

---

**Student Name:** \_\_\_\_\_ **Weight:** 8.3%

**Student ID:** \_\_\_\_\_ **Marks:** /36

## Assignment 5: Deep Learning

### Introduction

In this assignment, you will practice deep learning techniques on text and image data using fastai, one of the most famous deep learning Python libraries. You'll apply a language model on movie reviews and train a classifier to determine the sentiments of those reviews. You'll also train a convolutional neural network (CNN) to classify seedlings based on their images.

### Equipment and Materials

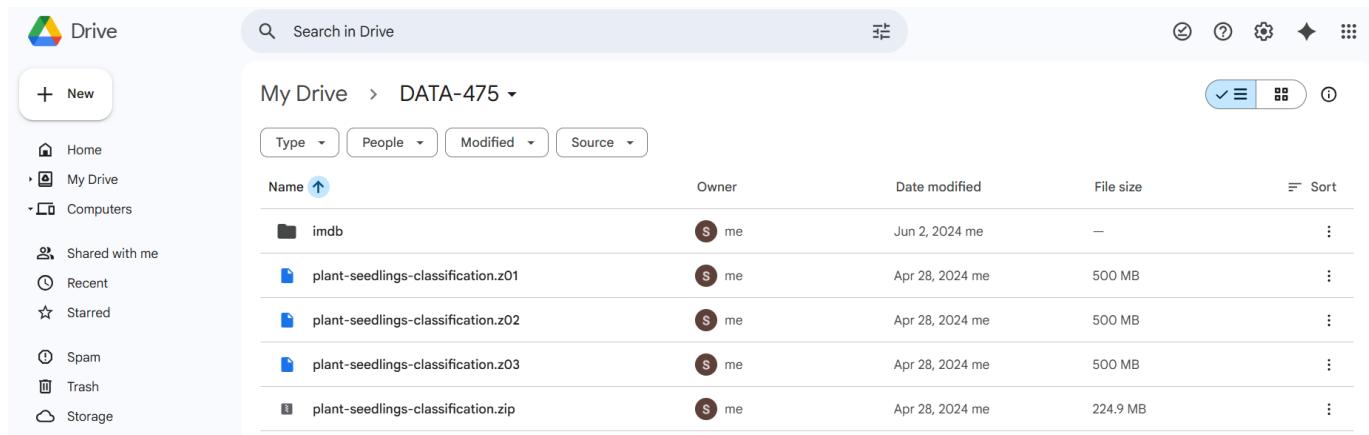
- BYOD laptop
- Google account
- Data files:
  - train.csv (from Assignment 4)
  - test.csv
  - plant-seedlings-classification.zip
  - plant-seedlings-classification.z01
  - plant-seedlings-classification.z02
  - plant-seedlings-classification.z03
- Assignment Activity 1 skeleton file:
  - language\_model.ipynb
- Assignment Activity 2 skeleton file:
  - image\_classification.ipynb

## Instructions

### Assignment Setup

1. Download all the data files listed in the Materials and Equipment section above.
2. Access your Google Drive at <https://drive.google.com/> and create a folder named **DATA-475** in the root.  
**Note:** If you don't have a Google account, you'll need to create one.
3. Upload the **plant-seedlings-classification.zip**, **plant-seedlings-classification.z01**, **plant-seedlings-classification.z02**, and **plant-seedlings-classification.z03** to your **DATA-475** folder.
4. Inside your **DATA-475** folder, create a folder named **imdb**.
5. Upload the **train.csv** and **test.csv** files to your **imdb** folder.

The layout of your DATA-475 should now resemble the screenshot below.



**Figure 1: Google Drive Layout**

© 2025 Google LLC, used with permission.  
Google and the Google logo are registered trademarks of Google LLC.

---

## Assignment Activity 1: Sentiment Analysis Using a Language Model

In this section, you'll continue working on the IMDb movie review dataset from your previous assignment, but this time you'll apply deep learning techniques in NLP to perform sentiment analysis on the textual data.

You need to:

- Reuse a pre-trained language model and fine tune it for movie reviews
- Create a classifier on top of the tuned language model to classify the sentiment of those reviews

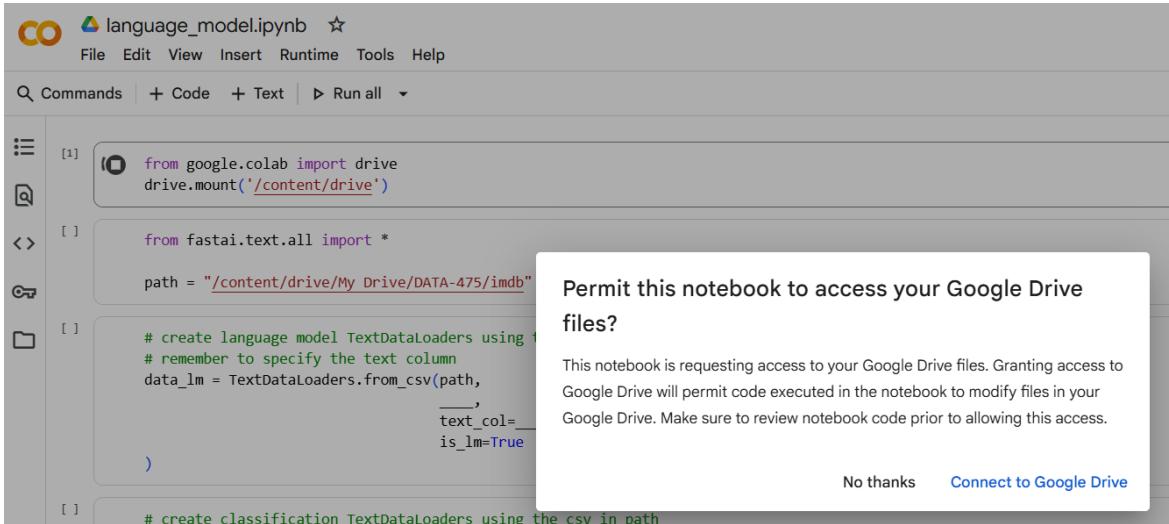
And answer the following questions

- Predict the next 20 words given the syntax: “**this movie talks about Canadian**”
- Classify the sentiment of two example reviews:
  - “This is a horrible movie.”
  - “This is a great movie.”
- Compute the accuracy on test data

### Instructions

1. Download the **language\_model.ipynb** skeleton file to your computer.
2. Go to <https://colab.research.google.com/> and sign in with your Google account credentials.
3. Select **File > Upload notebook** to upload the skeleton file.

4. Run the first cell to mount Google Drive to your notebook.
5. In the message pop-up window, select “Connect to Google Drive”.



The screenshot shows a Jupyter Notebook interface with a code cell containing Python code for mounting Google Drive. A permission dialog box is overlaid on the right side of the screen, asking "Permit this notebook to access your Google Drive files?". The dialog also provides a note about granting access to Google Drive files and includes "No thanks" and "Connect to Google Drive" buttons.

```

from google.colab import drive
drive.mount('/content/drive')

from fastai.text.all import *
path = "/content/drive/My Drive/DATA-475/imdb"

# create language model TextDataLoaders using
# remember to specify the text column
data_lm = TextDataLoaders.from_csv(path,
                                   text_col=...,
                                   is_lm=True
)
# create classification TextDataLoaders using the csv in path

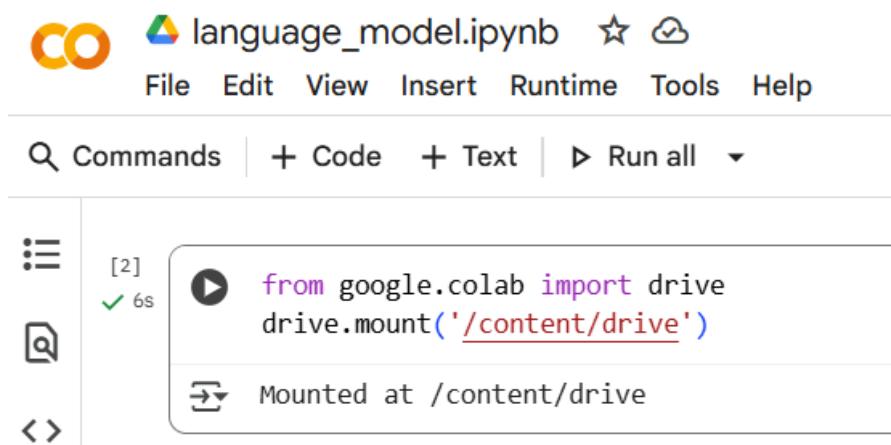
```

**Figure 2. Mount Google Drive**

© 2025 Google LLC, used with permission.

Google and the Google logo are registered trademarks of Google LLC.

6. Select your Google account when prompted to sign-in then select “Continue”.
7. If prompted to “Select what Google Drive for desktop can access” you can scroll to the bottom of the list and select “Continue”, you don’t need to give additional access here.
8. Once the Google Drive is mounted, a message appears, as shown in the image below.



The screenshot shows a Jupyter Notebook interface with a code cell containing Python code for mounting Google Drive. The output of the cell shows a green checkmark and the text "Mounted at /content/drive".

```

from google.colab import drive
drive.mount('/content/drive')

Mounted at /content/drive

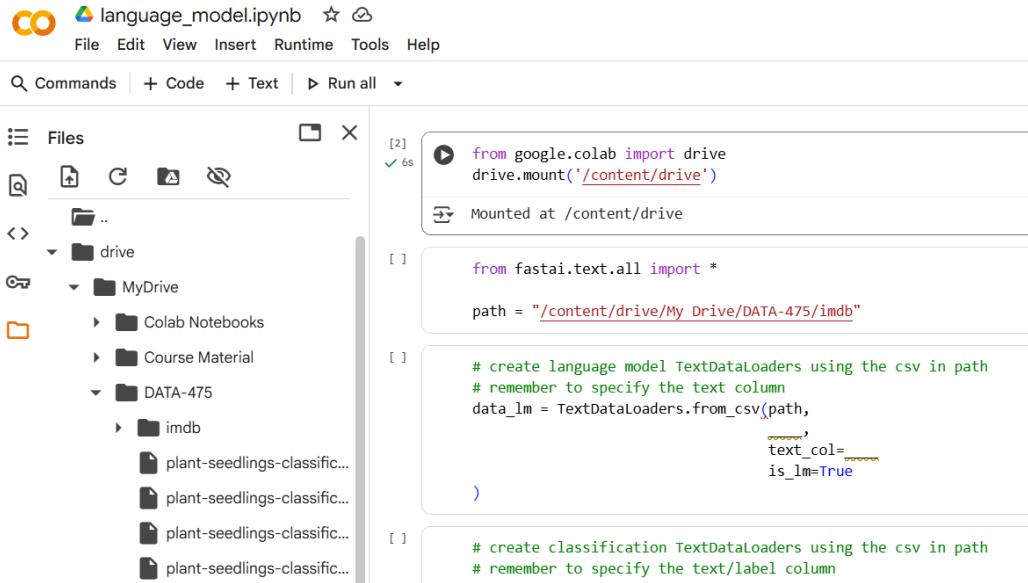
```

**Figure 3: Mount Google Drive Completed**

© 2025 Google LLC, used with permission.

Google and the Google logo are registered trademarks of Google LLC.

9. To confirm that the mounting was successful, click **Files** (the folder icon) on the left sidebar and look for the data files under **drive/My Drive/DATA-475**. It should resemble the image below.



The screenshot shows the Google Colab interface. On the left, the 'Files' sidebar displays a tree structure: .., drive, MyDrive, Colab Notebooks, Course Material, DATA-475, and imdb. Inside the imdb folder, there are several CSV files. On the right, a code cell contains Python code for mounting a drive and creating DataLoaders:

```

from google.colab import drive
drive.mount('/content/drive')

from fastai.text.all import *
path = "/content/drive/My Drive/DATA-475/imdb"

# create language model TextDataLoaders using the csv in path
# remember to specify the text column
data_lm = TextDataLoaders.from_csv(path,
text_col=,
is_lm=True)

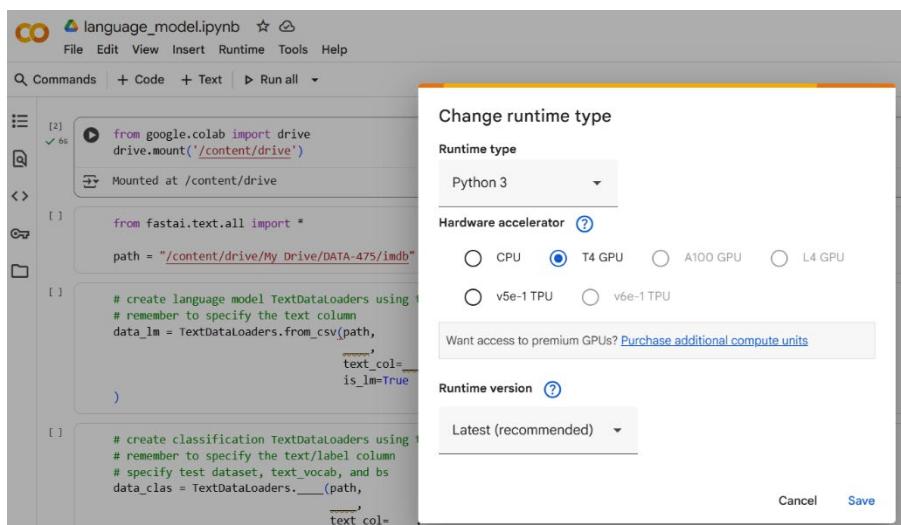
# create classification TextDataLoaders using the csv in path
# remember to specify the text/label column

```

**Figure 4: Find Data Files in Sidebar**

© 2025 Google LLC, used with permission.  
Google and the Google logo are registered trademarks of Google LLC.

10. Since deep learning algorithms require a lot of computational power, select **Runtime > Change runtime type**, and then confirm that **T4 GPU** is selected in the *Hardware accelerator* menu.

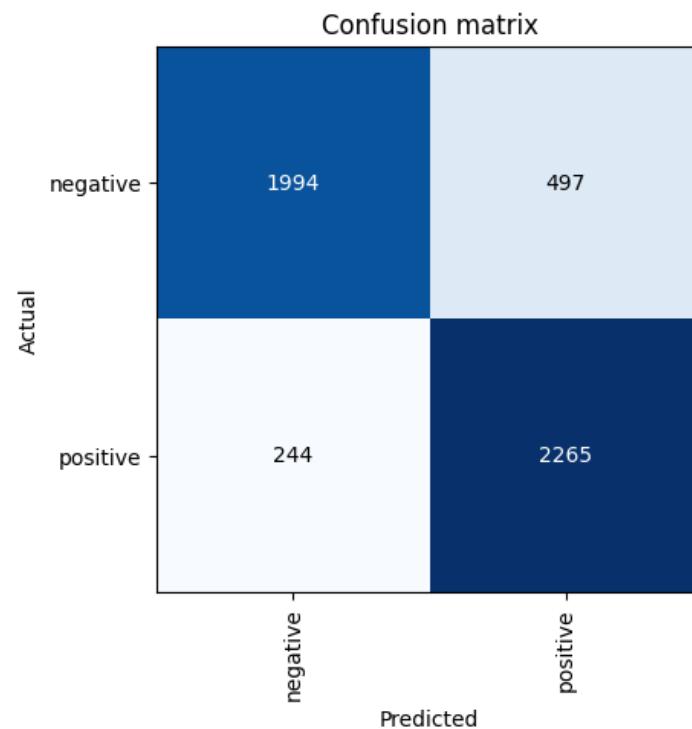


**Figure 5: Configure Hardware Accelerator**

© 2025 Google LLC, used with permission.  
Google and the Google logo are registered trademarks of Google LLC.

- 
11. Throughout the skeleton file, you will need to replace the \_\_\_\_\_ in each code block with the appropriate values.
  12. Create a `TextDataLoader` for a language model named `data_lm` using the training data in the `.csv` file in your `imdb` folder. Make sure you specify:
    - The path to the `.csv` file
    - The name of the training `.csv` file
    - The text column
  13. Create a `TextDataLoader` for a classification model named `data_clas` using the training data in the `.csv` file in your `imdb` folder. Make sure you specify:
    - The path to the `.csv` file
    - The name of the training `.csv` file
    - The text and label columns
    - The vocab as the vocab of training data in `data_lm`
    - The batch size (bs) as 32
  14. Create a language model learner with `AWD_LSTM` architecture, a `drop_mult` of 0.5, and ensure that the metric is accuracy.
  15. Find a proper learning rate and then fit one cycle.
  16. Save the encoder using ‘`model`’ as the argument.
  17. Run the cell to predict the next 20 words for “this movie talks about Canadian”
  18. Run the cell to create a text classifier learner and load the encoder you saved in step 16.
  19. Find a proper learning rate and tune the text classifier learner to achieve over 80% accuracy.
  20. Run the cell to predict the sentiment of “This is a horrible movie.”
  21. Run the cell to predict the sentiment of “This is a great movie.”
  22. Run the following cells to inspect the performance of your model. (Refer to the image below for a confusion matrix. The model has 85% accuracy on the test data.)

```
interp = ClassificationInterpretation.from_learner(  
    clas_learner,  
    ds_idx = 1  
)  
interp.plot_confusion_matrix()
```



**Figure 6: Example Confusion Matrix**

© 2025, Southern Alberta Institute of Technology

## Resources

- [fastai text.data](https://docs.fast.ai/text.data.html#textdataloaders) (<https://docs.fast.ai/text.data.html#textdataloaders>)
- [fastai text.learner](https://docs.fast.ai/text.learner.html#learner-convenience-functions) (<https://docs.fast.ai/text.learner.html#learner-convenience-functions>)

## Assignment Activity 2: Image Classification Using CNN

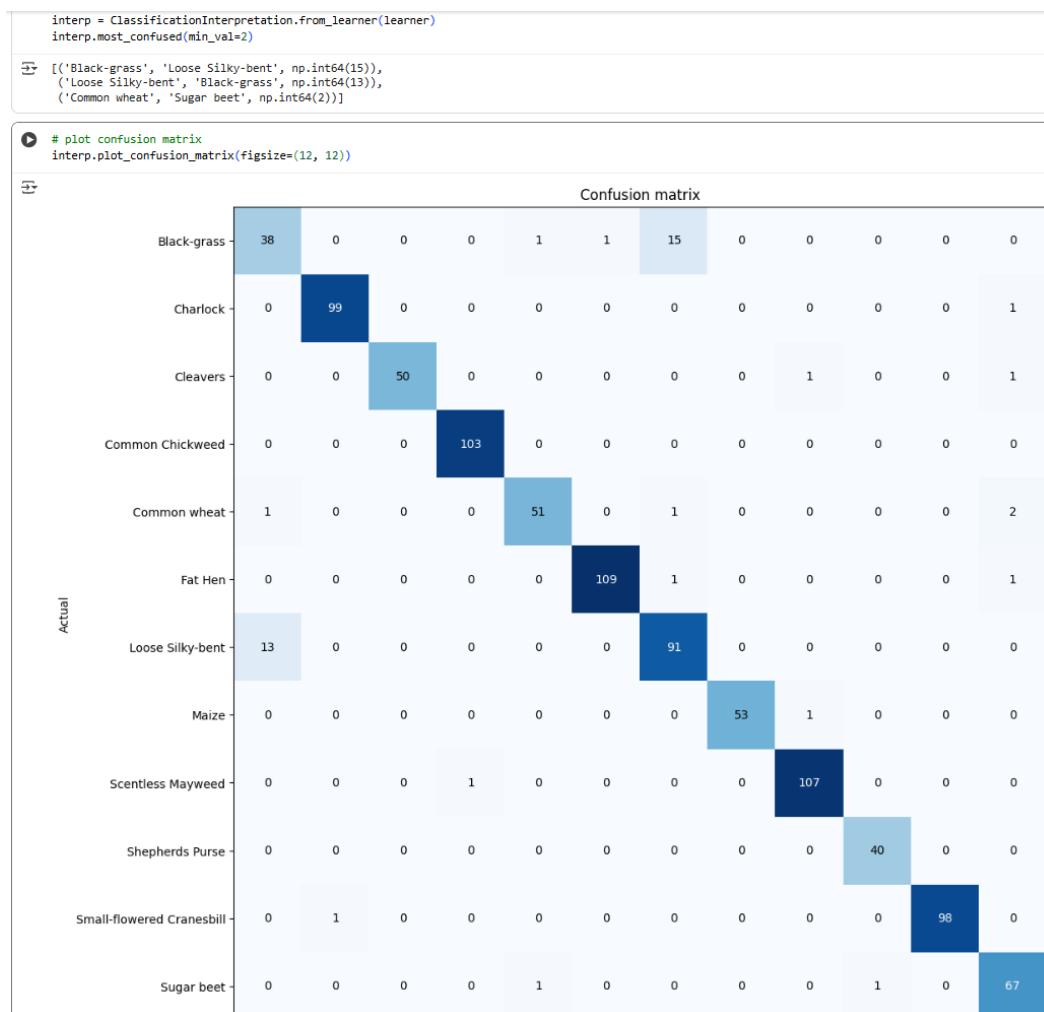
In this section, you'll apply deep learning techniques to a dataset containing images of 12 different seedlings. Your goal is to train a convolutional neural network (CNN) classifier to determine the type of seedlings based on their image.

### Instructions

1. Download the **image\_classification.ipynb** skeleton file to your computer.
2. Go to <https://colab.research.google.com/> and sign in with your Google account credentials.

- 
3. Select **File > Upload notebook** to upload the skeleton file.
  4. Mount your Google Drive and confirm that the runtime has GPU enabled by repeating Steps 4 to 10 from Assignment Activity 1.
  5. Run the second cell to unzip the files in a location named **plant-seedlings-classification**.
    - a. Ensure that this cells executes without error before proceeding.
  6. Run the third cell to print out all seedling labels.
  7. Run the fourth cell to sample one image per type for the seedling images.
  8. Throughout the skeleton file, you will need to replace the \_\_\_\_\_ in each code block with the appropriate values.
  9. Create an `ImageDataLoader` and specify:
    - The path to the folder containing **training images** in the unzipped plant-seedlings-classification folder
    - The portion for validation as 20%
    - The size of images (in pixels) to feed into the model
    - The transforms to augment the training data (flip randomly and allow flipping vertically) and normalize images using `imagenet_stats`.
      - Note: these augmentations are already coded for you.
  10. Create a CNN learner (i.e., a vision learner in fastai) named **learner** using ResNet34.
  11. Find a proper learning rate and then fit one cycle.
  12. Interpret the initial results and identify the most confused classes.
  13. Fine tune the model to achieve over **90% accuracy** on validation.
  14. Interpret the results and identify the most confused classes again.

The following screenshot shows a sample confusion matrix where the model achieves over 95% accuracy.



**Figure 7: Example Confusion Matrix**

© 2025, Southern Alberta Institute of Technology

## Resources

- [fastai vision.data](https://docs.fast.ai/vision.data.html#imagedataloaders) (<https://docs.fast.ai/vision.data.html#imagedataloaders>)
- [fastai vision.learner](https://docs.fast.ai/vision.learner.html#learner-convenience-functions) (<https://docs.fast.ai/vision.learner.html#learner-convenience-functions>)

## Marking Criteria

Criteria	Poor (0)	Satisfactory (1)	Good (2)	Excellent (3)
<b>Procedural Knowledge</b>	Selects inappropriate strategies or skills required by the task or makes critical errors in application.	Selects and applies appropriate strategies or skills but makes a number of non-critical errors in the application.	Selects and applies appropriate strategies and skills without significant errors.	Selects and applies appropriate strategies and skills without error and applies some in innovative ways.
<b>Software Use</b>	Uses software with limited competence.	Uses software with some competence.	Use software with considerable competence.	Uses software with a high degree of competence.
Criteria	Poor (0)	Satisfactory (2)	Good (4)	Excellent (6)
<b>Programming</b>	Unable to identify required data type or data structure. Unable to identify required control structure.	Able to apply require data type or data structure but does not produce correct results.  Able to apply required control structure but does not produce correct results.	Able to apply require data type or data structure and produce partially correct results.  Able to apply required control structure and produce partially correct results.	Able to apply required data type or data structure and produce correct results.  Able to apply required control structure and produce correct results.

Criteria	Poor (0)	Satisfactory (2)	Good (4)	Excellent (6)
<b>Testing</b>	<p>Unable to run program.</p> <p>The program produces incorrect results.</p> <p>Unable to organize the code.</p> <p>No documentation.</p>	<p>Able to run program correctly without any logic errors.</p> <p>The program produces correct results but does not display correctly. Does little check for errors and out of range data.</p> <p>The code is readable only by a person who already knows its purpose.</p> <p>Document is simple comments embedded in code with header separating the codes.</p>	<p>Able to run program correctly without any logic error and display inappropriate output.</p> <p>The program works and meets all specifications. Does some check for errors and out of range data.</p> <p>The code is fairly easy to read.</p> <p>Documentation is simple comments and header that useful in understanding the code.</p>	<p>Able to run program correctly without any logic error and display appropriate output.</p> <p>The program works and meets all specifications. Does exceptional checking for errors and out of range data.</p> <p>The code is extremely well organized and easy to follow.</p> <p>Documentation is well written and clearly explains what the code is accomplishing.</p>

## Assignment Assessment Scoring Sheet

### Assignment Activity 1: Sentiment Analysis Using a Language Model

Criteria	Poor (0)	Satisfactory (1)	Good (2)	Excellent (3)
Procedural Knowledge				
Use of Software				
Criteria	Poor (0)	Satisfactory (2)	Good (4)	Excellent (6)
Programming				
Testing				
Instructor Comments:				
Total Marks:	/18			

## Assignment Activity 2: Image Classification Using CNN

Criteria	Poor (0)	Satisfactory (1)	Good (2)	Excellent (3)
<b>Procedural Knowledge</b>				
<b>Use of Software</b>				
Criteria	Poor (0)	Satisfactory (2)	Good (4)	Excellent (6)
<b>Programming</b>				
<b>Testing</b>				
<b>Instructor Comments:</b>				
<b>Total Marks:</b>	/18			

### Comments