WALC 2023
Applied AI

Preventing Overfitting & DL Wrap-Up

Prof. Marcelo J. Rovai
rovai@unifei.edu.br

UNIFEI - Federal University of Itajuba, Brazil
TinyML4D Academic Network Co-Chair

Preventing Overfitting

Preventing Overfitting

+Data

+Data

+Data

+Data

Preventing Overfitting

+Data

+Data

+Data

+Data

What can we do if we need more data?

Preventing Overfitting

+Data

+Data

+Data

+Data

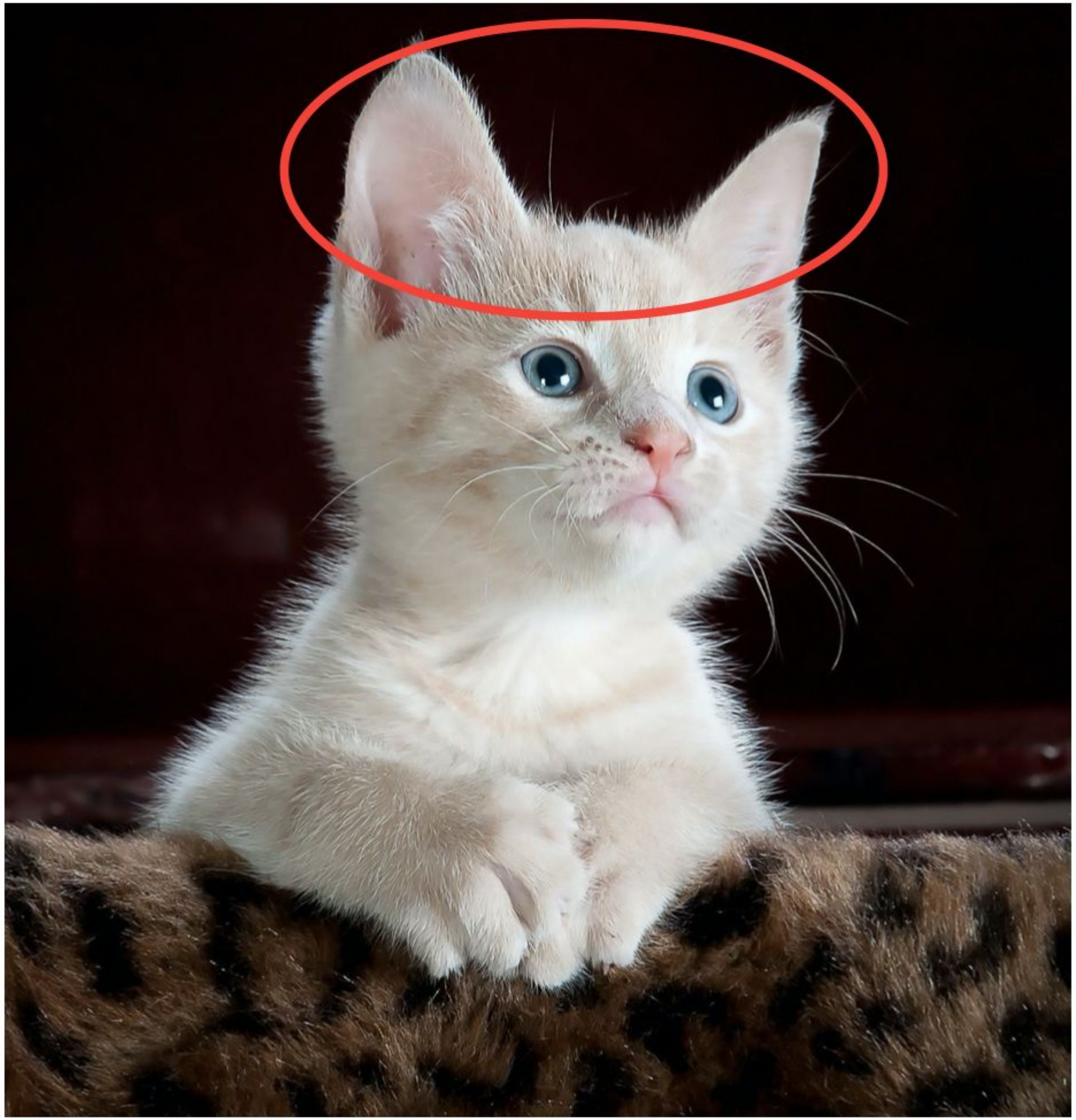
What can we do if we need more data?

- Data Augmentation (artificial)
- Transfer Learning
- Early Stopping
- Dropout Regularization

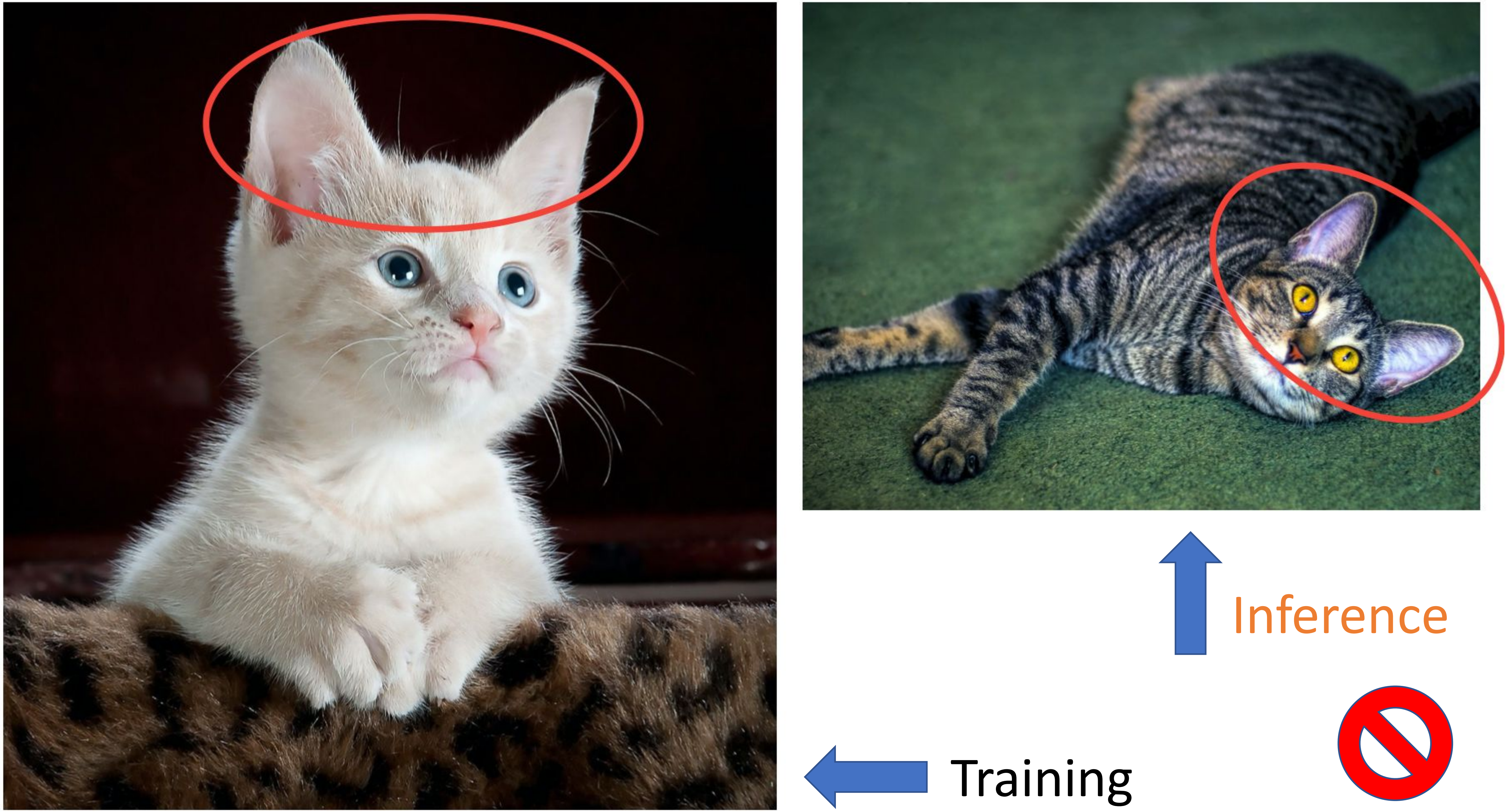
Preventing Overfitting

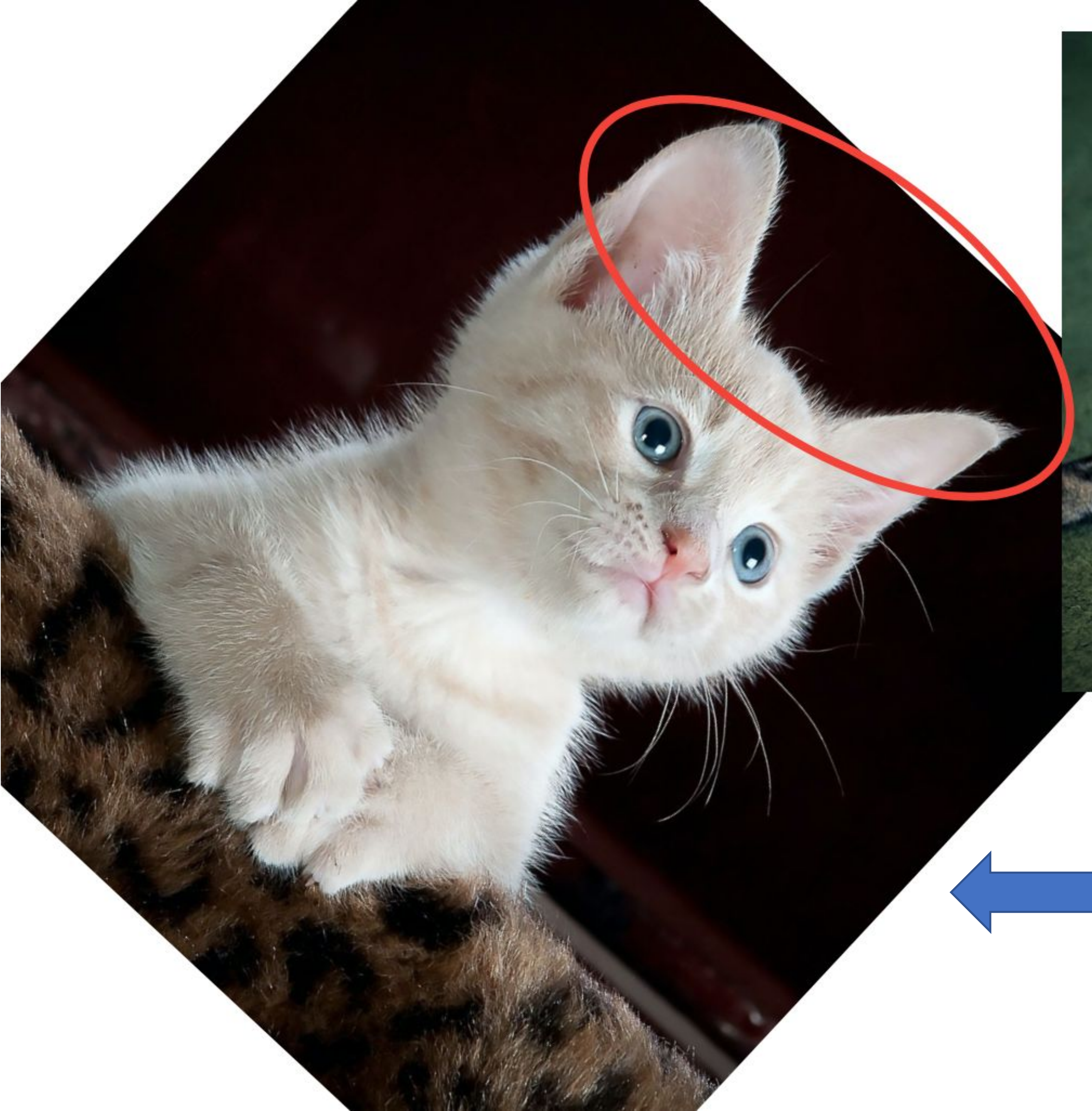
More Data, Data Augmentation (artificial)

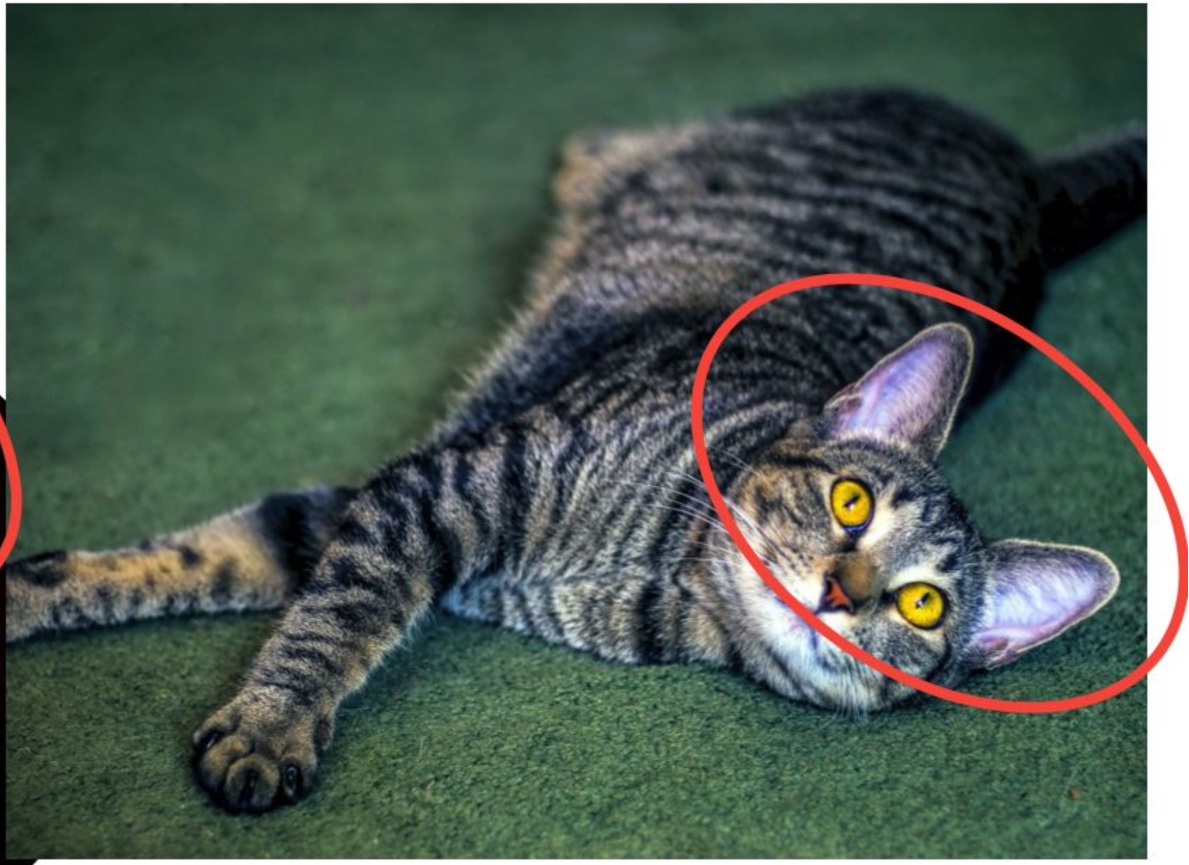
Overfitting generally occurs when there are a small number of training examples. Data augmentation takes the approach of generating additional training data from your existing examples by augmenting them using random transformations that yield believable-looking images. This helps expose the model to more aspects of the data and generalize better.



Training


Inference
Training



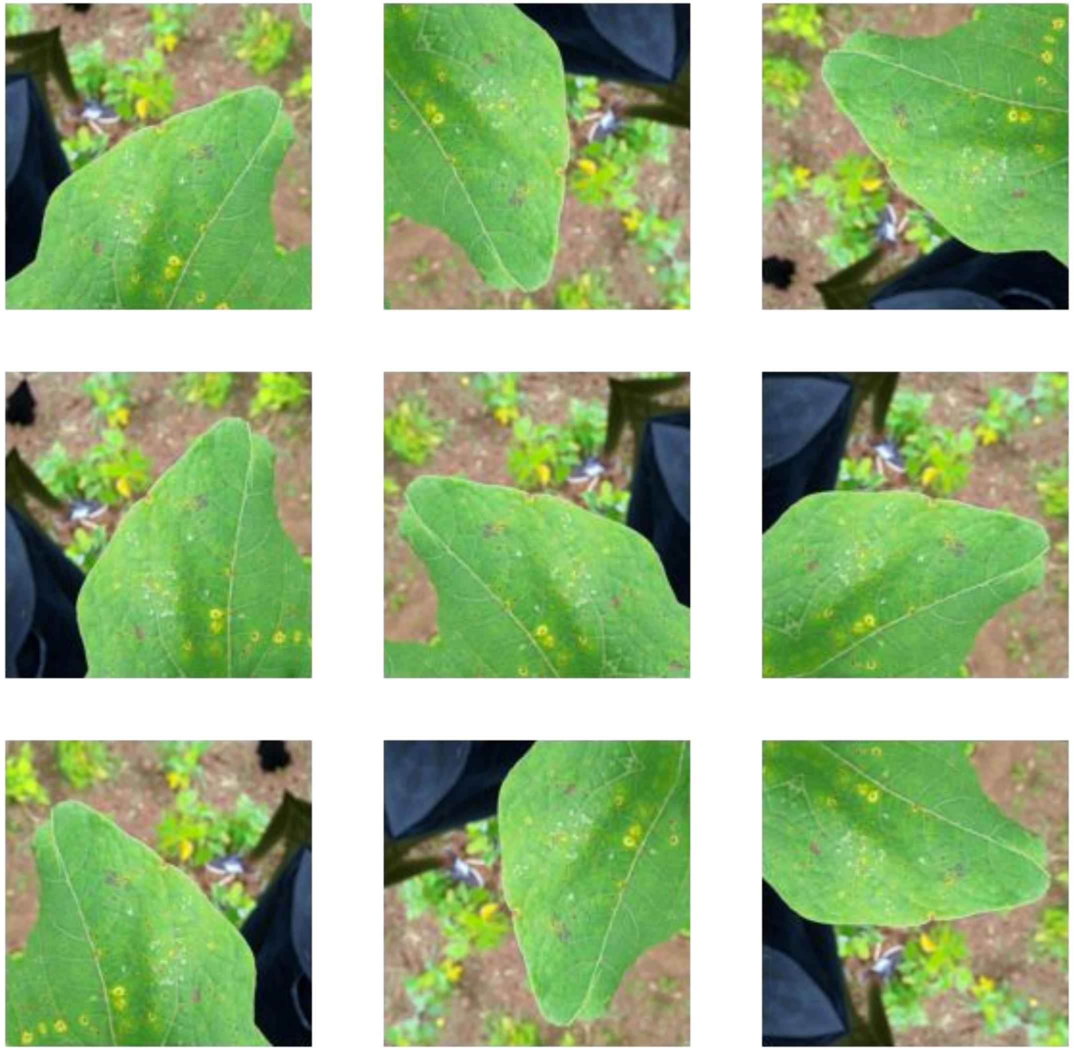


Training with
Data Augmentation

Using Keras preprocessing layers

```
data_augmentation = tf.keras.Sequential([
  layers.RandomFlip("horizontal_and_vertical"),
  layers.RandomRotation(0.2),
])
```

```
plt.figure(figsize=(10, 10))
for i in range(9):
  augmented_image = data_augmentation(image)
  ax = plt.subplot(3, 3, i + 1)
  plt.imshow(augmented_image[0])
  plt.axis("off")
```

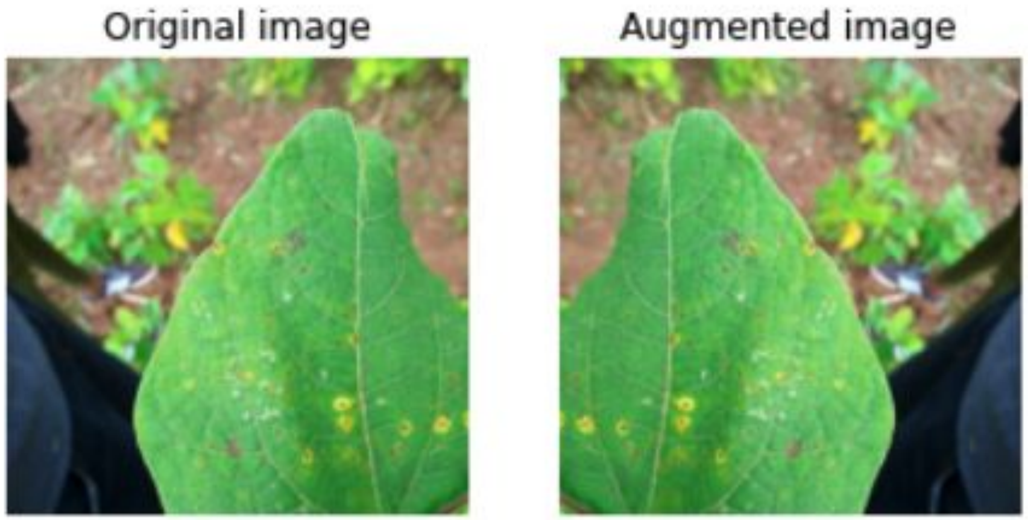


There are a variety of preprocessing layers you can use for data augmentation including:

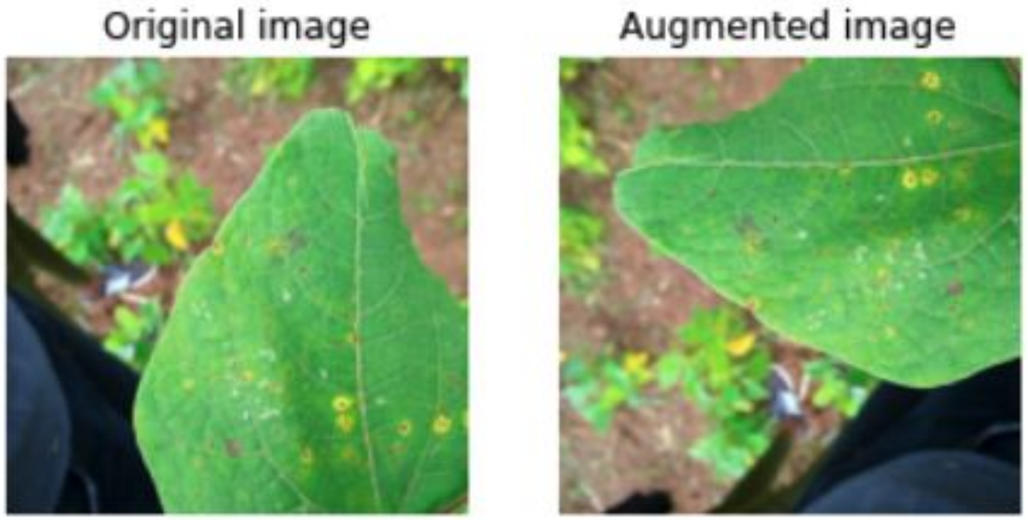
- tf.keras.layers.RandomContrast,
- tf.keras.layers.RandomCrop,
- tf.keras.layers.RandomZoom,
- and others.

Using tf.image

```
flipped = tf.image.flip_left_right(image)
visualize(image, flipped)
```

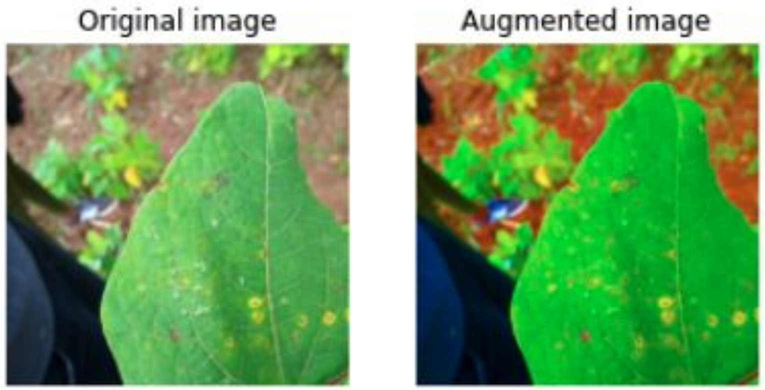



```
rotated = tf.image.rot90(image)
visualize(image, rotated)
```

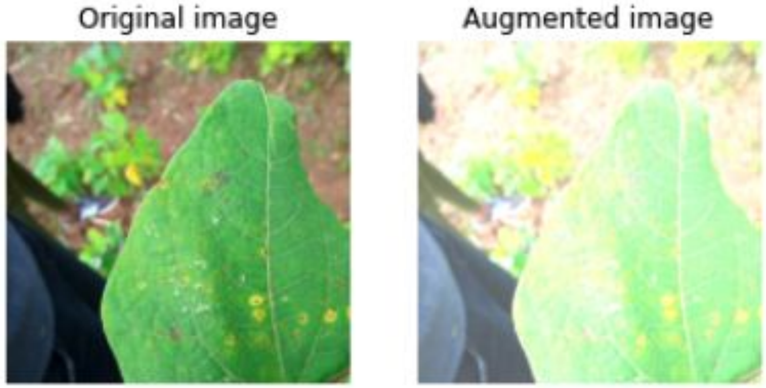


Using tf.image

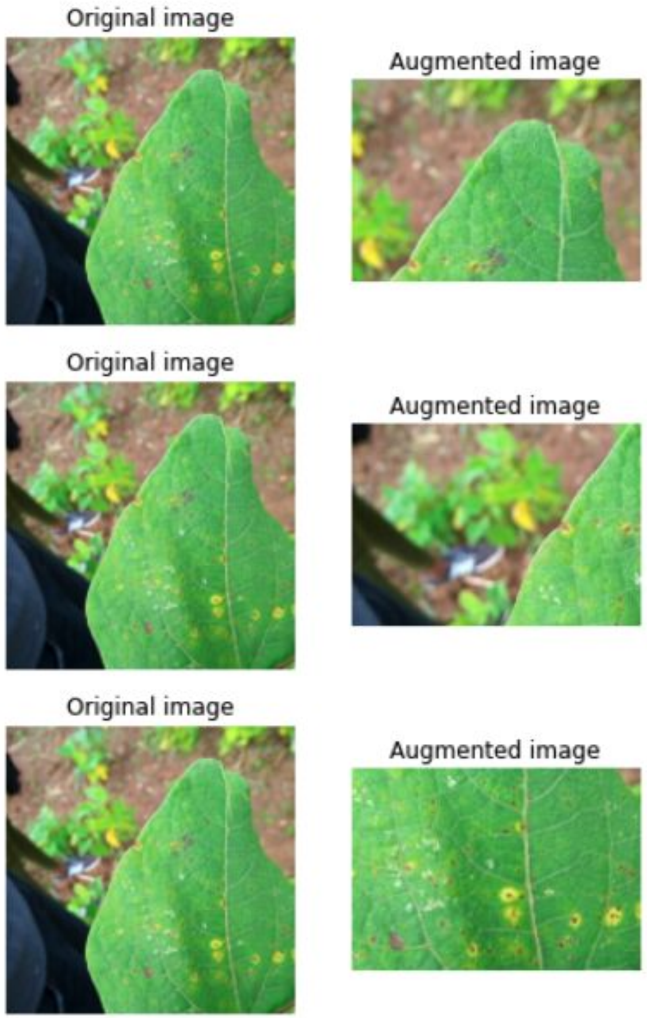
```
saturated = tf.image.adjust_saturation(image, 3)
visualize(image, saturated)
```




```
bright = tf.image.adjust_brightness(image, 0.4)
visualize(image, bright)
```



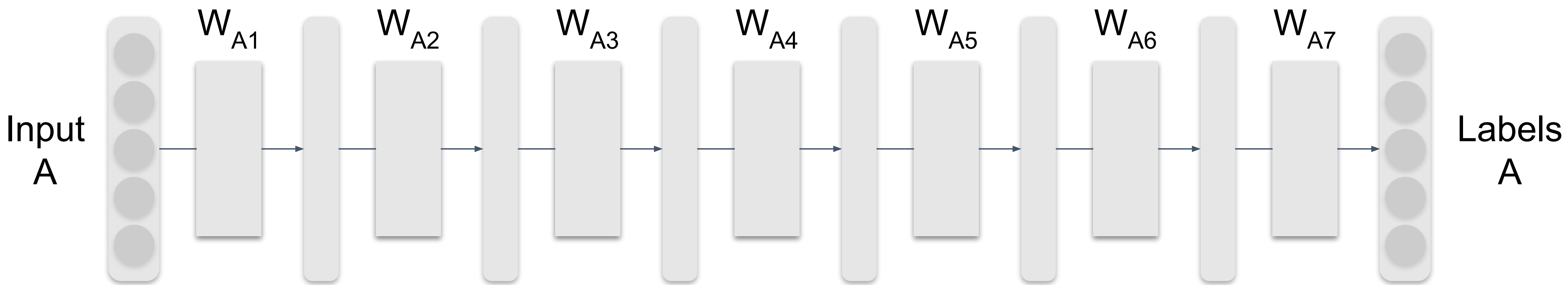

```
for i in range(3):
  seed = (i, 0)  # tuple of size (2,)
  stateless_random_crop = tf.image.stateless_random_crop(
      image, size=[210, 300, 3], seed=seed)
  visualize(image, stateless_random_crop)
```



Preventing Overfitting

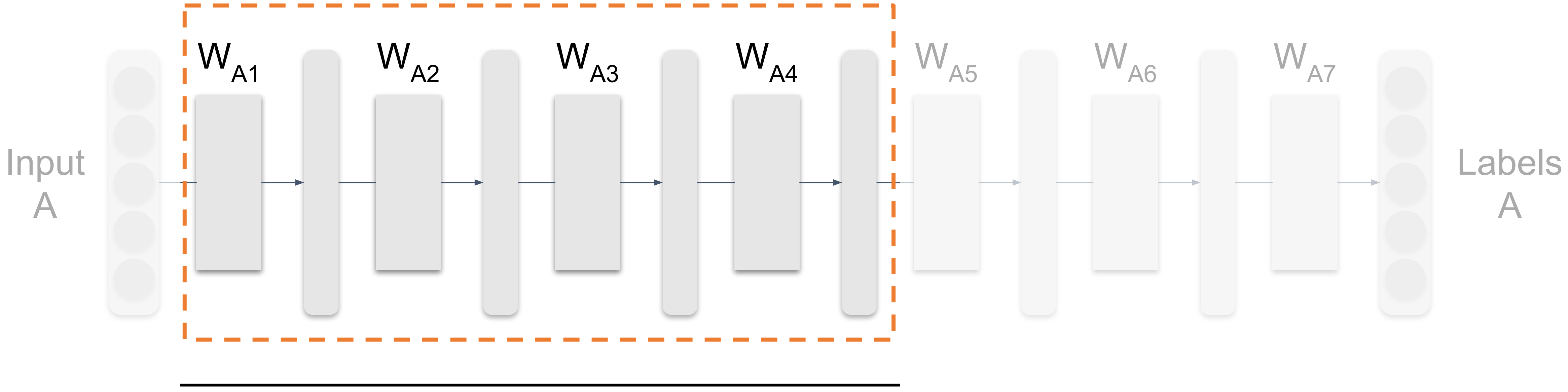
Transfer Learning

End Result of Training



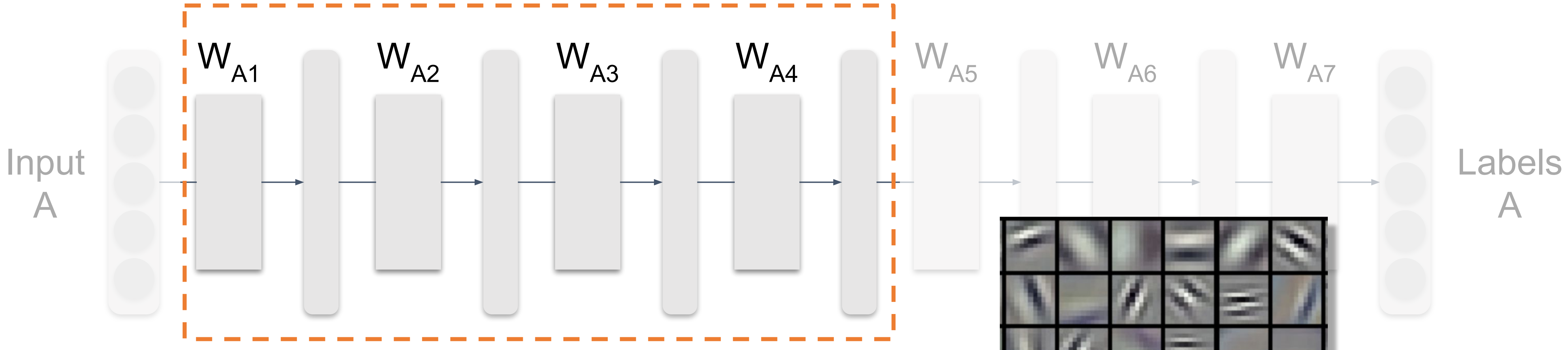
The end result of the training is to learn the weights of the neural network model.

End Result of Training

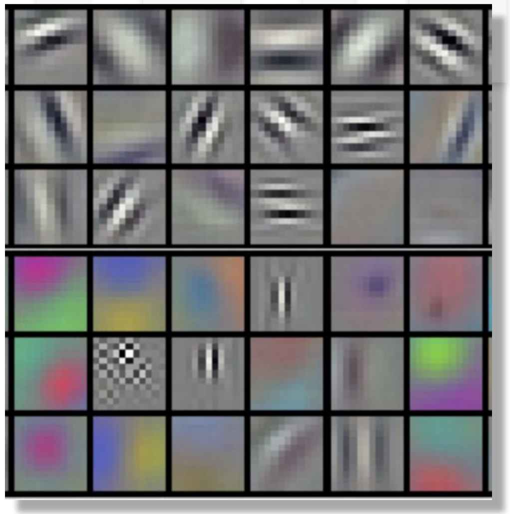


Learns ***general features*** irrespective of task

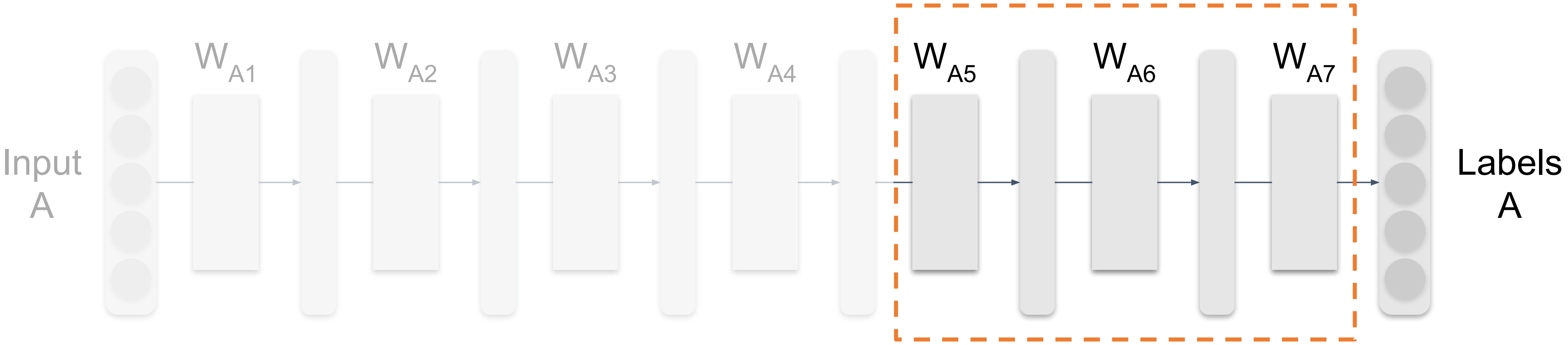
End Result of Training



Learns ***general features*** irrespective of task



End Result of Training

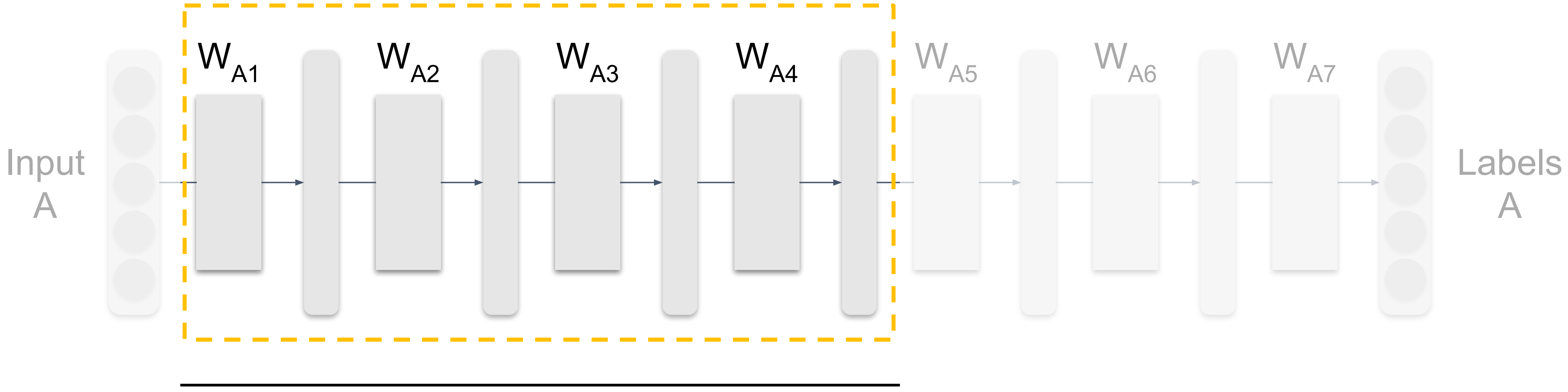


Task-specific
features


Source: Google

Transfer Learning

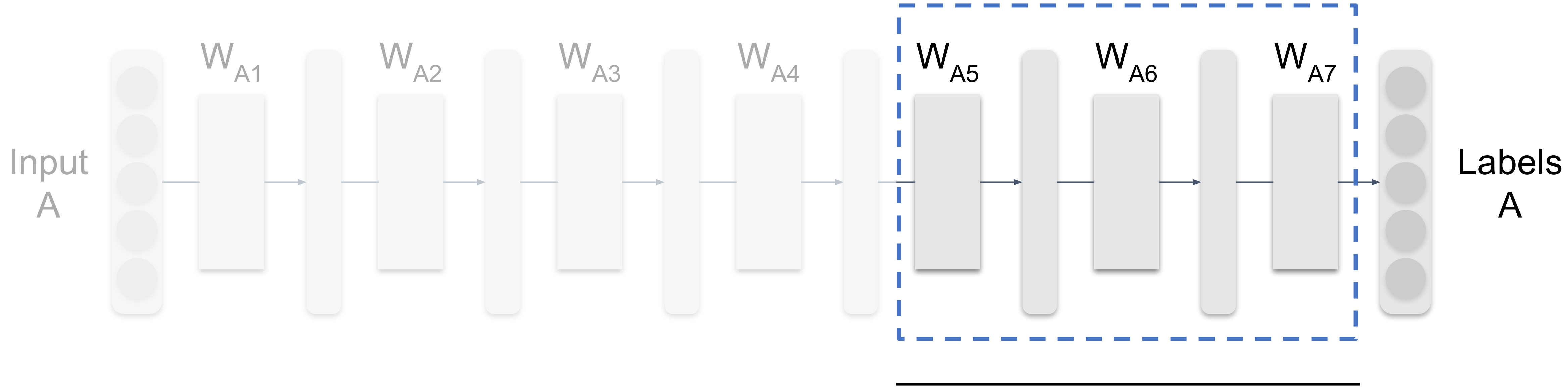
Reuse (freeze general feature extraction)



Learns ***general features*** irrespective of task

Transfer Learning

Train **only** last few layers

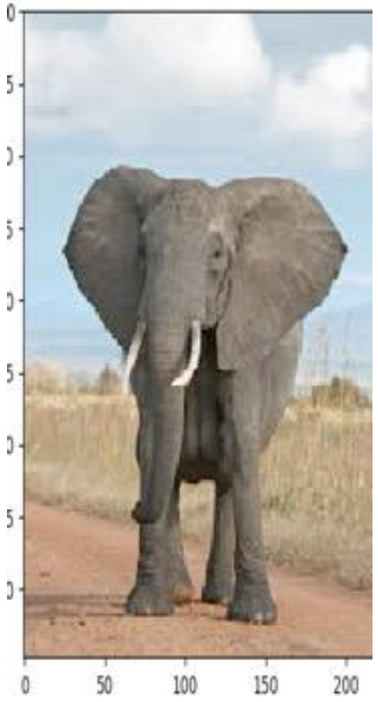


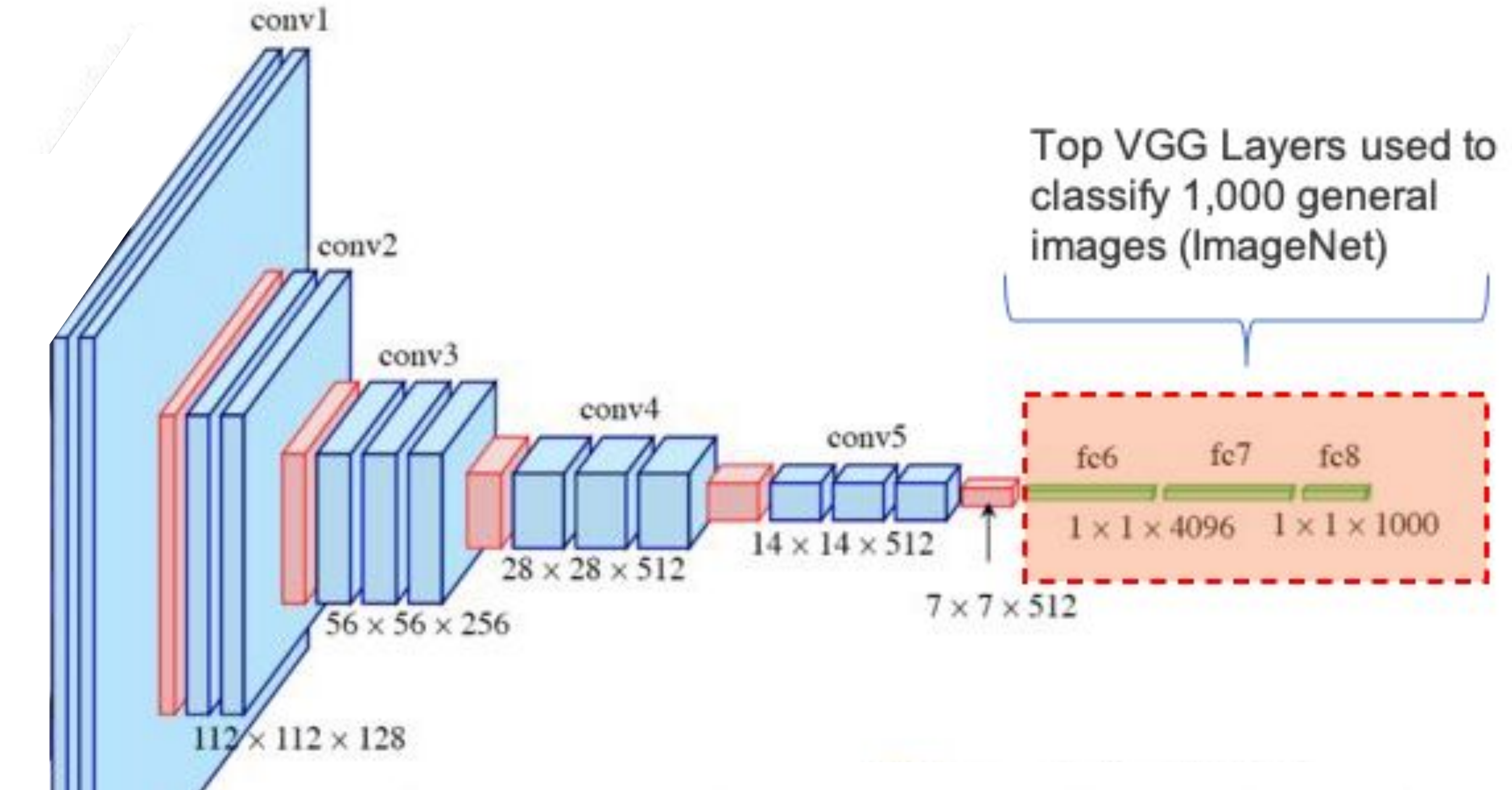
Task-specific features

covidXray

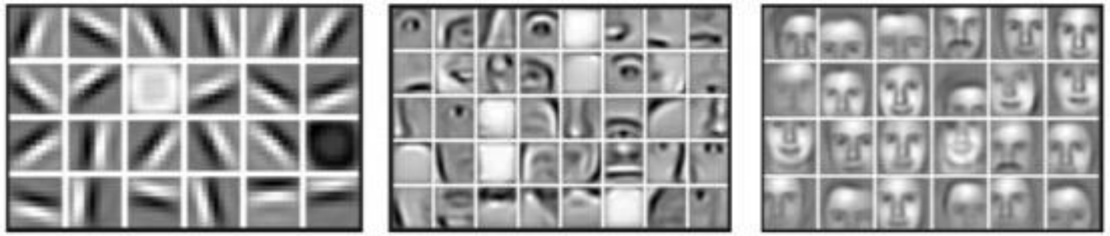
Detecting Covid-19 in Chest X-Ray images

VGG-16 Convolutional Neural Network Model



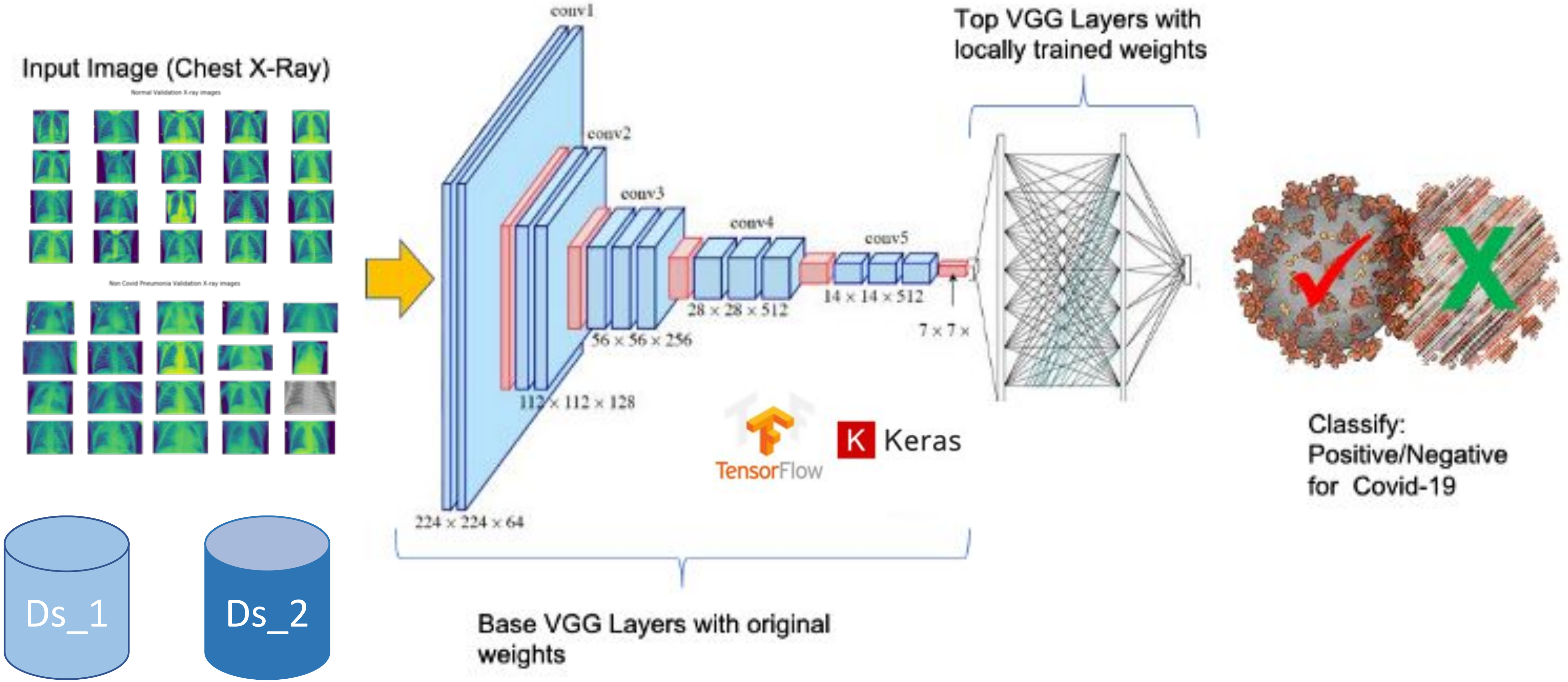


Elephant
93.0%

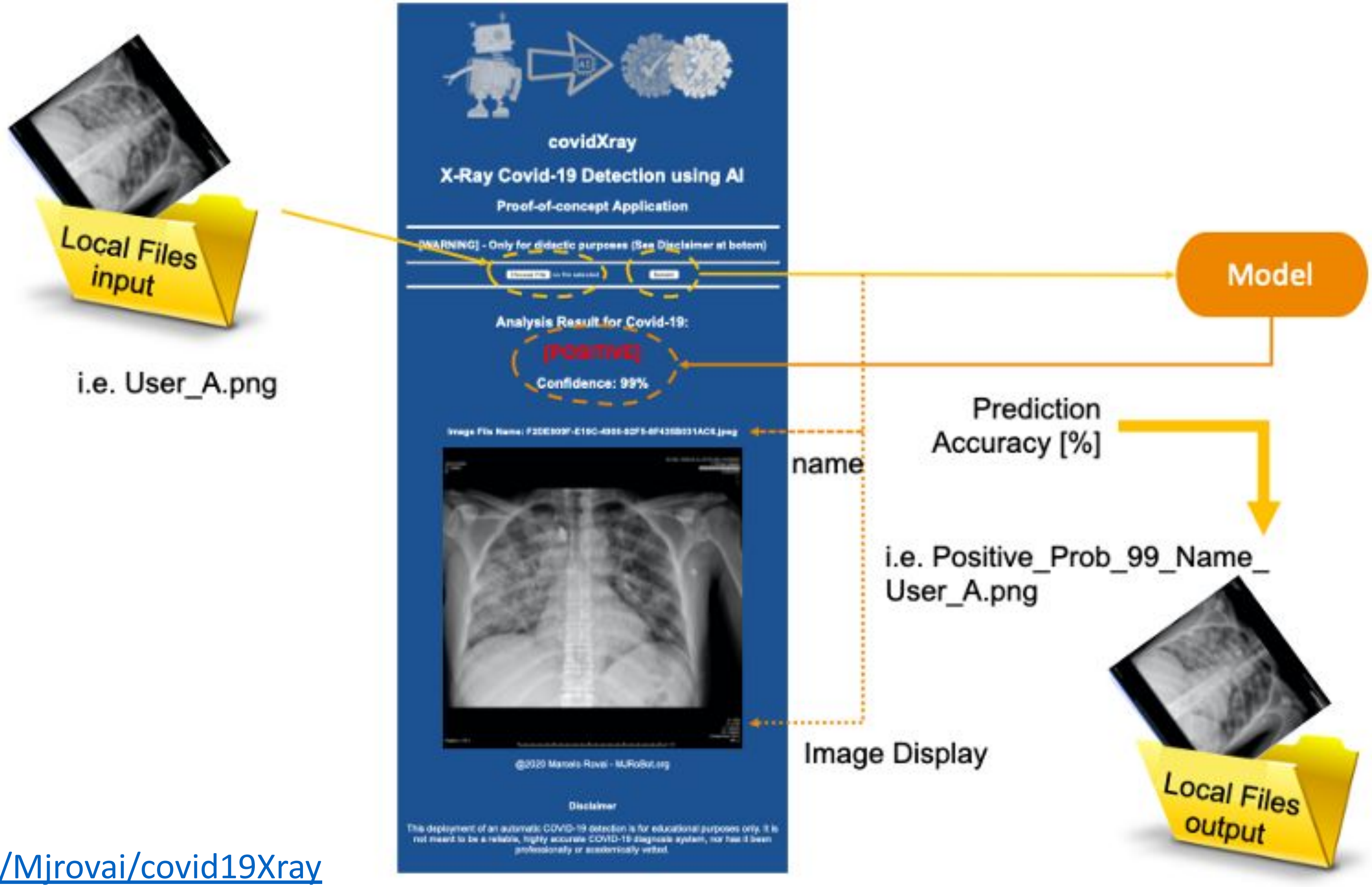


First Layer Representation Second Layer Representation Third Layer Representation

Training the model (Transfer Learning)



Inference



https://github.com/Mjrovai/covid19Xray

Preventing Overfitting

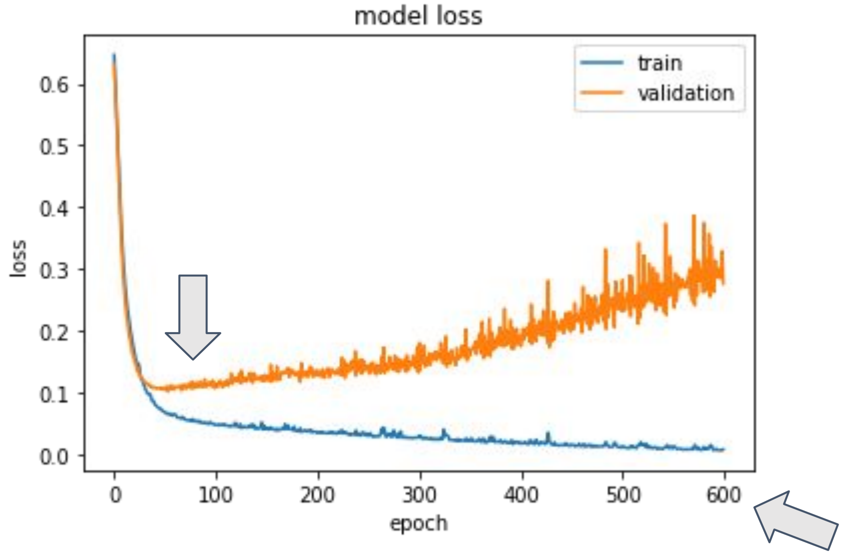
Early Stopping & Dropout Regularization

Early Stopping

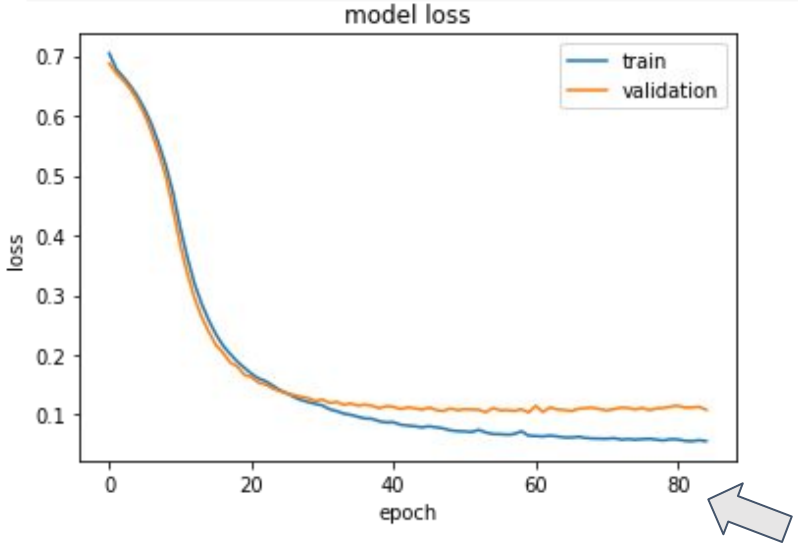
```
from tensorflow.keras.callbacks import EarlyStopping

early_stop = EarlyStopping(monitor='val_loss',
                           mode='min',
                           verbose=1,
                           patience=25)
```

```
history = model.fit(X_train,
                    y_train,
                    epochs=600,
                    validation_data=(X_test, y_test),
                    verbose=1
                    )
```

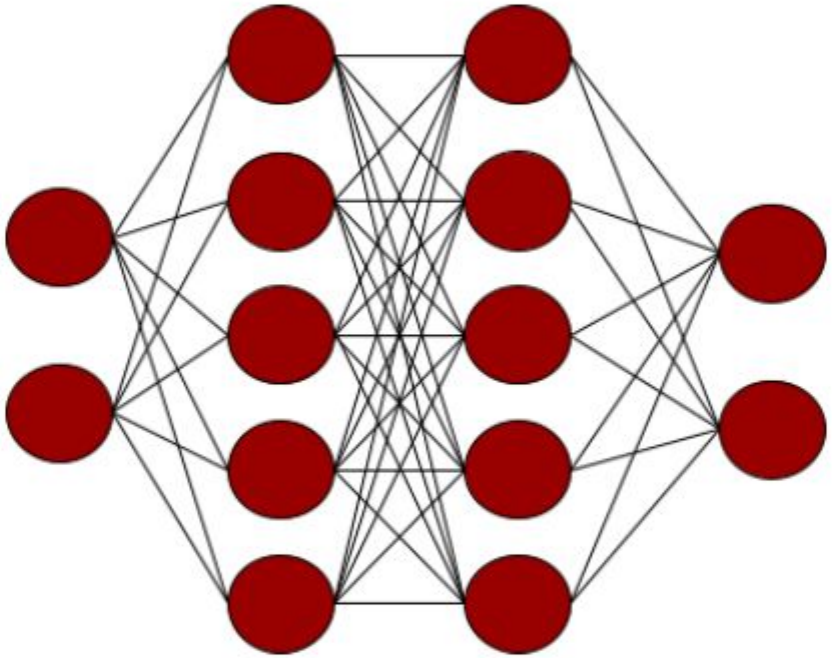



```
history = model.fit(x=X_train,
                    y=y_train,
                    epochs=600,
                    validation_data=(X_test, y_test),
                    verbose=1,
                    callbacks=[early_stop]
                    )
```



Dropout Regularization

```
model = tf.keras.models.Sequential([
                        tf.keras.layers.Flatten(input_shape=(28,28)),
                        tf.keras.layers.Dense(256, activation=tf.nn.relu),
                        tf.keras.layers.Dense(128, activation=tf.nn.relu),
                        tf.keras.layers.Dense(64, activation=tf.nn.relu),
                        tf.keras.layers.Dense(10, activation=tf.nn.softmax)])
```

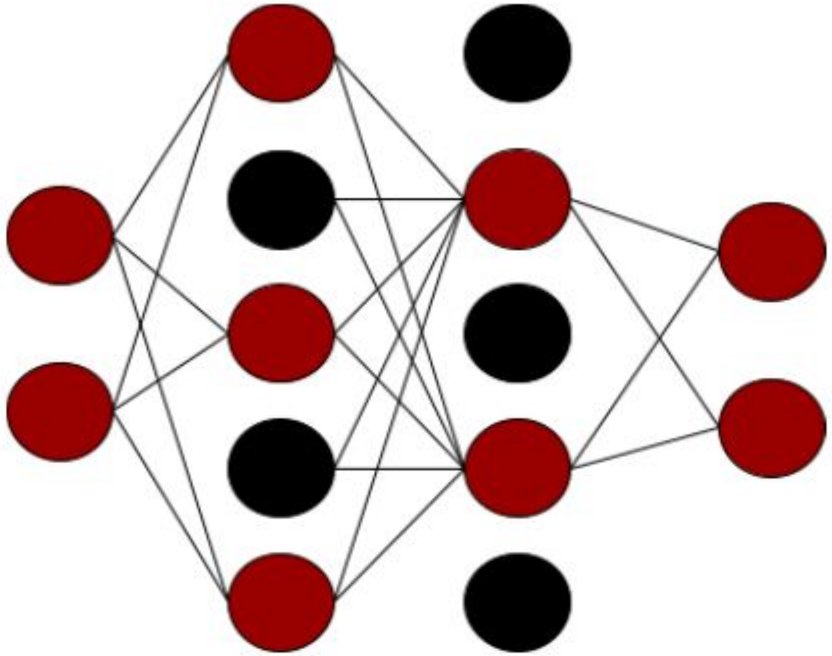


Fashion MNIST Dataset

- 20 Epochs
- 94.0% Accuracy on Train Data
- 88.5% Accuracy on Validation Data

Dropout Regularization

```
model = tf.keras.models.Sequential([
                                    tf.keras.layers.Flatten(input_shape=(28,28)),
                                    tf.keras.layers.Dense(256, activation=tf.nn.relu),
                                    tf.keras.layers.Dropout(0.2),
                                    tf.keras.layers.Dense(128, activation=tf.nn.relu),
                                    tf.keras.layers.Dropout(0.2),
                                    tf.keras.layers.Dense(64, activation=tf.nn.relu),
                                    tf.keras.layers.Dropout(0.2),
                                    tf.keras.layers.Dense(10, activation=tf.nn.softmax)])
```



Fashion MNIST Dataset

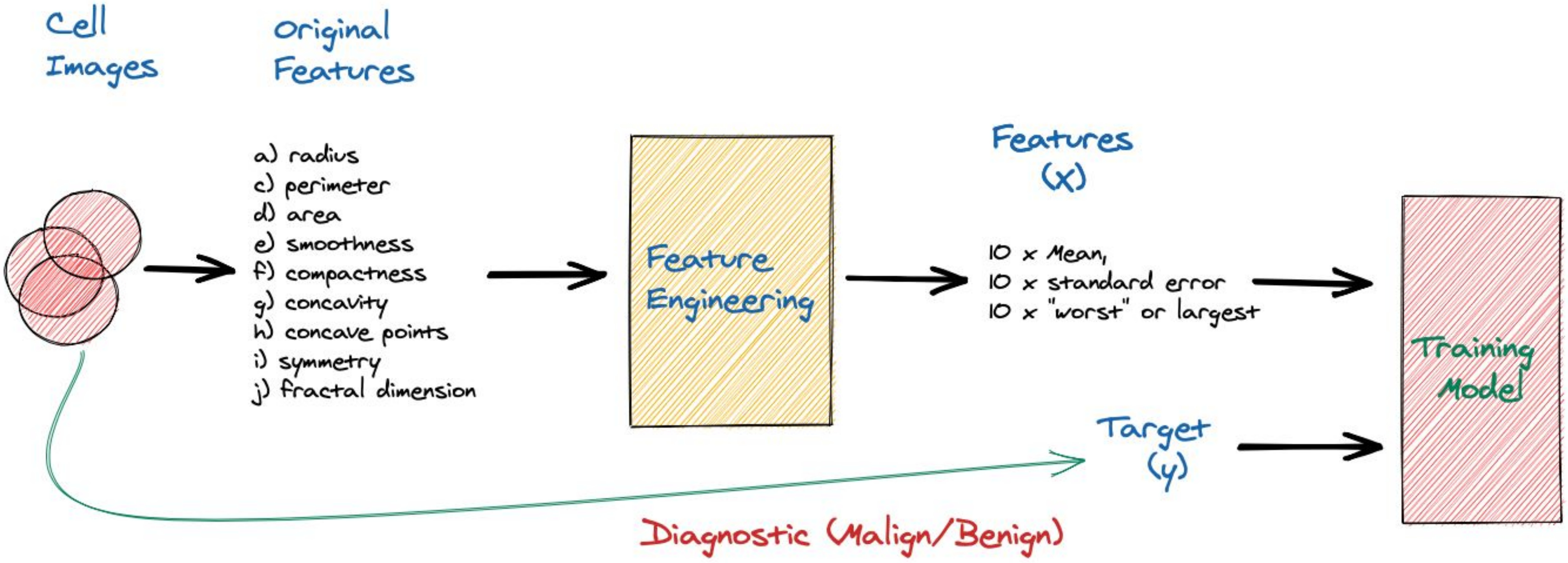
- 20 Epochs
- 89.5% Accuracy on Train Data
- 88.3% Accuracy on Validation Data

Removing a random number of neurons and connections (in this example, 20%), reduces the chances of the neurons becoming overspecialized and the model will generalize better, reducing the overfit.

Wisconsin Diagnostic Breast Cancer (WDBC)

Optional Homework

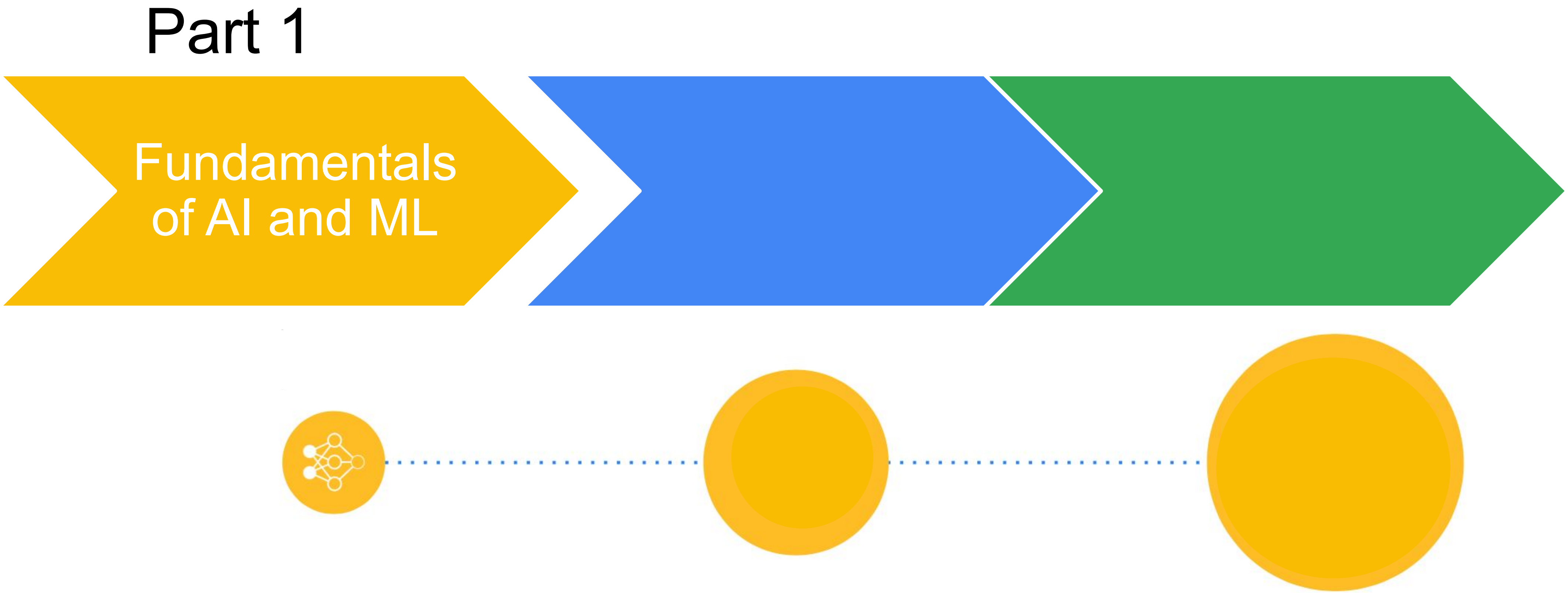
[Breast_Cancer_Classification.ipynb]()



UCI ML Breast Cancer Wisconsin (Diagnostic) datasets. https://goo.gl/U2Uwz2

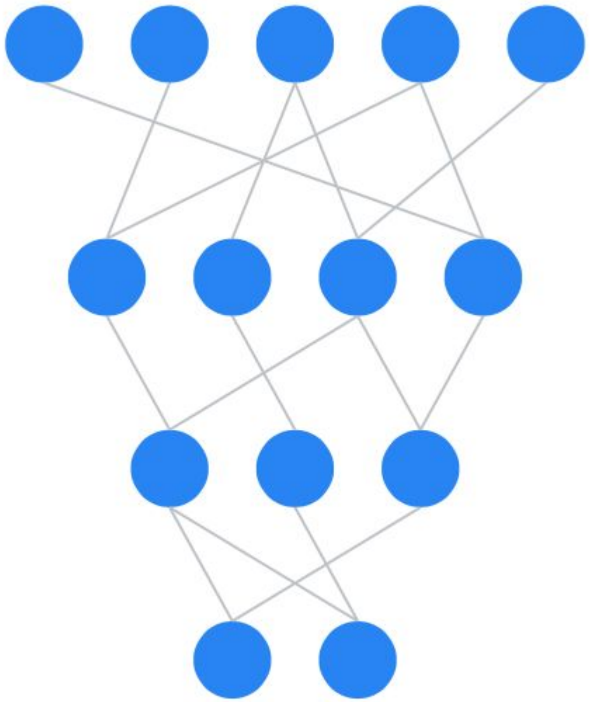
Deep Learning Wrap-Up

What have we learned so far?



In Part 1, while discussing what is the language of machine learning, we introduced ML with TensorFlow.

Total Recall from **Part 1**



“Language” for **Part 1**

Training Data

Neural Network

Training

Validation Data

Inference

Gradient Descent

Test Data

Features

Classification

Loss Function

Filters

Overfitting

Kernels

Regression

Data augmentation

CNNs

DNNs

Responsible AI

Preprocessing

“Language” for **Part 1**

Training Data

Neural Network

Training

Validation Data

Inference

Gradient Descent

Test Data

Features

Classification

Loss Function

Filters

Overfitting

Kernels

Regression

Data augmentation

CNNs

DNNs

Responsible AI

Preprocessing

“Language” for **Part 1**

Training Data

Neural Network

Training

Validation Data

Inference

Gradient Descent

Test Data

Features

Classification

Loss Function

Filters

Overfitting

Kernels

Regression

Data augmentation

CNNs

DNNs

Responsible AI

Preprocessing

“Language” for **Part 1**

Training Data

Neural Network

Training

Validation Data

Gradient Descent

Inference

Test Data

Features

Classification

Loss Function

Filters

Overfitting

Kernels

Regression

Data augmentation

CNNs

DNNs

Responsible AI

Preprocessing

What will we learn?

Part 1 Part 2

In Part 2, we will get a sneak peek into the variety of different TinyML applications, as keyword spotting (“Alexa”), gesture recognition, understand how to leverage the sensors, and so forth.

What will we learn?

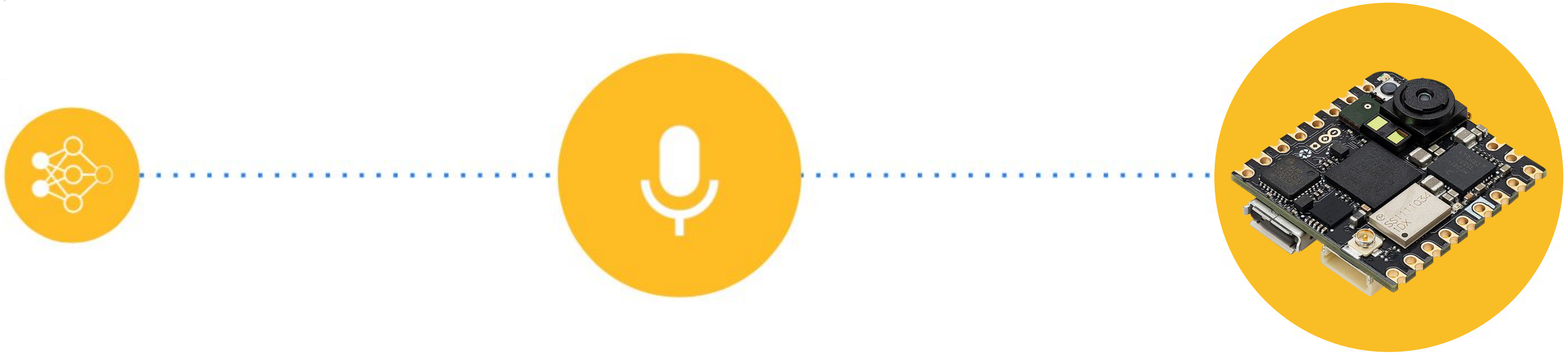
Part 1

Part 2

Fundamentals of AI and ML

Applications of TinyML

Deploying TinyML



In Part 2, we will **also** learn how to deploy models on a real microcontroller. Along the way we will explore the challenges unique to and amplified by TinyML (e.g., preprocessing, post-processing, dealing with resource constraints).

TensorFlow

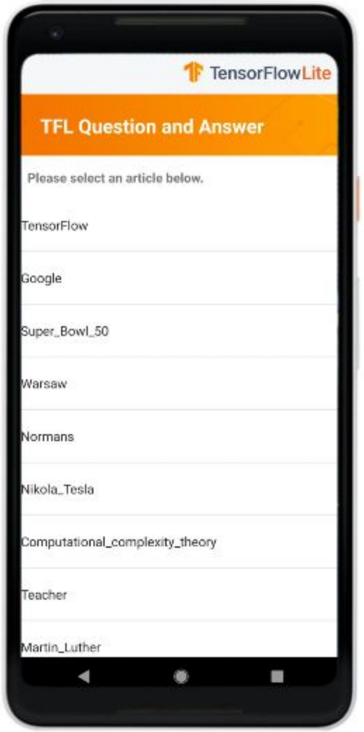
TensorFlow Lite

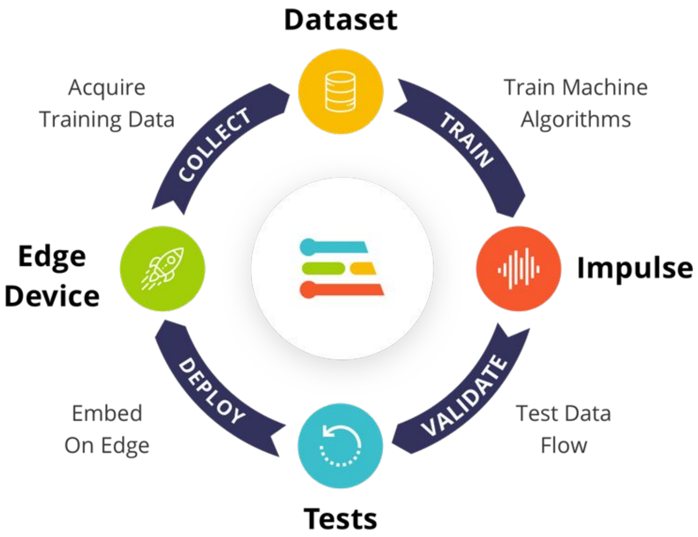
Train a model → Convert model → Optimize model → Deploy model at Edge → Make inferences at Edge

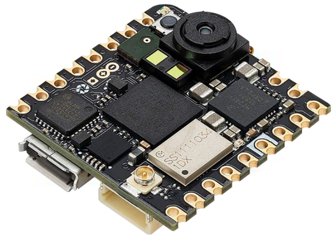
TensorFlow

TensorFlow Lite

Train a model → Convert model → Optimize model → Deploy model at Edge → Make inferences at Edge







Thanks

TINYML4D