

De La Salle University – Manila

Gokongwei College of Engineering

Department of Electronics and Computer Engineering

CpE Elective 3 Laboratory

LBYCPC4

Summative Laboratory Report

Prepared by:

Banal, Kenan A.

LBYCPC4 – EQ3

# TABLE OF CONTENTS

<Auto-update this table after writing the full content>

Table of Contents

[TABLE OF CONTENTS 2](#_Toc210034151)

[I. INTRODUCTION <Replace the guide text with your own introduction; Approx. 1-2 pages> 3](#_Toc210034152)

[II. METHODOLOGY, RESULTS, AND ANALYSIS <Main Body - Coverage of Topics; Many pages> 3](#_Toc210034153)

[Activity 1: Autoencoders 3](#_Toc210034154)

[1.1. Objectives 3](#_Toc210034155)

[1.2. Experimental Procedure 3](#_Toc210034156)

[1.3. Results and Analysis 3](#_Toc210034157)

[1.4. Answer to Guide Questions, Insights, and Reflections 15](#_Toc210034158)

[Activity 2: <Title> 15](#_Toc210034159)

[2.1. Objectives 15](#_Toc210034160)

[2.2. Experimental Procedure 15](#_Toc210034161)

[2.3. Results and Analysis 15](#_Toc210034162)

[2.4. Answer to Guide Questions, Insights, and Reflections 15](#_Toc210034163)

[<Add more activity sections as needed> 15](#_Toc210034164)

[III. CONCLUSION AND OVERALL UNDERSTANDING <Replace the guide text with your own conclusions. Approx. 1-2 pages> 15](#_Toc210034165)

[IV. REFERENCES <Use APA Citation Style> 16](#_Toc210034166)

Note: Remove all text in between angle brackets (i.e., < and >) as well as the brackets themselves upon submission. This note should also be removed.

# INTRODUCTION <Replace the guide text with your own introduction; Approx. 1-2 pages>

<This report serves as a summative assessment of your understanding and application of the concepts and theory in the design and development of generative deep learning models and the deployment of deep learning models on a single-board computer. It should demonstrate your ability to analyze problems, design and implement working models, and critically reflect on your learning.

In this section, write a complete introduction with the following guidelines:

* State the purpose of this report – e.g., to demonstrate your understanding and application of concepts learned in all laboratory activities.
* Discuss all the concepts and theories you have learned. Ideally, there should be one paragraph that discusses the concepts covered for each laboratory activity. Provide in-text citations that support the statements you have written in relation to the concepts or theories learned.
* In-text citations should refer to reputable sources and should have a corresponding entry in the references section.

>

# METHODOLOGY, RESULTS, AND ANALYSIS <Main Body - Coverage of Topics; Many pages>

## Activity 1: Autoencoders

### Objectives

<Enumerate the objectives as found on the laboratory activity. You may add objectives not mentioned in the activity.

1. Understand the working principle and architecture of autoencoders.
2. Build and train an autoencoder using deep learning framework.
3. Visualize the output of an autoencoder
4. Assess the performance of the implemented autoencoder

### Experimental Procedure

<In your own words, enumerate the step-by-step procedures done. Include illustrations as needed. Use proper figure captions and table numbering.

1. A script was prepared to download and preprocess the dataset so that …
2. … and so on …

>

### Results and Analysis

<Place the data and results required by the activity. Paste relevant snippet codes only (i.e., your code solution). Include illustrations as needed. Use proper figures and table labels. All required figures should have a reasonable solution to be readable. For each code snippet, figure, or table provided, include an explanation and analysis.

Code snippets

|  |
| --- |
| # Load the Wine dataset as a Numpy array  # Perform feature normalization using MinMaxScaler  # Perform train-test split with 20% test data  # Save the training data to variable X\_train  # Save the testing data to variable X\_test  # Save the target to variable y  ### --YOUR CODE HERE-- ###  wine = load\_wine()  X = wine.data  y = wine.target  scaler = MinMaxScaler()  X\_normalized = scaler.fit\_transform(X)  X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_normalized,                                    y, test\_size=0.2, random\_state=42)  y = y  print("X\_train shape:", X\_train.shape)  print("X\_test shape:", X\_test.shape)  print("y shape:", y.shape) |

Figure 1.1. Loading the Wine Dataset and Splitting the Dataset

Explain each of your code solutions…

|  |
| --- |
| # Create the autoencoder model based from the illustrated model plot  # Save the model as autoencoder\_ann variable  ### --YOUR CODE HERE-- ###  input\_dim = X\_train.shape[1]  autoencoder\_ann = Sequential([Input(shape=(input\_dim,)),                                Dense(3, activation="sigmoid"),                                Dense(input\_dim, activation="sigmoid")]) |

Figure 1.2 Creating the Autoencoder

A diagram of a code

AI-generated content may be incorrect.

Figure 1.2.1 Autoencoder Architecture

Explain each of your code solutions…

|  |
| --- |
| # Configure the network for training using the compile method  # Set the optimizer to Adam and determine the appropriate loss function  ### --YOUR CODE HERE-- ###  autoencoder\_ann.compile(optimizer='adam', loss='mse')  # Train the model using X\_train. Use validation with X\_test  # Set the batch size to 10 and the verbosity to 2.  # Ensure the validation loss to be no greater than 0.02  # Use early stopping to determine the appropriate training epochs  # Assign the output to hist\_autoencoder\_ann variable  ### --YOUR CODE HERE-- ###  from keras.callbacks import EarlyStopping  early\_stopping = EarlyStopping(monitor='val\_loss', mode='min', baseline=0.02)  hist\_autoencoder\_ann = autoencoder\_ann.fit(      X\_train, X\_train,      validation\_data=(X\_test, X\_test),      epochs=1000,      batch\_size=10,      verbose=2,      callbacks=[early\_stopping]  )  eval\_base = autoencoder\_ann.evaluate(X\_test, X\_test, verbose=0) |

Figure 1.3. Configuring the Autoencoder and Making of Early Stopper

Explain each of your code solutions…

|  |
| --- |
| # Extract the losses during training and validation  losses = hist\_autoencoder\_ann.history["loss"]  validation\_losses = hist\_autoencoder\_ann.history["val\_loss"]  epochs = range(1, len(losses) + 1)  # Plot the history of training and validation losses  plt.figure(figsize=(10, 5), constrained\_layout=True)  plt.subplot(1, 2, 1)  ### --YOUR CODE HERE-- ###  plt.plot(epochs, losses)  plt.title('Training Loss per Epoch')  plt.xlabel('Epoch')  plt.ylabel('Loss')  plt.grid(True)  plt.subplot(1, 2, 2)  plt.plot(epochs, validation\_losses)  plt.title('Validation Loss per Epoch')  plt.xlabel('Epoch')  plt.ylabel('Loss')  plt.grid(True)  plt.show() |

Figure 1.4. Plotting the Training and Validation loss

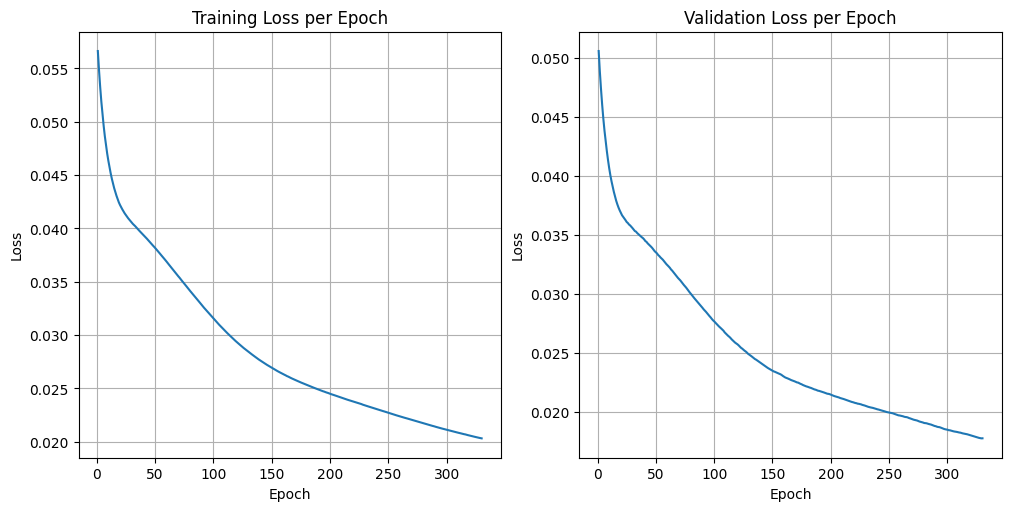


Figure 1.4.1. Training and Validation Loss per Epoch

Explain each of your code solutions…

|  |
| --- |
| # Extract the encoder subnetwork  encoder\_ann = Sequential(autoencoder\_ann.layers[:1])  # Obtain the output of the encoder for each dataset sample  # Save the results to X\_encoded variable  # The results should include both training and testing data  ### --YOUR CODE HERE-- ###  X\_encoded = encoder\_ann.predict(X\_normalized) |

Figure 1.5. Getting of Encoder Prediction

Explain each of your code solutions…

|  |
| --- |
| # Import PCA from Scikit-Learn  from sklearn.decomposition import PCA  X\_PCA = PCA(n\_components=3).fit\_transform(np.vstack((X\_train, X\_test)))  # Create a 3D plot of the PCA components for each dataset point  # Put appropriate title and axis labels  ### --YOUR CODE HERE-- ###  fig = plt.figure(1, figsize=(8, 6))  ax = fig.add\_subplot(111, projection="3d", elev=45, azim=-45)  ax.scatter(      X\_PCA[:, 0],      X\_PCA[:, 1],      X\_PCA[:, 2],      c=y,      s=20  )  # Set titles and axis labels  ax.set\_title("PCA output")  ax.set\_xlabel("Component 1")  ax.set\_ylabel("Component 2")  ax.set\_zlabel("Component 3")  plt.show() |

Figure 1.6. 3D Plotting the PCA Components

A graph of a function

AI-generated content may be incorrect.

Figure 1.6.1 3D Plot of he PCA Components

Explain each of your code solutions…

|  |
| --- |
| # Define another autoencoder with 2 nodes at hidden layer  ### --YOUR CODE HERE-- ###  autoencoder\_2\_nodes = Sequential([      Input(shape=(13,)),      Dense(2, activation='sigmoid'),      Dense(13, activation='sigmoid')  ])  autoencoder\_2\_nodes.summary()  # Configure and train the network  ### --YOUR CODE HERE-- ###  autoencoder\_2\_nodes.compile(      optimizer='adam',      loss='mse'  )  es = EarlyStopping(monitor='val\_loss', mode='min', baseline=0.02)  history\_2\_nodes = autoencoder\_2\_nodes.fit(      X\_train, X\_train,      epochs=1000,      validation\_data=(X\_test, X\_test),      verbose=2,      batch\_size=10,      callbacks=[es]  )  # Plot the history of training and validation losses  ### --YOUR CODE HERE-- ###  train\_losses = history\_2\_nodes.history["loss"]  val\_losses = history\_2\_nodes.history["val\_loss"]  epochs = range(1, len(train\_losses) + 1)  plt.figure(figsize=(10, 5), constrained\_layout=True)  plt.subplot(1, 2, 1)  plt.plot(epochs, train\_losses)  plt.xlabel("Epochs")  plt.ylabel("Training Loss")  plt.title("Training Loss per Epoch")  plt.subplot(1, 2, 2)  plt.plot(epochs, val\_losses)  plt.xlabel("Epochs")  plt.ylabel("Validation Loss")  plt.title("Validation Loss per Epoch")  plt.show()  # Obtain the output of the encoder for each dataset sample  ### --YOUR CODE HERE-- ###  encoder\_2\_nodes = Sequential(autoencoder\_2\_nodes.layers[:1])  X\_encoded\_2\_nodes = encoder\_2\_nodes.predict(X\_normalized)  plt.figure(figsize=(8, 6))  plt.scatter(      X\_encoded\_2\_nodes[:, 0],      X\_encoded\_2\_nodes[:, 1],      c=y,      s=20  )  plt.title("Encoder Output")  plt.xlabel("Component 1")  plt.ylabel("Component 2")  plt.show() |

Figure 1.7 Making Another Autoencoder but with 2D Plots

A graph of loss and loss

AI-generated content may be incorrect.

Figure 1.7.1 Training and Validation Loss per Epoch

A graph of colored dots

AI-generated content may be incorrect.

Figure 1.7.2 2D Plots of the Encoder

Explain each of your code solutions…

|  |
| --- |
| # Import functions and classes  from keras.datasets import fashion\_mnist  from keras.utils import to\_categorical  # Load the Fashion MNIST dataset  # Rescale the pixel values to be between 0 and 1  # Save the training images and labels to x\_train and y\_train variables  # Save the testing images and labels to x\_test and y\_test variables  # Plot ten (10) images from training set in a single row  ### --YOUR CODE HERE-- ###  (x\_train, y\_train), (x\_test, y\_test) = fashion\_mnist.load\_data()  x\_train = x\_train.astype('float32') / 255.0  x\_test = x\_test.astype('float32') / 255.0  class\_names = ['T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat',                 'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot']  fig, axes = plt.subplots(1, 10, figsize=(15, 2),                           subplot\_kw={'xticks':[], 'yticks':[]})  for i, ax in enumerate(axes.flat):      ax.imshow(x\_train[i], cmap='binary')      ax.set\_title(f'{class\_names[y\_train[i]]}', fontsize=10)  plt.suptitle('Original Fashion MNIST Images')  plt.tight\_layout()  plt.show() |

Figure 1.8. Loading and Splitting the Fashion MNIST dataset

A black and white picture of a chair

AI-generated content may be incorrect.

Figure 1.8.1 Loading and Splitting the Fashion MNIST dataset

Explain each of your code solutions…

|  |
| --- |
| # Apply noise to the image  x\_train\_noisy = x\_train + 0.2\*np.random.normal(loc=0, scale=1,                                                 size=x\_train.shape)  x\_test\_noisy = x\_test + 0.2\*np.random.normal(loc=0, scale=1,                                               size=x\_test.shape)  x\_train\_noisy = np.clip(x\_train\_noisy, 0, 1)  x\_test\_noisy = np.clip(x\_test\_noisy, 0, 1)  fig, axes = plt.subplots(1, 10, figsize=(10, 10),                           subplot\_kw={'xticks':[], 'yticks':[]},                           gridspec\_kw=dict(hspace=0.1, wspace=0.1))  for i, ax in enumerate(axes.flat):    ax.imshow(x\_train\_noisy[i], cmap='binary', interpolation='none') |

Figure 1.9. Adding Noise to the Images and Plotting it

Explain each of your code solutions…

|  |
| --- |
| # Import functions and classes  from keras.layers import Conv2D, MaxPooling2D, UpSampling2D  # Create the autoencoder model based from the illustrated model plot  # Save the model as autoencoder\_cnn variable  ### --YOUR CODE HERE-- ###  autoencoder\_cnn = Sequential([      Input(shape=(28, 28, 1)),      Conv2D(32, (3, 3), activation='relu', padding='same', name='conv2d\_1'),      MaxPooling2D((2, 2), padding='same', name='max\_pooling2d'),      Conv2D(32, (3, 3), activation='relu', padding='same', name='conv2d\_2'),      MaxPooling2D((2, 2), padding='same', name='code'),      Conv2D(32, (3, 3), activation='relu', padding='same', name='conv2d\_3'),      UpSampling2D((2, 2), name='up\_sampling2d'),      Conv2D(32, (3, 3), activation='relu', padding='same', name='conv2d\_4'),      UpSampling2D((2, 2), name='up\_sampling2d\_1'),      Conv2D(1, (3, 3), activation='sigmoid', padding='same', name='decode')  ])  autoencoder\_cnn.summary() |

Figure 1.10 Making a Denoising Autoencoder

A black and white diagram

AI-generated content may be incorrect.

Figure 1.10.1 Denoising Autoencoder Architecture

Explain each of your code solutions…

|  |
| --- |
| # Configure the network for training  ### --YOUR CODE HERE-- ###  autoencoder\_cnn.compile(optimizer='adam', loss='mse')  # Train the model. Perform validation as well  # Set the batch size to 100 and the verbosity to 2  # Ensure the validation loss to be no greater than 0.01  # Use early stopping to determine the appropriate training epochs  # Assign the output to hist\_autoencoder\_cnn variable  ### --YOUR CODE HERE-- ###  early\_stopping = EarlyStopping(      monitor='val\_loss',      baseline = 0.01,      mode = 'min'  )  hist\_autoencoder\_cnn = autoencoder\_cnn.fit(      x\_train\_noisy, x\_train\_noisy,      validation\_data=(x\_test\_noisy, x\_test\_noisy),      epochs=900,      batch\_size=100,      verbose=2,      callbacks=[early\_stopping]  ) |

Figure 1.11 Configuring the Autoencoder Parameters with Early Stopper

Explain each of your code solutions…

|  |
| --- |
| # Extract the losses during training and validation  losses = hist\_autoencoder\_cnn.history["loss"]  validation\_losses = hist\_autoencoder\_cnn.history["val\_loss"]  epochs = range(1, len(losses) + 1)  # Plot the history of training and validation losses  plt.figure(figsize=(10, 5), constrained\_layout=True)  plt.subplot(1, 2, 1)  ### --YOUR CODE HERE-- ###  plt.plot(epochs, losses)  plt.xlabel("Epochs")  plt.ylabel("Training Loss")  plt.title("Training Loss per Epoch")  plt.subplot(1, 2, 2)  plt.plot(epochs, validation\_losses)  plt.xlabel("Epochs")  plt.ylabel("Validation Loss")  plt.title("Validation Loss per Epoch")  plt.show() |

Figure 1.12 Plotting the Training and Validation Loss of Autoencoder

A graph of a loss

AI-generated content may be incorrect.

Figure 1.12.1 Training and Validation Loss of Autoencoder

Explain each of your code solutions…

|  |
| --- |
| # Obtain model output for ten (10) input noisy images  # Display both the input noisy images and the output denoised image in two rows  # The first row contains the input noisy images  # The second row contains the output denoised images  ### --YOUR CODE HERE-- ###  denoised\_images = autoencoder\_cnn.predict(x\_test\_noisy[:10])  fig, axes = plt.subplots(2, 10, figsize=(15, 4))  for i in range(10):    axes[0,i].imshow((x\_test\_noisy[i]).reshape(28,28), cmap='gray')    axes[0,i].set\_title("Noisy Img")    axes[1,i].imshow((denoised\_images[i]).reshape(28,28), cmap='gray')    axes[1,i].set\_title("Denoised Img")  plt.suptitle("Image Denoising")  plt.show() |

Figure 1.13. Displaying the Noisy and Denoised Images

A collage of images of a person's body

AI-generated content may be incorrect.

Figure 1.13.1. Image Denoising Dataset

Explain each of your code solutions…

|  |
| --- |
| ### --YOUR CODE HERE-- ###  from sklearn.metrics import mean\_absolute\_error  decoded\_images = autoencoder\_cnn.predict(x\_test\_noisy)  errors = []  for i in range(len(x\_test)):      mae = mean\_absolute\_error(          x\_test[i].reshape(-1),          decoded\_images[i].reshape(-1)      )      errors.append(mae)  errors = np.array(errors)  avg\_error = np.mean(errors)  print("Average MAE:", avg\_error)  plt.hist(errors, bins=50)  plt.title("Distribution of MAE")  plt.xlabel("MAE")  plt.ylabel("Number of Images")  plt.show() |

Figure 1.14. Displaying the Average MAE and its Image Distribution

A graph of a distribution of a number

AI-generated content may be incorrect.

Figure 1.14.1 MAE vs Number of Images Distribution

Explain each of your code solutions…

|  |
| --- |
| ### --YOUR CODE HERE-- ###  import requests  from PIL import Image  from io import BytesIO  clothing\_urls = [    "https://storage.googleapis.com/kagglesdsdata/datasets/929774/1572891/images\_original/00805d0e-7fe5-4251-b577-86065e4f6587.jpg?X-Goog-Algorithm=GOOG4-RSA-SHA256&X-Goog-Credential=databundle-worker-v2%40kaggle-161607.iam.gserviceaccount.com%2F20250914%2Fauto%2Fstorage%2Fgoog4\_request&X-Goog-Date=20250914T111709Z&X-Goog-Expires=345600&X-Goog-SignedHeaders=host&X-Goog-Signature=",    "https://storage.googleapis.com/kagglesdsdata/datasets/139630/329006/fashion-dataset/images/10020.jpg?X-Goog-Algorithm=GOOG4-RSA-SHA256&X-Goog-Credential=databundle-worker-v2%40kaggle-161607.iam.gserviceaccount.com%2F20250914%2Fauto%2Fstorage%2Fgoog4\_request&X-Goog-Date=20250914T102554Z&X-Goog-Expires=345600&X-Goog-SignedHeaders=host&X-Goog-Signature=",    "https://storage.googleapis.com/kagglesdsdata/datasets/139630/329006/fashion-dataset/images/10029.jpg?X-Goog-Algorithm=GOOG4-RSA-SHA256&X-Goog-Credential=databundle-worker-v2%40kaggle-161607.iam.gserviceaccount.com%2F20250914%2Fauto%2Fstorage%2Fgoog4\_request&X-Goog-Date=20250914T102554Z&X-Goog-Expires=345600&X-Goog-SignedHeaders=host&X-Goog-Signature=",    "https://storage.googleapis.com/kagglesdsdata/datasets/139630/329006/fashion-dataset/images/10013.jpg?X-Goog-Algorithm=GOOG4-RSA-SHA256&X-Goog-Credential=databundle-worker-v2%40kaggle-161607.iam.gserviceaccount.com%2F20250914%2Fauto%2Fstorage%2Fgoog4\_request&X-Goog-Date=20250914T102554Z&X-Goog-Expires=345600&X-Goog-SignedHeaders=host&X-Goog-Signature=",    "https://storage.googleapis.com/kagglesdsdata/datasets/929774/1572891/images\_original/002eb5b8-6541-42a3-9596-0d94f7b866ae.jpg?X-Goog-Algorithm=GOOG4-RSA-SHA256&X-Goog-Credential=databundle-worker-v2%40kaggle-161607.iam.gserviceaccount.com%2F20250914%2Fauto%2Fstorage%2Fgoog4\_request&X-Goog-Date=20250914T111709Z&X-Goog-Expires=345600&X-Goog-SignedHeaders=host&X-Goog-Signature=",  ]  x\_external = []  for url in clothing\_urls:    response = requests.get(url, timeout=10)    img = Image.open(BytesIO(response.content))    img = img.convert("L")    img = img.resize((28, 28))    img\_array = np.array(img) / 255.0    x\_external.append(img\_array.reshape(28, 28))  x\_external = np.array(x\_external)  x\_external\_noisy = x\_external + 0.2 \* np.random.normal(      loc=0, scale=1, size=x\_external.shape  )  x\_external\_noisy = np.clip(x\_external\_noisy, 0, 1)  decoded\_external = autoencoder\_cnn.predict(x\_external\_noisy)  fig, axes = plt.subplots(2, 5, figsize=(12, 5))  for i in range(5):      axes[0, i].imshow(x\_external\_noisy[i].reshape(28, 28), cmap="binary")      axes[0, i].set\_title("Noisy")      axes[1, i].imshow(decoded\_external[i].reshape(28, 28), cmap="binary")      axes[1, i].set\_title("Denoised")  plt.suptitle("Noisy and Denoised Images")  plt.show() |

Figure 1.15. Displaying the 5 Tested Noisy to Denoised Images

A group of images of a person

AI-generated content may be incorrect.

Figure 1.15. Five Noisy to Denoised Images

Explain each of your code solutions…

### Answer to Guide Questions, Insights, and Reflections

<Enumerate point-by-point answers to guide questions, followed by insight and learning reflections.

A.6. Can you distinguish the clusters?

The data shows that …

A.8. … and so on …

My insights in this activity…

>

## Activity 2: <Title>

### Objectives

<…>

### Experimental Procedure

<…>

### Results and Analysis

<…>

### Answer to Guide Questions, Insights, and Reflections

<…>

## <Add more activity sections as needed>

|  |
| --- |
| ### --YOUR CODE HERE-- ###  import your mom |

Figure 1.15. Displaying the 5 Tested Noisy to Denoised Images

Figures and Tables

A graph with a line

Description automatically generated

Figure x. Training loss for the autoencoder…

Explain and provide analysis of each plot or model illustration…

Table x. Model accuracy comparison…

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |

Explain and provide analysis of each table provided…

>

# CONCLUSION AND OVERALL UNDERSTANDING <Replace the guide text with your own conclusions. Approx. 1-2 pages>

* < Summarize your findings on how the fundamental concepts and theory apply to the results of the activity conducted.
* Mention the strengths and weaknesses of applying the concepts and theories in laboratory activities based on your actual model implementation experience.
* Discuss the most significant learning moments and challenges you faced in applying the concepts and how you overcame them.
* Briefly mention how the knowledge gained in these modules will contribute to your understanding of the advanced topics in the future.
* Provide a concluding statement summarizing your overall understanding and your recommendations.>

# REFERENCES <Use APA Citation Style>

* Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press.

**RUBRIC FOR SUMMATIVE LABORATORY REPORT ASSESSMENT:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **CRITERIA** | **EXEMPLARY**  **90-100** | **SATISFACTORY**  **80-89** | **DEVELOPING**  **70-79** | **BEGINNING**  **Below 70** | **WEIGHT** |
| Coverage of Topics | All modules are thoroughly addressed with clear explanations and relevant examples from lab work. | Most modules are addressed adequately with explanations and some examples from lab work. | Some modules are addressed superficially or key concepts are missing. Examples from lab work are limited or unclear. | Many modules are missing or poorly addressed. Understanding of key concepts is not demonstrated. Few or no examples from lab work are provided. | 20% |
| Experiment Procedure ***(SO-PI: B1)*** | Laboratory procedure is logically correct including the method used for evaluating results. | Most of the Laboratory procedure, including the method used for evaluating results, are logically correct. | Only around half of the laboratory procedure, including the method used for evaluating results, is logically correct. | Less than half of the laboratory procedure, including the method used for evaluating results, is logically correct. | 20% |
| Experimental Data  ***(SO-PI: B1)*** | Relevant code snippets from lab work are provided for most key concepts. Data gathered is complete, correct and well documented. Tables and graphs are used as required in the procedure. | Relevant code snippets from lab work are provided for most key concepts. Most of the data expected by the experiment are well documented and correct. Missing one required table or graph. | Code snippets are limited, poorly commented, or not always clearly relevant to the concepts being explained. Around 50% of the data are correct and well documented. Missing two to three required tables or graphs. | Few or no relevant code snippets are provided. The data gathered is severely lacking and many of those gathered are questionable. All the required tables and graphs are missing. | 20% |
| Analysis of Experimental Data  ***(SO-PI: B2)*** | All of the data, including errors were analyzed and interpreted correctly using appropriate theories. Clear and extensive evidence of using appropriate probabilistic, statistical, and data analysis techniques. | Most of the data were analyzed and interpreted correctly using appropriate theories. Clear evidence of using appropriate probabilistic, statistical, and data analysis techniques. | Only around 50% of the data were analyzed and interpreted. Some evidence of using appropriate probabilistic, statistical, and data analysis techniques. | Less than half of the data were analyzed and interpreted. Little or no evidence of appropriate probabilistic, statistical, and data analysis techniques. | 20% |
| Concrete understanding   - writing the conclusion  ***(SO-PI: B2)*** | Summarizes the experiment/activity, cites data or output, cites the causes of error, and suggests a recommendation. Conclusion answers all the objectives. Conclusion made is in agreement with the data gathered. | Demonstrates a good overall understanding of most concepts covered and provides some reflection on the learning experience and challenges faced. Conclusion generally answers most of the objectives. Conclusion is stated on the basis of data gathered. | Conclusion is brief and missing significant pieces of information. Conclusion answers only some of the objectives. Conclusion made has some relation with the data gathered. | Demonstrates a limited understanding of the concepts covered and provides little to no meaningful reflection on the learning experience. There is no connection between the conclusion and the objectives and/or data gathered. | 20% |
|  |  |  |  | TOTAL: | 100% |