*Week 12*

*Introduction to Computer Vision-Based Deep Learning and Image Classification*

*Tuesday, March 28th 5:30 PM – 6:45 PM*

***DUE: Tuesday, April 4th***

Diagram

Description automatically generated

**Figure 1. Artificial Intelligence, Machine Learning, and Deep Learning**

*DESCRIPTION*

The popularity of recent technological advancements such as robots, autonomous cars, and artificial intelligence (AI)-assisted tools (e.g., ChatGPT) has sparked an interest in AI globally. AI essentially corresponds to the ability to program computers to mimic human brain functionality. Machine learning (ML) is a subset of AI which helps train computers by extracting important features from training datasets and train computers to learn common patterns. Deep learning (DL) is a technique of ML which relies on deep neural networks (DNN) to automatically learn important information from training datasets. These DNNs are similar to the neurons in our brains that are used to learn and transfer information. Computer vision is a common application of DL where we train computers to accurately identify, locate, or segment objects of interest using visual features. Convolutional neural networks (CNN) are DNNs that rely on convolutions to train DL models for computer vision applications. This lab will introduce students on how to install DL libraries on a GPU enabled computer, use DL frameworks for training models, and train computer vision models for identifying images.

*PREREQUISITES*

1. Complete Labs 1, 2, and 3

*LEARNING OBJECTIVES*

By the end of this course, students will:

1. Upload DL Jupyter Notebook into Google Colab
2. Import benchmark datasets available through the installed DL libraries
3. Create a CNN architecture to train DL models using benchmark datasets for training
4. Use Transfer Learning approach for training DL models by utilizing state-of-the-art pre-trained models
5. Install DL libraries on the edge device
6. Deploy trained models on the edge device
7. Complete the Homework Assignment

*ASSIGNMENT*

**Objective 1: Upload DL Jupyter Notebook into Google Colab**

1. Power ON the edge device
2. Launch the Chromium browser
3. Login into Brightspace
4. Download the jupyter notebook named “CGT575\_Edge\_Device\_Lab4\_Skeleton.ipynb” from Brightspace
5. In a new tab open Google Colab: <https://colab.research.google.com>
6. Login to your Google account
7. Now you should be able to see Google Colab as shown in **figure 2**. Click on the “Upload” button that is circled in **RED** in **figure 2**.

A screenshot of a computer

Description automatically generated

**Figure 2. Start Google Colab**

1. Then click on “choose a file” button shown in **figure 3,** click Open to upload the .ipynb jupyter notebook that was downloaded from Brightspace.

Graphical user interface, application, website

Description automatically generated

**Figure 3. Load a Jupyter Notebook into Google Colab**

1. After you have uploaded the jupyter notebook on Google Colab, you will be able to run the code blocks in the same manner as with Jupyter Notebooks (**figure 4**). Before running the code, first follow the instructions on step 10.

Graphical user interface, text

Description automatically generated

**Figure 4. Lab4 Notebook in Google Colab**

1. Now click on “Runtime” in the top menu and then click on “Change runtime type” as circled in **RED** in **figure 5** below.

Graphical user interface

Description automatically generated

**Figure 5. Change Runtime**

1. Change the “Hardware accelerator” to GPU for training the models much faster and click on “save” as shown in **figure 6** below:

Graphical user interface, application

Description automatically generated

**Figure 6. Change runtime to GPU**

**Objective 2: Import benchmark datasets available through the installed DL libraries**

1. The code blocks have been created within the notebook. You may run each code block by clicking the “play” button on the top left of each code block as circled in RED in **figure 7**.

Graphical user interface, application

Description automatically generated

**Figure 7. Importing libraries for deep learning**

1. After importing the relevant libraries, proceed with importing the MNIST benchmark dataset for identifying pictures of 10 numbers (i.e., 0, 1, 2, 3, 4, 5, 6, 7, 8, 9). The code shown in **figure 8** will help you import the correct dataset in the correct shape required for training models.
2. After importing the dataset and checking the correct shape, proceed to reshape and normalize the dataset as shown in bottom two code blocks in **figure 8**.

Graphical user interface, text

Description automatically generated

**Figure 8. Importing MNIST dataset**

**Objective 3: Create a CNN architecture to train DL models using benchmark datasets for training**

1. Once the dataset has been imported, you will create a simple CNN model as shown in **figure 9**. Read the commented code which will explain what each line is creating using Keras library within the tensorflow DL framework. This custom model is comprised of 3 convolutional layers and 2 dense layers at the end. The size of the final layer is “10” as you are training a model to identify 10 different classes.
2. After creating the CNN model, show the summary of the model by running model.summary()

Text

Description automatically generated

**Figure 9. Building a custom CNN**

1. After creating the CNN model, you will now Compile it by setting the Hyperparameters such as the optimizer and loss function. These hyperparameters govern how the model will train, and the accuracy will be assessed (**figure 10)**:
   1. The ADAM optimizer is used as this appropriate for the current application
   2. The Cross-entropy loss function helps each layer reduce the loss and improve accuracy
   3. The Metrics used are “accuracy” as this will help track the improvement in accuracy as the model will train
   4. The “model.fit” command will start the training of the model by using the training and testing data that was loaded in Step 11. The model is being trained for 10 epochs.

Graphical user interface, text

Description automatically generated

**Figure 10. Training the CNN model**

1. Once the model has trained, we must evaluate its performance by printing the accuracy and plotting the training and validation accuracy and loss plots (**figure 11**).

Graphical user interface, text, application

Description automatically generated

**Figure 11. Evaluating the CNN model**

1. Once the model is evaluated, you must save the model by running the code block shown in **figure 12** below.

Graphical user interface, text, application

Description automatically generated

**Figure 12. Saving the CNN model**

1. The model will be saved in the current directory. In order to download it, click on the folders icon on the left side of the screen as shown in **figure 13** below:

Graphical user interface, text, application

Description automatically generated

**Figure 13. Locate the saved CNN model**

1. Now click on the three dots next to the “mnist\_custom.h5” file and click on Download as shown in **figure 14**. You will need to load downloaded model files in Objective 6 of this Lab.

Graphical user interface

Description automatically generated

**Figure 14. Download the saved CNN model**

1. Finally, you are required to answer a simple question for reporting what you have observed during the training of the model (**figure 15**):

Graphical user interface, text, application

Description automatically generated

**Figure 15. Question to be answered**

**Objective 4: Use Transfer Learning approach for training DL models by utilizing state-of-the-art pre-trained models**

1. Once the custom CNN model has been trained and evaluated, Transfer Learning will be used to train the models. Scroll down to the Transfer Learning section of the Google Colab Notebook as shown in **figure 16**:

Text

Description automatically generated

**Figure 16. Transfer learning**

1. You may run all the code blocks and become familiar with how Transfer Learning models are trained.
2. The VGG16 pre-trained model will be used, and you will evaluate the trained model in the same manner as before.
3. Once the model is trained and evaluated, you must save the model and download it as shown above in **Objective 3 Steps 4 – 6**.

**Objective 5: Install DL libraries on the edge device**

1. Installing TensorFlow
   1. Open the terminal
   2. Type sudo apt-get update and press “enter” (enter password if prompted)
   3. Type sudo apt-get install libhdf5-serial-dev hdf5-tools libhdf5-dev zlib1g-dev zip libjpeg8-dev liblapack-dev libblas-dev gfortran and press “enter”
   4. Create new anaconda environment with Python3.9 named “dl” by typing conda create -n dl python=3.9 and press “enter” (it will take time, answer “y” to proceed))
   5. Activate the conda environment by typing conda activate dl and press “enter”
   6. Then type pip install -U numpy==1.21.1 future==0.18.2 mock==3.0.5 keras\_preprocessing==1.1.2 keras\_applications==1.0.8 gast==0.4.0 protobuf pybind11 cython pkgconfig packaging h5py==3.6.0 and press “enter”
   7. Type pip install chardet and press “enter”
   8. Now type pip install -U numpy grpcio absl-py py-cpuinfo psutil portpicker six mock requests gast h5py astor termcolor protobuf keras-applications keras-preprocessing wrapt google-pasta setuptools testresources and press “enter”
   9. Install TensorFlow by typing pip install --extra-index-url https://developer.download.nvidia.com/compute/redist/jp/v461 tensorflow==2.11.0rc2 and press “enter” (wait for libraries to install)
   10. To confirm installation, enter python by typing python and press “enter”
   11. Then type import tensorflow as tf and press “enter”
   12. Finally type print(tf.\_\_version\_\_) and press “enter”
   13. The output should be: “2.11.0-rc2”
   14. Type “cntrl+z” to exit python
   15. Install jupyter notebooks in this conda environment by typing conda install -c anaconda jupyter and press “enter” (answer “y” when prompted, wait for libraries to install)
   16. Type conda install nbconvert==5.4.1 and press “enter” (wait for libraries to install)

**Objective 6: Deploy trained models on the edge device**

1. Locate the “mnist\_custom.h5” model that was downloaded from Google Colab and copy it to your home folder
2. Activate the conda environment by typing conda activate dl and press “enter”
3. Launch the jupyter notebook within the new conda environment named “dl” by typing jupyter notebook and pressing “enter”
4. Create new notebook
5. Name the Jupyter Notebook as “Lab4.ipynb”
6. Import the TensorFlow library as shown in **figure 17** below.
7. Load the model using the tf.keras.models.load\_model(“mnist\_custom.h5”) as shown in figure 10 below.
8. Load the MNIST dataset and evaluate the model as shown in **figure 17** below.

Text

Description automatically generated

**Figure 17. Deploying model onto edge device**

**Objective 7: Homework Tasks**

1. Repeat **Objectives 2 and 3** using the Fashion MNIST dataset which has already been imported in the notebook (**figure 18**). Train the models using Google Colab in the same manner using the custom CNN model.

Graphical user interface, text

Description automatically generated

**Figure 18. Homework training custom CNN using Fashion MNIST dataset**

1. Repeat **Objectives 4 and 5** using the Fashion MNIST dataset which has already been import-ed in the notebook (**figure 19**). Train the models using Google Colab in the same manner using the custom CNN model.

Graphical user interface, text

Description automatically generated

**Figure 19. Homework transfer learning using Fashion MNIST dataset**

1. Finally, submit the following on Brightspace:
   1. “**CGT575\_Edge\_Device\_Lab4\_Skeleton.ipynb**”
   2. “**Lab4.ipynb**”

*REFERENCES / ADDITIONAL RESOURCES*

1. Reference: <https://docs.nvidia.com/deeplearning/frameworks/install-tf-jetson-platform/index.html#install>