



AOI02-Aidea AOI Project

Hsueh-Ting Chu

朱學亭老師

- 現職：亞洲大學資工系副教授(兼AI研究組長)
- EMAIL: htchu.taiwan@gmail.com
- FB: <https://www.facebook.com/htchu.taiwan>





ARTIFICIAL INTELLIGENCE
COLLABORATION PLATFORM

Aidea

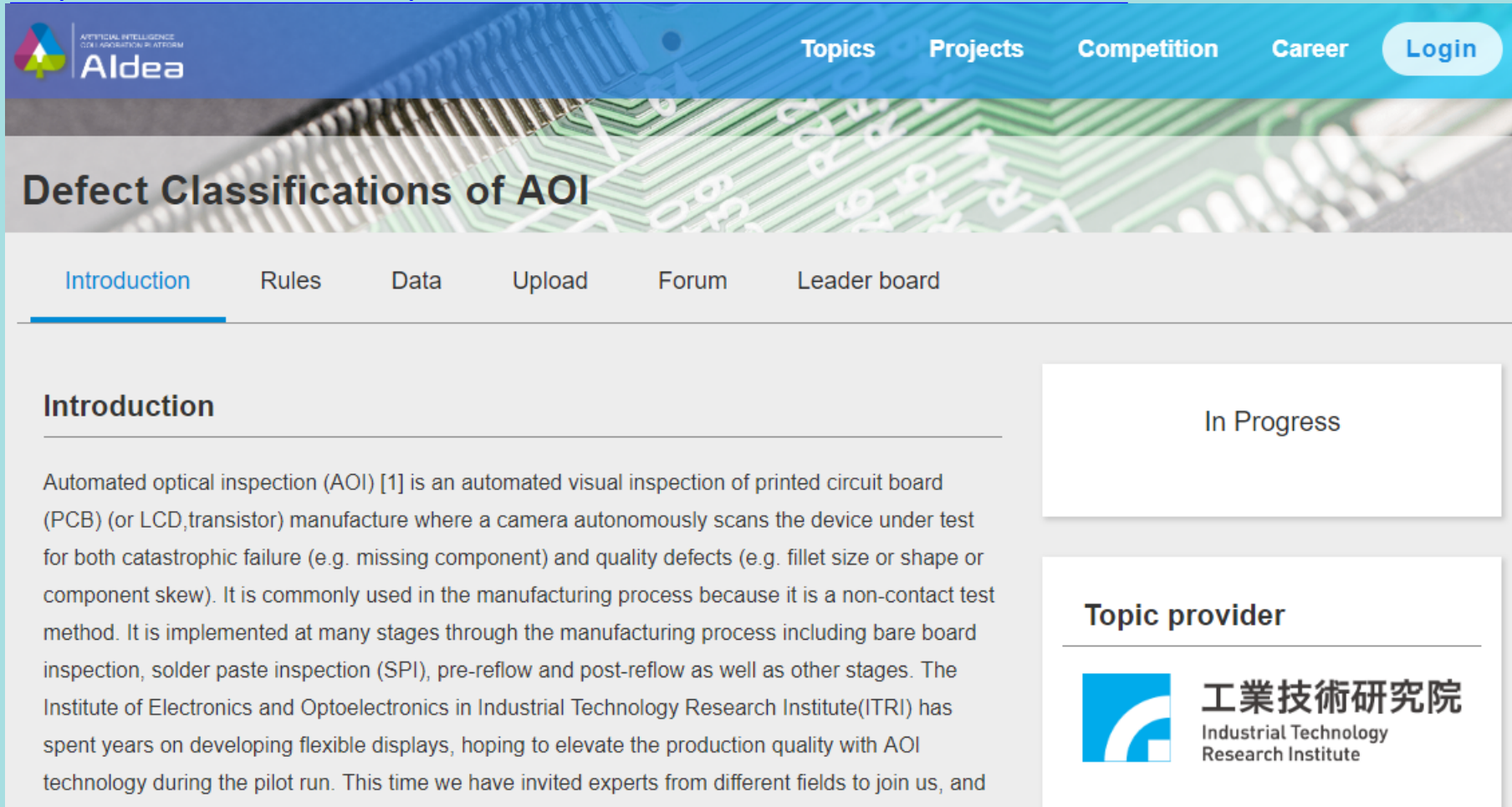
Aidea platform

- Artificial Intelligence Collaboration Platform by ITRI(Industrial Technology Research Institute)
- Solving real AI industrial issues in Taiwan
- Building AI industrial datasets in Taiwan



Open Topic: Defect Classifications of AOI

<https://aidea-web.tw/topic/a49e3f76-69c9-4a4a-bcfc-c882840b3f27>



The screenshot shows the AIDEA website interface. At the top, there is a navigation bar with the AIDEA logo (Artificial Intelligence Collaboration Platform) and links for Topics, Projects, Competition, Career, and a Login button. Below the navigation bar is a banner image of a circuit board. The main heading is 'Defect Classifications of AOI'. Underneath, there are tabs for Introduction, Rules, Data, Upload, Forum, and Leader board. The 'Introduction' tab is selected. The content area on the left contains an 'Introduction' section with a paragraph about Automated optical inspection (AOI). On the right, there is a box labeled 'In Progress' and a 'Topic provider' section featuring the logo and name of the Industrial Technology Research Institute (ITRI).

Defect Classifications of AOI


[Introduction](#) [Rules](#) [Data](#) [Upload](#) [Forum](#) [Leader board](#)

Introduction

Automated optical inspection (AOI) [1] is an automated visual inspection of printed circuit board (PCB) (or LCD, transistor) manufacture where a camera autonomously scans the device under test for both catastrophic failure (e.g. missing component) and quality defects (e.g. fillet size or shape or component skew). It is commonly used in the manufacturing process because it is a non-contact test method. It is implemented at many stages through the manufacturing process including bare board inspection, solder paste inspection (SPI), pre-reflow and post-reflow as well as other stages. The Institute of Electronics and Optoelectronics in Industrial Technology Research Institute (ITRI) has spent years on developing flexible displays, hoping to elevate the production quality with AOI technology during the pilot run. This time we have invited experts from different fields to join us, and

In Progress

Topic provider

 **工業技術研究院**
Industrial Technology
Research Institute



The top three exceeding Baseline score (**0.998521**) on Private Leaderboard would be awarded

Closed Project: Defect Classifications of AOI

The screenshot shows the Aldea AI platform interface. At the top, there is a navigation bar with links for Topics, Projects, Competition, Career, and a Login button. Below this, a banner for the "Asia University / Summer Program" is displayed. A timeline indicates key dates: "Begin 06/15", "Register end 07/01", and "Team-up end 07/28". A secondary navigation bar includes links for Introduction, Rules, Data, Upload, Forum, Team up, and Leader board. The "Introduction" section is active, providing a definition of Automated Optical Inspection (AOI). To the right, a "Team up" section features a digital timer showing 05 days, 20 hours, 02 minutes, and 41 seconds.

Aldea
ARTIFICIAL INTELLIGENCE
COLLABORATION PLATFORM

Topics Projects Competition Career Login

Asia University / Summer Program

Begin 06/15 Register end 07/01 Team-up end 07/28

Introduction Rules Data Upload Forum Team up Leader board

Introduction

Automated optical inspection (AOI) [1] is an automated visual inspection of printed circuit board (PCB) (or LCD, transistor) manufacture where a camera autonomously scans the device under test for both catastrophic failure (e.g. missing component) and quality defects (e.g. fillet size or shape or component skew). It is commonly used in the manufacturing process because it is a non-contact test

Team up

05 20 02 41
days hrs mins secs

AOI Study

- Same dataset:
 - Open Topic: Defect Classifications of AOI
 - Closed Project: Defect Classifications of AOI
- Dataset is available from the Open Topic.
- Closed Project is for ranking.



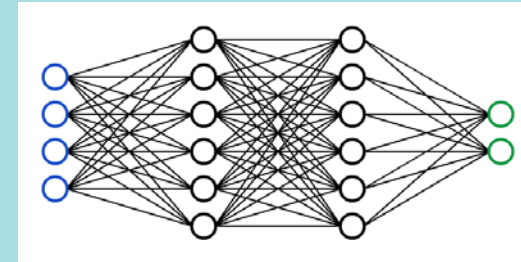
AOI data description

There are 6 categories included in image data that the issue offers
(1 normal category + 5 defect categories)

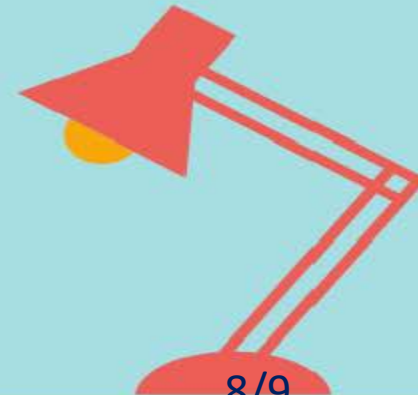
The download data file (aoi_data.zip) includes:

- train_images: image data for training (PNG format), 2,528 images in total.
- train.csv: includes 2 columns, ID and Label.
 - ID: the image filename
 - Label: defect classification category
(0: normal, 1: void, 2: horizontal defect, 3: vertical defect, 4: edge defect, 5: particle)
- test_images: image data for testing (PNG format), 10,142 images in total.
- test.csv: includes 2 columns, ID and Label.
 - ID: the image filename
 - Label: Nan

AOI Workflow

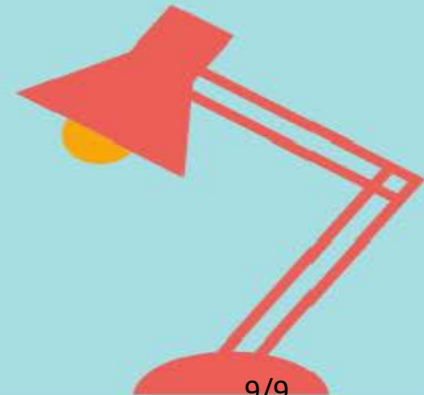


(A) Setup TF 2.0 (B) Mounting (optional) (C) Input training data (D) Model training and inference (E) Output test result



Colab api

- **Magic commands**
 - `%tensorflow_version 2.x`
 - `%matplotlib inline`
- **Google Colab API**
 - `from google.colab import drive`
 - `drive.mount('/content/drive')`

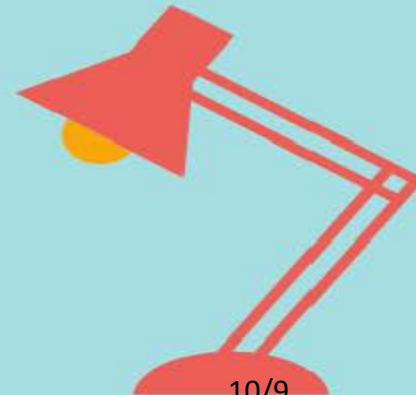


NumPy



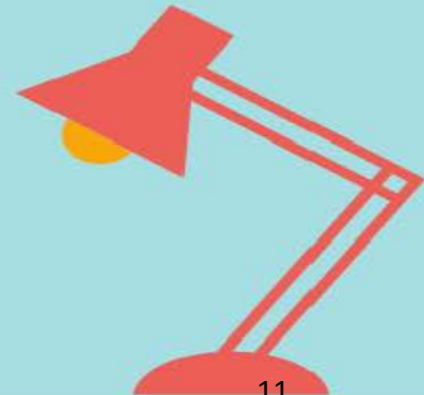
NumPy

- `import numpy as np`
- `a = np.array([2,3,4])`
- `b = np.array([(1.5,2,3), (4,5,6)])`
- `c = np.zeros((3,4))`
- `d = np.ones(3, dtype=np.int32)`



numpy.argmax

- Returns the indices of the maximum values along an axis
- `>>> a = np.arange(6).reshape(2,3) + 10`
- `>>> a`
- `array([[10, 11, 12],`
- `[13, 14, 15]])`
- `>>> np.argmax(a)`
- `5`
- `>>> np.argmax(a, axis=0)`
- `array([1, 1, 1])`
- `>>> np.argmax(a, axis=1)`
- `array([2, 2])`

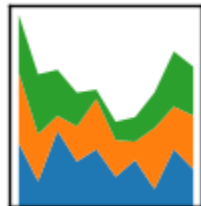
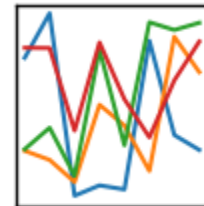
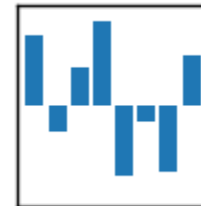


Pandas

- Python Data Analysis Library
- `import pandas as pd`
- `df_train = pd.read_csv("train.csv")`
- `df_out.to_csv("A.csv", index=False)`

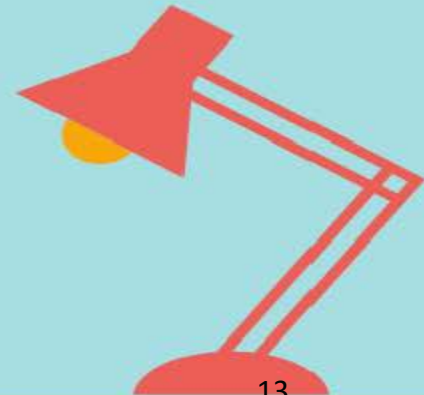
pandas

$$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$$



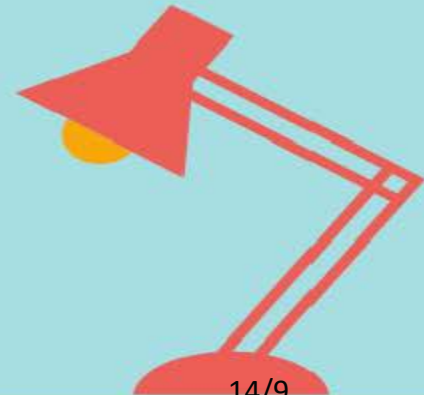
Data Structures

- List \Leftrightarrow NumPy array
- List \Leftrightarrow Pandas DataFrame
- NumPy array \Leftrightarrow Pandas DataFrame



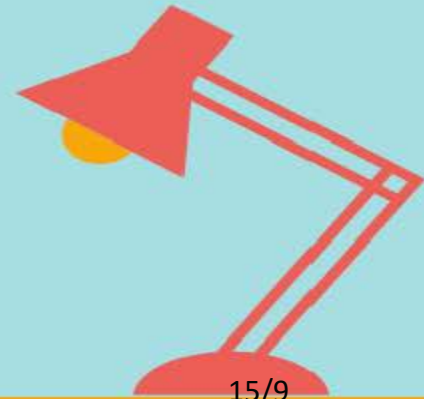
Python Data Visualization Libraries

- Matplotlib
- Seaborn
- `pandas.DataFrame.plot`



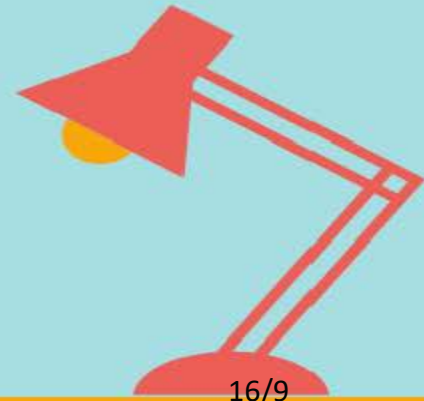
Matplotlib

- Matplotlib is a Python 2D plotting library
- `import matplotlib.pyplot as plt`
- `fig, ax = plt.subplots()`
- `ax.plot(x, y)`
- `ax.set_xlim(0, 10)`
- `ax.set_ylim(-1, 1)`
- `plt.show()`



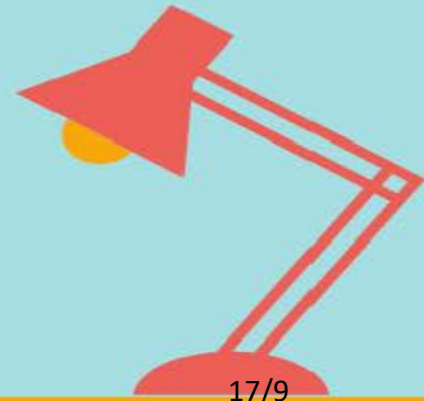
Seaborn heatmap

- seaborn: statistical data visualization
- `import numpy as np; np.random.seed(0)`
- `import seaborn as sns;`
- `sns.set()`
- `uniform_data = np.random.rand(10, 12)`
- `ax = sns.heatmap(uniform_data)`



Python Image Libraries

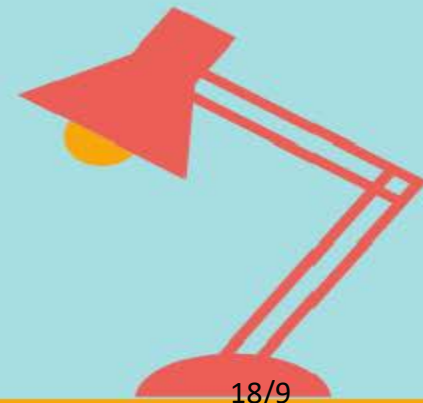
- PIL: Python Imaging Library
- Pillow: the friendly PIL fork
- scikit-image: Image processing in Python
- OpenCV: Open source computer vision
- `tf.keras.preprocessing.image`



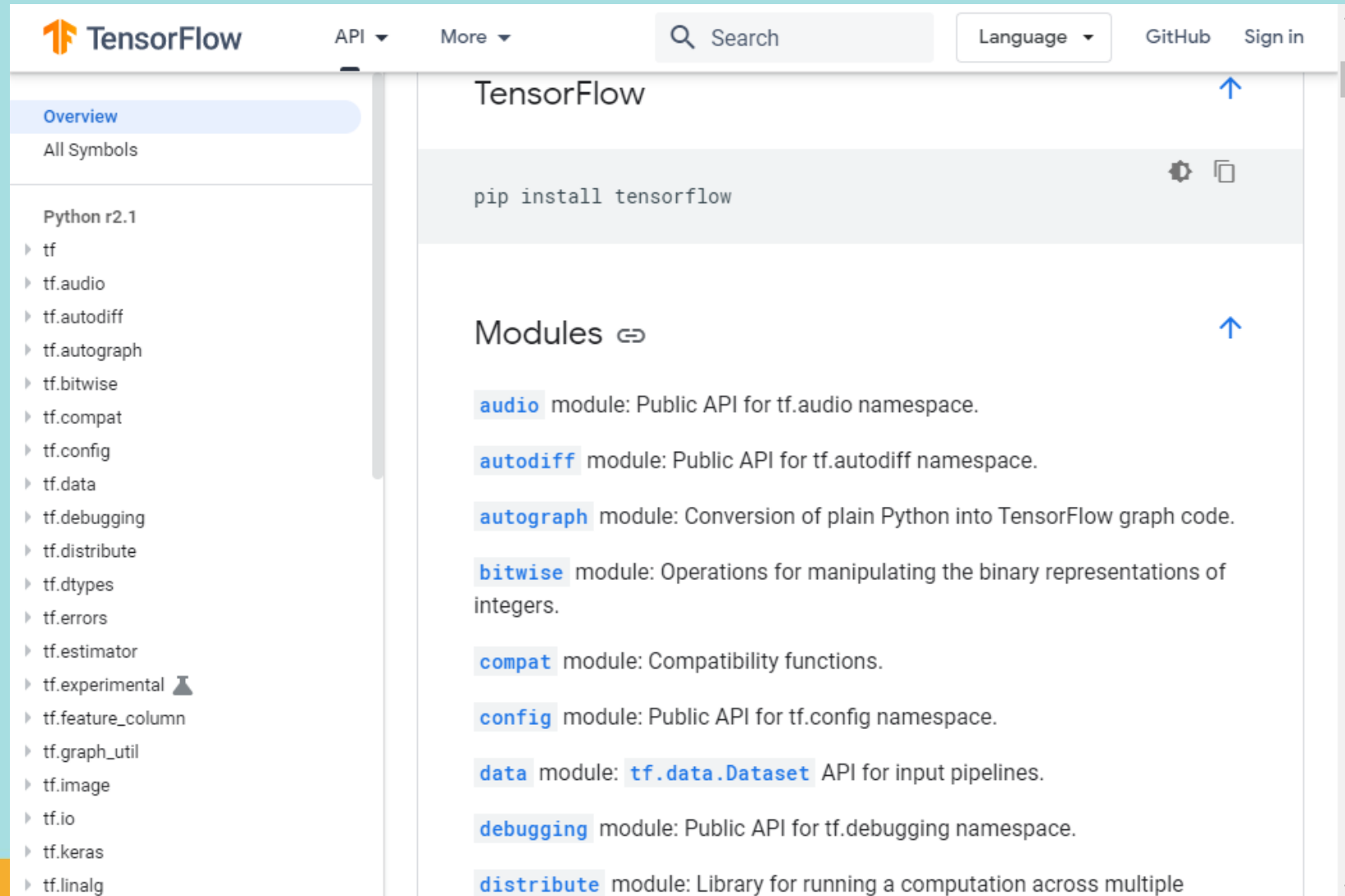
scikit-learn



- scikit-learn: machine learning in Python
- `from sklearn.metrics import confusion_matrix`
- `from sklearn.model_selection import train_test_split`
- `X_train, X_test, y_train, y_test = train_test_split(`
- `... X, y, test_size=0.33, random_state=42)`



TensorFlow 2.x



The screenshot shows the TensorFlow 2.x API documentation page. The header includes the TensorFlow logo, navigation links for 'API' and 'More', a search bar, and links for 'Language', 'GitHub', and 'Sign in'. The left sidebar contains a navigation menu with 'Overview' and 'All Symbols' at the top, followed by a section for 'Python r2.1' listing various modules like 'tf', 'tf.audio', 'tf.autodiff', etc. The main content area is titled 'TensorFlow' and features a code block for 'pip install tensorflow'. Below this, the 'Modules' section lists several modules with their descriptions: 'audio' (Public API for tf.audio namespace), 'autodiff' (Public API for tf.autodiff namespace), 'autograph' (Conversion of plain Python into TensorFlow graph code), 'bitwise' (Operations for manipulating the binary representations of integers), 'compat' (Compatibility functions), 'config' (Public API for tf.config namespace), 'data' (tf.data.Dataset API for input pipelines), 'debugging' (Public API for tf.debugging namespace), and 'distribute' (Library for running a computation across multiple).

TensorFlow

Overview
All Symbols

Python r2.1

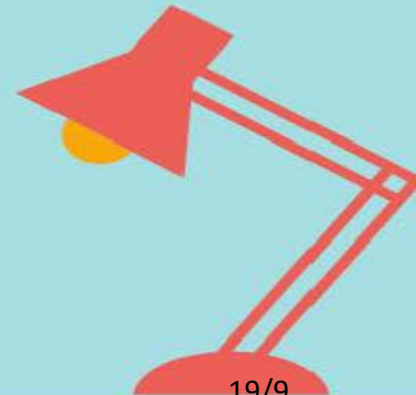
- tf
- tf.audio
- tf.autodiff
- tf.autograph
- tf.bitwise
- tf.compat
- tf.config
- tf.data
- tf.debugging
- tf.distribute
- tf.dtypes
- tf.errors
- tf.estimator
- tf.experimental
- tf.feature_column
- tf.graph_util
- tf.image
- tf.io
- tf.keras
- tf.linalg

TensorFlow

```
pip install tensorflow
```

Modules

- audio** module: Public API for tf.audio namespace.
- autodiff** module: Public API for tf.autodiff namespace.
- autograph** module: Conversion of plain Python into TensorFlow graph code.
- bitwise** module: Operations for manipulating the binary representations of integers.
- compat** module: Compatibility functions.
- config** module: Public API for tf.config namespace.
- data** module: **tf.data.Dataset** API for input pipelines.
- debugging** module: Public API for tf.debugging namespace.
- distribute** module: Library for running a computation across multiple



TensorFlow Tutorials

For beginners

The best place to start is with the user-friendly Sequential API. You can create models by plugging together building blocks. Run the “Hello World” example below, then visit the [tutorials](#) to learn more.

To learn ML, check out our [education page](#). Begin with curated curriculums to improve your skills in foundational ML areas.

```
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)
```

For experts

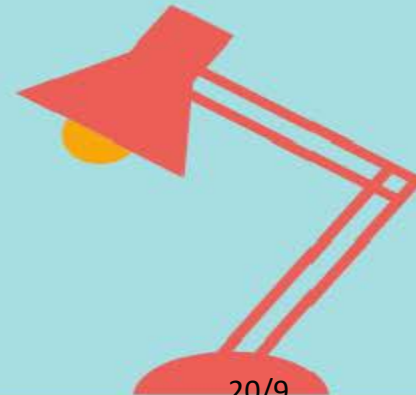
The Subclassing API provides a define-by-run interface for advanced research. Create a class for your model, then write the forward pass imperatively. Easily author custom layers, activations, and training loops. Run the “Hello World” example below, then visit the [tutorials](#) to learn more.

```
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))
```



Sample

```
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)
```



Thanks!

Q&A

