*Week 7*

*Deploying Deep Learning Models on Web and Smartphone Applications Using Streamlit API*

*Thursday, February22nd 5:30 PM – 6:45 PM*

***DUE: Thursday, February 29th***





**Figure 1. Web and Smartphone Applications for Real-Time Deep Learning (**[**https://tinyurl.com/bdh36zyb**](https://tinyurl.com/bdh36zyb)**,** [**https://tinyurl.com/3fc9wby6**](https://tinyurl.com/3fc9wby6)**)**

*DESCRIPTION*

Although training deep learning models and extracting image meta data are important skills that can be applied for various domains, it is important to deploy and extend these skills for the end-user by developing a web or smartphone application. Therefore, in this lab, you will be using the Streamlit Python API for developing a web application. You will be deploying a deep learning model, allowing users input images, make predictions, and map geocoordinates.

*PREREQUISITES*

1. Complete Labs 1, 2, 3, 4, and 5

*LEARNING OBJECTIVES*

By the end of this lab, students will complete the following:

1. Setup Streamlit onto the edge device and launch a basic application
2. Create additional pages
3. Deploy the image classification model trained in an earlier lab onto the web application
4. Create a map by extracting image metadata on the web application
5. Homework Assignment

*ASSIGNMENT*

**Objective 1: Setup Streamlit onto the edge device and launch a basic application**

1. Power ON the edge device
2. First install Visual Studio Code on the edge device **if you do not have it installed already**:
   1. Open the browser
   2. Go to: <https://code.visualstudio.com/download>
   3. Download the version circled in **RED** below

Graphical user interface

Description automatically generated with low confidence

**Figure 2. Installing Visual Studio Code**

1. Now launch the terminal
2. Activate the anaconda environment and install Streamlit (**figure 3**):
   1. Activate the anaconda environment used in the last two labs by typing conda activate dl and press “enter”
   2. To install Streamlit, type pip install streamlit==0.84.2 and press “enter”

Text

Description automatically generated

**Figure 3. Install Streamlit**

* 1. Type pip install streamlit-folium and press “enter”
  2. Type pip install plotly and press “enter”

1. Navigate to the correct folder and create a Python file (**figure 4**):
   1. Create a new directory called lab 6 by typing mkdir lab6 and press “enter”
   2. Navigate to the new directory by typing cd lab6 and press “enter”
   3. Type gedit main.py and press “enter”
   4. First copy the code shown in **figure 5**.
   5. Save the file and close the Python window

Text

Description automatically generated

**Figure 4. Create a Lab 6 Folder and enter the dl anaconda environment**

Text

Description automatically generated

**Figure 5. main.py starter code**

1. To launch the application, type streamlit run main.py in the terminal and press “enter” (**figure 6**). The browser will automatically launch, and the application home page will appear, as shown in **figure 7**.

Text

Description automatically generated

**Figure 6. Run the application**

Graphical user interface, text, application

Description automatically generated

**Figure 7. Application homepage after starter code**

1. Open Visual Studio Code and open the main.py file within the Lab6 folder. This will allow you to edit the code while you are also using the terminal.
2. Import additional libraries as shown in **figure 8**:

Text

Description automatically generated

**Figure 8. Import additional libraries**

1. After adding code, you do not need to stop and run the application. Simply go to your browser where the application is running and click on the “Rerun” button on the top right corner of the page as shown in **figure 9**.

Graphical user interface

Description automatically generated with medium confidence

**Figure 9. Rerun application**

1. Reference: <https://docs.streamlit.io/library/get-started/create-an-app>
2. **NOTE: After every modification made to the code in main.py Python file, save the code by typing “ctrl+s” and press “enter”. You will only be able to rerun the application from the browser (shown in figure 9) after saving the code.**

**Objective 2: Create additional pages**

1. Now you will create three pages for the web application
2. Create a sidebar as shown in **figure 10** by using the Streamlit sidebar function: st.sidebar.selectbox(). Type the code shown in **figure 10** within the main.py Python file after the line with st.title().



**Figure 10. Create sidebar**

1. You will create 3 pages in this lab: “Homepage”, “Deep Learning”, and “Mapping”
2. Once you rerun the application, the sidebar will appear on the left side of the screen as shown in figure 11.

Graphical user interface, application

Description automatically generated

**Figure 11. Sidebar**

1. Create a Python function named pages() as shown in figure 12 after the line with st.write(). This function will be used to design the different pages in your application. The three pages that were created were “Homepage”, “Deep Learning”, and “Mapping”. The code for designing the “Homepage” has been provided where your application will show an interactive map. You will popular the code for designing “Deep Learning” and “Mapping” pages in Objectives 3 and 4, respectively.

Text

Description automatically generated

**Figure 12. Function to modify the different pages**

1. Call the Python function at the end of the file as circled in **RED** in **figure 12**.
2. Now the application will show the Homepage with a map as shown in **figure 13**.

A picture containing map

Description automatically generated

**Figure 13. Homepage with map of Purdue University**

**Objective 3: Deploy the image classification model trained in an earlier lab onto the web application**

1. Import additional libraries required at the top of the Python file as circled in **RED** in **figure 14**:
   1. VGG16 is the Image Classification model which is trained to identify everyday items
   2. image is used for loading images
   3. preprocess\_input is used for preprocessing the image for making predictions
   4. decode\_predictions will help identify the classes and their probabilities

Text

Description automatically generated

**Figure 14. Add additional libraries for deep learning**

1. Download an image from Google of any object (e.g., car, dog, cat, etc.) and save the image into the Lab6 folder using a simple name (e.g., car.jpg).
2. Within the pages() function where task is “Deep Learning”, you will now use a deep learning model to make predictions and load the predictions on the Web Application (**figure 15**):
   1. Load the VGG16 model with ImageNet weights using the model = VGG16(weights='imagenet')
   2. Provide the image path using the image name that was used for saving the file and place it as shown in figure 15 where img\_path = ‘car.jpg’
   3. Load the image in correct color format and target size: img = image.load\_img(img\_path, color\_mode='rgb', target\_size=(224, 224))
   4. Convert the PIL image into a numpy array using the code line: x = image.img\_to\_array(img)
   5. Expand the dimensions of the image: x = np.expand\_dims(x, axis=0)
   6. Preprocess the image: x = preprocess\_input(x)
   7. Obtain the features after predicting the item in the image: features = model.predict(x)
   8. Identify the class name and probability: p = decode\_predictions(features)

Text

Description automatically generated

**Figure 15. Deep Learning page**

* 1. To display the image on the website, use the st.image() function from the Streamlit API
  2. Finally, output the predictions correctly using st.write() function as shown in **figure 15**

1. Rerun the application and navigate to the Deep Learning page from the sidebar (**figure 16**)

Graphical user interface, application

Description automatically generated

**Figure 16. Select Deep Learning Tab in the sidebar**

1. Finally, you will see your image and the predictions from the deep learning model on a web application as shown in **figure 17**

Graphical user interface, website

Description automatically generated

**Figure 17. Result of the Deep Learning page**

1. Reference: <https://towardsdatascience.com/how-to-use-a-pre-trained-model-vgg-for-image-classification-8dd7c4a4a517>
2. Now you will also allow users to upload images for deep learning-based image classification by adding code in this step below the part circled in **RED** in **figure 15**:
   1. Upload the image by using the st.file\_uploader() function as shown in **figure 18**. The if statement is added to confirm if whether or not a file is uploaded.
   2. Using the PIL library open the image using the Image.open() function as shown in figure

Text

Description automatically generated

**Figure 18. Allow users to upload images and read the image**

* 1. Resize and preprocess the uploaded image as shown in **figure 19**.

Text

Description automatically generated

**Figure 19. Image preprocessing for deep learning**

* 1. Make the predictions and display them on the web application using the code shown in **figure 20**.

Graphical user interface, text

Description automatically generated

**Figure 20. Made predictions from uploaded image**

* 1. The overall code for the “Deep Learning” page is shown in **figure 21**. **NOTE: Make sure the indentation of the code is correct.**

Text

Description automatically generated

**Figure 21. Overall code for step 7**

* 1. Users can upload images by clicking on the “Browse Files” button circled in **RED** in **figure 22**.

Graphical user interface, website

Description automatically generated

**Figure 22. Deep Learning page**

**Objective 4: Create a map on the web application**

1. In this part of the lab, you will be creating a map on the “Mapping” page. Additionally, you will be adding a marker on the map corresponding to the geolocation (**figure 23**):
   1. The latitude and longitude values are shown
   2. As you created maps using folium in Lab 5, use folium to create a map using the folium.Map() function
   3. Create a marker on the map using the folium.Marker() function and replace the popup and tooltip text with “Lab6”
   4. To display the map, use the folium\_static() function

Text

Description automatically generated

**Figure 23. Code for the Mapping page of the web application**

1. Finally, the Mapping page should look like the image shown in **figure 24**.

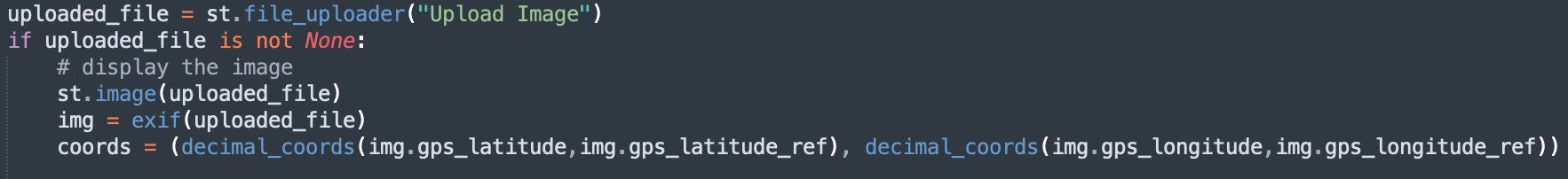
Map

Description automatically generated

**Figure 24. Mapping page**

**Objective 5: Homework Tasks**

1. Modify to task = st.sidebar.selectbox() function from **figure 10** to add an additional page called “Homework”
2. You will also have to modify the pages() function by adding the line elif task == “Homework” at the end of the function in the same way the “Deep Learning” and “Mapping” pages were created (**figure 12**)
3. The page should allow users to upload image files using the uploaded\_file = st.file\_uploader(“Upload Image”) function
4. Upload one of the images that were provided for Lab 5 (**figure 25**)
5. Display the image on the page (**figure 25**)
6. Extract the geocoordinates (**figure 25**)



**Figure 25. Functions for extracting geocoordinates from files.**

1. Using the geocoordinates, map them and drop a marker as you did in **Objective 4**
2. The final “Homework” page is shown in **figure 26**.

Graphical user interface, application

Description automatically generated

**Figure 26. Homework page**

1. You will submit:
   1. Screenshots for each of the pages that were created in this lab
   2. The “main.py” Python file

*REFERENCES / ADDITIONAL RESOURCES*

1. Creating dashboards in Streamlit: <https://towardsdatascience.com/a-multi-page-interactive-dashboard-with-streamlit-and-plotly-c3182443871a>
2. Interactive dashboard in Streamlit: <https://www.turing.com/kb/how-to-build-an-interactive-dashboard-in-python-using-streamlit>
3. Bubble chart dashboard in Streamlit: <https://towardsdatascience.com/building-a-dashboard-in-under-5-minutes-with-streamlit-fd0c906ff886>