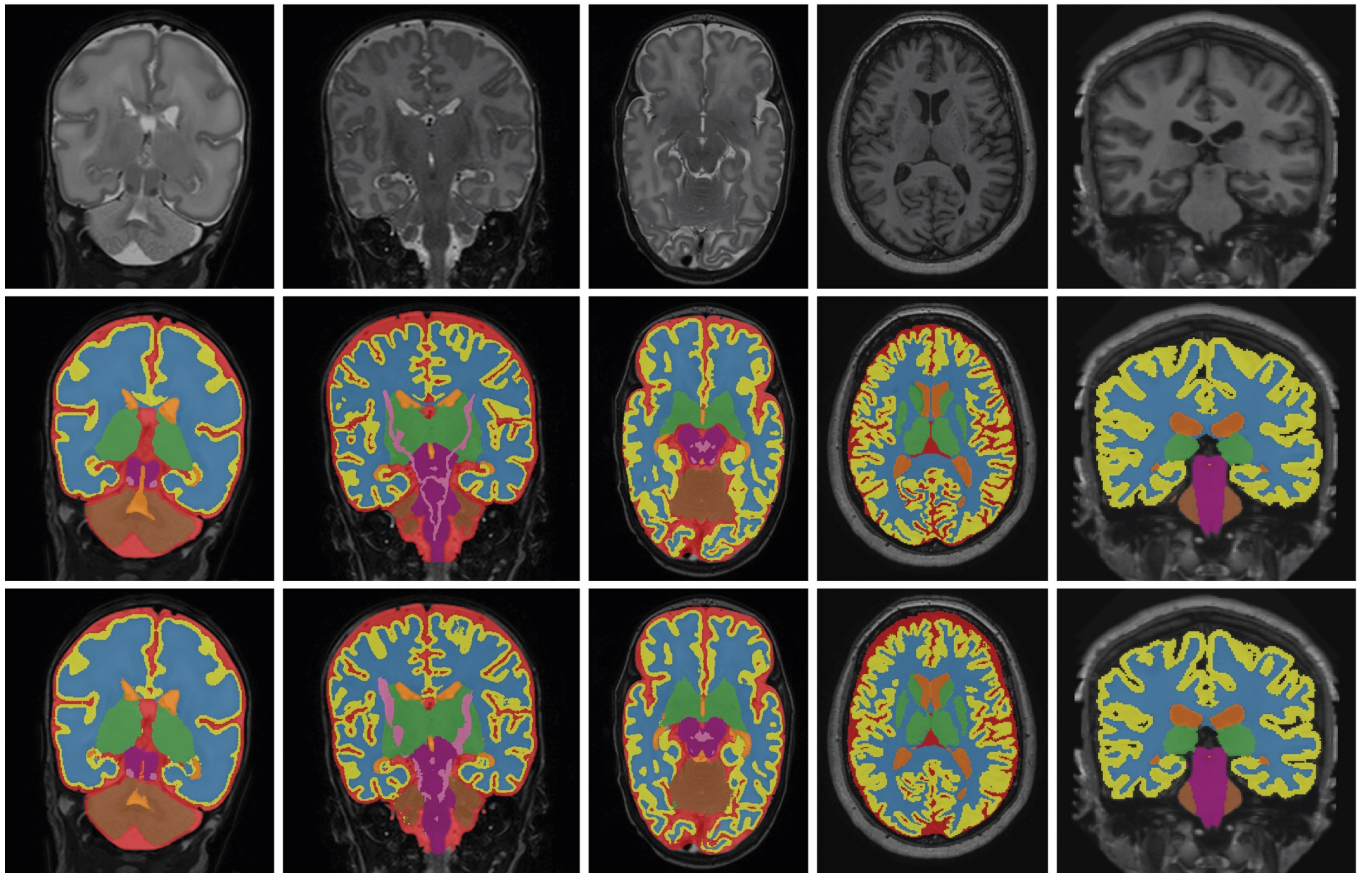


Why Segment?

-> Precisely Identify Boundaries



Why Segment?
-> Determine Areas or Volumes



Why Segment?
-> Extract features when direct classification is not feasible

Segment Bones → Calculate lengths and areas → Simple Classifier for Genetic or Growth Disorder

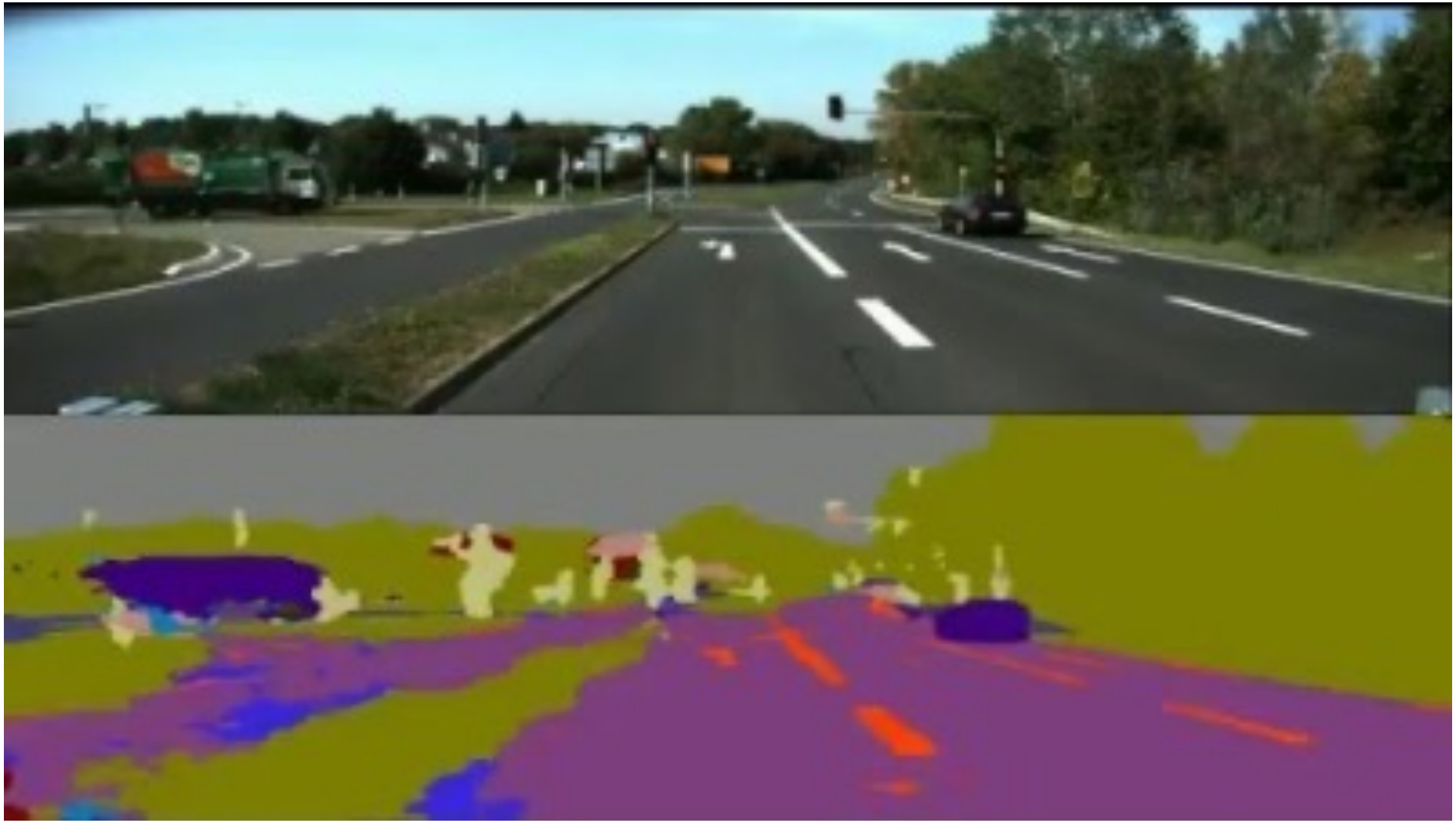
Segmentation of Colon Polyps in Real Time

What do they all have in common?

- Predictions are made with a convolutional neural network (CNN), which is a neural network with *convolutional* (rather than *fully-connected*) layers.

- These models are not trained from scratch. Instead of beginning with random parameters, we begin with parameters learned from a different, usually non-medical task.

Approach: Start with SegNet, then re-train to identify polyps



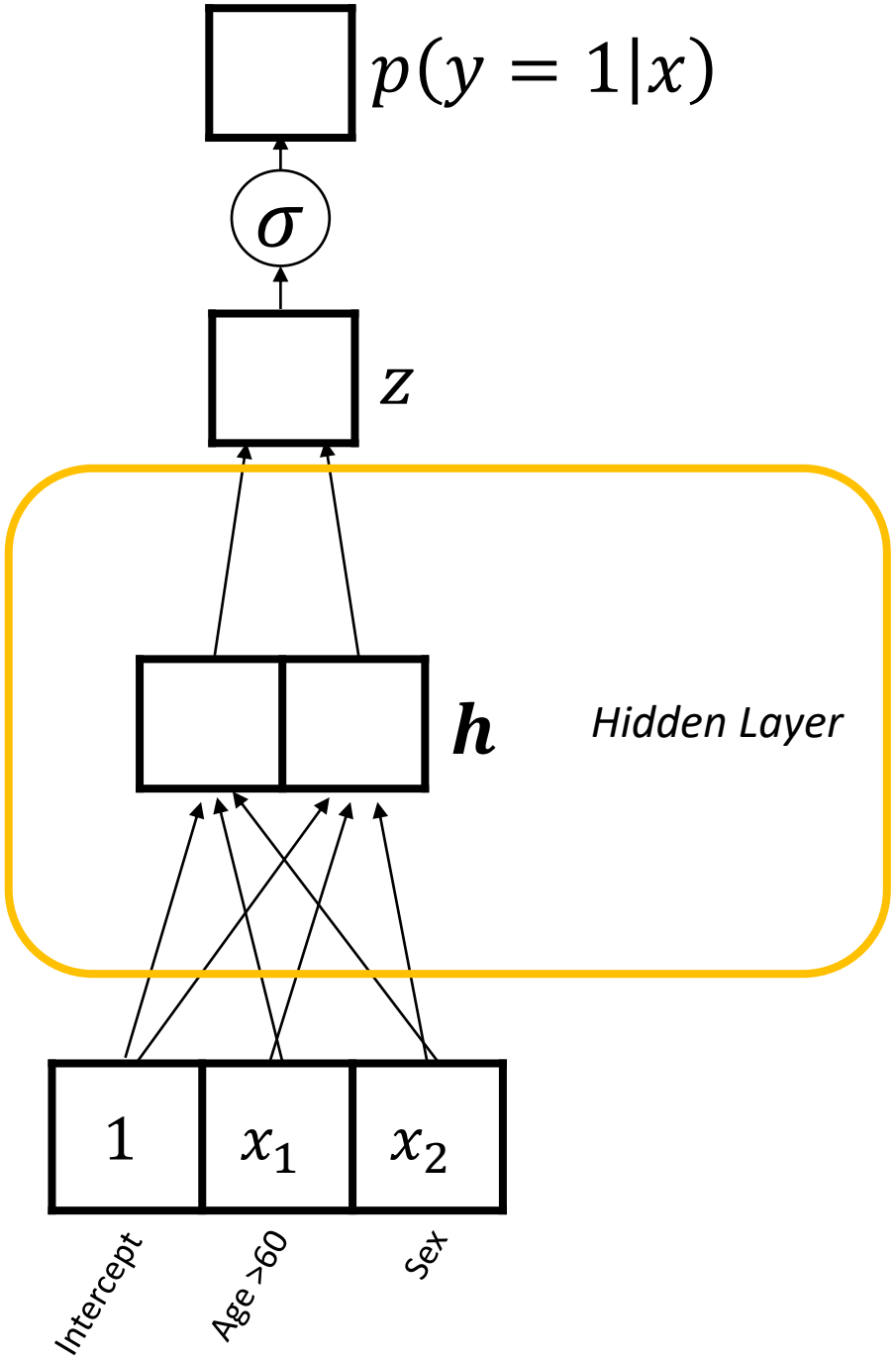
What do they all have in common?

- Predictions are made with a convolutional neural network (CNN), which is a neural network with *convolutional* (rather than *fully-connected*) layers.

- These models are not trained from scratch. Instead of beginning with random parameters, we begin with parameters learned from a different, usually non-medical task.
  - Why is this a good idea?
  - How can we develop our own CNN model?

How do CNNs work?

Hierarchical feature extraction (like the MLP).



Recall: the MLP

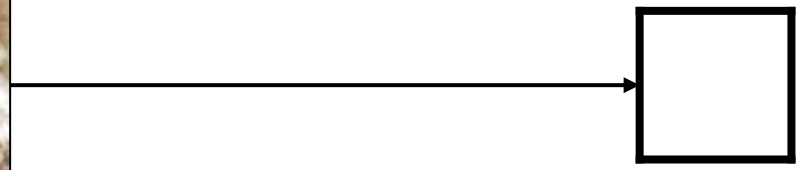
The MLP allowed us to detect *complex, nonlinear* features…

- older male OR younger female
- high OR low blood pressure
- specific patient profiles
- different handwritten digit styles

…by first identifying simpler features.

The model makes predictions about the complex features based on which simple features are detected.

The relationship between individual pixels and the outcome is not linear.

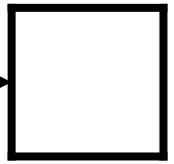


y, associated value or label

x, data/features for a subject or patient

End goal: predict y from x

The relationship between individual pixels and the outcome is not linear.



y, associated value or label

x, data/features for a subject or patient

End goal: predict y from x

We must build a hierarchy of features.

pixels

low-level motifs

high-level motifs

dog

label

End goal: predict *dog* from *pixels*

We must build a hierarchy of features.

pixels

high-level motifs are *specific arrangements* of low-level motifs

low-level motifs

- edges
- shapes
- textures

high-level motifs

- eyes
- ears
- paws

dog

label

End goal: predict *dog* from *pixels*

We must build a hierarchy of features.

pixels

low-level motifs

- edges
- shapes
- textures

high-level motifs

- microaneurysm
- exudates
- hemorrhage

retinopathy

label

End goal: predict *retinopathy* from *pixels*

We must build a hierarchy of features.

pixels

low-level motifs are common to *all* naturalistic images…

low-level motifs

- **edges**
- **shapes**
- **textures**

high-level motifs

- microaneurysm
- exudates
- hemorrhage

retinopathy

label

End goal: predict *retinopathy* from *pixels*

Why start with a pre-trained model?

- Instead of beginning with random parameters, we begin with parameters learned from a different, usually non-medical task.

- This is because these parameters are *better than random*: many features that are helpful in classifying naturalistic images are also helpful in classifying medical images.

- This is particularly true *early* in the feature hierarchy (i.e. low-level motifs; early layers).

How do we build our CNN?

Copy, modify, fine-tune. We will focus on whole-image classification.

Step 1. Download a trained image classification model.

Pre-trained Models

Neural nets work best when they have many parameters, making them powerful function approximators. However, this means they must be trained on very large datasets. Because training models from scratch can be a very computationally intensive process requiring days or even weeks, we provide various pre-trained models, as listed below. These CNNs have been trained on the ILSVRC-2012-CLS image classification dataset.

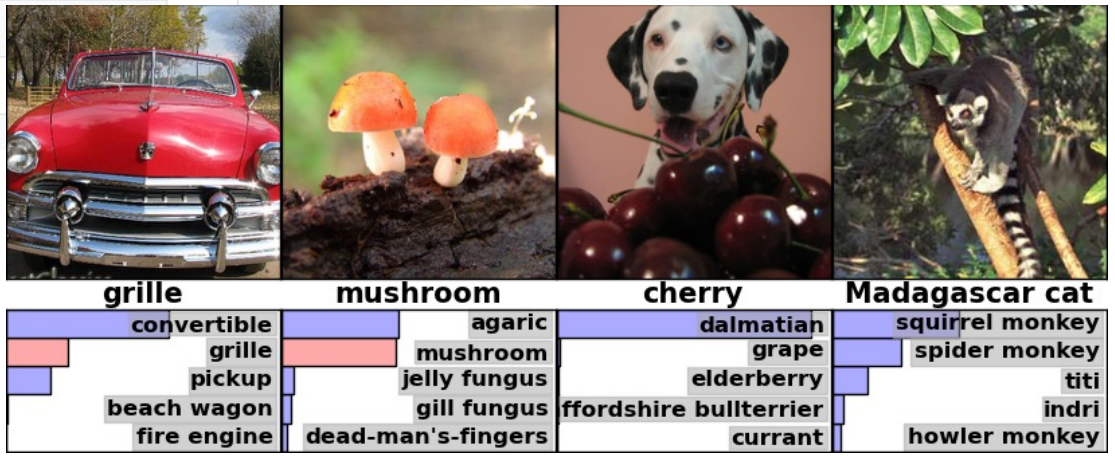
In the table below, we list each model, the corresponding TensorFlow model file, the link to the model checkpoint, and the top 1 and top 5 accuracy (on the imagenet test set). Note that the VGG and ResNet V1 parameters have been converted from their original caffe formats (here and here), whereas the Inception and ResNet V2 parameters have been trained internally at Google. Also be aware that these accuracies were computed by evaluating using a single image crop. Some academic papers report higher accuracy by using multiple crops at multiple scales.

Model	TF-Slim File	Checkpoint	Top-1 Accuracy	Top-5 Accuracy
Inception V1	Code	inception_v1_2016_08_28.tar.gz	69.8	89.6
Inception V2	Code	inception_v2_2016_08_28.tar.gz	73.9	
Inception V3	Code	inception_v3_2016_08_28.tar.gz	78.0	
Inception V4	Code	inception_v4_2016_09_09.tar.gz	80.2	

https://github.com/tensorflow/models/tree/master/research/slim#Pretrained

Python Code:
Defines the model architecture

Checkpoint File:
Contains parameters of the trained model

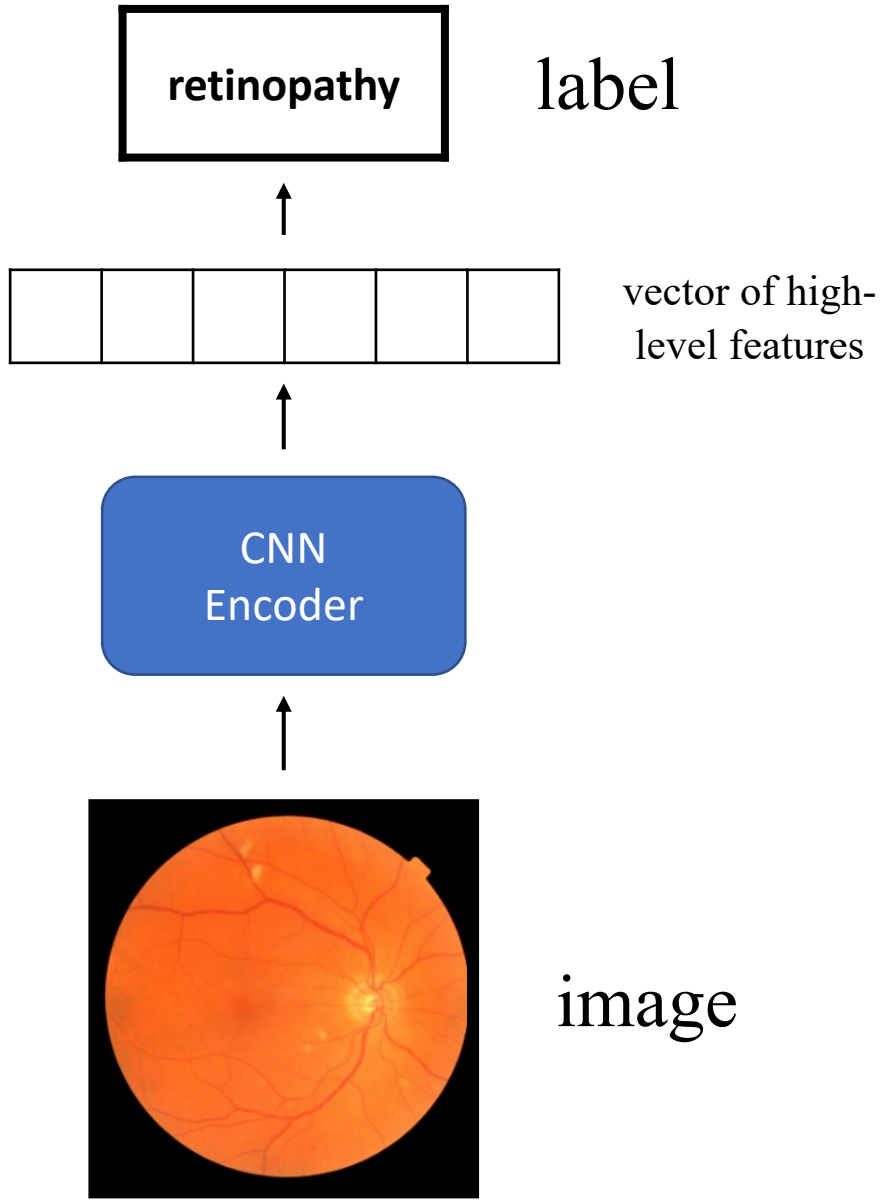


We can divide this model into two pieces:

1. A CNN *image encoder* that converts the raw image to a vector of high-level motifs / features.

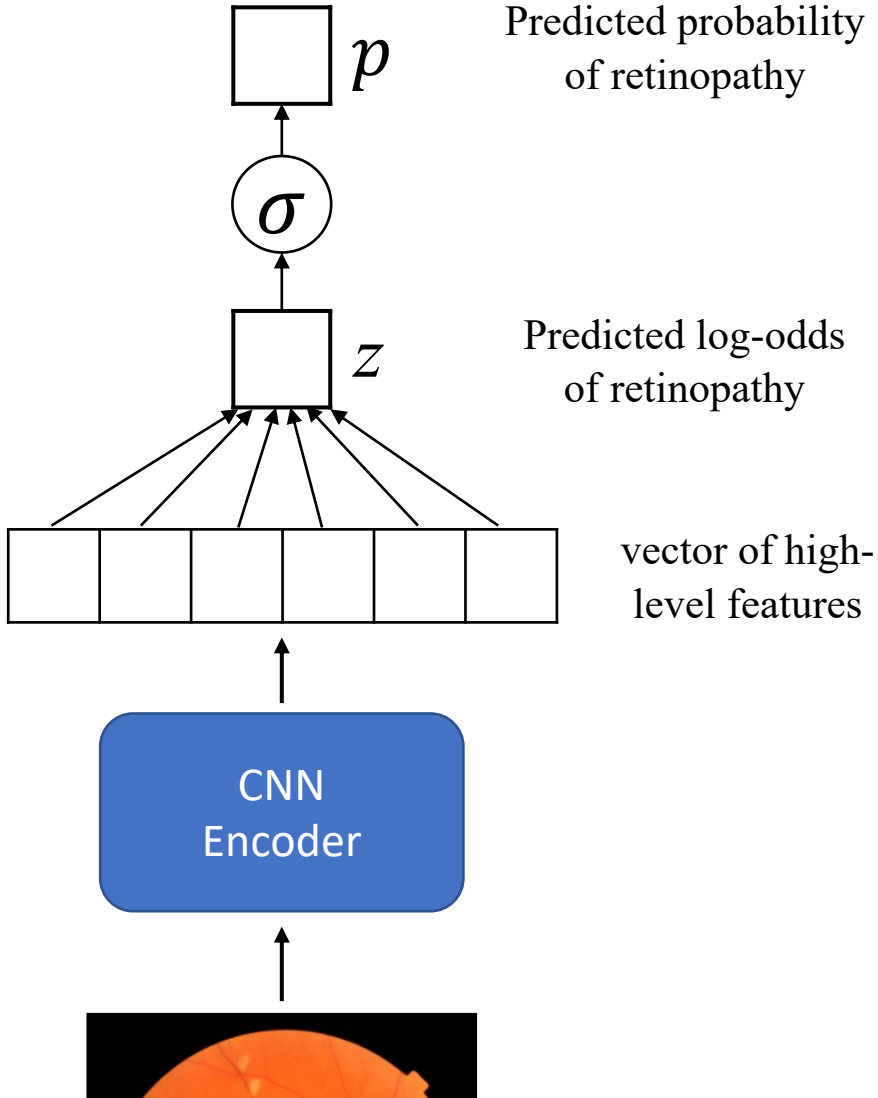
2. A *final layer*, or *prediction head* – this is a logistic regression model – that makes predictions about the label from these high-level features.

- We will reuse the encoder but replace the prediction head, since it is specific to the previous (non-medical) task.



(Enhance!) Inspecting the prediction head:

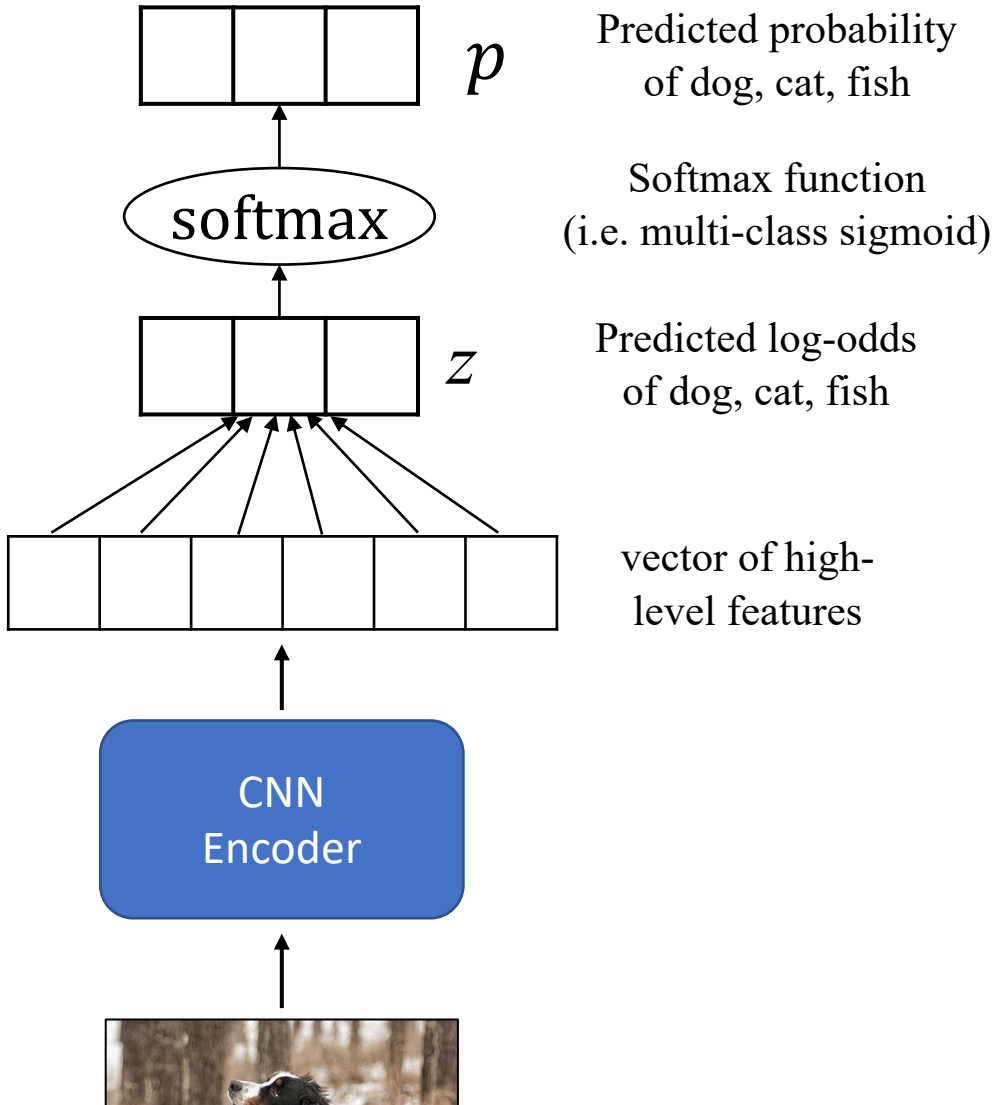
In this example, our prediction head is simple logistic regression on the features extracted by the CNN encoder.



(Enhance!) Inspecting the prediction head:

In this example, our prediction head is simple logistic regression on the features extracted by the CNN encoder.

Typically, we start with a multi-class (rather than binary) prediction head.

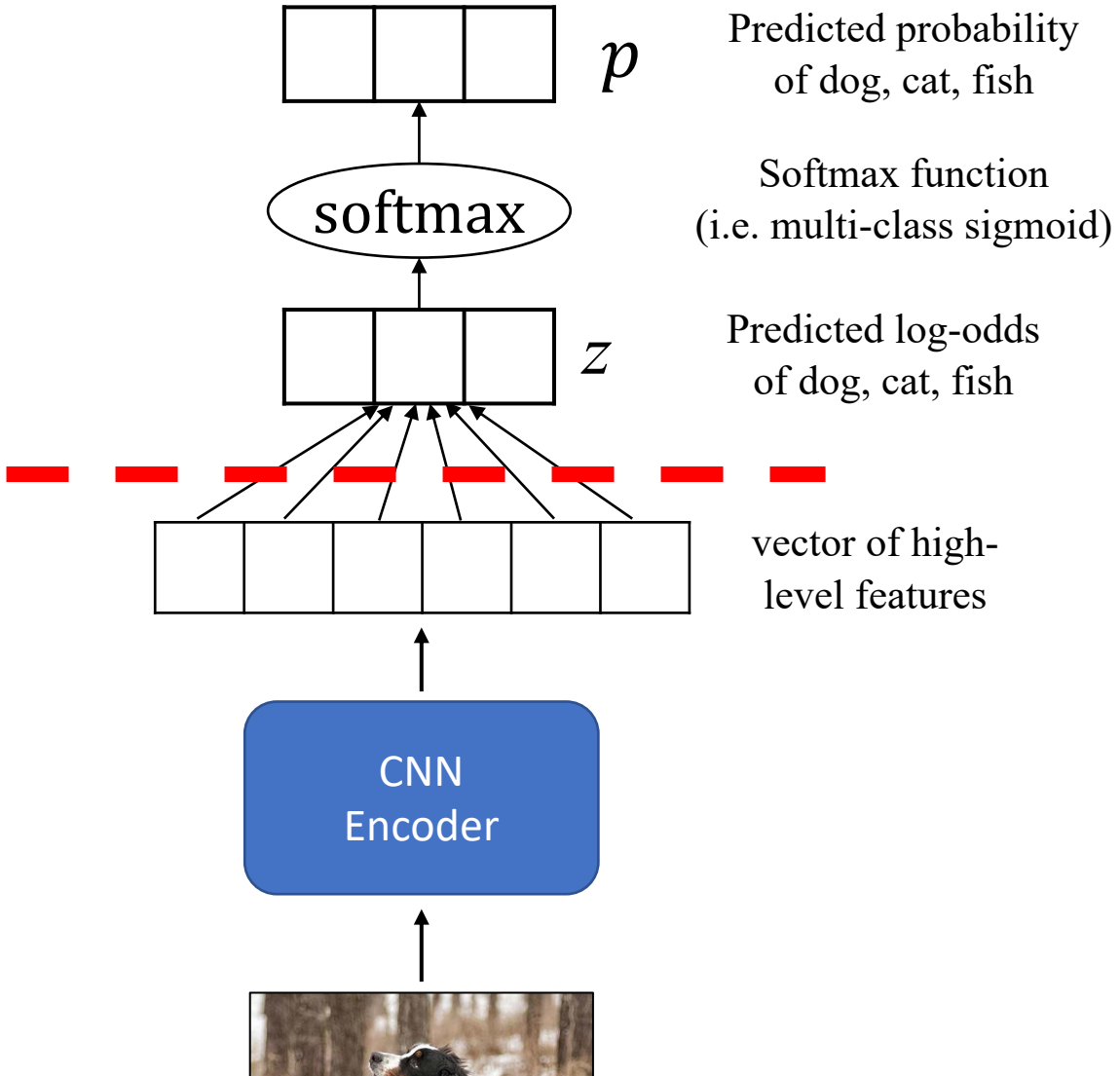


Step 2: Replace the prediction head.

In this example, our prediction head is simple logistic regression on the features extracted by the CNN encoder.

Typically, we start with a multi-class (rather than binary) prediction head.

We want to make predictions about a new set of labels, so we cut off the prediction head…

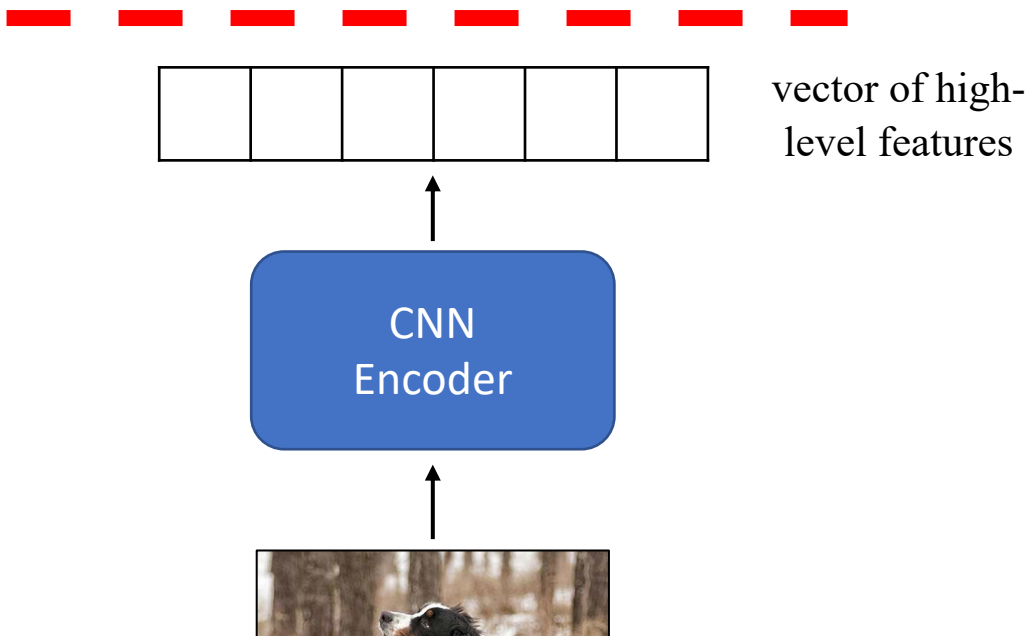


Step 2: Replace the prediction head.

In this example, our prediction head is simple logistic regression on the features extracted by the CNN encoder.

Typically, we start with a multi-class (rather than binary) prediction head.

We want to make predictions about a new set of labels, so we cut off the prediction head…



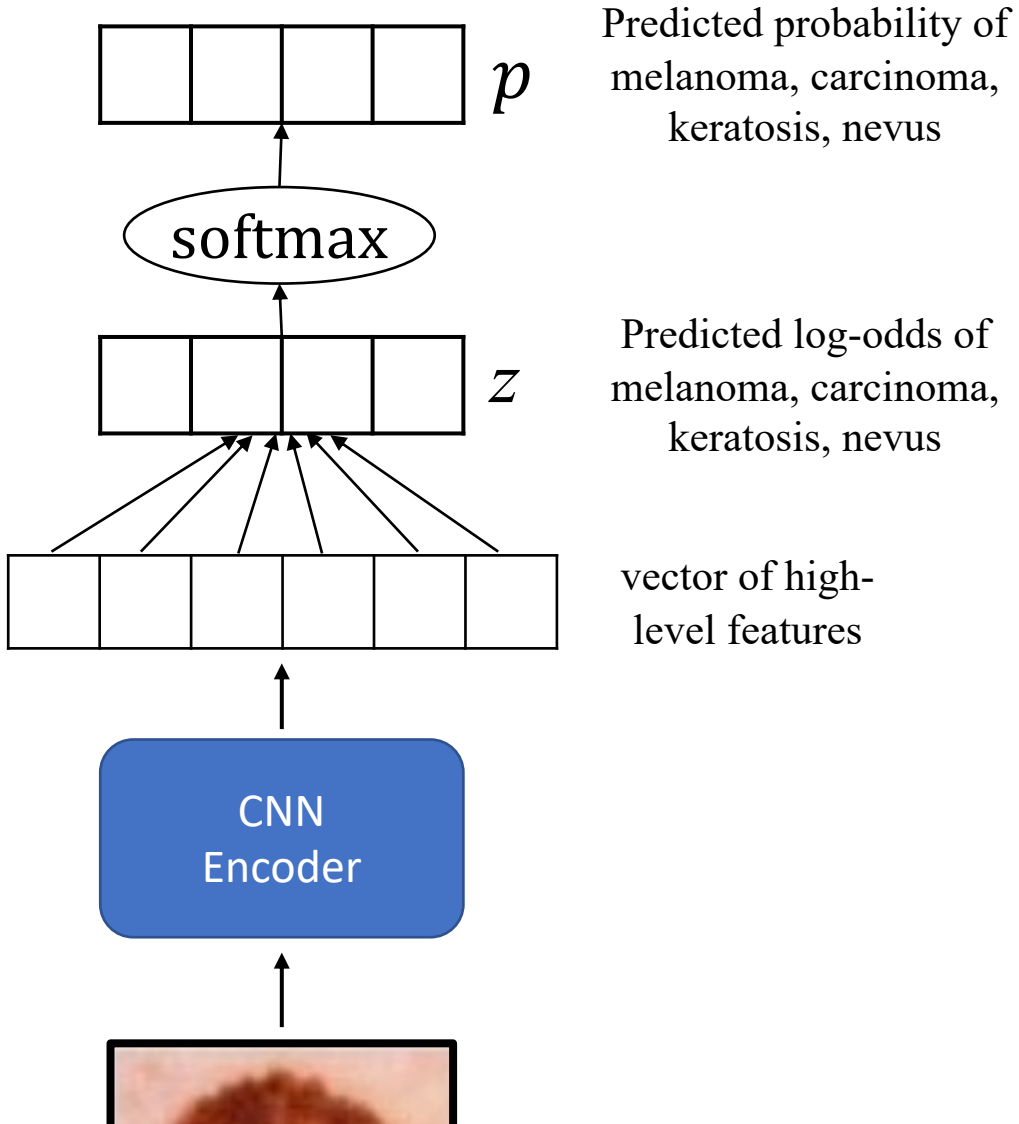
Step 2: Replace the prediction head.

In this example, our prediction head is simple logistic regression on the features extracted by the CNN encoder.

Typically, we start with a multi-class (rather than binary) prediction head.

We want to make predictions about a new set of labels, so we cut off the prediction head…

…and replace it with a new one appropriate for our new prediction task.



Step 3: Train it on your data.

- Finally, you train the model with your dataset of medical images.
- You train the same way you'd train any other model.*
- The prediction head is new, so parameters in this layer are learned entirely from your data.
- The CNN encoder is not new, so we say parameters in the encoder are *fine-tuned* to your data.
- You'll need a big dataset (e.g. 1k), but since you started with a pre-trained model, not as big as if you trained from scratch (e.g. 1M).
- This process (steps 1-3) is called *transfer learning*.

*with a few caveats

Some notes on fine-tuning:

1) fine-tuning a pre-trained model tends to be **at least as good as learning from scratch**
   (empirical result)

2) freeze early layers and fine-tune later layers
   more data → fine-tune more layers

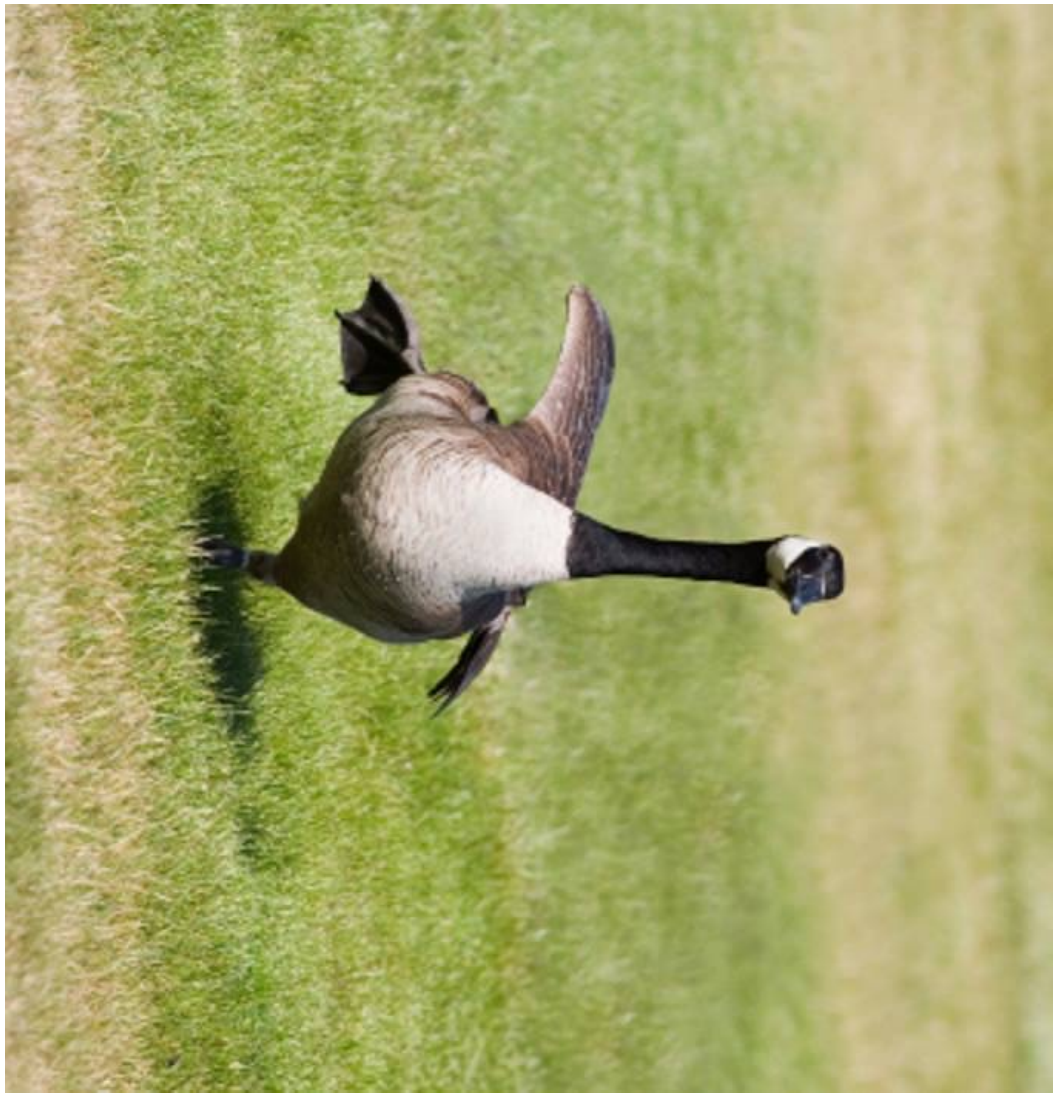
3) best tuning depth depends on the application, and should be explored as a hyperparameter

Image Data Preprocessing

…also called data augmentation

CNN: I know this one, it’s a goose!

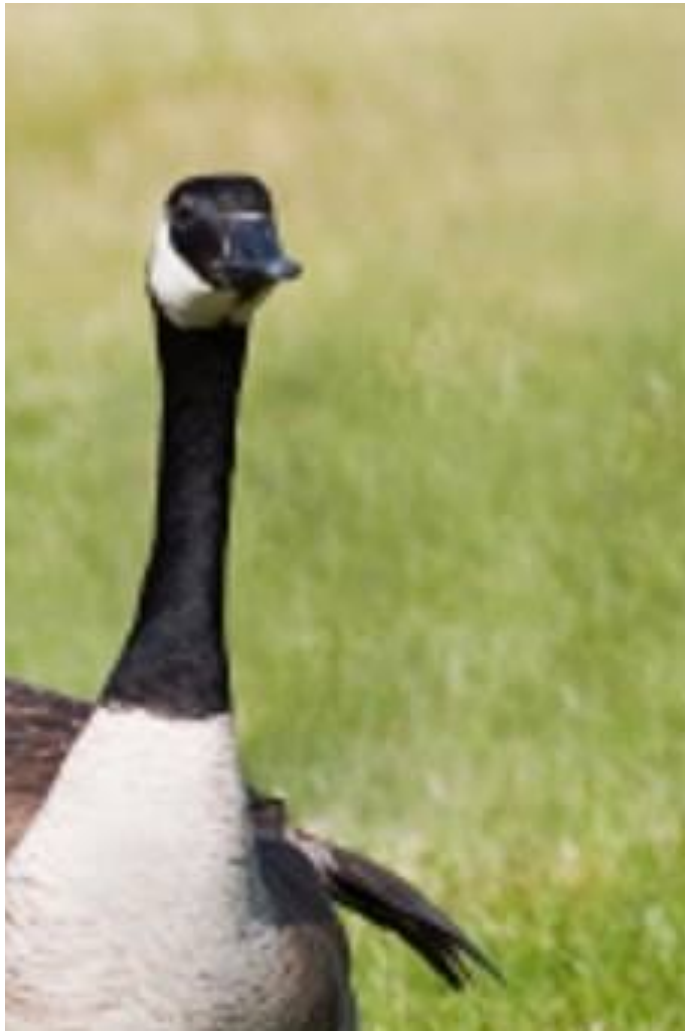
CNN: I have no idea what this is.



CNN: I’ve never seen anything like this in my life.

CNN: I don’t see anything.

CNN: I think this is a space shuttle.



Solution: Augment your dataset

- Apply random rotations, crops, brightness adjustments, stretching, etc
- The details depend on the application.

Summary

- CNNs can classify images or detect objects within images, including medically relevant findings.
- CNNs work through hierarchical extraction of increasingly complex features.
- We can re-purpose a trained CNN to classify medical images by:
  - Finding and downloading code and parameters defining the existing model
  - Replacing the *final layer* (i.e., *prediction head*) with a new one suitable for our task
  - Training the model with our dataset to *learn* parameters in the prediction head and *fine-tune* parameters in the CNN encoder
- Applying random rotations, crops, brightness adjustments, etc during training makes the model robust to these variations if/when they are encountered