Hochschule Wismar University of Applied Sciences Technology, Business and Design

Fakultät für Ingenieurwissenschaften

Bereich Elektrotechnik und Informatik

Prof. Dr.-Ing. habil. Olaf Simanski , Niklas Huhs

Special Aspects of Automation

Image Recognition Coupling Matlab with Python

Group 11.1

Anshaj Malhotra, 509954

Sujal Pranami, 508650

Paritosh Patel, 510925

Introduction

- **Definition:** Image recognition is a key technology in AI used in security, healthcare, and automation.
- **Objective:** This project integrates **MATLAB and Python** to build an image recognition system.
- **Workflow:**
    1. Train a **Convolutional Neural Network (CNN)** in Python using **TensorFlow/Keras**.
    2. Convert the trained model to **ONNX (Open Neural Network Exchange)** format.
    3. Import the ONNX model into **MATLAB** for further analysis and simulation.

Theoretical Background

- **Convolutional Neural Networks (CNNs)**
  - Specialized deep learning models for image recognition.
  - Extract features through **convolutional layers, pooling layers, and fully connected layers**.
- **CIFAR-10 Dataset**
  - Benchmark dataset with **60,000 images across 10 categories**.
  - Images classified as **airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck**.
- **ONNX Format**
  - Standard for **cross-platform deep learning model exchange**.
  - Enables **seamless transition between Python and MATLAB**.

Model Training in Python

- **Implementation:**
    - Built a CNN with **three convolutional layers, batch normalization, dropout, and pooling layers**.
    - Trained on **CIFAR-10 dataset** using **TensorFlow/Keras**.
- **Key Hyperparameters:**
    - **Batch size:** 64
    - **Learning rate:** 0.001
    - **Epochs:** 20
- **Code Snippets**

1. Load and preprocess the CIFAR-10 dataset

```
(x_train, y_train), (x_test, y_test) = cifar10.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0
```

2. CNN Model Training

```
model = Sequential([
    Conv2D(32, (3,3), activation='relu', input_shape=(32,32,3)),
    MaxPooling2D((2,2)),
    Conv2D(64, (3,3), activation='relu'),
    MaxPooling2D((2,2)),
    Flatten(),
    Dense(64, activation='relu'),
    Dense(10, activation='softmax')
])
```

3.Compile the model

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

4. Add early stopping to prevent overfitting

```
early_stopping = EarlyStopping(
monitor='val_loss', patience=3, restore_best_weights=True
)
```

5. Train the model with augmented data

```
history = model.fit(
datagen.flow(x_train, y_train, batch_size=64),
epochs=20,
validation_data=(x_test, y_test),
callbacks=[early_stopping])
```

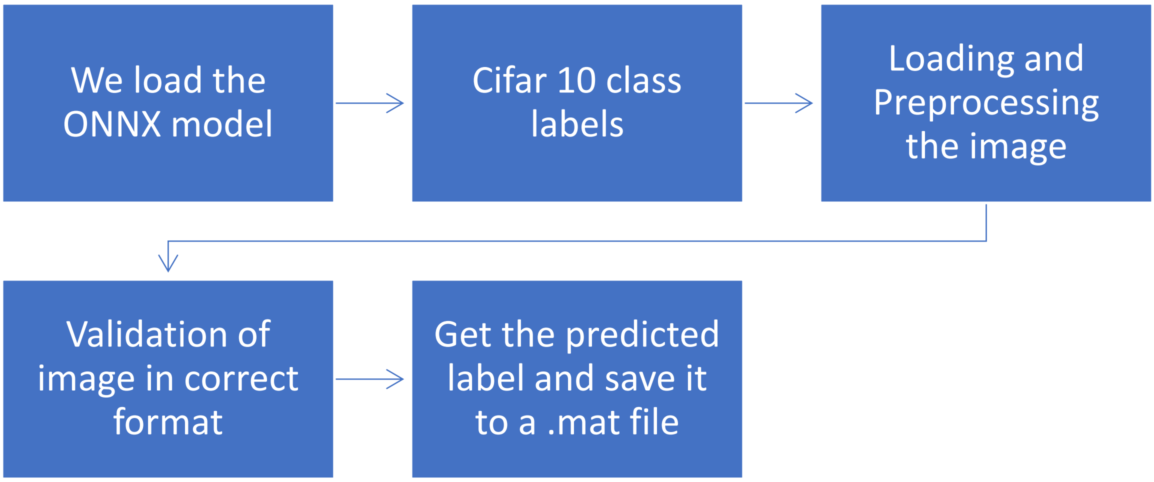
6. *Save in HDF5 .h5 format*

```
h5_file_path = 'cifar10_model.h5'
model.save(h5_file_path)
print(f"Model saved in .h5 format at: {h5_file_path}")
```

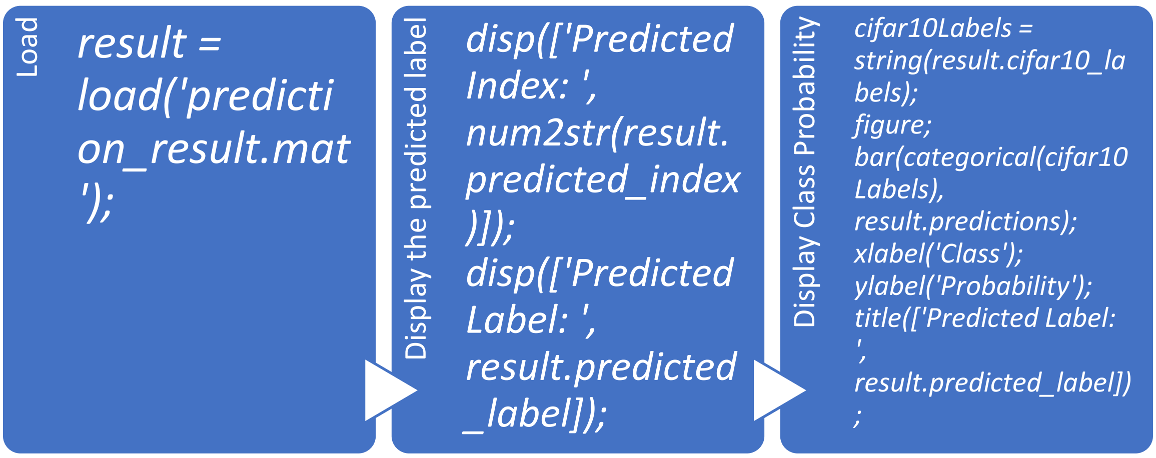
7. Conversion .h5 file to ONNX

```
import tf2onnx

model = tf.keras.models.load_model('cifar10_model.h5')

onnx_model, _ = tf2onnx.convert.from_keras(model)

with open("cifar10_model.onnx", "wb") as f:

   f.write(onnx_model.SerializeToString())
```

8. Predict_ONNX



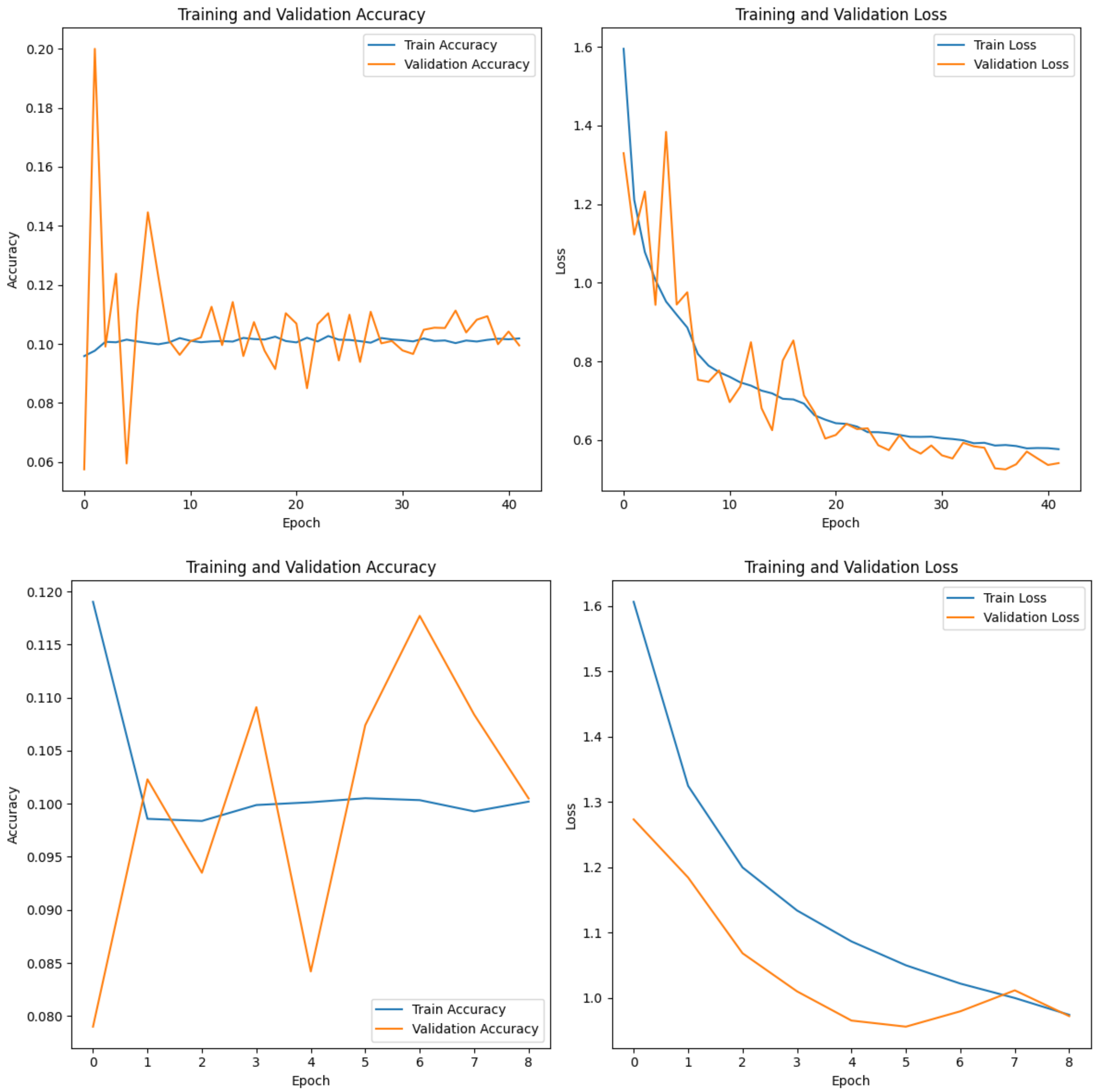
9.Displaying the results in Matlab



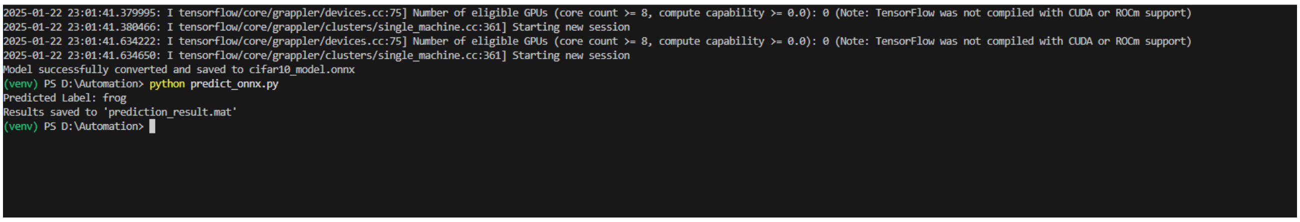
10. Snapshots from Code Run

```
PS D:\Automation> venv\Scripts\activate
(venv) PS D:\Automation> python cifar10_model.py
2025-01-22 22:32:01.235410: I tensorflow/core/util/port.cc:113] oneDNN custom operations are on. You may see slightly different numerical results due to floating-point round-off errors from different computation orders. To tur
n them off, set the environment variable `TF_ENABLE_ONEDNN_OPTS=0`.
WARNING:tensorflow:From D:\Automation\venv\Lib\site-packages\keras\src\losses.py:2976: The name tf.losses.sparse_softmax_cross_entropy is deprecated. Please use tf.compat.v1.losses.sparse_softmax_cross_entropy instead.

WARNING:tensorflow:From D:\Automation\venv\Lib\site-packages\keras\src\backend.py:1398: The name tf.executing_eagerly_outside_functions is deprecated. Please use tf.compat.v1.executing_eagerly_outside_functions instead.

WARNING:tensorflow:From D:\Automation\venv\Lib\site-packages\keras\src\layers\pooling\max_pooling2d.py:161: The name tf.nn.max_pool is deprecated. Please use tf.nn.max_pool2d instead.

2025-01-22 22:32:52.168926: I tensorflow/core/platform/cpu_feature_guard.cc:182] This TensorFlow binary is optimized to use available CPU instructions in performance-critical operations.
To enable the following instructions: SSE SSE2 SSE3 SSE4.1 SSE4.2 AVX2 FMA, in other operations, rebuild TensorFlow with the appropriate compiler flags.
WARNING:tensorflow:From D:\Automation\venv\Lib\site-packages\keras\src\optimizers\__init__.py:309: The name tf.train.Optimizer is deprecated. Please use tf.compat.v1.train.Optimizer instead.

Epoch 1/20
WARNING:tensorflow:From D:\Automation\venv\Lib\site-packages\keras\src\utils\tf_utils.py:492: The name tf.ragged.RaggedTensorValue is deprecated. Please use tf.compat.v1.ragged.RaggedTensorValue instead.

WARNING:tensorflow:From D:\Automation\venv\Lib\site-packages\keras\src\engine\base_layer_utils.py:384: The name tf.executing_eagerly_outside_functions is deprecated. Please use tf.compat.v1.executing_eagerly_outside_functions
instead.

782/782 [==============================] - 27s 30ms/step - loss: 1.6064 - accuracy: 0.1190 - val_loss: 1.2734 - val_accuracy: 0.0790
Epoch 2/20
782/782 [==============================] - 23s 29ms/step - loss: 1.3247 - accuracy: 0.0986 - val_loss: 1.1845 - val_accuracy: 0.1023
Epoch 3/20
782/782 [==============================] - 22s 28ms/step - loss: 1.1999 - accuracy: 0.0984 - val_loss: 1.0685 - val_accuracy: 0.0935
Epoch 4/20
782/782 [==============================] - 22s 28ms/step - loss: 1.1341 - accuracy: 0.0999 - val_loss: 1.0103 - val_accuracy: 0.1091
Epoch 5/20
782/782 [==============================] - 22s 28ms/step - loss: 1.0867 - accuracy: 0.1001 - val_loss: 0.9656 - val_accuracy: 0.0842
Epoch 6/20
782/782 [==============================] - 22s 28ms/step - loss: 1.0502 - accuracy: 0.1005 - val_loss: 0.9562 - val_accuracy: 0.1074
Epoch 7/20
782/782 [==============================] - 22s 28ms/step - loss: 1.0222 - accuracy: 0.1003 - val_loss: 0.9798 - val_accuracy: 0.1177
Epoch 8/20
782/782 [==============================] - 22s 28ms/step - loss: 1.0001 - accuracy: 0.0993 - val_loss: 1.0118 - val_accuracy: 0.1084
Epoch 9/20
782/782 [==============================] - 22s 28ms/step - loss: 0.9743 - accuracy: 0.1002 - val_loss: 0.9724 - val_accuracy: 0.1005
Model saved in TensorFlow SavedModel format at: cifar10_saved_model
Model saved in .keras format at: cifar10_model.keras
D:\Automation\venv\Lib\site-packages\keras\src\engine\training.py:3103: UserWarning: You are saving your model as an HDF5 file via `model.save()`. This file format is considered legacy. We recommend using instead the native Ke
ras format, e.g. `model.save('my_model.keras')`.
  saving_api.save_model(
Model saved in .h5 format at: cifar10_model.h5
(venv) PS D:\Automation>
```

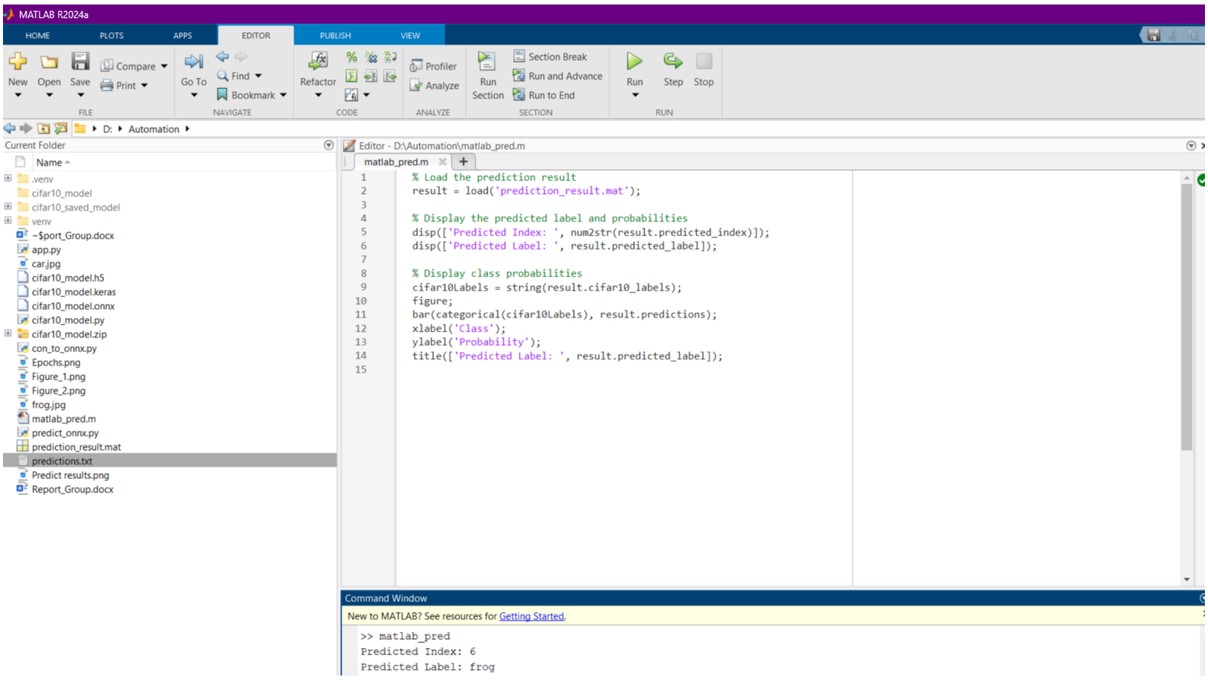
- **Epochs Run**



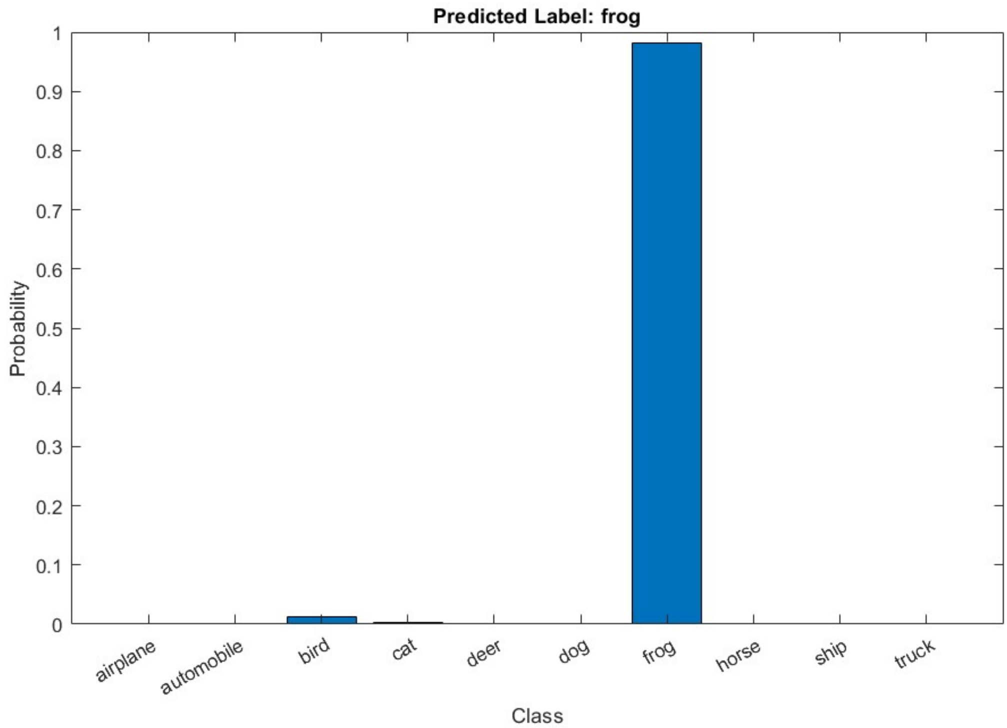
- **Validation Plots in 40 epochs & 8 epochs**



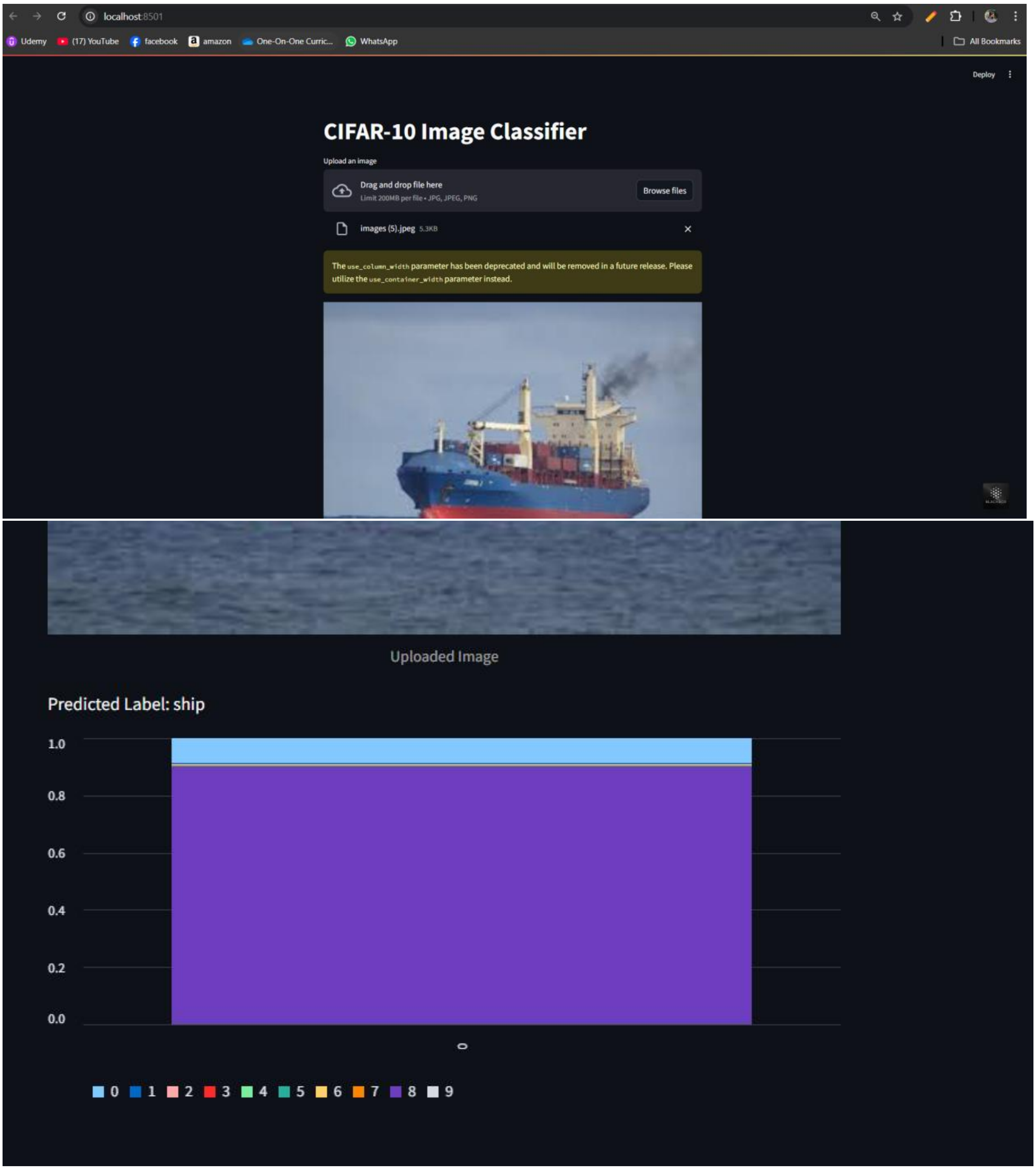
- **Predicted Label**



- **Displaying of Results in Matlab**



- **Probability Plot**



- Made a StreamLit App For validation of CNN model over a Web Application