

Introduction to Transfer Learning

Thien Tran

MLDA@EEE

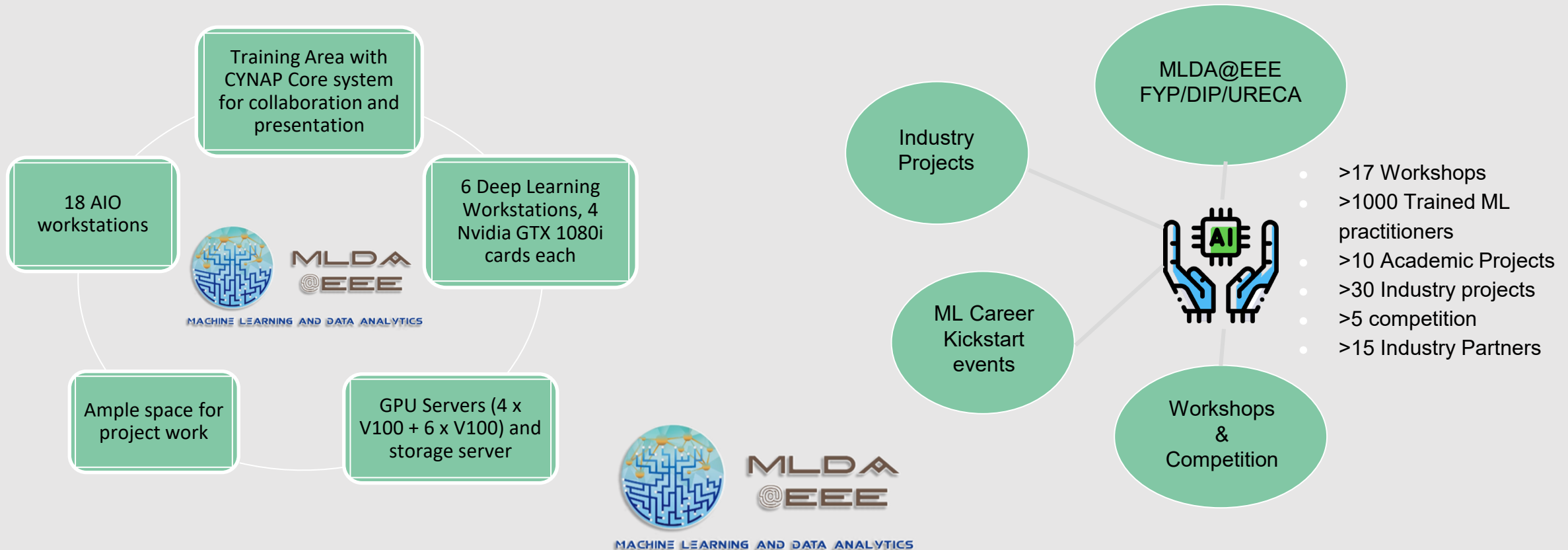


MLDA
@EEE

MACHINE LEARNING AND DATA ANALYTICS

Our Mission

Provide an integrated platform for EEE/IEM students to learn and implement Machine Learning, Data Science & AI, as well as facilitate connections with the industry.

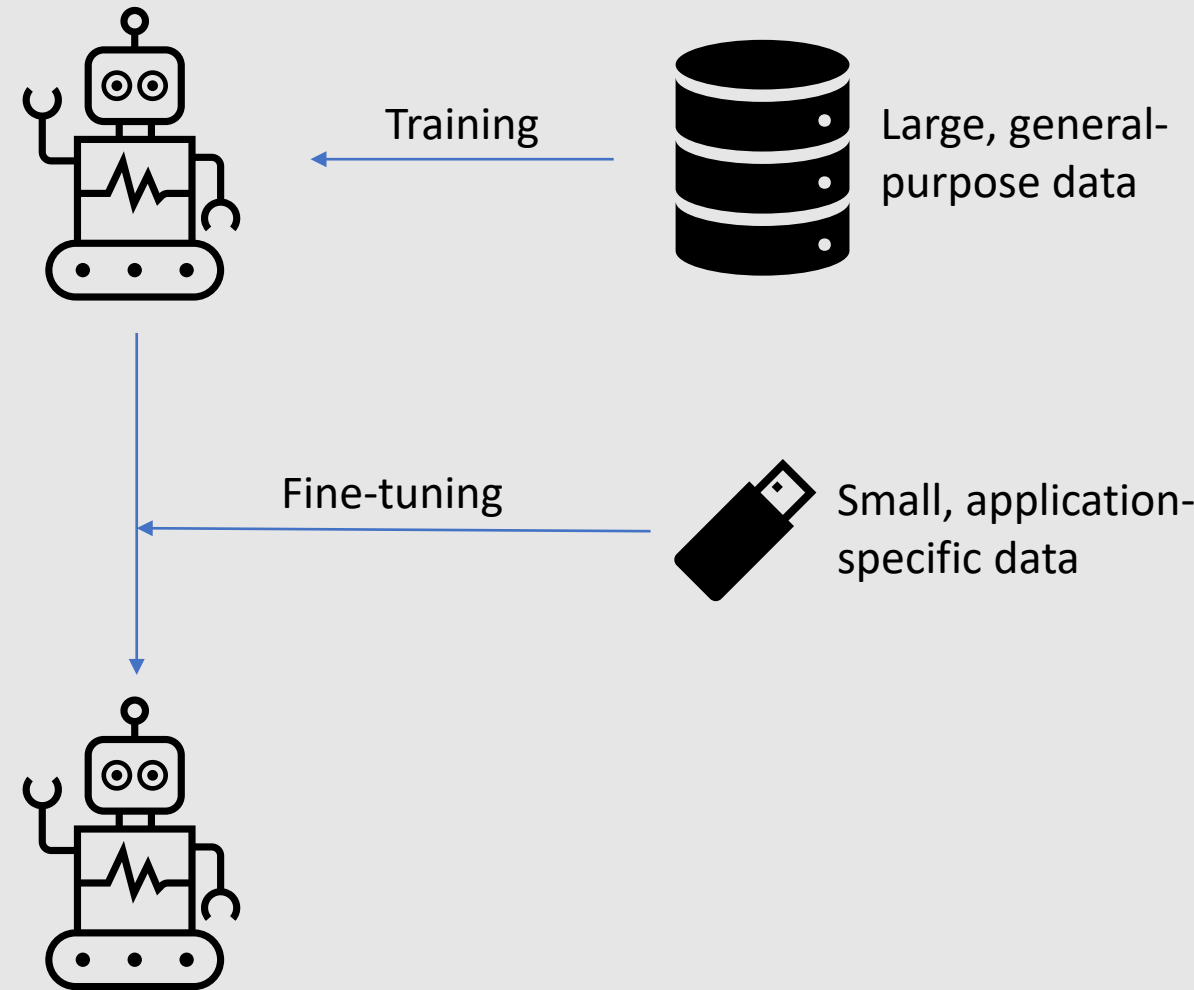


Agenda

- Theory of Transfer learning
- Transfer Learning workflow
- Hands-on: implement Transfer Learning with TensorFlow Keras

What is Transfer Learning

- Apply knowledge learned from **one task** to another **related task**



What is Transfer Learning



Car



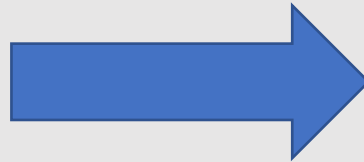
Truck



No car



No truck



Motivations



Lack of training data

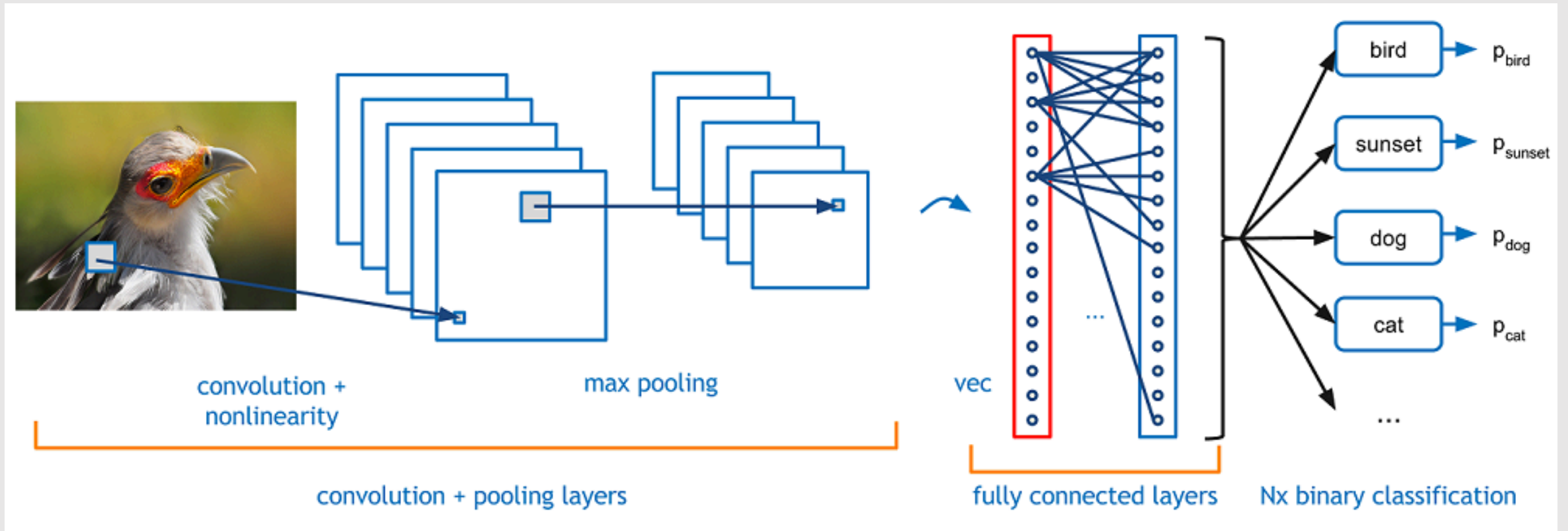


Reduce training time



More robust to unseen data

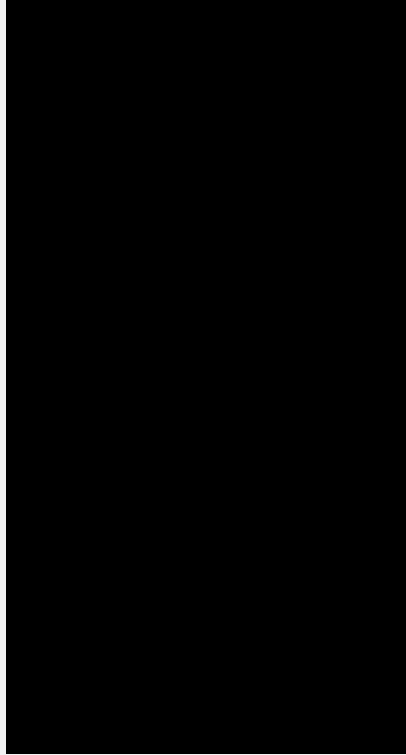
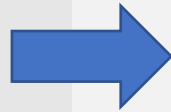
Transfer Learning – Learning the features



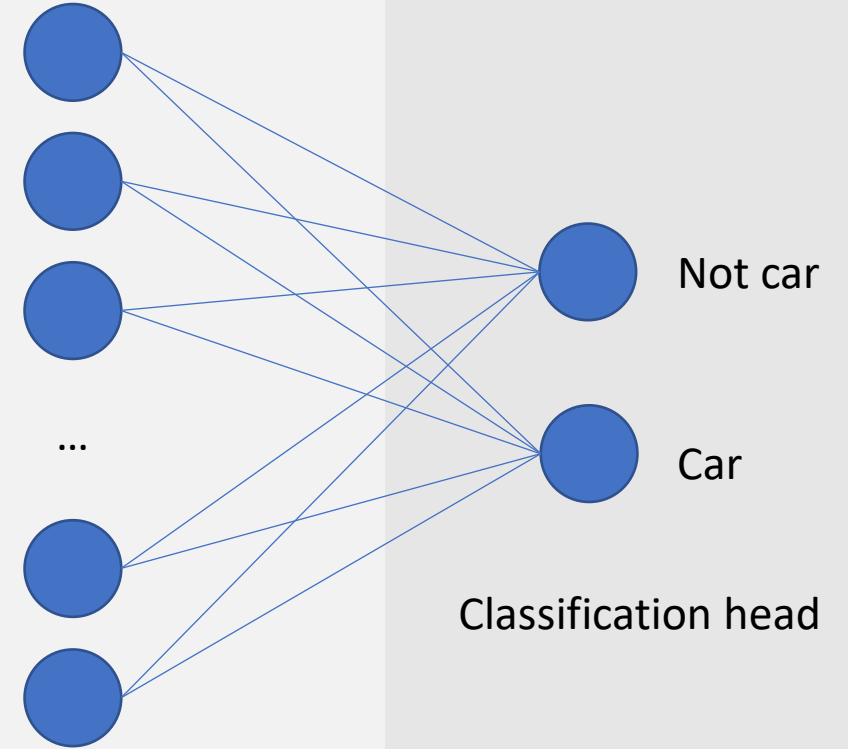
Transfer Learning – Black box



Input – Rank 3 Tensor



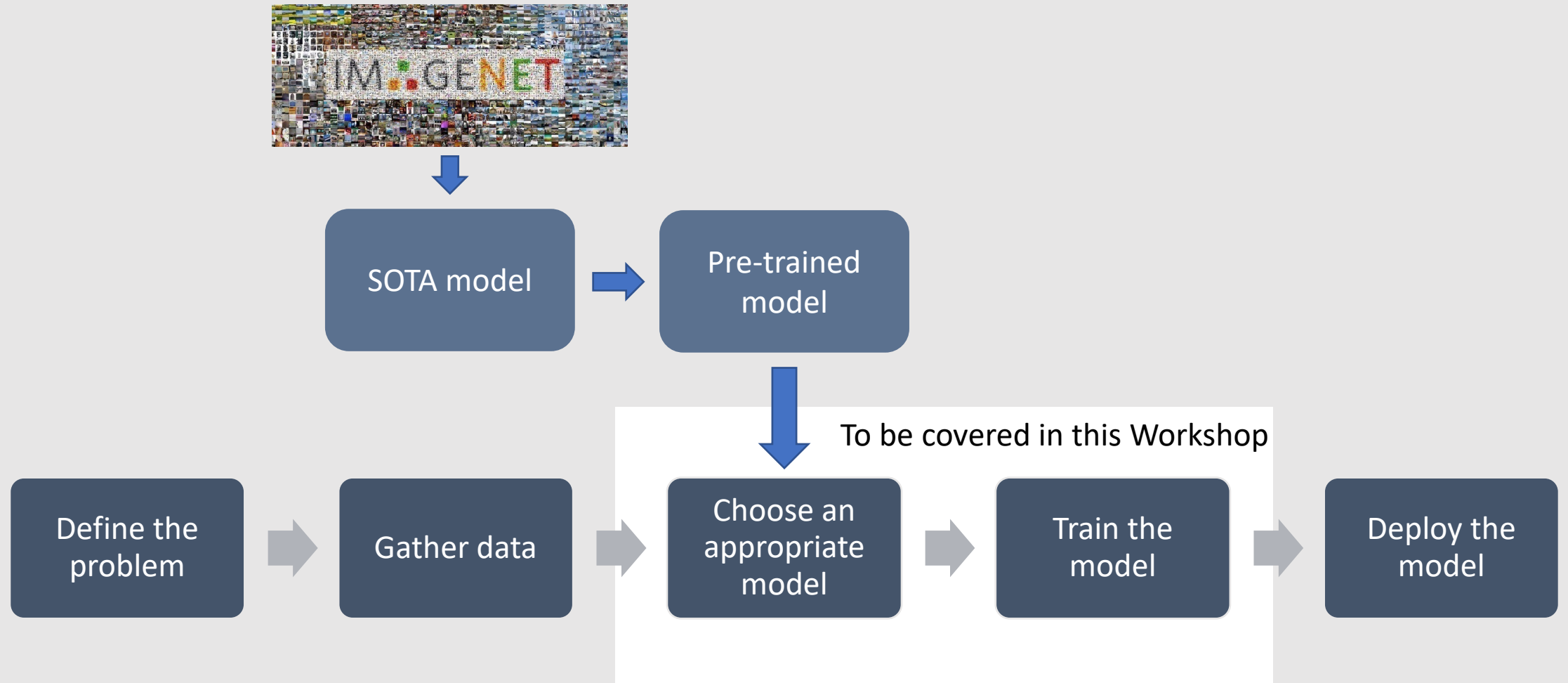
Black box – Pre-trained model



Output - Features vector

Classification head

Sample Transfer Learning Workflow



Upstream Training - ImageNet

- ImageNet: a database of images for visual object recognition research
 - 14 million images, hand-annotated
 - 20,000 categories (classes)
 - The standard dataset for evaluating neural network architecture in research
 - <http://image-net.org/explore>



Upstream Training - ImageNet

Geological formation, formation

(geology) the geological features of the earth

1808 pictures 86.24% Popularity Percentile Wordnet IDs

Numbers in brackets: (the number of synsets in the subtree).

- ImageNet 2011 Fall Release (32326)
 - plant, flora, plant life (4486)
 - geological formation, formation (17)
 - aquifer (0)
 - beach (1)
 - cave (3)
 - cliff, drop, drop-off (2)
 - delta (0)
 - diapir (0)
 - folium (0)
 - foreshore (0)
 - ice mass (10)
 - lakefront (0)
 - massif (0)
 - monocline (0)
 - mouth (0)
 - natural depression, depression (0)
 - natural elevation, elevation (41)
 - oceanfront (0)
 - range, mountain range, range of mountains (0)
 - relict (0)
 - ridge, ridgeline (2)
 - ridge (0)
 - shore (7)
 - slope, incline, side (17)
 - spring, fountain, outflow, outpouring (0)
 - talus, scree (0)
 - vein, mineral vein (1)
 - volcanic crater, crater (2)
 - wall (0)

Treemap Visualization Images of the Synset Downloads

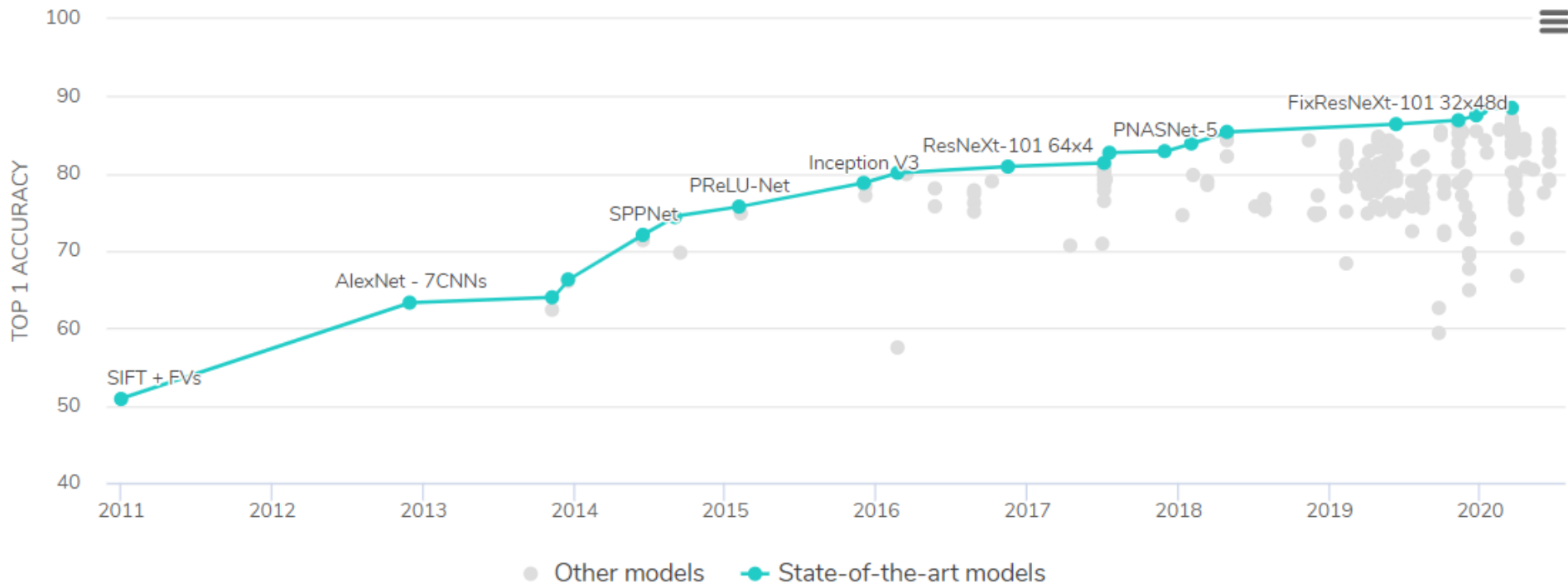
ImageNet 2011 Fall Release > Geological formation, formation

Natural 	Slope 	Shore
	Ice 	Water
	Vein 	Delta
	Massif 	Foreshore
	Talus 	Volcanic
	Mouth 	Beach
	Lakefront 	Diapir
	Wall 	Cliff
	Monocline 	Aquifer
	Oceanfront 	Cave
		Spring
		Ridge

SOTA models

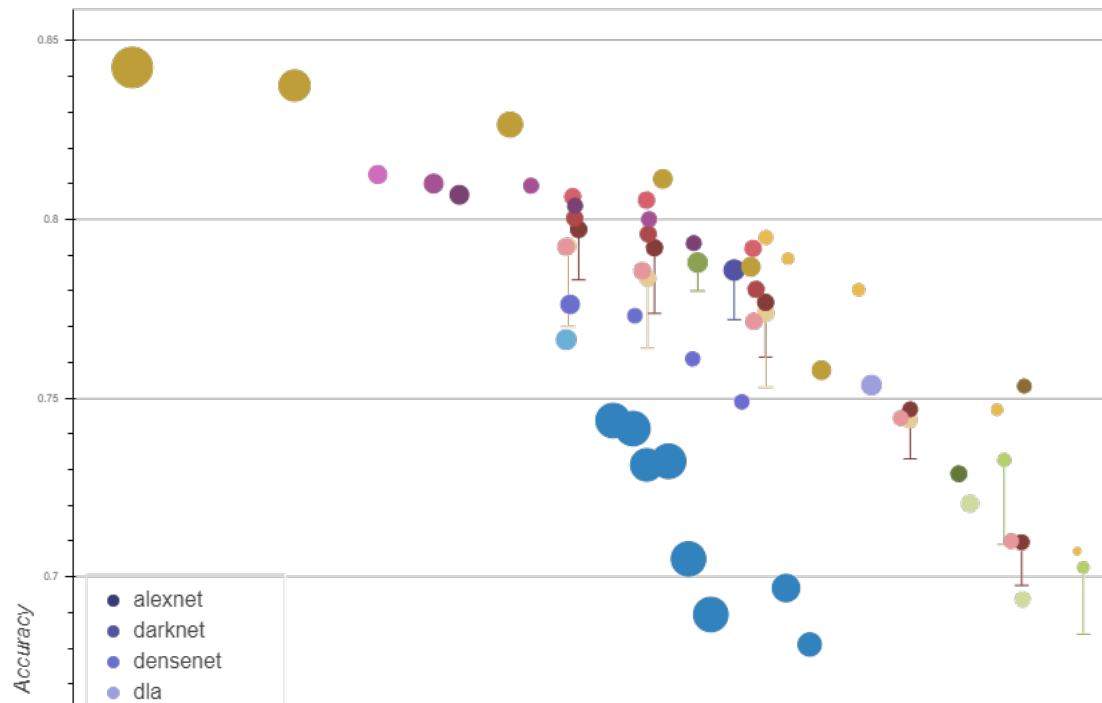
Model	Top-1 accuracy
AlexNet (2012)	63.3%
VGG-19 (2015)	74.5%
ResNet-152 (2016)	77.8%
ResNeXt-101 (2017)	80.9%
EfficientNet-B7 (2019)	84.4%
FixEfficientNet-L2 (2020)	88.5%

Image Classification on ImageNet

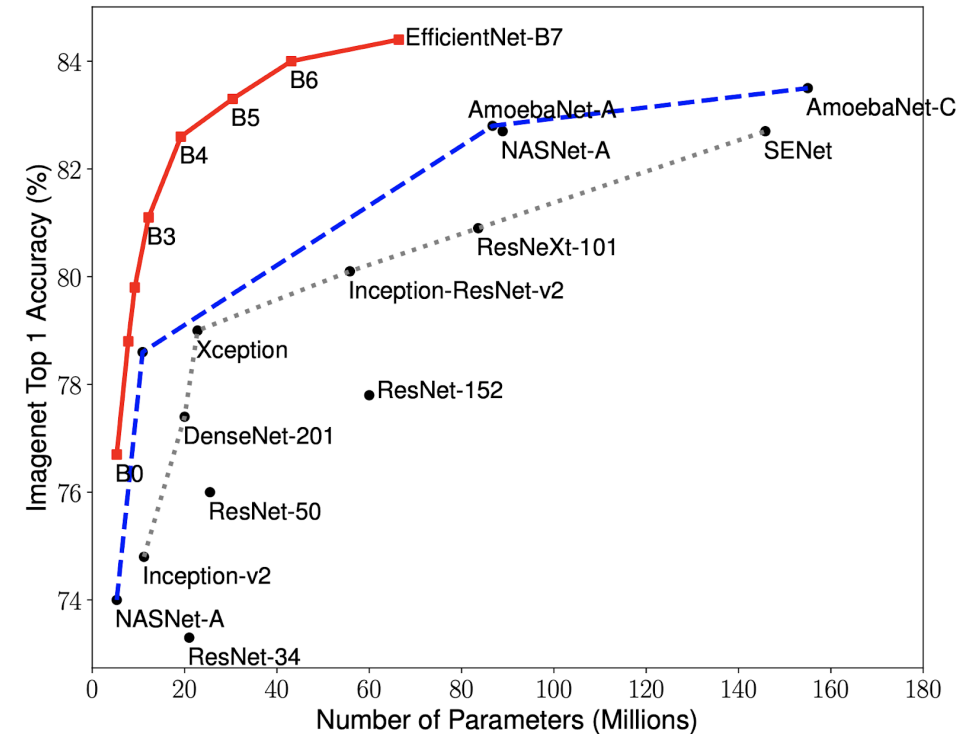


Source: <https://paperswithcode.com/sota/image-classification-on-imagenet>

Choose a model



Accuracy vs Throughput Trade-off



Model size

Define the
problem

Gather data

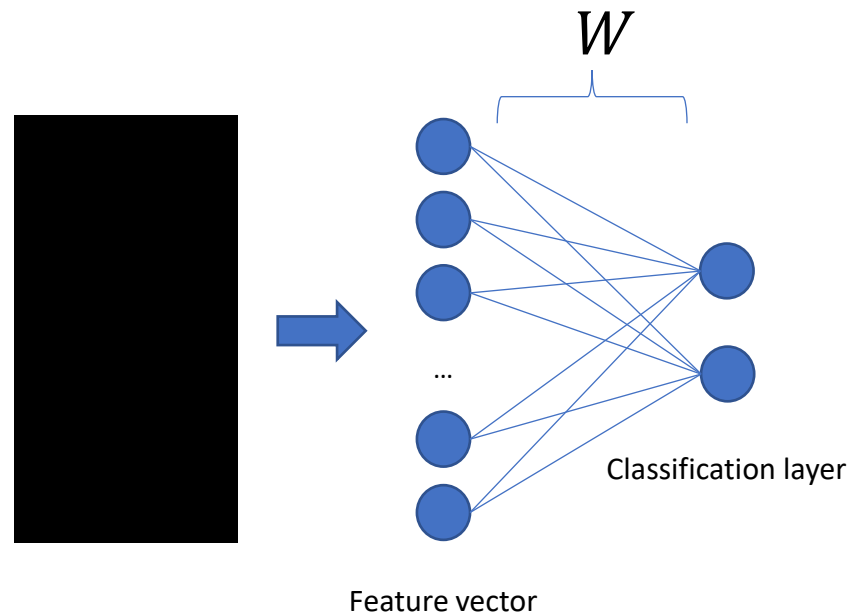
Choose an
appropriate
model

Train the
model

Deploy the
model

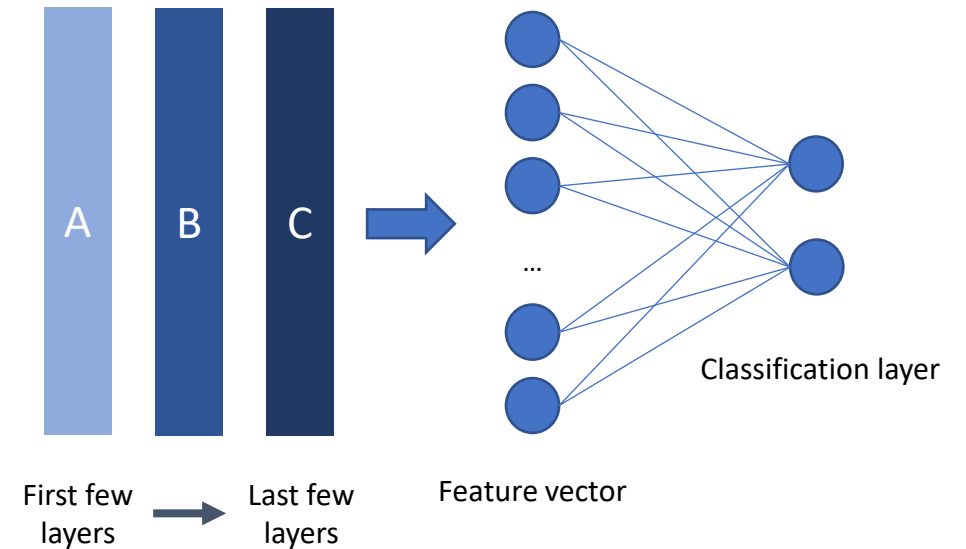
Classification layer and Fine-tuning

Train the classification layer only



Train the whole network

Train selectively the last few layers



Define the
problem

Gather data

Choose an
appropriate
model

Train the
model

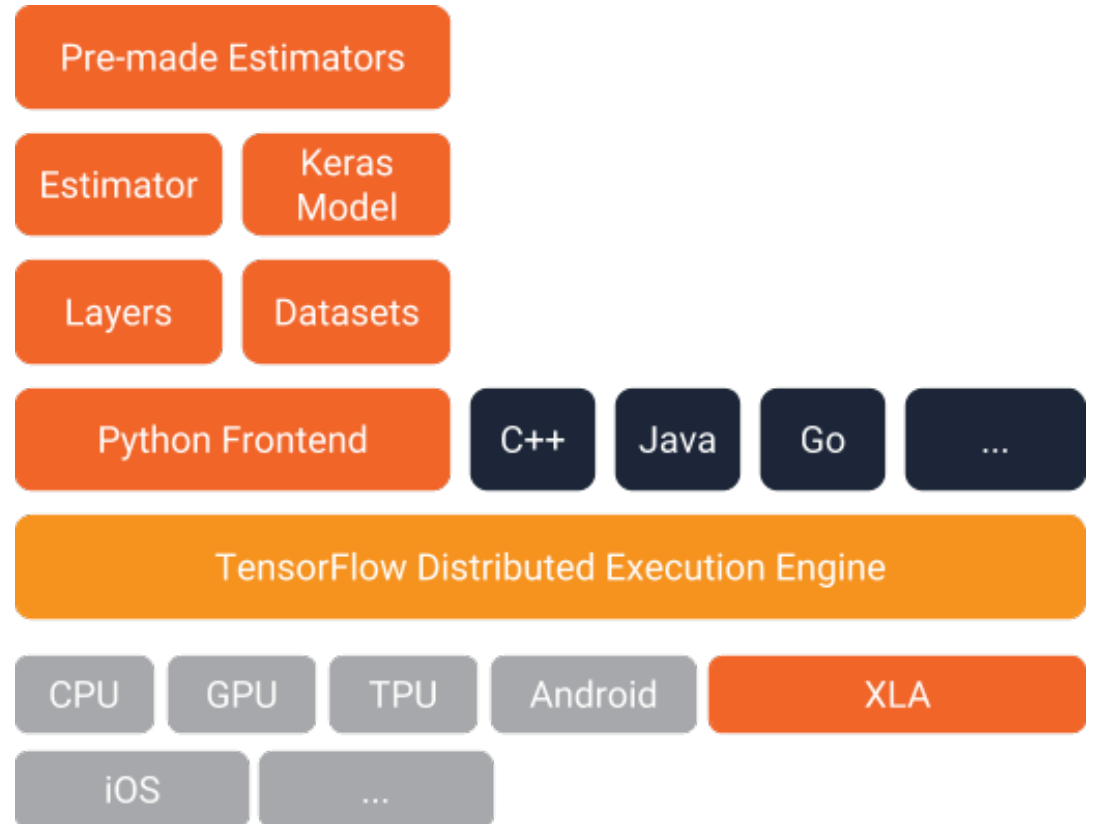
Deploy the
model

TensorFlow and Keras

TensorFlow's **high-level APIs** are based on the Keras API standard for defining and training neural networks.

Keras enables **fast** prototyping, state-of-the-art research, and production—all with **user-friendly APIs**.

TensorFlow Architecture



TensorFlow and Keras

```
import tensorflow as tf
mnist = tf.keras.datasets.mnist

(x_train, y_train), (x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0

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

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

model.fit(x_train, y_train, epochs=5)
model.evaluate(x_test, y_test)
```

[Run code now](#)[Try in Google's interactive notebook](#)

```
class MyModel(tf.keras.Model):
    def __init__(self):
        super(MyModel, self).__init__()
        self.conv1 = Conv2D(32, 3, activation='relu')
        self.flatten = Flatten()
        self.d1 = Dense(128, activation='relu')
        self.d2 = Dense(10, activation='softmax')

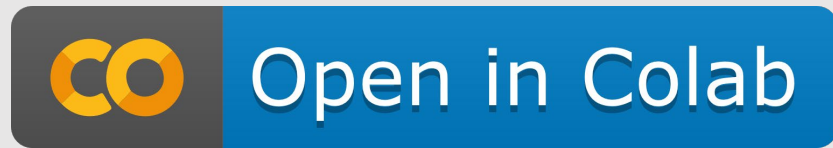
    def call(self, x):
        x = self.conv1(x)
        x = self.flatten(x)
        x = self.d1(x)
        return self.d2(x)

model = MyModel()

with tf.GradientTape() as tape:
    logits = model(images)
    loss_value = loss(logits, labels)
    grads = tape.gradient(loss_value, model.trainable_variables)
    optimizer.apply_gradients(zip(grads, model.trainable_variables))
```

[Run code now](#)[Try in Google's interactive notebook](#)

Hands-on session



<https://github.com/MLDA-NTU/Transfer-Learning-DL2020>