National College of Ireland

Configuration Manual of Offloading the Computational Overhead of AI-application to the edge devices for face mask detection using Hybrid Computing Framework

MSc Research Project
Cloud Computing

Sumit Verma
Student ID: 20230851

School of Computing
National College of Ireland

Supervisor: Jitendra Kumar Sharma

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Sumit Verma
Student ID:	20230851
Programme:	Cloud Computing
Year:	2022
Module:	MSc Research Project
Supervisor:	Jitendra Kumar Sharma
Submission Due Date:	15/08/2022
Project Title:	Configuration Manual of Offloading the Computational Overhead of AI-application to the edge devices for face mask detection using Hybrid Computing Framework
Word Count:	1048
Page Count:	9

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Sumit Verma
Date:	15th August 2022

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual of Offloading the Computational Overhead of AI-application to the edge devices for face mask detection using Hybrid Computing Framework

Sumit Verma
20230851

1 Introduction

In order to fully configure the project "Offloading the Computational Overhead of AI-application to the Edge Devices for Face Mask Detection Using Hybrid Computing Framework," this paper was prepared. The main goal of this project is to use hybrid computing to offload the computational burden for face mask identification to the edge devices. This document supplements the project paper by focusing on the program's technical features in order to help a new user comprehend them and, if necessary, replicate them. It describes all the steps involved in data extraction and pre-processing in addition to the model-building process. The application's hardware prerequisites have also been specified.

1.1 Research Objective

The following research goals were the main focus of this study.

- To put into practice the architecture for cloud and hybrid computing.
- Face Mask Detection Application Implementation
- Implement the idea of offloading and compare how well the cloud and hybrid computing architecture function.
- Talk about the advantages and disadvantages of each strategy.

2 Section 2

We used Google Colab as our cloud platform for this project in order to train the models utilized in the application. The interface of Google Colab, which is mostly used in machine learning and data science workspaces and is primarily tailored for frameworks like TensorFlow and Keras, is remarkably similar to that of Jupyter Notebook. The second part of the project involves the use of local machines, where the GUI application will be used to shift the processing burden to edge devices.

2.1 Cloud Server Configuration

Cloud Server Configuration	
Operating System (OS)	Ubuntu 22.04
Main Memory (RAM)	16 GB
CPU Cores	8
Hard Disk	500 GB
Tools and Libraries	Webrtc, ngrok, TensorFlow, keras, pandas and matplotlib

Figure 1: Server Configuration

2.2 Software Configuration

The Google Colab software environment and the Jupiter notebook for the deep learning models are used to configure the software for this research project. All of the models were programmed in the Python programming language using version 3.6.3. While conducting this research, a large number of Python libraries were used. Tersonflow (2.5.0), Keras, Matplotlib (3.3.3), Sklearn, Pandas, Numpy, and other libraries are utilized.

3 Data Gathering

To use deep learning models, the first step is to gather data. The data should include photographs of people wearing and not wearing masks, which have been gathered from Kaggle (Face Mask Detection Dataset, 2022). The dataset includes 7500 RGB photos that are divided into with mask and without mask categoriesWang et al. (2019). There are around 3700 photographs of faces with masks and 3800 images of faces without masks in the dataset, which is balanced. the sample dataset with labels that have and don't have masks.

4 Data Transformation

4.1 Data Pre-processing Image Augmentation

The facemask data needs to be pre-processed, where the height and width of images are set to (160X160) size, in order to improve the model performances and outcomes. Photos have been enhanced by storing the existing images with two additional camera

perspectives and performing rotation and flip operations at various angles. TensorFlow has built-in functionality for data augmentation.Zhang et al. (2018).

```
[ ]  #data-augmentation
     data_augmentation = keras.Sequential(
       [
         layers.RandomFlip("horizontal",
                           input_shape=(img_height,
                                        img_width,
                                        3)),
         layers.RandomRotation(0.1),
         layers.RandomZoom(0.1),
       ]
     )

[ ]  plt.figure(figsize=(10, 10))
     for images, _ in train_ds.take(1):
       for i in range(9):
         augmented_images = data_augmentation(images)
         ax = plt.subplot(3, 3, i + 1)
         plt.imshow(augmented_images[0].numpy().astype("uint8"))
         plt.axis("off")
```

Figure 2: Data Pre-processing and Image Augmentation

Results after Data Augmentation

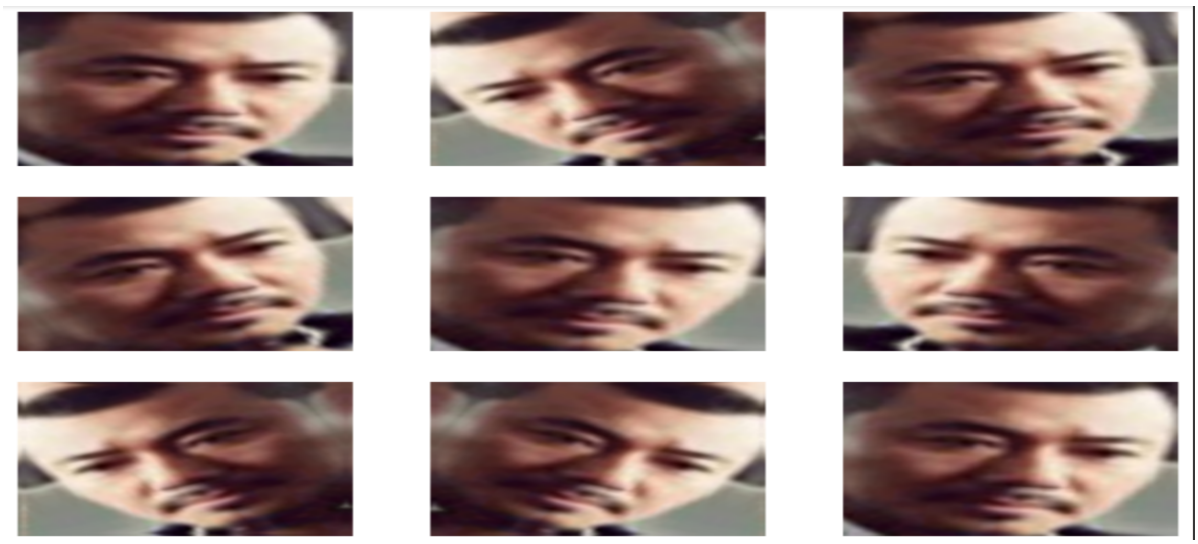

Figure 3: Data Pre-processing and Image Augmentation Results

4.2 Exploratory Data Analysis

Here, we used Python libraries to complete the data analysis.

```python
#Data visualization EDA use python library
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 10))
for images, labels in train_ds.take(1):
  for i in range(9):
    ax = plt.subplot(3, 3, i + 1)
    plt.imshow(images[i].numpy().astype("uint8"))
    plt.title(class_names[labels[i]])
    plt.axis("off")
```

Figure 4: Data visualization EDA using python library

4.3 Model Training

There are two different models that have been used to accurately classify photos with and without masks. MobileNet and Convolutional Neural Network (CNN) are the models utilized in this study. among which the pre-trained mobilenet model is. TensorFlow and the Keras library have been used to train the model, with a total of 10 epochs.

```python
[ ] #Adding different layers for the model training
    inputs = tf.keras.Input(shape=(160, 160, 3))
    x = data_augmentation(inputs)
    x = normalization_layer(x)
    x = base_model(x, training=False)
    x = global_average_layer(x)
    x = tf.keras.layers.Dropout(0.2)(x)
    outputs = prediction_layer(x)
    model = tf.keras.Model(inputs, outputs)
```

Figure 5: different layers for the model training

Constructing the model using the loss function and dam optimiser(optimizer, loss function, learning rate)

```
[ ] #compile the model using Adam optimiser and loss function(optimizer, loss function, learning rate)
    base_learning_rate = 0.0001
    model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=base_learning_rate),
                  loss=tf.keras.losses.BinaryCrossentropy(from_logits=True),
                  metrics=['accuracy'])

[ ] model.summary()

    Model: "model_1"
    _________________________________________________________________
     Layer (type)                 Output Shape              Param #
    =================================================================
     input_4 (InputLayer)         [(None, 160, 160, 3)]     0

     sequential_1 (Sequential)    (None, 160, 160, 3)       0
```

Figure 6: Compiling the model

4.4 Graph Validation

Accuracy and loss will be validated through the use of graph validation.

```
# Graph of validation acuracay and validation loss
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

loss = history.history['loss']
val_loss = history.history['val_loss']

epochs_range = range(epochs)

plt.figure(figsize=(8, 8))
plt.subplot(1, 2, 1)
plt.plot(epochs_range, acc, label='Training Accuracy')
plt.plot(epochs_range, val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')

plt.subplot(1, 2, 2)
plt.plot(epochs_range, loss, label='Training Loss')
plt.plot(epochs_range, val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
plt.show()
```

Figure 7: Graph validation

Results after validation

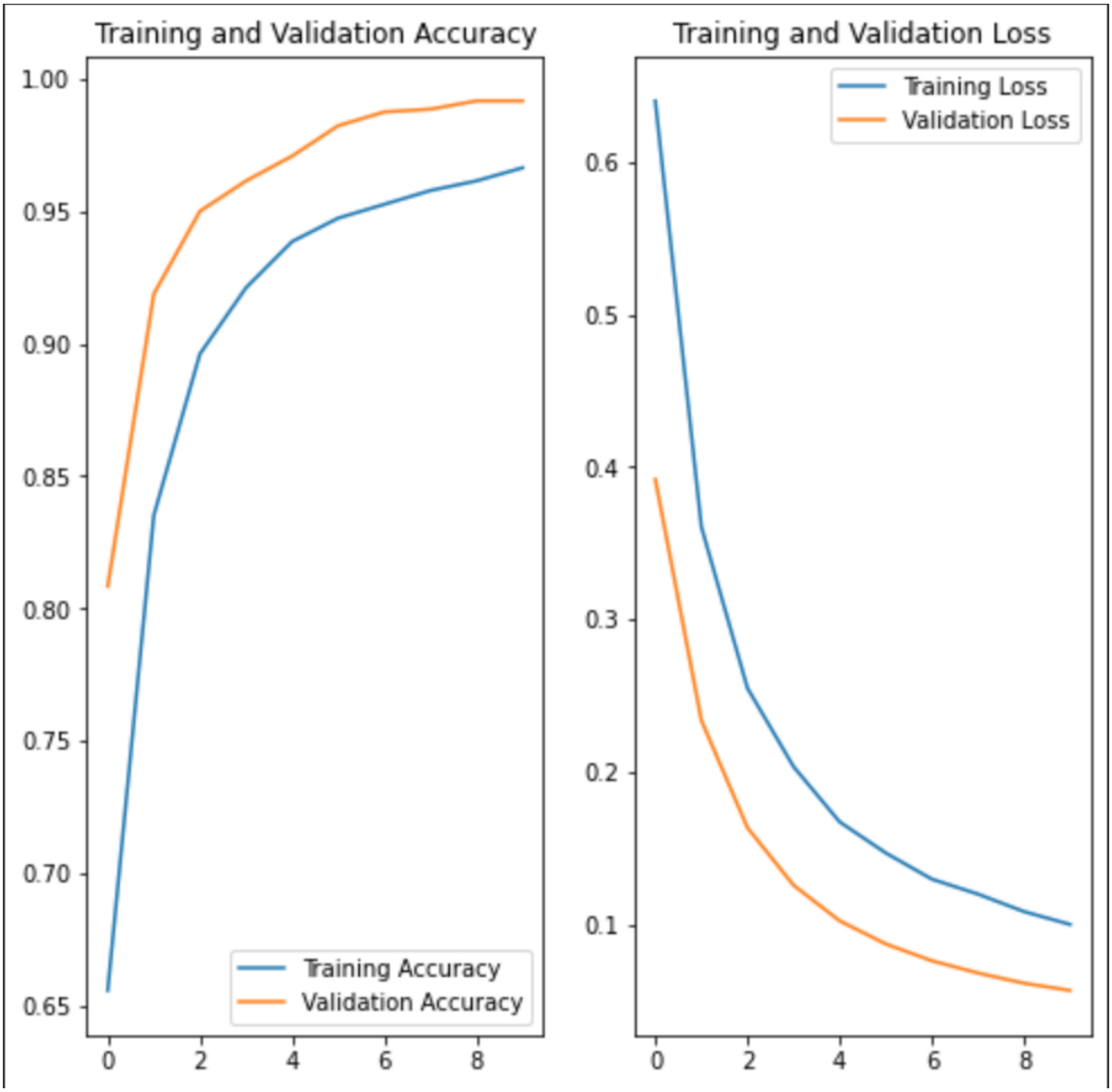

Figure 8: Graph validation Results

5 Implementation

The application will be created and deployed across both architectures after the best face mask classification model has been determined (Cloud-server and Hybrid Computing). These architectures are implemented with the aid of a number of tools and technologies.

Before moving forward with either architecture, various prerequisites need to be confirmed.

1. Install python and check the version.

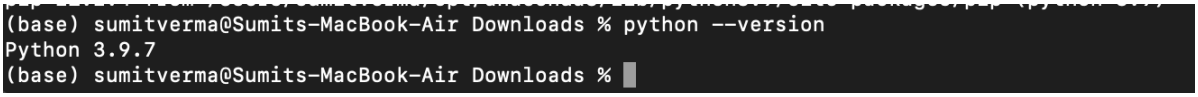

Figure 9: Python version

2. Install pip and check the version.

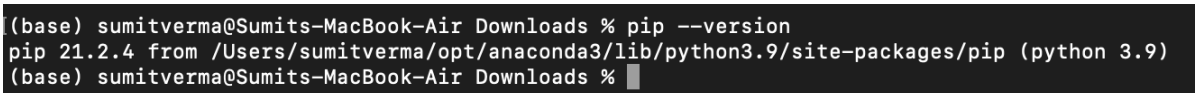

Figure 10: Pip version

3. For the real-time video-based application, WebRTC is also required.
4.Install ngrok since the system needs https to access the webcam.

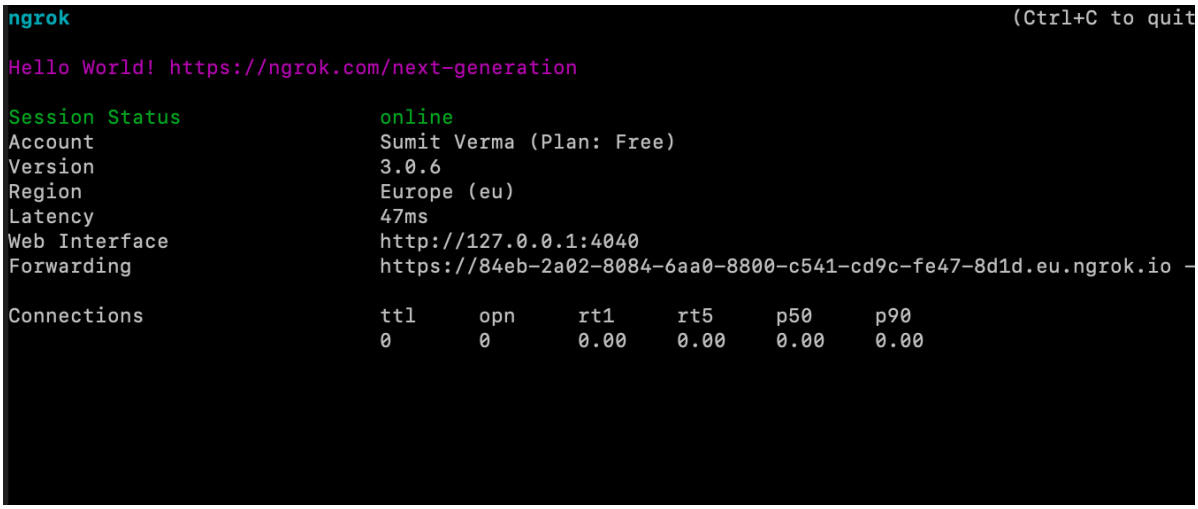

Figure 11: Ngrok

5.1 Implementation of Cloud-Based Architecture

For the cloud-based architecture, the edge device browser sends the video stream frame by frame to the cloud server. Web-RTC is used for this communication between the edge device browser and the cloud server.

The image is preprocessed on the server before being sent to the model for inference. Because this transfer and processing take place for each frame, the load on the system grows as more application users are added.

5.2 Implementation of Hybrid-Computing Based Architecture

The edge device downloads the AI assets (model and logic) on the client side and puts them in the local storage of the browserShahidinejad and Ghobaei-Arani (2020). No data is transferred to the cloud after the model is downloaded; instead, the system runs the model on the client side as needed (locally).

Due to the elimination of network latency in the transfer of video frames from client to server and vice versa in hybrid computing architecture. Because there is no calculation burden on the server connected with each user, the system is extremely scalable.

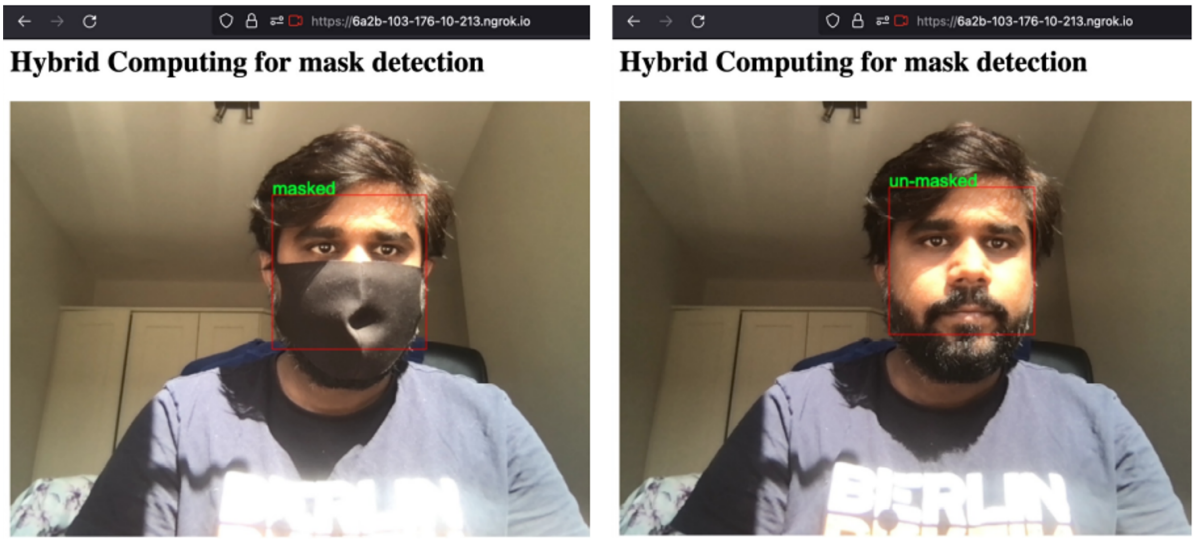

Figure 12: Identification of Face Mask using Laptop (Hybrid Computing)

6 Evaluation

Here, we conducted two studies in which CPU usage was calculated as the number of users rose. For any form of processing in this design, the cloud server is completely dependent. Consequently, a notable increase in CPU consumption has been seen as the number of users has increased.

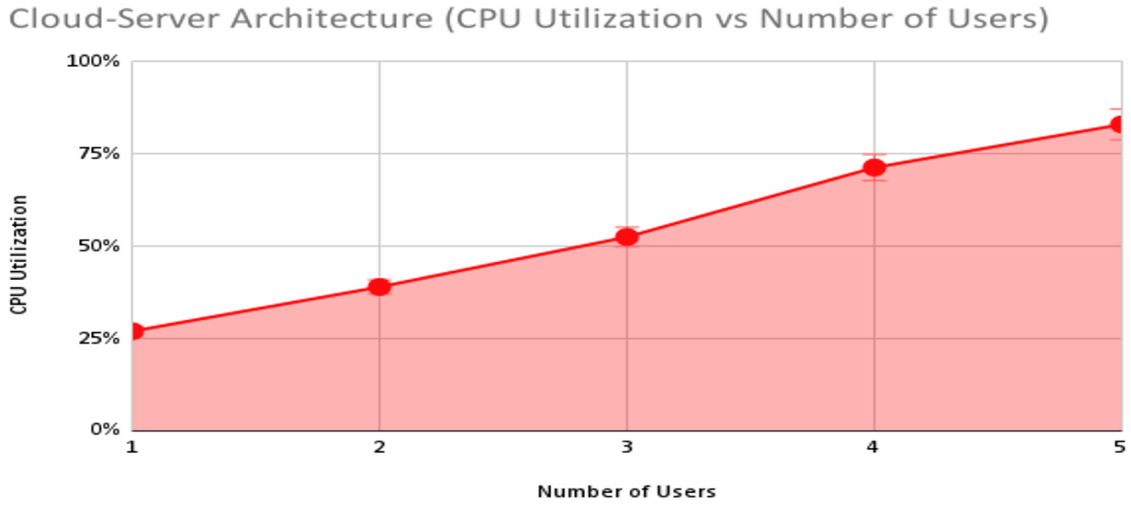

Figure 13: CPU Utilization

Utilization of network bandwidth. Because the program solely relies on the cloud server, a significant amount of bandwidth is used to receive and transmit back video

frames. The network consumption of the cloud server with regard to various user counts is shown on a line graph. On cloud servers, a noticeable rise in network usage has been seen along with an increase in users. Due to network connection latency problems, a delay in the video frames has been seen when the application is operating.

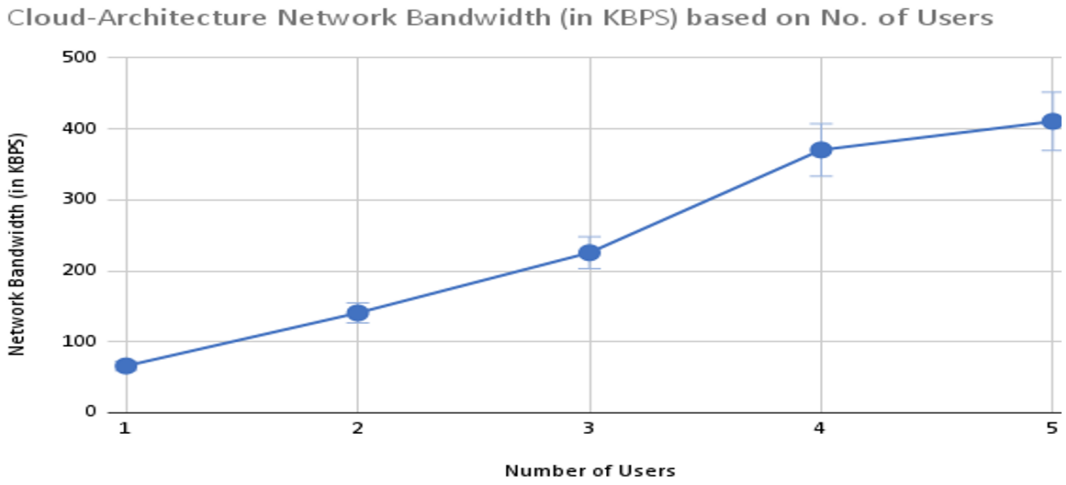

Figure 14: Network Bandwidth

References

Shahidinejad, A. and Ghobaei-Arani, M. (2020). Joint computation offloading and resource provisioning for e dge-cloud computing environment: A machine learning-based approach, *Software: Practice and Experience* **50**(12): 2212–2230.

Wang, J., Pan, J., Esposito, F., Calyam, P., Yang, Z. and Mohapatra, P. (2019). Edge cloud offloading algorithms: Issues, methods, and perspectives, *ACM Computing Surveys (CSUR)* **52**(1): 1–23.

Zhang, J., Zhou, Z., Li, S., Gan, L., Zhang, X., Qi, L., Xu, X. and Dou, W. (2018). Hybrid computation offloading for smart home automation in mobile cloud computing, *Personal and Ubiquitous Computing* **22**(1): 121–134.