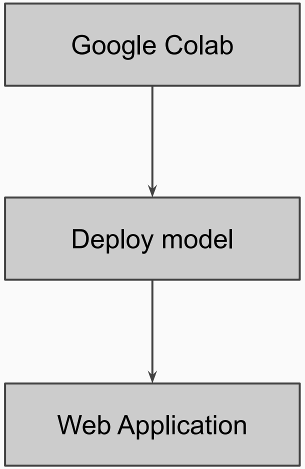
# IoTFuse 2020 Data Workshop

## Transfer Learning for IoT

*Presented by Madhu Tennakoon*

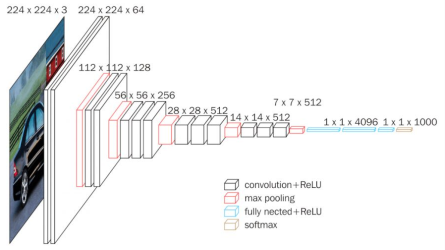
# Goals

- Design a transfer learning experiment on Google Colab
- Deploy a computer vision service
- Design a web app to consume the web service



# What we will use

- Keras with Tensorflow backend
  - Easy to use
  - API for using VGG, Inception, and ResNet
- ML deployment service
- Pretrained image-classifier model
- Web app



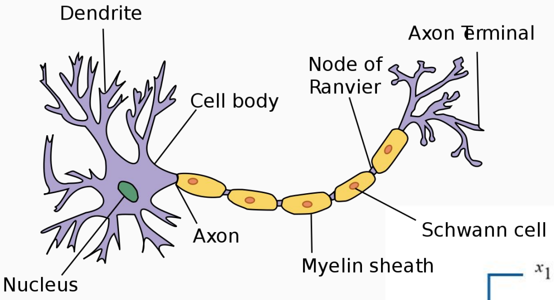
# What we will need

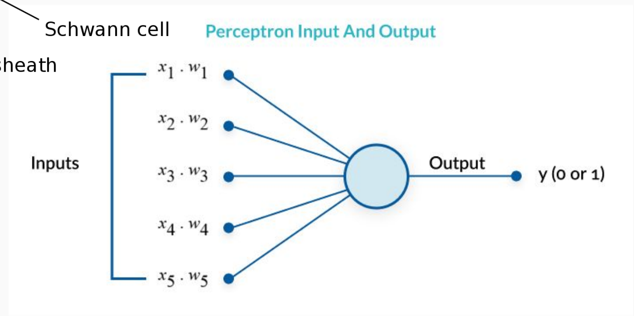
- Web Browser
- **Google account** to kick off experimentation with Google Colab

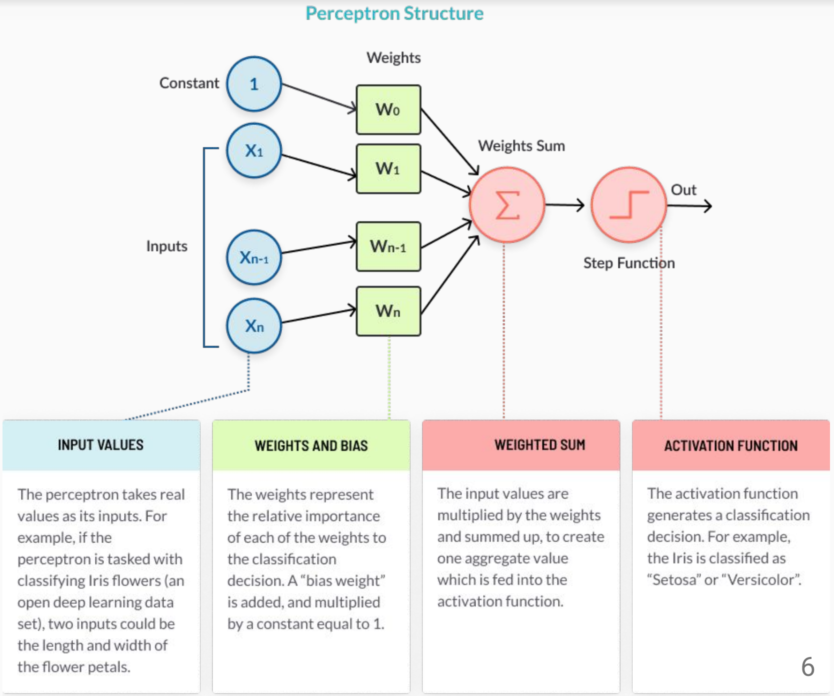
# Part I: Crash Course - Neural Networks

# The neuron

- Modeled after the essential unit of the human brain
- Can have multiple inputs
- Each input has weights that indicate its significance

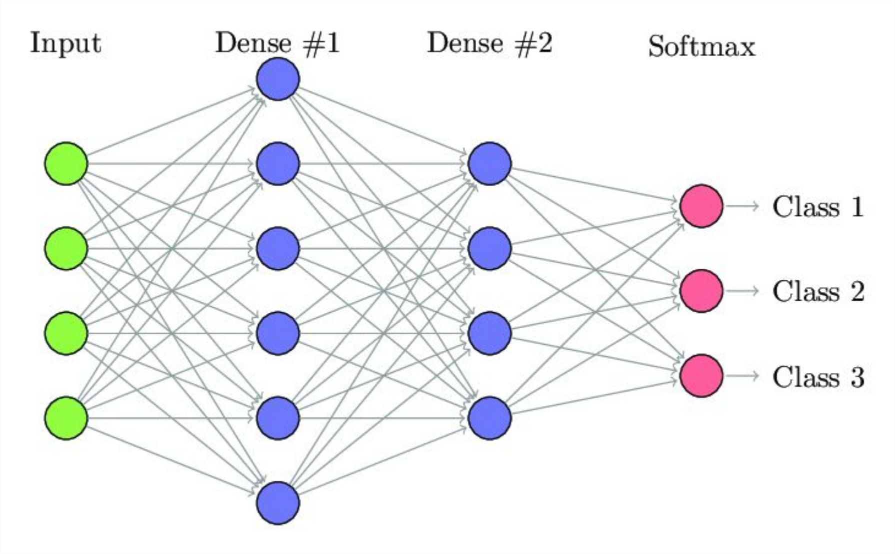


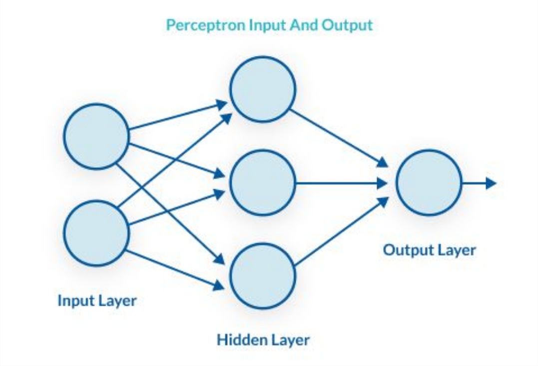



# Neural Network

`Keras layers: Dense, Softmax`



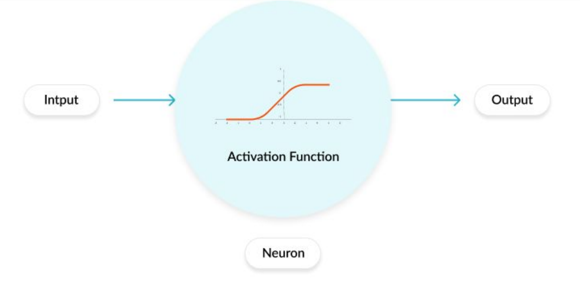

- Composed of layers of neurons
- Can be used as regressors or **classifiers**



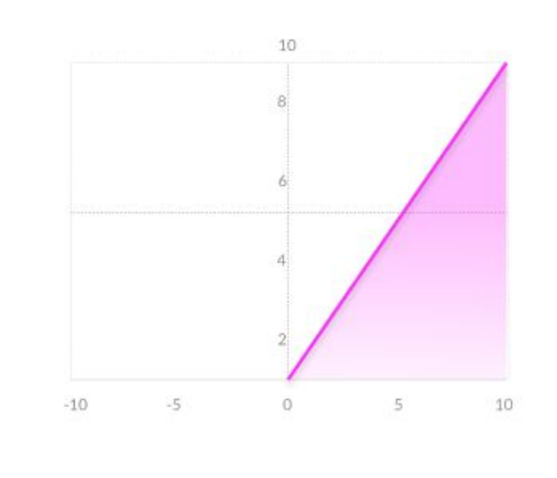
# Activation Function

Keras layer: Conv2D

- Determines output of a neuron
- On/off: is a neuron *fired* or not?



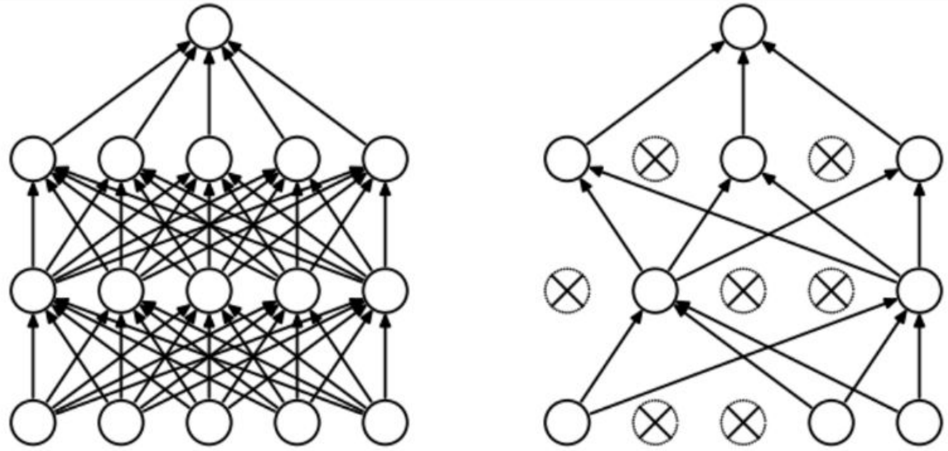

ReLU




- **R**ectified **L**inear **U**nit
- Computationally efficient
- Allows faster convergence

# Dropout

Keras layer: Conv2D



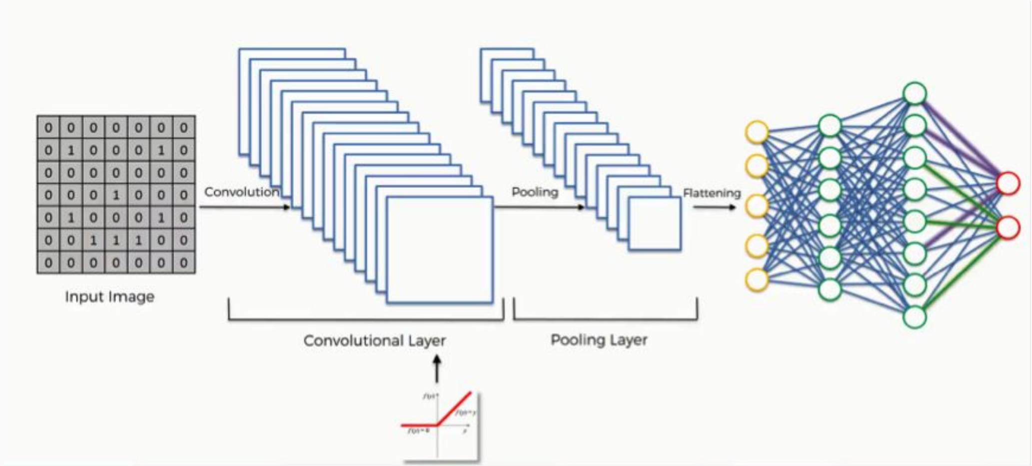
(a) Standard Neural Net (b) After applying dropout.

- We randomly remove neurons during model training
- Prevents "over-fitting”

# Part II: Crash Course - Convolutional Neural Networks

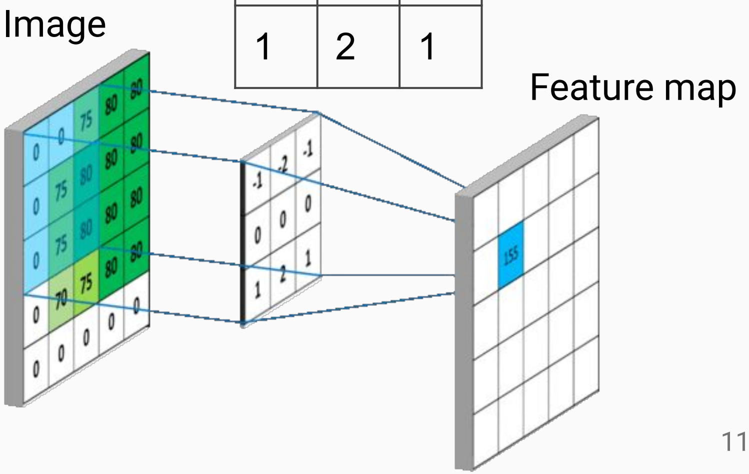
# Convolutional Neural Network

Keras layer: `Conv2D`




3x3 Kernel

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

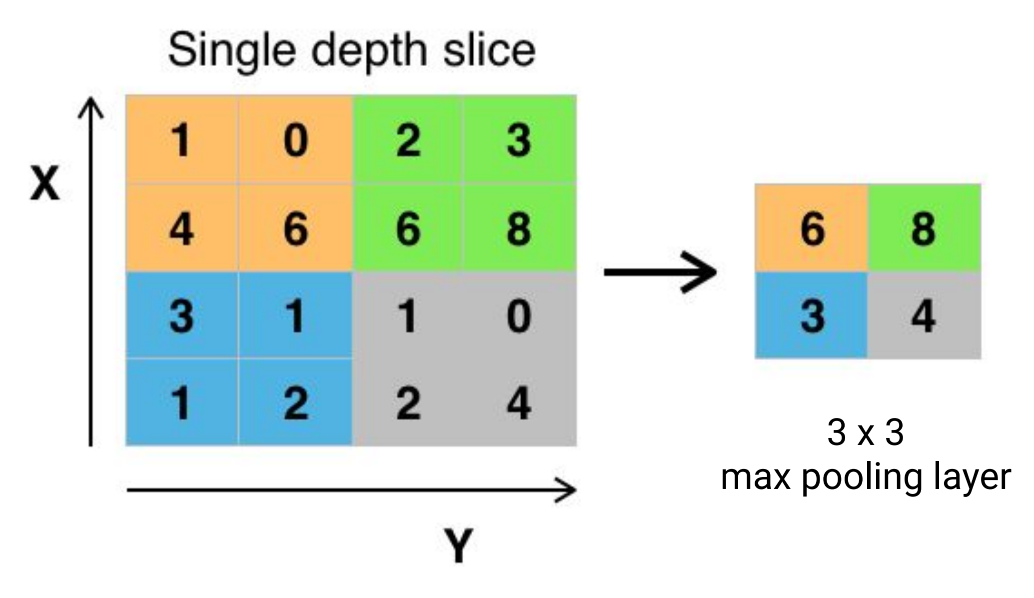



Each convolution layer:

- Extracts higher-level features
- Maps the input into a smaller feature space

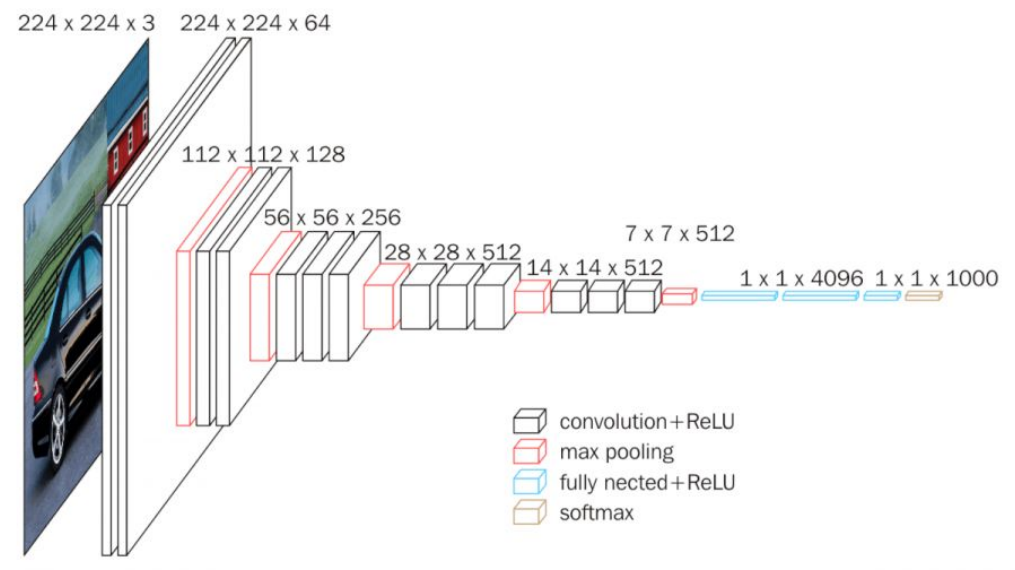
# Max Pooling

`Keras layer: AveragePooling2D`




- Replaces highest values in an output matrix with the local maximum
- Reduces resolution → Less computation
- Helps in optimization

# An example: VGG16

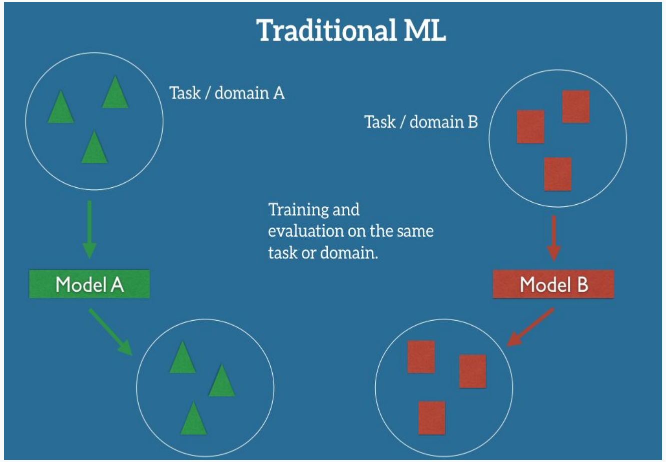


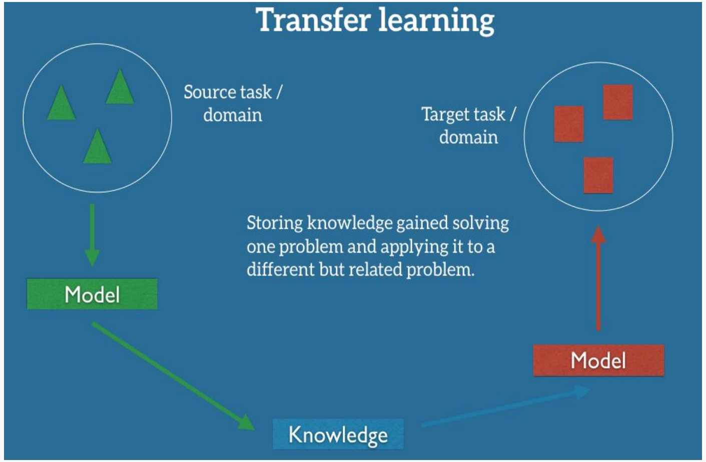

- Achieves 92.7% top-5 test accuracy in ImageNet
- Trained with 14M images
- 1000 classes

# Part III: What is Transfer Learning?

# Transfer Learning

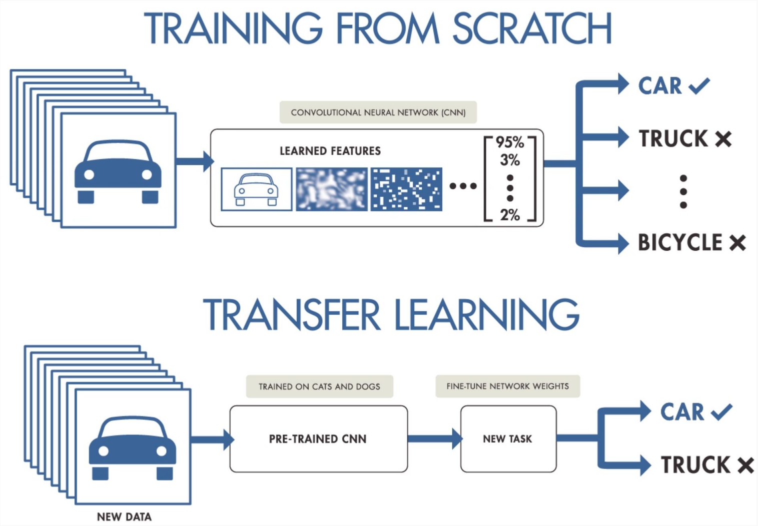
- A technique for taking a network that has already been trained for a domain and adapting it to your domain
- We can exploit what was previously learned to improve the model's generalization in another setting



# Transfer Learning

*“One of the most powerful ideas in deep learning is that you can take knowledge the neural network has learned from one task and apply that knowledge to a separate task”  -- Andrew Ng*

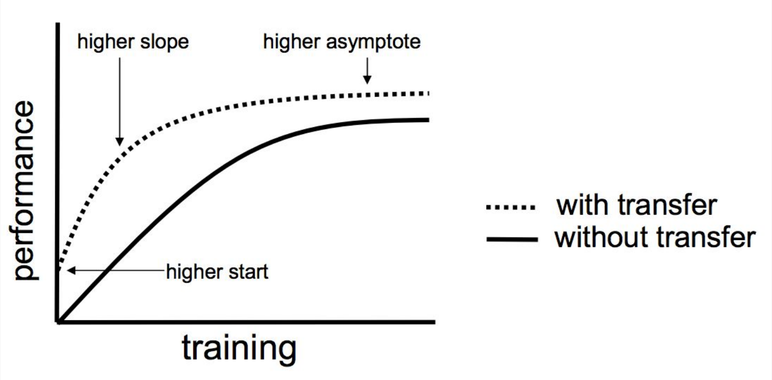


# Transfer Learning Applications

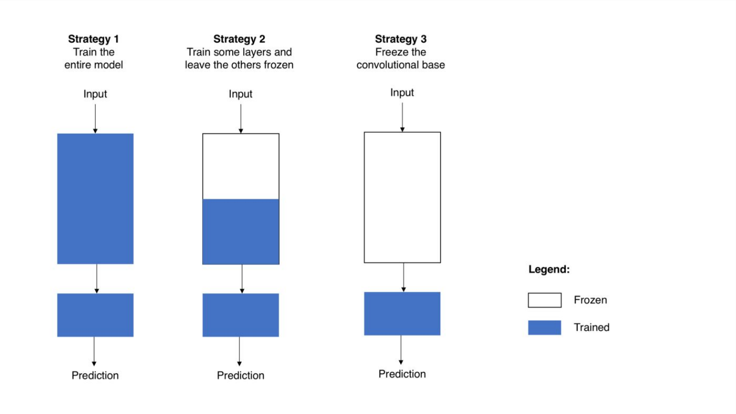
- Natural language translation
- Using financial anomaly detection models to find mineral gold deposits
- Very common in Computer Vision: you would rarely train from scratch anymore

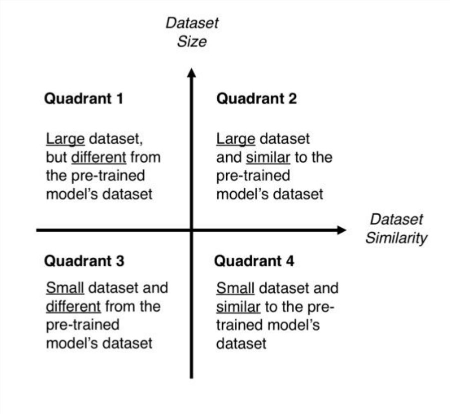
# Why we need it

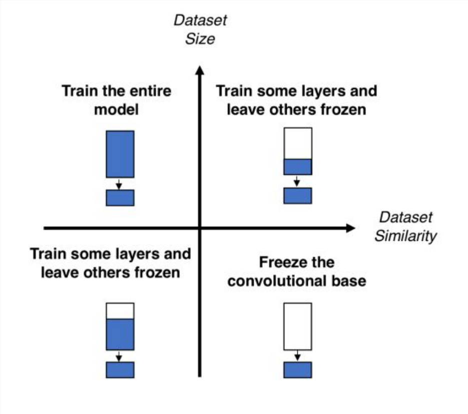
- Deep convolutional neural network models may take days or even weeks to train on very large datasets.
- Dynamic domains with limited datasets are common in the IoT space.
- Speed of training and higher generalization capability is crucial.
- We can get closer with transfer learning.



# When to use it



# Our Task: Learning from COVID-19 Chest X-Rays

# ML experimentation workflow

1. Understanding your data
2. Formulating the problem - is it a classification problem or a regression problem?
3. Data preparation - Do we augment data? How do we create training and testing data?
4. Learning and predicting - start with a pretrained convolutional neural network
5. Performance evaluation - choosing the right metrics; accuracy, precision, recall

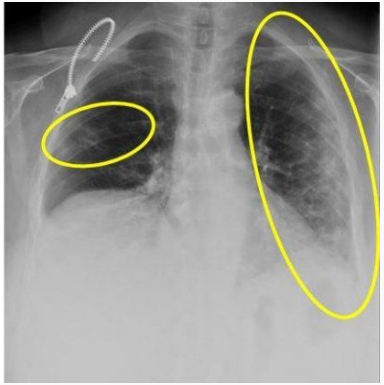
# Part I: Understanding your data

# Understanding your data

- What are you trying to predict?
- Which datasets do you use?
- How many classes exist?
- What assumptions are being made of the data?

*Healthy*

*Infected*



# Understanding your data

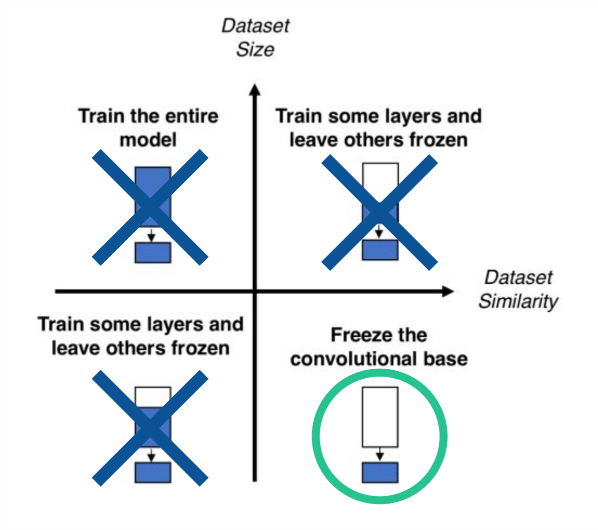
**Caution!**

1. The goal of this tutorial is **not** medical image analysis; it's only to demonstrate the potential of transfer learning
2. This is a very small dataset
3. We have to be aware of biases
   a. Is there patient overlap? If the COVID-positive dataset is from adults whose ages range from 30~70, whereas the healthy dataset is from children, the detector might be learning the difference between anatomies of adults and children (bone structure, medical device artifacts, identification tags).
   b. The detector may learn subtleties in the image quality (blur, brightness)

# Part II: Setting up the problem

# Setting up the problem

- Do we have enough data?
- Is the problem similar to a prior learning task?



# Setting up the problem

Binary classification problem: **infected** (positive) vs. **non-infected** (negative)

| | **PREDICTION** | |
|---|---|---|
| **ACTUAL** | ***Negative*** | ***Positive*** |
| ***Negative*** | TN | FP |
| ***Positive*** | FN | TP |

Patients we falsely diagnose (EXPENSIVE)

Patients we fail to diagnose (BAD)

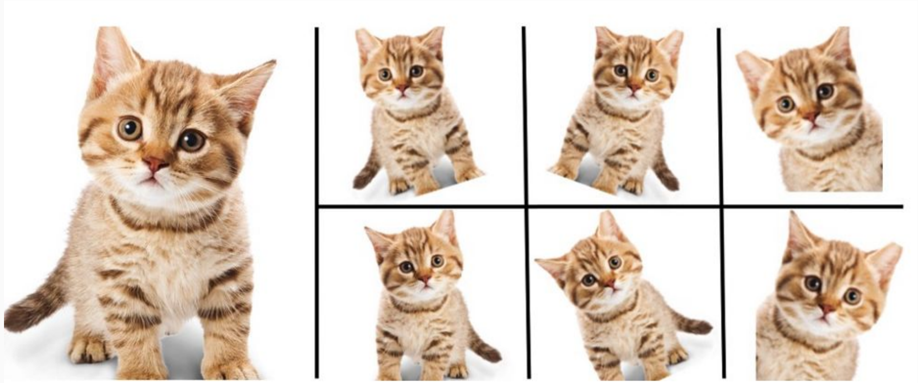
**Recall**: TP / ( TP + FN )

**Precision**: TP / ( TP + FP )

# Part III: Data Preparation

# Data Augmentation

Rotate Flip Scale Translate



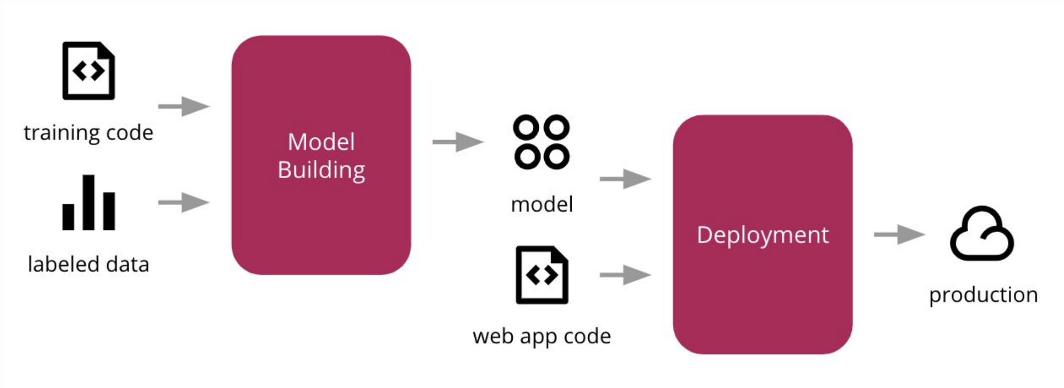
- Helps increase the amount of relevant data
- Makes your model more versatile

# Machine Learning Experimentation

# Part 2: Model Deployment

# Model deployment

1. Wrapping up the model in an API
2. Creating a web service
3. Web service input: **features**
4. Web service output: **prediction**



# Part 3: Creating visualizations

# References


- https://hbr.org/2016/12/how-to-make-better-predictions-when-you-dont-have-enough-data?platform=hootsuite
- https://github.com/lindawangg/COVID-Net/
- https://healthcare-in-europe.com/en/news/imaging-the-coronavirus-disease-covid-19.html