CEC Continuing Education Center

DesignNews

Getting Hands-On With Automated Inspection Concepts Using AI-Based Smart Cameras

# Develop an Automated Visual Inspection Workflow

Sponsored by

DigiKey

informa markets

# Webinar Logistics

- Turn on your system sound to hear the streaming presentation.
- If you have technical problems, click “Help” or submit a question asking for assistance.
- Participate in ‘Attendee Chat’ by maximizing the chat widget in your dock.

# Dr. Don Wilcher

Visit 'Lecturer Profile' in your console for more details.

# Course Kit and Materials

**Pixy2 CMUCAM5**

**Pan/Tilt2 Servo Motor Kit for Pixy2**

**Arduino Uno Rev 3**

**M5Stack AI Camera**

**M5GO IoT Starter Kit V2.7**

## Agenda:

- Workflow Explained
  a) Definition
  b) Approaches
- Automated Visual Inspection Systems Workflow Examples
- Transfer Learning Theory
  a) History
  b) ChatGPT-LLM Python Code
- Lab: Building an Automated Visual Inspection Workflow and Report using Python

## Seminal Research Perspective

“Inspections are performed in virtually every production system. Their purpose is to verify that the production operations were carried out properly and that the production output meets the expectations of the customer” (Ben-Gal et al., 2002).

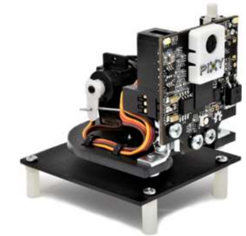
# Workflow Explained: Definition

- A sequence of steps or processes through which a piece of work passes.
  a) passes from initiation
  b) to completion
- Defines the specific, repeatable steps required to complete:
  a) task
  b) achieve a goal.
- Workflows are designed to ensure that tasks are carried out efficiently, consistently
  a) according to predefined rules or
  b) guidelines

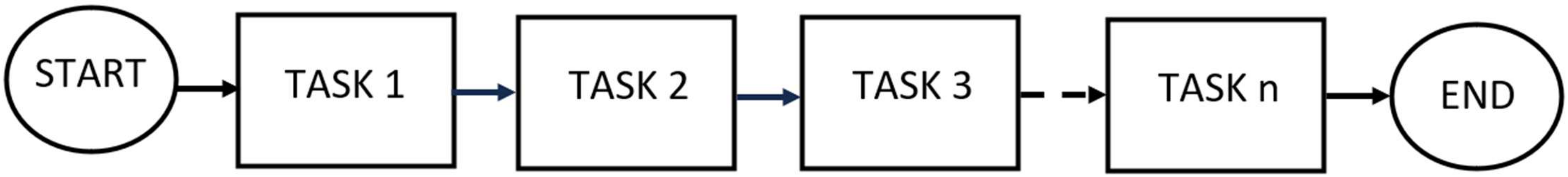
# Workflow Explained: Approaches

- Sequential Workflow:
  Linear Process – Tasks are performed in a set order, one after the other.
- Parallel Workflow:
  Multiple tasks are performed simultaneously.
- Conditional Workflow:
  The path of the workflow changes based on certain conditions or decisions.
- Rule-based Workflow:
  Specific rules dictate how tasks are processed and moved through the workflow.

# Workflow Explained: Approaches. . .

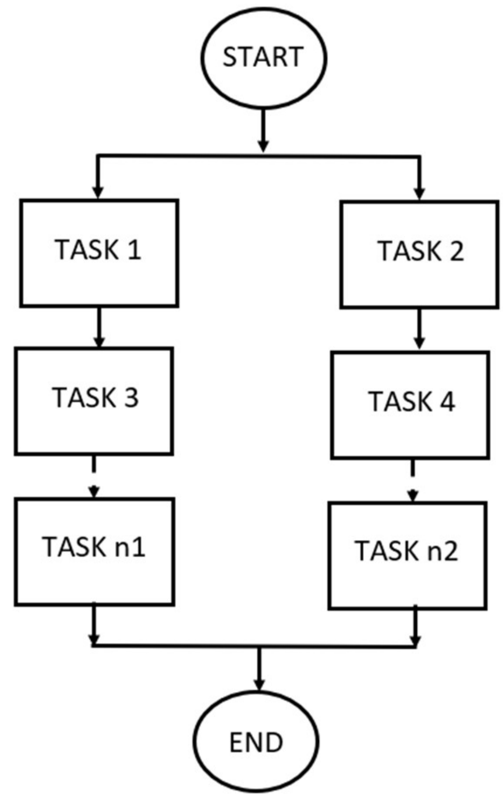


Sequential Workflow



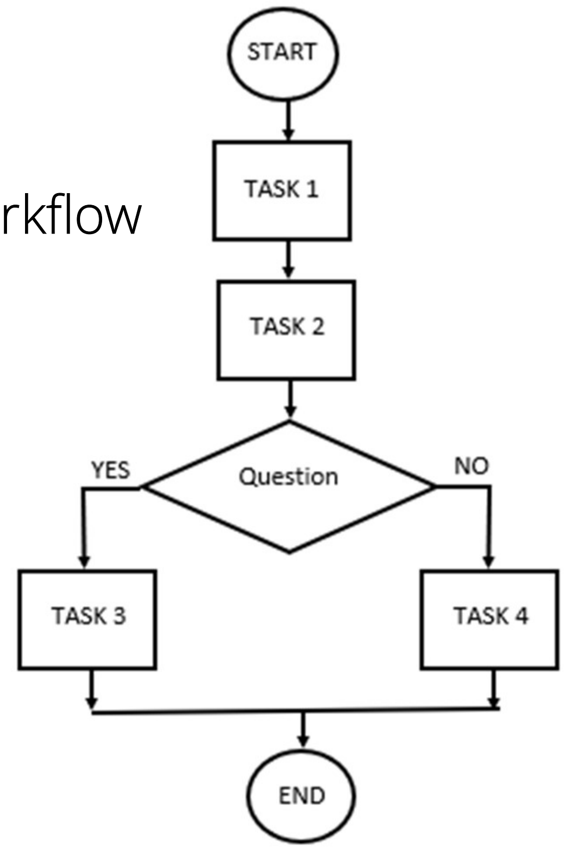
# Workflow Explained: Approaches. . .

Parallel Workflow



# Workflow Explained: Approaches. . .

Conditional Workflow

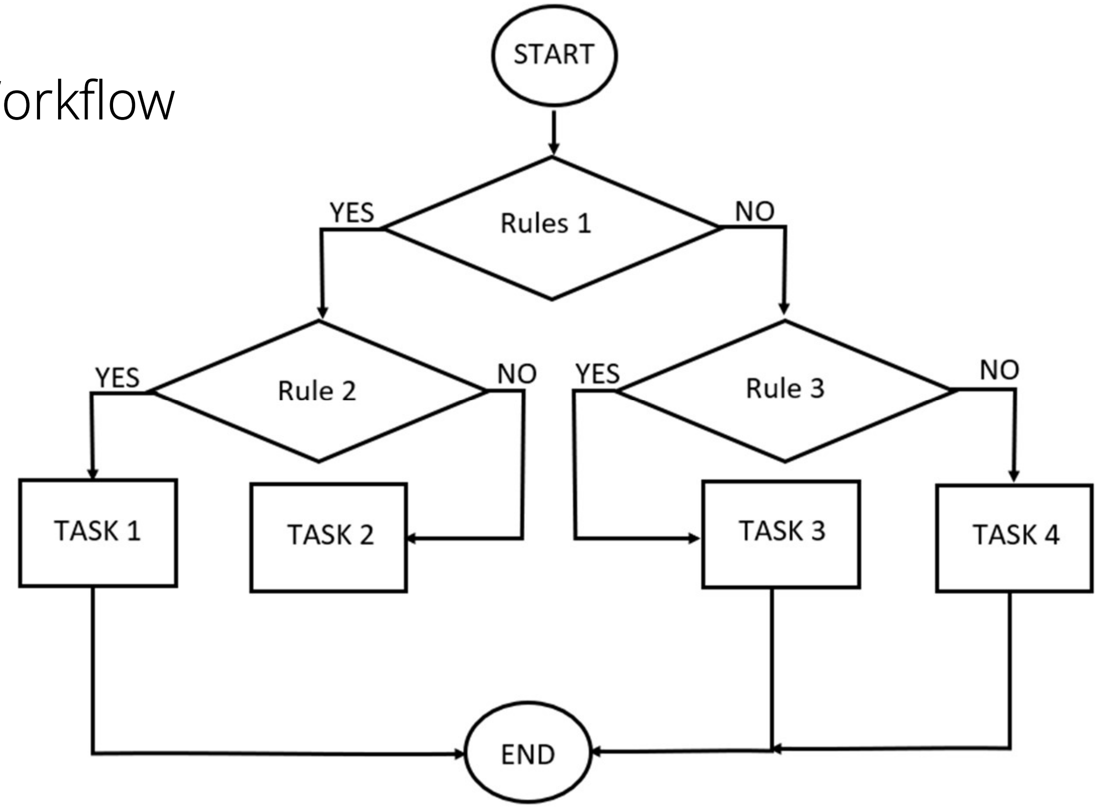


Continuing Education Center

DigiKey

# Workflow Explained: Approaches. . .

Rule-based Workflow

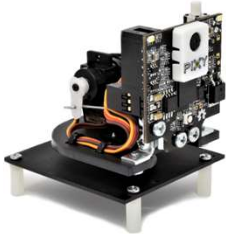


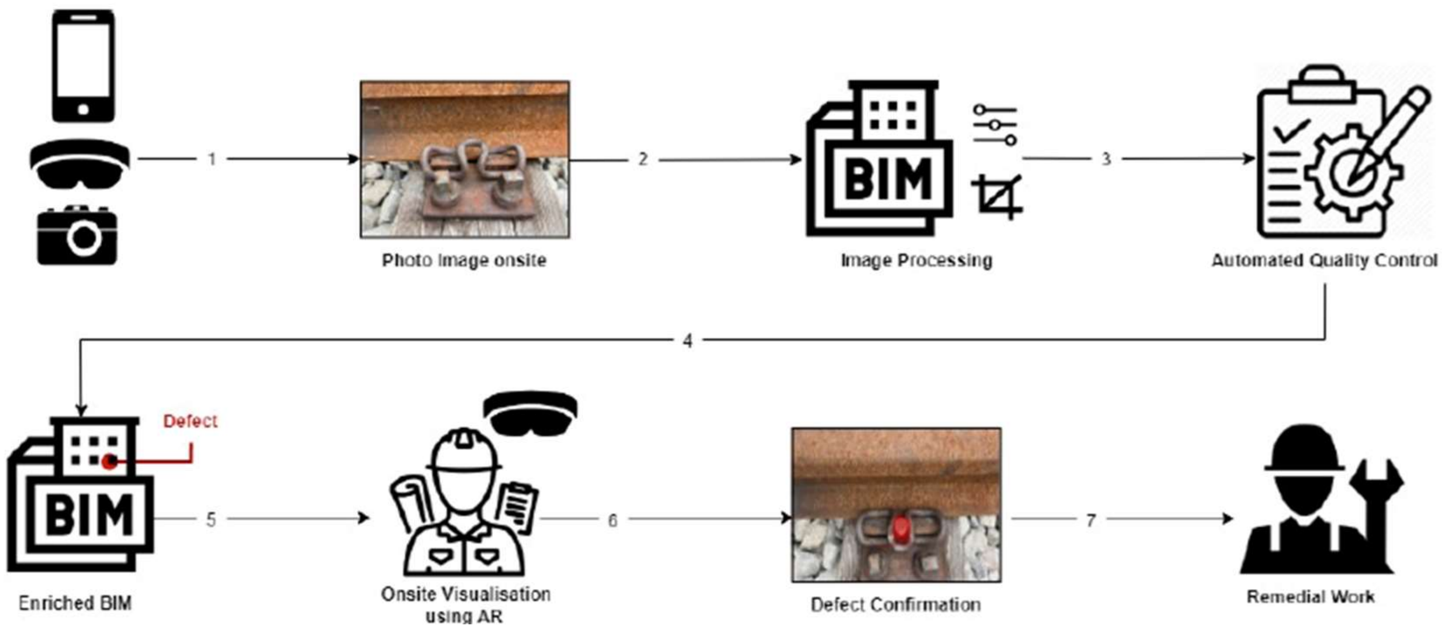
# Question 1

**Which workflow has a linear process?**

a) Parallel
b) Conditional
c) Rule-based
d) Sequential

# Automated Visual Inspection Systems Workflow Examples



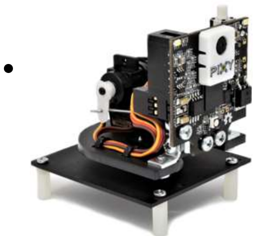


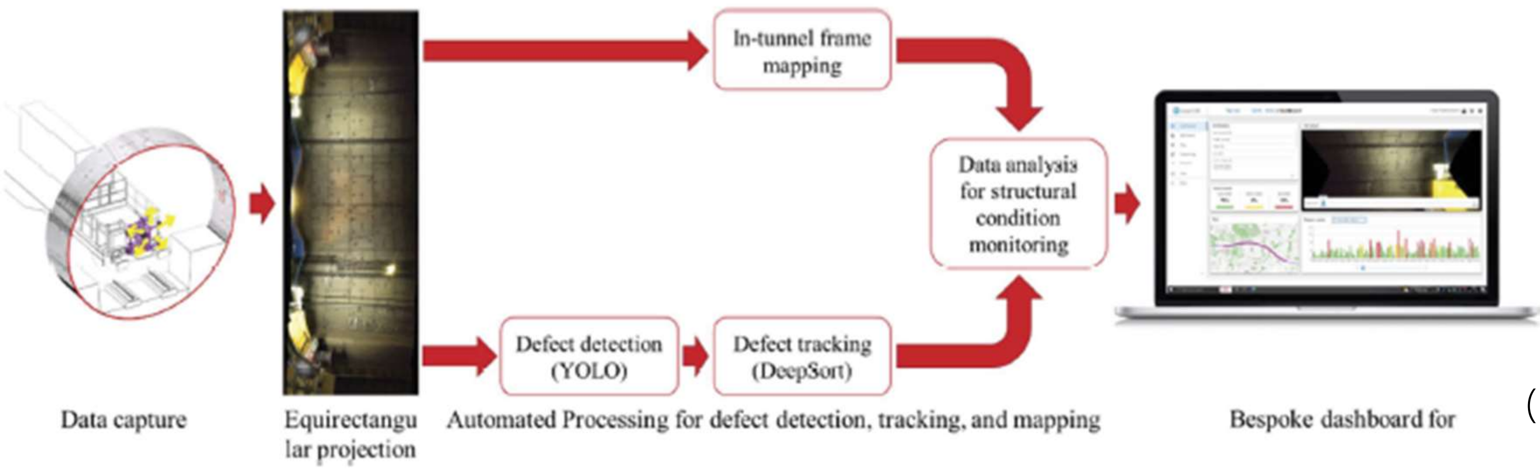

Sequential Workflow

(Gounaridou et al., 2023)

Coinstruction phase digital Twin mOdel (COGITO) project. Semi-automated visual inspection system for detecting railway structure defects.

# Automated Visual Inspection Systems Workflow Examples. . .





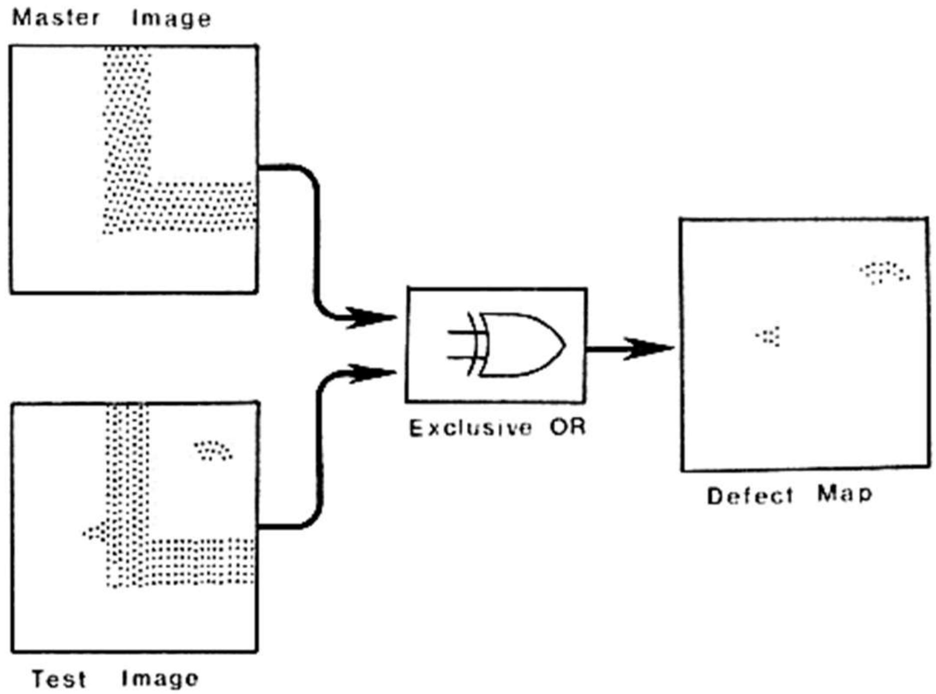

Parallel Workflow

(Panella et al., 2023)

Go Pro Hero 9 and lighting system – Equirectangular projection device

End-to-End development of an automated visual inspection workflow. Semi-automated visual inspection system for detecting tunnel defects.

# Automated Visual Inspection Systems Workflow Examples. . .




Conditional-based Workflow

(Chin & Harlow, 1992)

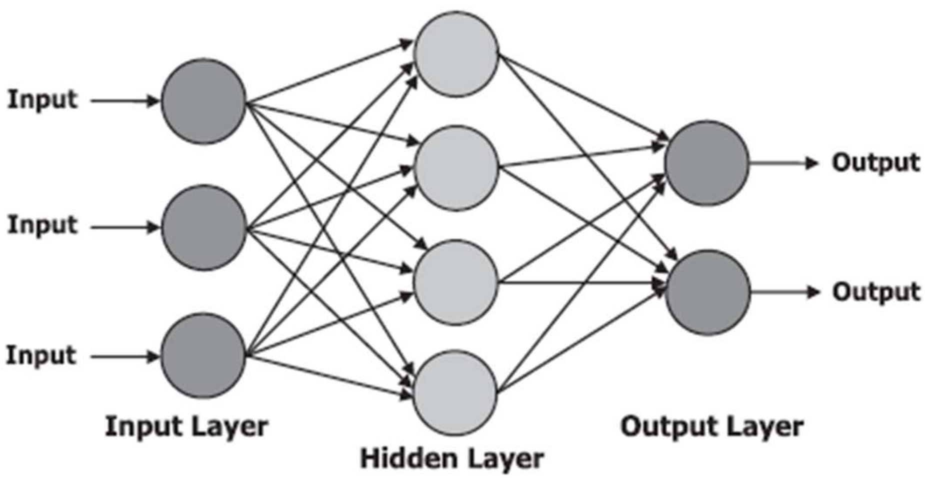
Automated Visual Inspection System for detecting Printed Circuit Board (PCB) defects.

# Transfer Learning Theory: History

- A Pre-trained neural network trained for Task 1. Transfer Learning (TL) allows for
  - a) shorter training time
  - b) known as positive transfer learning
- TL is a learning method using multistage neural networks named Deep Neural Networks (DNNs).
- It has been noted that pioneering work on TL took place in the early 1990s (Bozinovski, 2019).

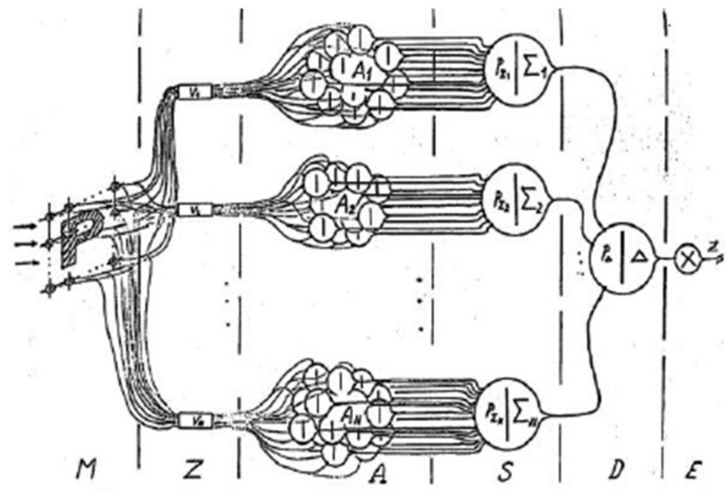
# Transfer Learning Theory: . .
# History

A Neural Network, Typical




(Rahimi & Nazemizadeh, 2013)

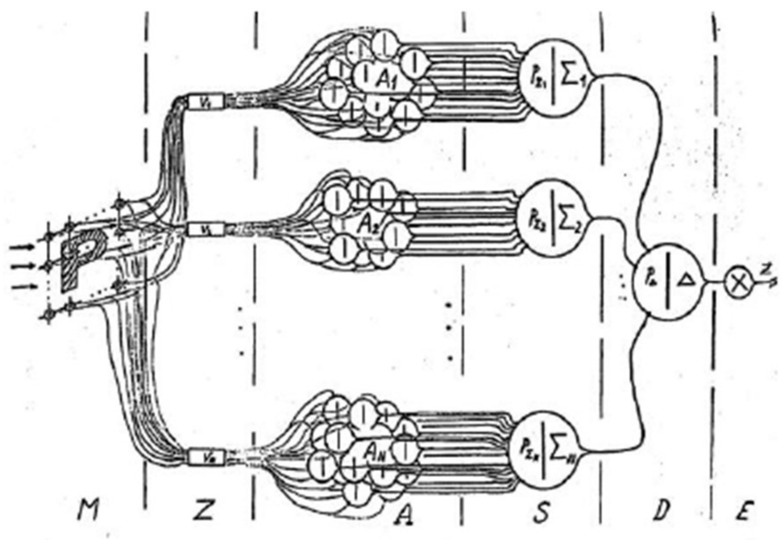
A 5-Layer Neural Network Used in
Supervised Learning For Pattern Recognition




(Bozinovski, 2020, 1974, 1995)

# Transfer Learning Theory: . . . History

A 5-Layer Neural Network Used in Supervised Learning For Pattern Recognition




(Bozinovski, 2020, 1974, 1995)

Definition of Layers:

M- layer: Sensor Layer (Binary 0 or 1)
Z –layer: Feature Extraction (Classes are aligned with Features)
A – layer: Associated Units (Associated Weights)
S – layer: Computes some functions
D – layer: Arbiter – Selects a S element –the usual way of computing a maximum function.
E-layer: Output interface layer used to activate some device that is controlled by the neural network.

# Transfer Learning Theory: . . . History

Dataset used with the 5-Layer Neural Network Model for the Supervised Learning in Pattern Recognition Experiment


IBM29

(Bozinovski, 2020, 1974, 1955)

Pattern Recognition Experiment Parameters.

- TL experiment carried out on decade of images of letters A, B, E, F, and T using an IM29 terminal.
- Such an experiment was used to predict the length of the teaching sequence and the efficiency of the training. (Important Note regarding the experiment's outcome).
- Experiment establishes TL applications like Teachable Machines with multiple dimensional arrays for large datasets using TensorFlow library.

# Question 2

**TL is a learning method using a single neural network named Deep Neural Networks (DNN).**

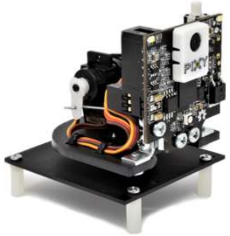
**a) True**

**b) False**

CEC Continuing Education Center

DigiKey

# Transfer Learning Theory: . . .
# ChatGPT LLM Python Code



Here is a prompt that can generate Python code that can be used to explore TL Theory.

> Please write a software agent in Python that models Transfer Learning.

# Transfer Learning Theory: . . . ChatGPT LLM Python Code

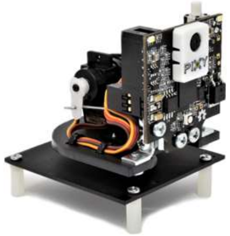
```python
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, GlobalAveragePooling2D
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.datasets import cifar10

# Load the CIFAR-10 dataset
(x_train, y_train), (x_test, y_test) = cifar10.load_data()

# Normalize the data
x_train, x_test = x_train / 255.0, x_test / 255.0

# Load the MobileNetV2 model pre-trained on ImageNet, excluding the top layer
base_model = tf.keras.applications.MobileNetV2(input_shape=(32, 32, 3), include_top=False, weights='imagenet')

# Freeze the base model
base_model.trainable = False

# Create a new model on top
model = Sequential([
    base_model,
    GlobalAveragePooling2D(),
```

TL –Python Code (Partial): Code copied from ChatGPT and pasted into Colaboratory Environment

CEC Continuing Education Center

DigiKey

# Transfer Learning Theory: . . .
# ChatGPT LLM Python Code



Install the Tensorflow library before running the TL-Python Code !

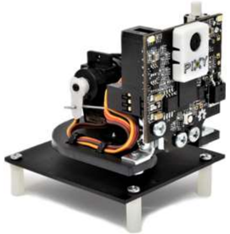
```
1   pip install tensorflow
```

# Transfer Learning Theory: . . .
# ChatGPT LLM Python Code

Generated Output from executed code

```
Downloading data from https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz
170498071/170498071 [==============================] - 6s 0us/step
WARNING:tensorflow:`input_shape` is undefined or non-square, or `rows` is not in [96, 128, 160, 192, 224]. Weights for input shape (224, 224) will be
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/mobilenet_v2/mobilenet_v2_weights_tf_dim_ordering_tf_kernels_1.0_22
9406464/9406464 [==============================] - 1s 0us/step
Epoch 1/10
782/782 [==============================] - 85s 105ms/step - loss: 1.9155 - accuracy: 0.1166 - val_loss: 1.8535 - val_accuracy: 0.1080
Epoch 2/10
782/782 [==============================] - 83s 106ms/step - loss: 1.8309 - accuracy: 0.1197 - val_loss: 1.8281 - val_accuracy: 0.1451
Epoch 3/10
782/782 [==============================] - 82s 105ms/step - loss: 1.8066 - accuracy: 0.1189 - val_loss: 1.8060 - val_accuracy: 0.1742
Epoch 4/10
782/782 [==============================] - 83s 106ms/step - loss: 1.7903 - accuracy: 0.1206 - val_loss: 1.7935 - val_accuracy: 0.1393
Epoch 5/10
782/782 [==============================] - 93s 119ms/step - loss: 1.7791 - accuracy: 0.1156 - val_loss: 1.7964 - val_accuracy: 0.1445
Epoch 6/10
782/782 [==============================] - 83s 106ms/step - loss: 1.7702 - accuracy: 0.1168 - val_loss: 1.7789 - val_accuracy: 0.1344
Epoch 7/10
782/782 [==============================] - 82s 105ms/step - loss: 1.7584 - accuracy: 0.1154 - val_loss: 1.7718 - val_accuracy: 0.1353
Epoch 8/10
782/782 [==============================] - 83s 106ms/step - loss: 1.7545 - accuracy: 0.1136 - val_loss: 1.7833 - val_accuracy: 0.1006
Epoch 9/10
782/782 [==============================] - 82s 105ms/step - loss: 1.7506 - accuracy: 0.1138 - val_loss: 1.7749 - val_accuracy: 0.1049
```

# Transfer Learning Theory: . . .
# ChatGPT LLM Python Code




## What is CIFAR-10?

- The Canadian Institute For Advanced Research (CIFAR) -10 is a set of images that can be used to teach a computer how to recognize objects.
- The dataset contains 60,00 32x32 color images in ten different classes.

## What is Mobilenet-v2?

- A Convolutional Neural Network (CNN) that is 53 layers deep.
- The pretrained CNN can classify images into 1000 object categories.

CIFAR-10 and Mobilent-v2 datasets

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/mobilenet_v2/mobilenet_v2_weights_tf_dim_ordering_tf_kernels_1.0_22
9406464/9406464 [==============================] - 1s 0us/step
```

# Question 3

**The Canadian Institute For Advanced Research (CIFAR) – 20 is a set of images that can be used to teach a computer how to recognize objects.**

**a) False**
**b) True**

Continuing Education Center

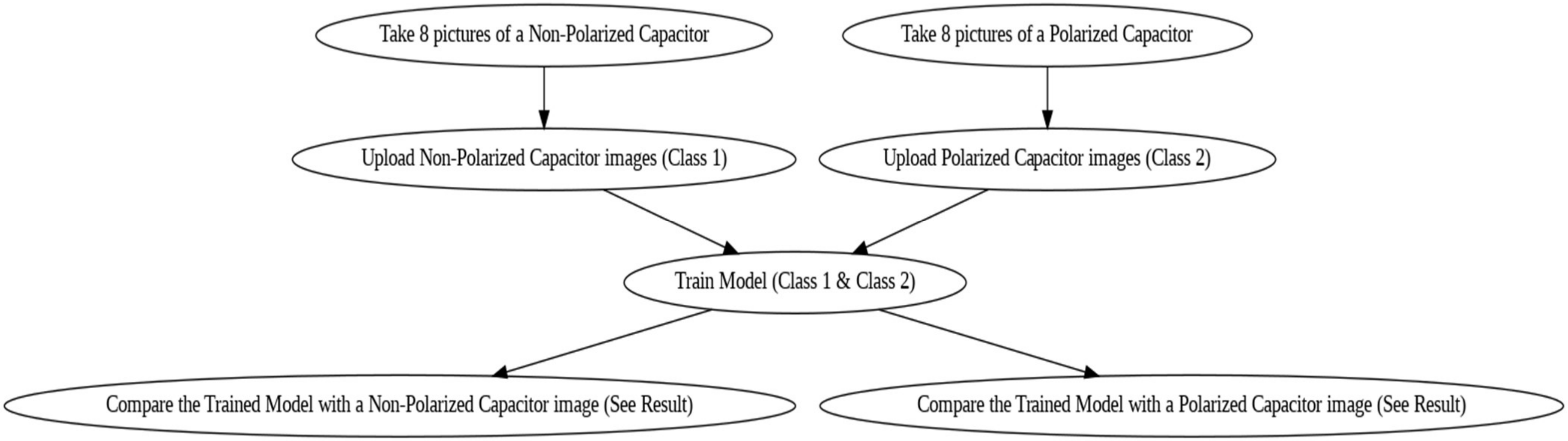
# Transfer Learning Theory: . . . ChatGPT LLM Python Code

## Final Test Results

```
Epoch 10/10
782/782 [==============================] - 190s 243ms/step - loss: 1.7244 - accuracy: 0.1087 - val_loss: 1.6038 - val_accuracy: 0.1053
313/313 [==============================] - 12s 37ms/step - loss: 1.6038 - accuracy: 0.1053
Test accuracy: 0.1053
```

- The test accuracy of 0.1053 (10.53%) obtained from running the Python code is significantly low for CIFAR-10 dataset.
- The outcome results suggest that the model is not performing well on the test set.

# Lab: Build an Automated Visual Inspection Workflow and Report using Python



## Lab: Build an Automated Visual Inspection Workflow and Report using Python. . .

**Lab Objectives:**

- Participants will learn to document requirements for a Teachable Machine Workflow diagram.
- Participants will learn to prompt ChatGPT LLM to create a Teachable Machine Workflow diagram software agent in Python code.
- Participants will learn to execute the Python code within the Colaboratory Environment.
- Participants will learn to create the training model results report using Python within the Colaboratory Environment.

# Lab: Build an Automated Visual Inspection Workflow and Report using Python. . .

Here is a prompt that can generate Python code that can be used to create a workflow diagram.

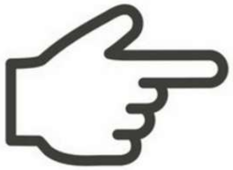
Please create a software agent using Python to create a Teachable Machine Visual Inspection Workflow diagram. The workflow diagram parameters are listed next.

a) Take 8 pictures of a Non-Polarized Capacitor
b) Take 8 pictures of a Polarized Capacitor
c) Upload Non-Polarized Capacitor images (Class 1)
d) Upload Polarized Capacitor images (Class 2)
e) Train Model (Class 1Class 2)
f) Compare the Trained Model with a Non-Polarized Capacitor image (See Result)
g) Compare the Trained Model with a Polarized Capacitor image (See Result)

CEC Continuing Education Center

DigiKey

Click the Play Button to run code



# Lab: Build an Automated Visual Inspection Workflow and Report using Python. . .

```
import graphviz

# Create a new Digraph object
dot = graphviz.Digraph(comment='Teachable Machine Visual Inspection Workflow')

# Define the nodes in the workflow
dot.node('A', 'Take 8 pictures of a Non-Polarized Capacitor')
dot.node('B', 'Take 8 pictures of a Polarized Capacitor')
dot.node('C', 'Upload Non-Polarized Capacitor images (Class 1)')
dot.node('D', 'Upload Polarized Capacitor images (Class 2)')
dot.node('E', 'Train Model (Class 1 & Class 2)')
dot.node('F', 'Compare the Trained Model with a Non-Polarized Capacitor image (See Result)')
dot.node('G', 'Compare the Trained Model with a Polarized Capacitor image (See Result)')

# Define the edges between nodes to create the workflow
dot.edge('A', 'C')
dot.edge('B', 'D')
dot.edge('C', 'E')
dot.edge('D', 'E')
dot.edge('E', 'F')
dot.edge('E', 'G')

# Save the diagram to a file
dot.render('teachable_machine_workflow', format='png')

# Print the DOT source code for the graph
print(dot.source)
```


TL –Python Code:
Code copied from ChatGPT and pasted into Colaboratory Environment

# Question 4

**In reviewing slide 32, which line of code displays the dot source on the screen?**

- a) 16
- b) 7
- c) 24
- d) 27

Continuing Education Center

# Lab: Build an Automated Visual Inspection Workflow and Report using Python. . .

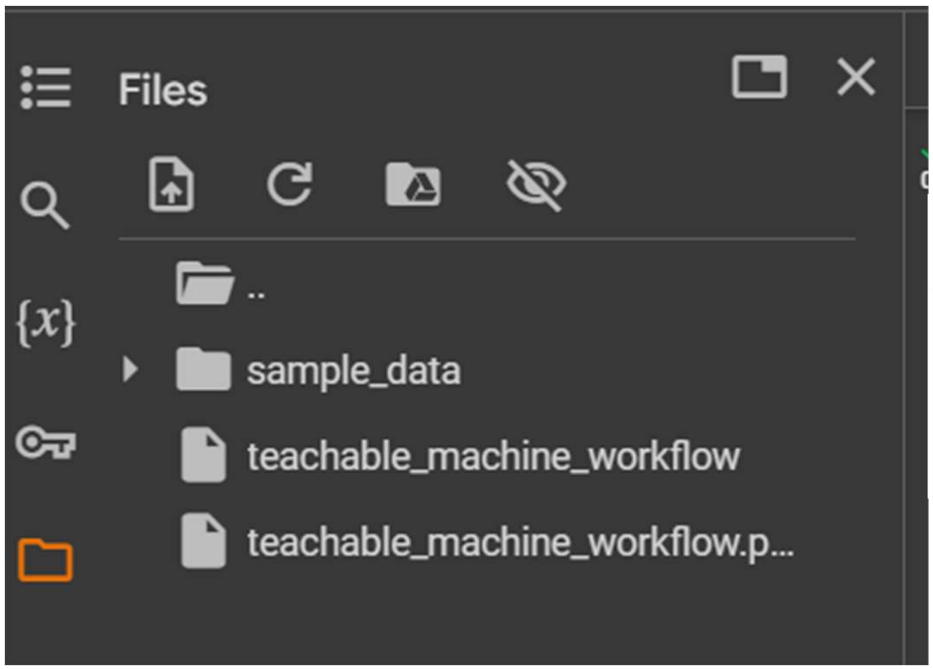
```python
# Print the DOT source code for the graph
print(dot.source)
```

```
// Teachable Machine Visual Inspection Workflow
digraph {
	A [label="Take 8 pictures of a Non-Polarized Capacitor"]
	B [label="Take 8 pictures of a Polarized Capacitor"]
	C [label="Upload Non-Polarized Capacitor images (Class 1)"]
	D [label="Upload Polarized Capacitor images (Class 2)"]
	E [label="Train Model (Class 1 & Class 2)"]
	F [label="Compare the Trained Model with a Non-Polarized Capacitor image (See Result)"]
	G [label="Compare the Trained Model with a Polarized Capacitor image (See Result)"]
	A -> C
	B -> D
	C -> E
	D -> E
	E -> F
	E -> G
}
```

DOT Source Code: Results

# Lab: Build an Automated Visual Inspection Workflow and Report using Python. . .

**Click the folder icon to open window**

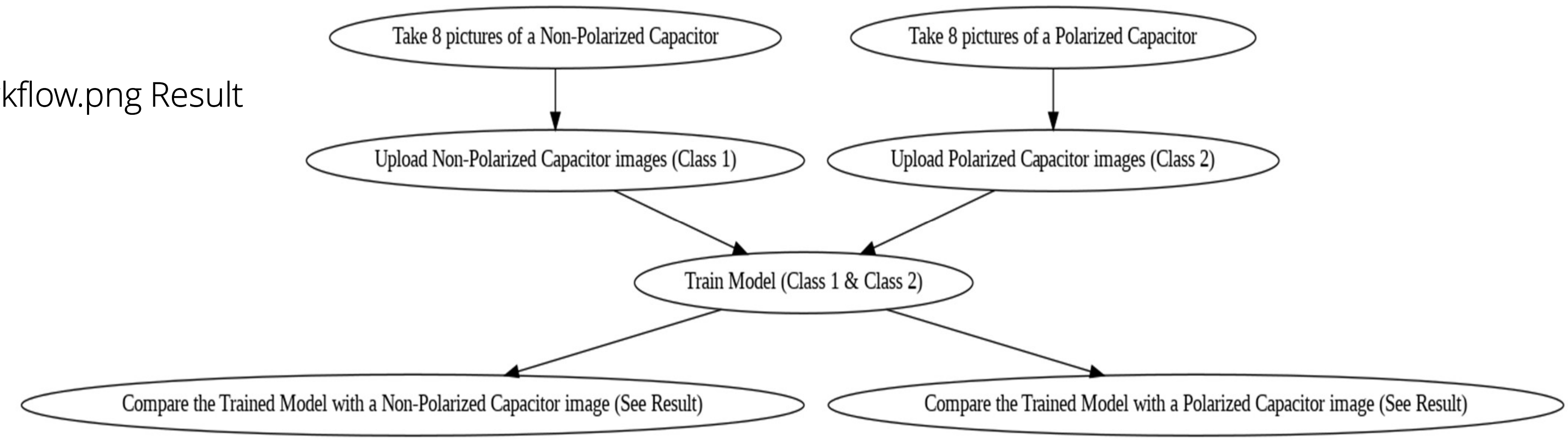



**DOT Source file**

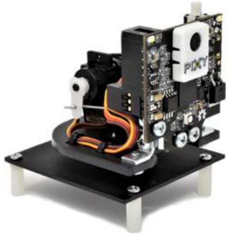
**Workflow diagram.png**

# Lab: Build an Automated Visual Inspection Workflow and Report using Python. . .

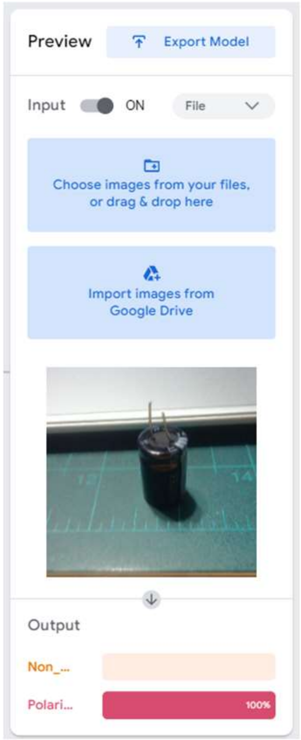
Workflow.png Result

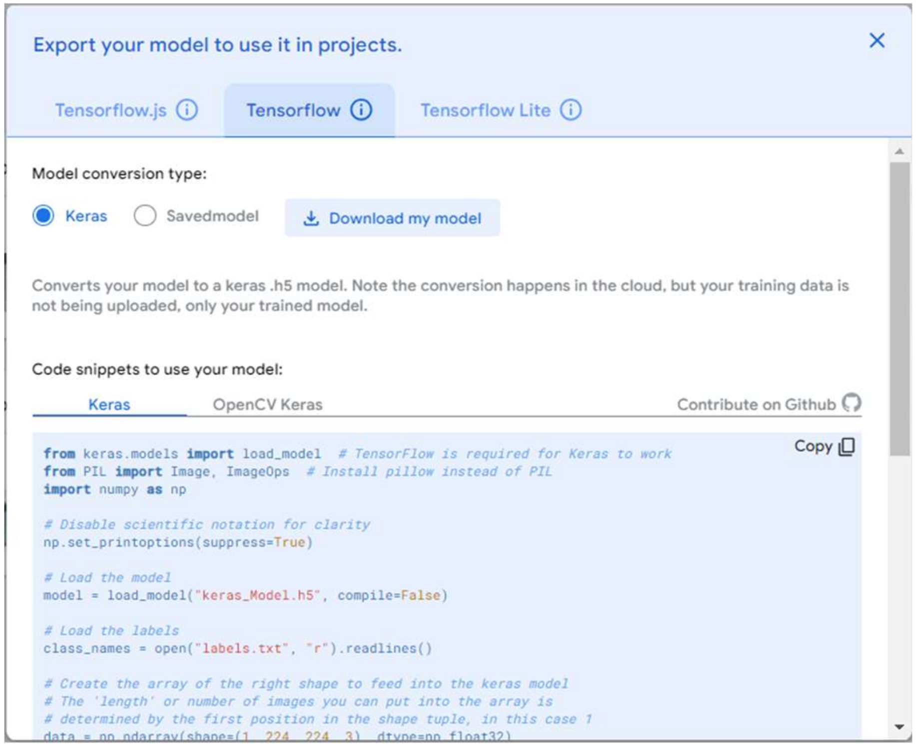


# Lab: Build an Automated Visual Inspection Workflow and Report using Python. . .



**Click the Export Model button**

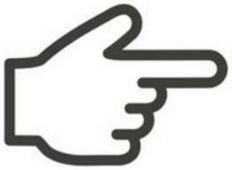


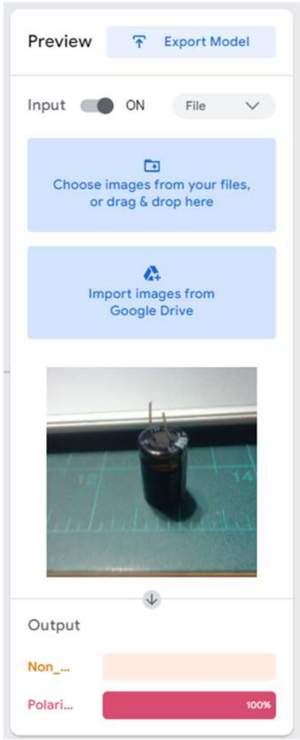



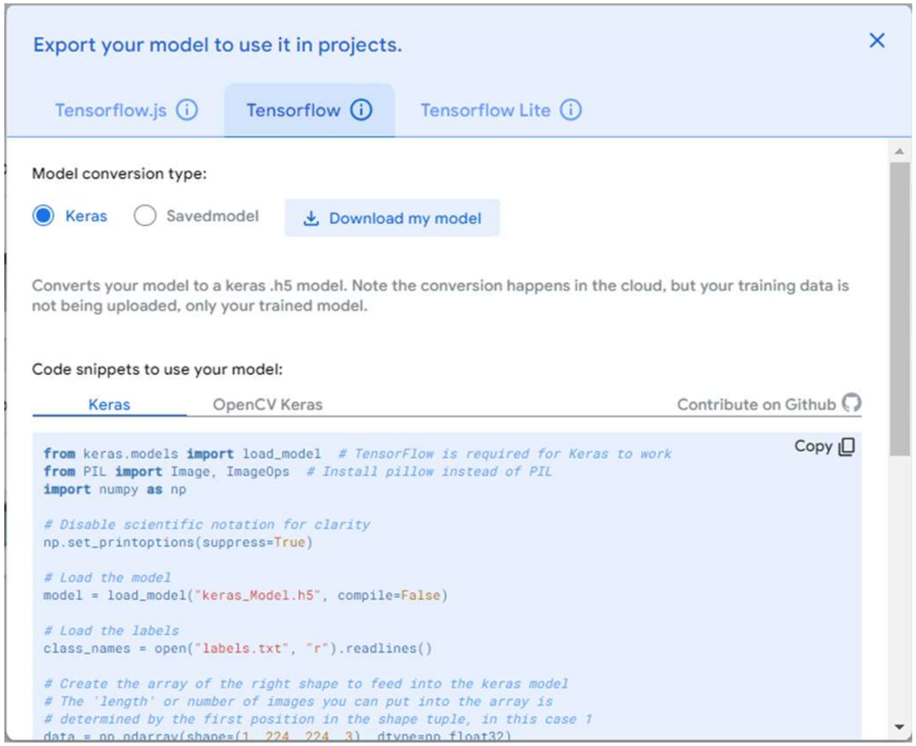

**Copy and paste the code into Colaboratory**

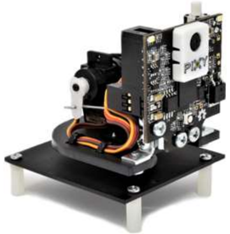
# Lab: Build an Automated Visual Inspection Workflow and Report using Python. . .

**Click the Export Model button**

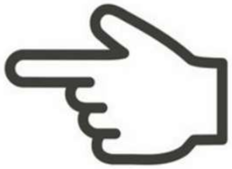




**Click the download my model button**



CEC Continuing Education Center

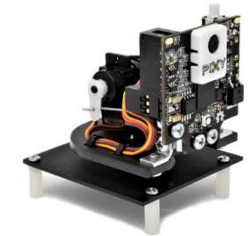
DigiKey

# Lab: Build an Automated Visual Inspection Workflow and Report using Python. . .

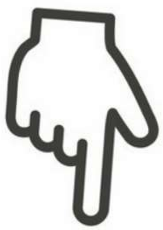
Partial Keras - Python code copied from Teachable Machine (slide 33)

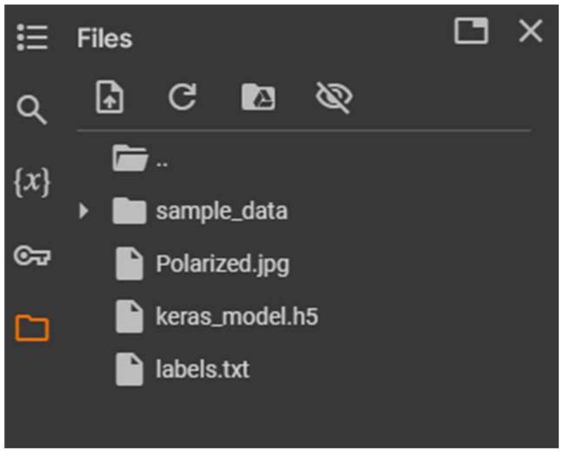
```
from keras.models import load_model  # TensorFlow is required for Keras to work
from PIL import Image, ImageOps  # Install pillow instead of PIL
import numpy as np
import matplotlib.pyplot as plt

# Disable scientific notation for clarity
np.set_printoptions(suppress=True)

# Load the model
model = load_model("keras_model.h5", compile=False)

# Load the labels
class_names = [line.strip() for line in open("labels.txt", "r").readlines()]

# Create the array of the right shape to feed into the keras model
data = np.ndarray(shape=(1, 224, 224, 3), dtype=np.float32)

# Replace this with the path to your image
image_path = "Polarized.jpg"
```

# Lab: Build an Automated Visual Inspection Workflow and Report using Python. . .



**Click the Upload Icon to add these files to Colaboratory**






```
# Load the model
model = load_model("keras_model.h5", compile=False)
```

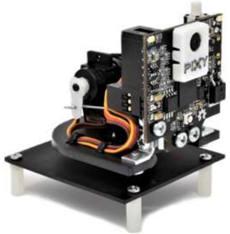
```
# Load the labels
class_names = [line.strip() for line in open("labels.txt", "r").readlines()]
```

```
# Replace this with the path to your image
image_path = "Polarized.jpg"
image = Image.open(image_path).convert("RGB")
```

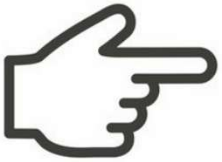
Note 1: The Polarized.jpg is an image selected from the pictures taken with the webcam.

Note 2: The kera_model.h5 and labels.txt files are from clicking the Download the Model button ( slide 34)

# Lab: Build an Automated Visual Inspection Workflow and Report using Python. . .



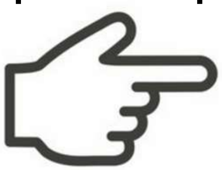
**Click the Play button to run the Python code.**



```python
from keras.models import load_model  # TensorFlow is required for Keras to work
from PIL import Image, ImageOps  # Install pillow instead of PIL
import numpy as np
import matplotlib.pyplot as plt

# Disable scientific notation for clarity
np.set_printoptions(suppress=True)
```

# Lab: Build an Automated Visual Inspection Workflow and Report using Python. . .

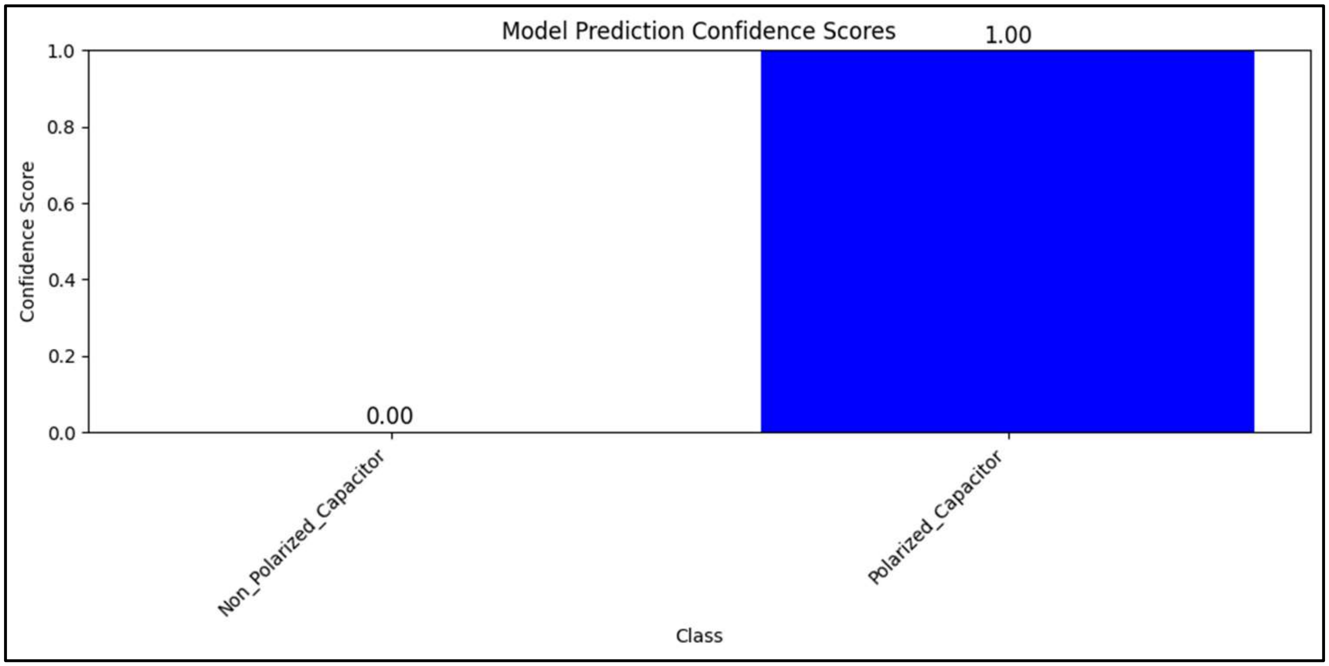
**Right clicking on Chart within Colaboratory and selecting Copy image, picture can be pasted into an inspection report**



Note : Same process used for Polarized Capacitor data can be implemented for Non-Polarized Capacitor information.

Output Results from executed Python Code

```
1/1 [==============================] - 1s 1s/step
Class: Polarized_Capacitor
Confidence Score: 0.9999963045120239
```



# Question 5

**In reviewing slide 42, what is the confidence score predicted on the chart?**

- **a) 0**
- **b) 1**
- **c) 0.99996**
- **d) none of the above**

# Thank you for attending

Please consider the resources below:


Ben-Gal, I, Herer, Y. T., & Raz, T. (2002). Self-correcting inspection procedure under errors. *IIE Transactions, 34*, 529 – 540. https://www.academia.edu/12922699/Self-correcting_inspection_procedure_under_inspection_errors

Bozinovski, S. (2020). Reminder of the first paper on transfer learning in neural networks, 1976. *Informatics 44*, 291-302. https://www.researchgate.net/publication/346435488_Reminder_of_the_First_Paper_on_Transfer_Learning_in_Neural_Networks_1976

Chin, R.T., & Harlow, C. A. (1992). Automated visual inspection: A survey. IEEE Transactions On Pattern Analysis and Machine Intelligence, 4 (6), 557-573. https://ieeexplore.ieee.org/document/4767309

Gounaridou, A., Pantraki, E., Dimitriadis, A.T., Ioaannidis, D., & Tzovaras, D. (2023). Semi-automated visual quality control inspection during construction or renovation of railways using deep learning techniques and augmented reality visualization. *Proceedings of the 23rd International Conference On Construction Applications of Virtual Reality*, 865 -976. https://www.researchgate.net/publication/378535268_Semi-Automated_Visual_Quality_Control_Inspection_During_Construction_or_Renovation_of_Railways_Using_Deep_Learning_Techniques_and_Augmented_Reality_Visualization

Panella, F., Lucy, J., Fisk, E., Huang, S.T., & Loo, Y. (2023). Computer vision and machine learning for cost-effective automated visual inspection of tunnels: A case study. https://www.taylorfrancis.com/chapters/oa-edit/10.1201/9781003348030-340/computer-vision-machine-learning-cost-effective-fully-automated-visual-inspection-tunnels-case-study-panella-lucy-fisk-huang-loo

CEC Continuing Education Center

DigiKey

# Thank you for attending

Please consider the resources below:

Rahimi, H.N., & Nazemizadeh, M. (2013). Dynamic analysis and intelligent control techniques for flexible manipulators: A review. *Advanced Robotics*, 1- 14. https://www.academia.edu/32830488/Dynamic_analysis_and_intelligent_control_techniques_for_flexible_manipulators_a_review

GitHub Code: https://github.com/DWilcher/DesignNews-WebinarCode/blob/main/June_24_Webinar_code.zip

CEC Continuing Education Center

DesignNews

# Thank You

Sponsored by

DigiKey

informa markets