

Cover sheet for submission of work for assessment



UNIT DETAILS

Unit name	Intelligent Systems	Class day/time		Office use only
Unit code	COS30018	Assignment no.		Due date
Name of lecturer/teacher	Dr. Sue Han Lee			
Tutor/marker's name				Faculty or school date stamp

STUDENT(S)

Family Name(s)	Given Name(s)	Student ID Number(s)
(1) Mohammad Mobin		101228706
(2) Masrur Rahman		101214608
(3) Noor ul Ain		102763334
(4)		
(5)		
(6)		

DECLARATION AND STATEMENT OF AUTHORSHIP

- I/we have not impersonated, or allowed myself/ourselves to be impersonated by any person for the purposes of this assessment.
- This assessment is my/our original work and no part of it has been copied from any other source except where due acknowledgement is made.
- No part of this assessment has been written for me/us by any other person except where such collaboration has been authorised by the lecturer/teacher concerned.
- I/we have not previously submitted this work for this or any other course/unit.
- I/we give permission for my/our assessment response to be reproduced, communicated, compared and archived for plagiarism detection, benchmarking or educational purposes.

I/we understand that:

- Plagiarism is the presentation of the work, idea or creation of another person as though it is your own. It is a form of cheating and is a very serious academic offence that may lead to exclusion from the University. Plagiarised material can be drawn from, and presented in, written, graphic and visual form, including electronic data and oral presentations. Plagiarism occurs when the origin of the material used is not appropriately cited.

Student signature/s

I/we declare that I/we have read and understood the declaration and statement of authorship.

(1) Mubin	(4)
(2) Masrur	(5)
(3) Noor	(6)

Further information relating to the penalties for plagiarism, which range from a formal caution to expulsion from the University is contained on the Current Students website at www.swin.edu.au/student/

Copies of this form can be downloaded from the Student Forms web page at www.swinburne.edu.au/studentforms/

Cover Sheet Information

<i>Project Title: Smart Meeting Minutes</i>		
<i>Team Name: Silverback Tech.</i>		
	<i>ID Number</i>	<i>Name</i>
1	101228706	Mohammad Mobin
2	101214608	Masrur Rahman Zahin
3	102763334	Noor Ul Ain Khurshid

Introduction

This project is based on Audio Transcription. We are to build 2 different deep learning models and compare them based on their evaluations. This project falls under the category of machine learning. We have a goal, dataset, model, and evaluation as any machine learning problem. In our case, our goal is to build a model which recognises words from the audio input. A basic supervised learning model requires the dataset and its labels or supervisory signal. The dataset must be split into training and testing sets and fed to our neural network. Just like a regular model requires lots of training data to learn a specific class, we have training data for each word. Each word is a class. Multiple speakers speak the same word, which has their waveform; although different speakers, the waveform appears similar. We feed the neural network different waveforms of the same word to be able to recognise that word from a different speaker. Our dataset is in audio form with a folder of different words containing multiple speakers speaking out the word. Our program converts the audio to its respective spectrogram, which becomes our data to be trained and tested, i.e. the Xtrain and Xtest variables. Then we attach them with their respective labels, which are the words, i.e. the Ytrain and Ytest variables.

Overall System Architecture

We are building two architectures: CNN and CNN, based on transfer learning

CNN

For CNN, the name speaks for itself; a few layers of convolution and pooling layers are kept for feature learning and reduction, reducing the image's complexity, learning only the essential features, and then predicting using a fully connected layer. We are using three layers of convolution layer and a pooling layer with a final dense layer.

CNN via Transfer Learning

For the second model, CNN architecture was used in conjunction with a pre-trained model using the ImageNet database. The model consisted of one Baselayer followed by MaxPooling and Dropout layers, and finally, four Dense layers to learn all the features from the image. Softmax was used on the final activation layer. Adam was used as the optimiser, and categorical_crossentropy was used for the loss function. Early stopping was also implemented to prevent overfitting with the patience of 10 epochs.

Data collections and annotations

Dataset collections were the most challenging part of our task since our coding and everything related to our work depends on the quality and quantity of the dataset, its cleanliness and usefulness for our implementation methods.

We have looked over three different datasets:

Our first dataset was a hefty file of around 75 gigabytes. It was good, except the dataset was in sentences; we needed single words. Initially, the team thought of splitting it using a code, but the folder was too large for our computers to handle and work with and thus causing our computers to crash several times while working with it. However, by that time, we had found our second dataset. Due to all this, we have not looked into the data labelling method for our dataset.

Our second dataset looked very promising. It was clean, well organised, and instrumental in our implementation strategy. The annotations were done by the CSV files provided, where one column referred to the link of the audio file, and the other referred to the word(the label). The dataset had no sentences, and we proceeded with it till last week. It also had separate CSV files for train and test, meaning the splitting has already been done for us. However, due to computer lagging issues of two team members, we thought to write codes out of theory and debug it all later since this way we could focus on debugging together. During the final week, everything was debugged except for the audio to image conversion; we faced several issues:

First, Mubin took a portion of the dataset, three classes to be specific, to test out the code for audio to image conversion; the audio was in a .opus format, and it worked well printing out the spectrogram in google colab. Later, when Masrur ran the codes, it gave an unknown format error, so we wrote code to convert it all to .wav format; it still did not work and gave errors. Masrur tried a random wav file from the internet, and it works, we assumed there was something wrong with the conversion method, and we could not find any other method of conversion from the opus file. We decided to discard the dataset since the only option left is to use colab, and we are in no time to upload heavy files into the colab.

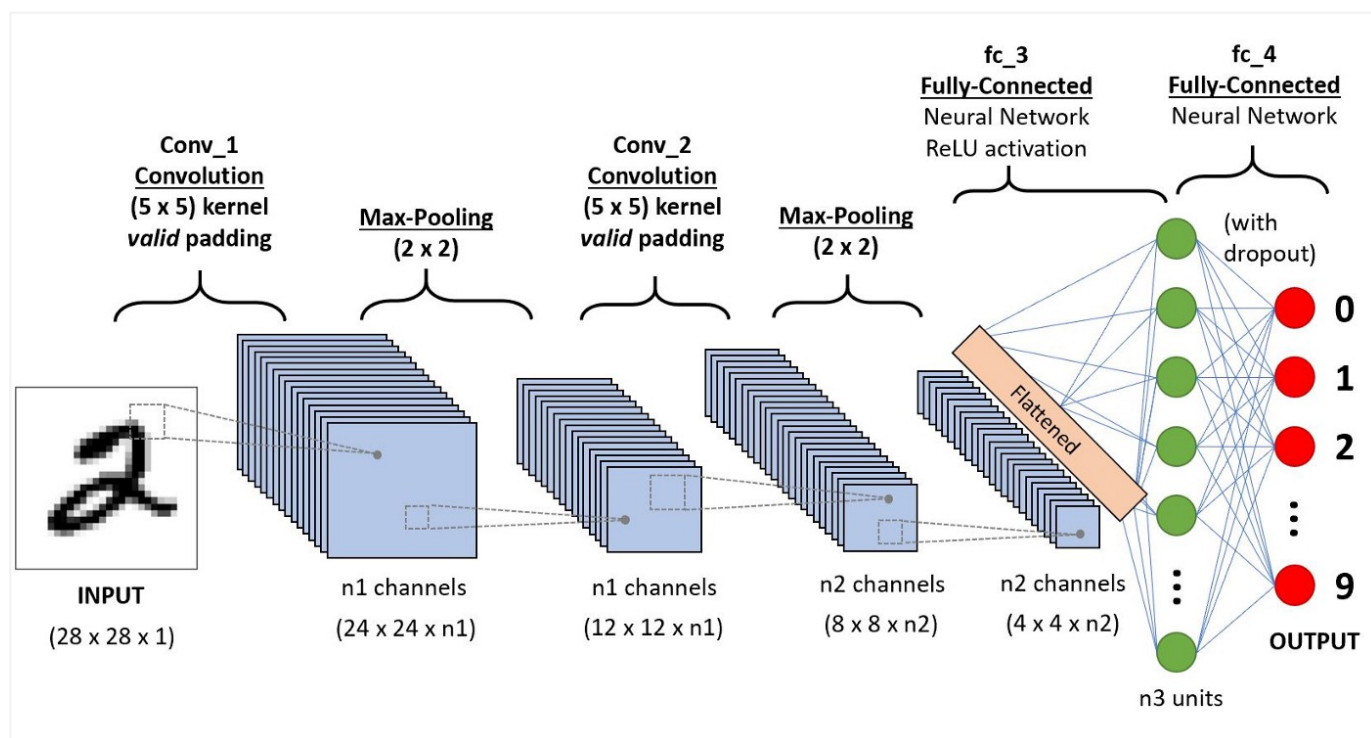
The third dataset was found on the same day we encountered problems with our second dataset. This dataset was also clean and organised well. Instead of CSV files, there were text files, with each line being the audio link. We wrote code to split the directory to store the label names. We quickly fine-tuned our code to match our new dataset and began audio conversion to one class. To our surprise, the conversion worked, and we saved up multiple spectrograms and checked to see if they were different, and the conversion worked fine. Later we converted all the train and test audios to images and were finally ready for training the model since our training parameters were ready.

Implemented Machine Learning Techniques For Automated Speech Recognition

The machine learning technique used for both models is **Supervised Learning**; it is a subcategory of machine learning which can be defined by its use of labelled input and output data; these data are later used to train algorithms that can accurately classify or predict the outcomes. As input data is fed into the model, the weights are adjusted until the model is appropriately fitted during the cross-validation phase. Supervised Learning can be categorised into two types of problems; **Classification** and **Regression**.

Classification uses algorithms that categorise test results into specific groups. It recognises certain entities in the dataset and tries to come to some conclusions about how they should be labelled or defined. Regression is mainly used to explore the relationship between dependent and independent variables in a probabilistic manner [1].

There are many supervised learning algorithms, but the algorithm used for our project was a **Neural Network algorithm**, to be more specific **Convolutional Neural Networks (CNN)**. CNN is used for image recognition, pattern identification, or computer vision. These networks use linear algebra principles, notably matrix multiplication, to find patterns in images. A CNN employs a technique similar to a multilayer perceptron but with fewer processing requirements. An input layer, an output layer, and a hidden layer with several convolutional layers, pooling layers, fully connected layers, and normalising layers make up a CNN's layers [2].



Scenarios/Examples To Demonstrate How The System Works

CNN

First, we take each of the text files and extract the audio files and their labels and store them in a variable as lists

```
train_files = []
y_train = []

path = "data/"
f = open('training_list.txt', 'r')

for line in f:
    train_path = line.split("\n")
    train_files.append(path + train_path[0])
    label = line.split("/")
    y_train.append(label[0])

f.close()
```

```
test_files = []
y_test = []

f = open("testing_list.txt", 'r')

for line in f:
    test_path = line.split("\n")
    test_files.append(path + test_path[0])
    label = line.split("/")
    y_test.append(label[0])

f.close()
```

Later, we use a library called Librosa to run into every audio in the list, convert it to spectrogram, and download the images in a new folder called image along with their respective label folder.


```

#save up the spectrogram images in the test folder for training the model

from matplotlib.backends.backend_agg import FigureCanvasAgg as FigureCanvas
imageNo = 1
for fileName in test_files:
    #write something like count number of files in a label folder and run a loop placing all those images in that folder
    y,sr=librosa.load(fileName) #load the file
    spectrogram = librosa.feature.melspectrogram(y=y, sr=sr,n_mels=128)
    spectrogram = librosa.power_to_db(spectrogram)
    spectrogram = spectrogram.astype(np.float32)

    out = librosa.core.spectrum.stft(y, n_fft = window_size, hop_length = hop_length, window=window)
    out = 2 * np.abs(out) / np.sum(window)

    fig = plt.figure()
    canvas = FigureCanvas(fig)
    ax = fig.add_subplot(111)
    p = librosa.display.specshow(librosa.amplitude_to_db(out, ref=np.max), ax=ax, y_axis='log', x_axis='time')

    label = fileName.split("/")
    path = 'images/test/{0}/'.format(label[1])

    #downloading few spectrogram images
    fig.savefig(path + str(imageNo) + '.png')
    imageNo += 1

```

Because of spectrogram images, we use an image data generator to rescale and process the primary training and testing variables to start training.

```

# Image rescaling for preprocessing
from keras.preprocessing.image import ImageDataGenerator as IDG
trainDatagen = IDG(rescale = 1/255.0)
testDatagen = IDG(rescale = 1/255.0)

trainGen = trainDatagen.flow_from_directory(
    train_dir,
    target_size=(height, width),
    batch_size=batchSize,
    class_mode='categorical')

testGen = testDatagen.flow_from_directory(
    test_dir,
    target_size=(height, width),
    batch_size=batchSize,
    class_mode='categorical')

```

Then we build up the architecture of the CNN, compile it up and train it with our data.

```
from keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, BatchNormalization, Dropout
from keras.models import Sequential, load_model

model = Sequential([
    #first set of convolution and pooling layer
    Conv2D(32,(3,3), activation = 'relu', input_shape = (height , width, 3)),
    BatchNormalization(),
    MaxPooling2D(2,2),
    Dropout(0.2),

    #second set of convolution and pooling layer
    Conv2D(64,(3,3), activation = 'relu', padding='same'),
    BatchNormalization(),
    MaxPooling2D(2,2),
    Dropout(0.2),

    #third set of convolution and pooling layer
    Conv2D(128,(3,3), activation = 'relu', padding='same'),
    BatchNormalization(),
    MaxPooling2D(2,2),
    Dropout(0.2),

    #FC layer
    Flatten(),
    Dense(128, activation = 'relu'),
    Dropout(0.2),
    #final layer, input is number of classes
    Dense(36, activation = 'softmax')
])

#model compilation
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics = ['accuracy'])
model.summary()
```

Below is the architecture of the transfer learning technique

```
model = Sequential()
model.add(base_model)
model.add(BatchNormalization(renorm=True))
model.add(MaxPooling2D(2,2))
model.add(Dense(512, activation='relu'))
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.5))

model.add(Flatten())
model.add(Dense(128, activation = 'relu'))
model.add(Dense(36, activation = 'softmax'))

model.compile(optimizer='adam', loss='categorical_crossentropy', metrics = ['accuracy'])
model.summary()
```

Critical Analysis Of The Implementation

As we know, CNN is deep learning neural network model preferably used for image classification and is very good at extracting features from images and learning them.

In general, image data has many pixels; the traditional fully connected layers would be too much for the model to handle; here is where CNN comes in basically, it takes in the image, uses a convolution layer to extract some features using a kernel or filter and then reduces them using pooling layer, it then keeps repeating the previous two steps depending on how complex the features are, then provides the important features to the fully connected layers for processing.

In other words, CNN takes an image, shortens it to just essential features and proceeds with fully connected layers.

Our data was audio; we had to convert it into an image; its spectrogram; every audio has a waveform image. The idea was to use the waveform of multiple speakers of the same word, which would be similar, and the model would learn the similar features so that during prediction, the model would predict that specific word.

We provided three layers of convolution and pooling layers with the activation function of Relu. And the final layer with the number of nodes same as that of the number of classes with the activation function of softmax since we were working on multi-class classification.

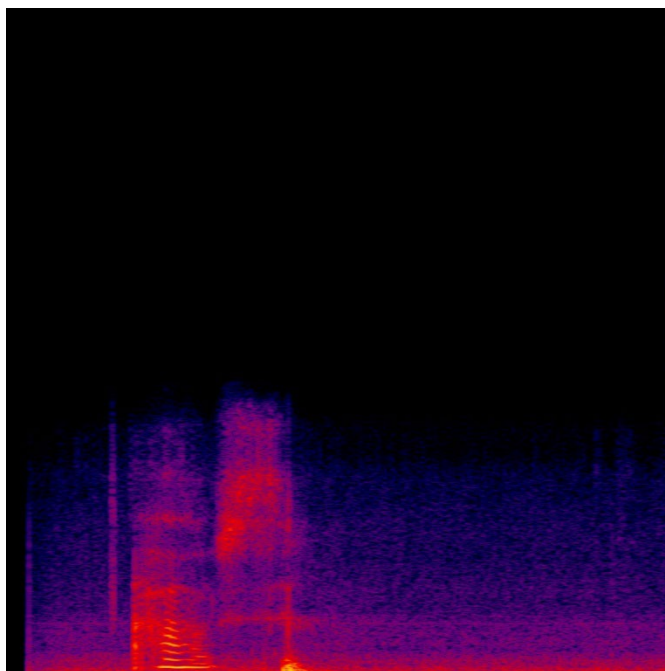
For the transfer learning model, we used the duplicate dataset audio files that have been converted into images previously.

Later we wrote code for evaluation using a confusion matrix and plotting graphs of two models and compared them

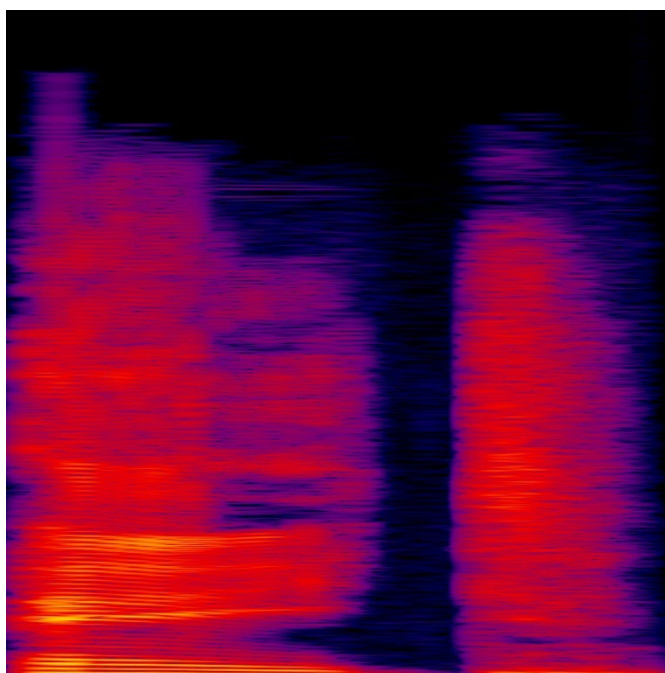
Practical Application Of The Description

Following are a few spectrogram images:

House



Eight

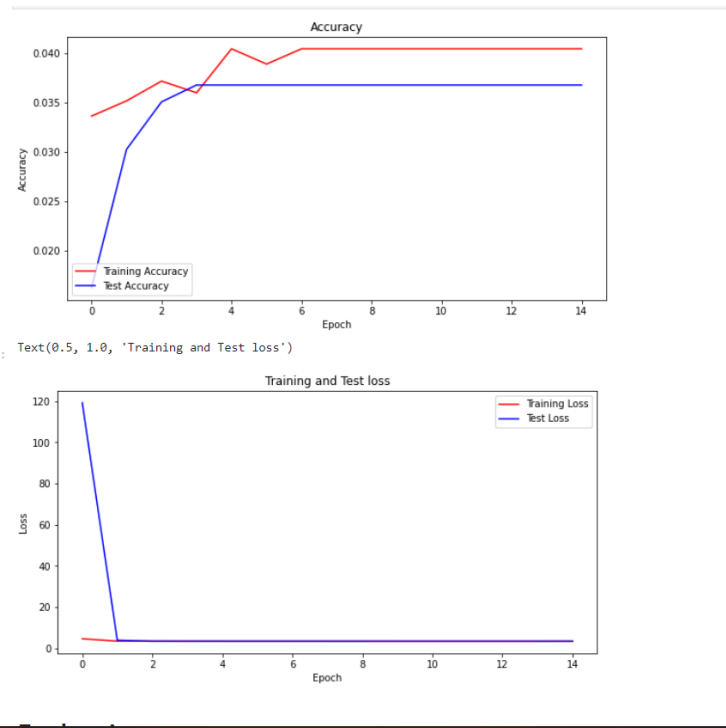


Model summary of the CNN

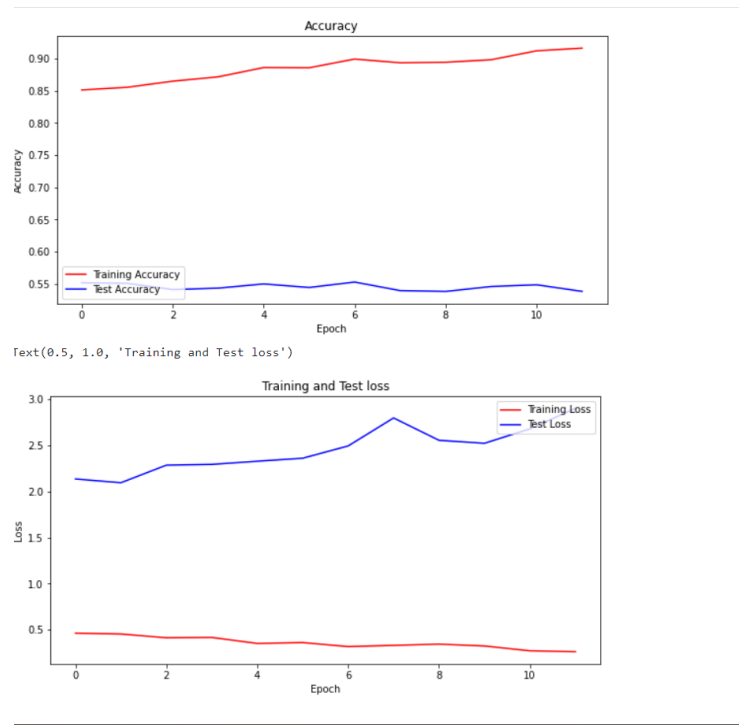
Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 254, 254, 32)	896
batch_normalization (Batch Normalization)	(None, 254, 254, 32)	128
max_pooling2d (MaxPooling2D)	(None, 127, 127, 32)	0
dropout (Dropout)	(None, 127, 127, 32)	0
conv2d_1 (Conv2D)	(None, 127, 127, 64)	18496
batch_normalization_1 (Batch Normalization)	(None, 127, 127, 64)	256
max_pooling2d_1 (MaxPooling2D)	(None, 63, 63, 64)	0
dropout_1 (Dropout)	(None, 63, 63, 64)	0
conv2d_2 (Conv2D)	(None, 63, 63, 128)	73856
batch_normalization_2 (Batch Normalization)	(None, 63, 63, 128)	512
max_pooling2d_2 (MaxPooling2D)	(None, 31, 31, 128)	0
dropout_2 (Dropout)	(None, 31, 31, 128)	0
flatten (Flatten)	(None, 123008)	0
dense (Dense)	(None, 128)	15745152
dropout_3 (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 36)	4644
Total params: 15,843,940		
Trainable params: 15,843,492		
Non-trainable params: 448		

CNN evaluation



Transfer learning evaluation



Model summary of the transfer learning

Model: "sequential"

Layer (type)	Output Shape	Param #
inception_resnet_v2 (Functional)	(None, 9, 9, 1536)	54336736
batch_normalization_203 (Batch Normalization)	(None, 9, 9, 1536)	10752
max_pooling2d_4 (MaxPooling2D)	(None, 4, 4, 1536)	0
dense (Dense)	(None, 4, 4, 512)	786944
dense_1 (Dense)	(None, 4, 4, 256)	131328
dropout (Dropout)	(None, 4, 4, 256)	0
flatten (Flatten)	(None, 4096)	0
dense_2 (Dense)	(None, 128)	524416
dense_3 (Dense)	(None, 36)	4644
Total params: 55,794,820		
Trainable params: 1,450,404		
Non-trainable params: 54,344,416		

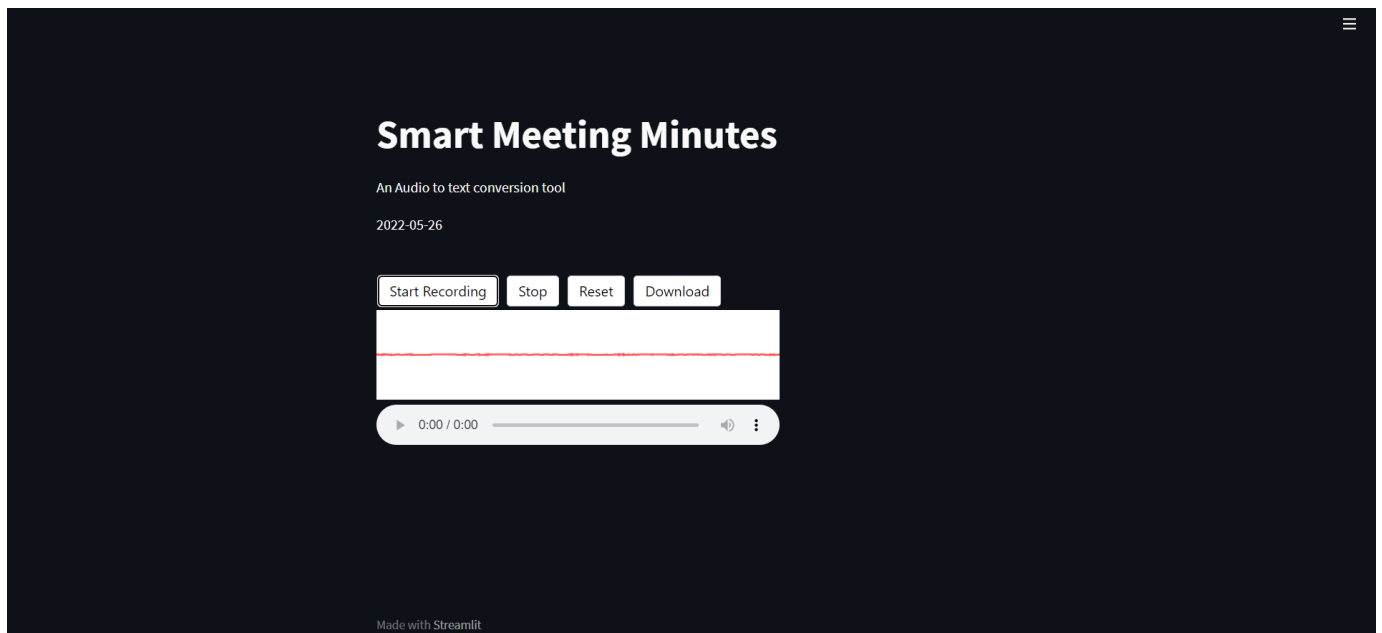
Training of the CNN

```
Epoch 1/15
344/344 [=====] - 89s 244ms/step - loss: 4.6806 - accuracy: 0.0336 - val_loss: 119.2171 - val_accuracy: 0.0162
Epoch 2/15
344/344 [=====] - 75s 218ms/step - loss: 3.5367 - accuracy: 0.0352 - val_loss: 3.9651 - val_accuracy: 0.0303
Epoch 3/15
344/344 [=====] - 80s 232ms/step - loss: 3.5195 - accuracy: 0.0372 - val_loss: 3.5163 - val_accuracy: 0.0351
Epoch 4/15
344/344 [=====] - 72s 208ms/step - loss: 3.5095 - accuracy: 0.0360 - val_loss: 3.5093 - val_accuracy: 0.0368
Epoch 5/15
344/344 [=====] - 70s 205ms/step - loss: 3.5035 - accuracy: 0.0404 - val_loss: 3.5050 - val_accuracy: 0.0368
Epoch 6/15
344/344 [=====] - 74s 214ms/step - loss: 3.4998 - accuracy: 0.0389 - val_loss: 3.5025 - val_accuracy: 0.0368
Epoch 7/15
344/344 [=====] - 71s 206ms/step - loss: 3.4974 - accuracy: 0.0404 - val_loss: 3.5007 - val_accuracy: 0.0368
Epoch 8/15
344/344 [=====] - 70s 204ms/step - loss: 3.4958 - accuracy: 0.0404 - val_loss: 3.4995 - val_accuracy: 0.0368
Epoch 9/15
344/344 [=====] - 71s 206ms/step - loss: 3.4946 - accuracy: 0.0404 - val_loss: 3.4988 - val_accuracy: 0.0368
Epoch 10/15
344/344 [=====] - 71s 206ms/step - loss: 3.4938 - accuracy: 0.0404 - val_loss: 3.4981 - val_accuracy: 0.0368
Epoch 11/15
344/344 [=====] - 71s 206ms/step - loss: 3.4931 - accuracy: 0.0404 - val_loss: 3.4976 - val_accuracy: 0.0368
Epoch 12/15
344/344 [=====] - 70s 205ms/step - loss: 3.4927 - accuracy: 0.0404 - val_loss: 3.4972 - val_accuracy: 0.0368
Epoch 13/15
344/344 [=====] - 71s 205ms/step - loss: 3.4923 - accuracy: 0.0404 - val_loss: 3.4968 - val_accuracy: 0.0368
Epoch 14/15
344/344 [=====] - 71s 205ms/step - loss: 3.4920 - accuracy: 0.0404 - val_loss: 3.4966 - val_accuracy: 0.0368
Epoch 15/15
344/344 [=====] - 71s 205ms/step - loss: 3.5134 - accuracy: 0.0404 - val_loss: 3.4963 - val_accuracy: 0.0368
```

Training of the Transfer learning

```
Epoch 1/15
343/343 [=====] - 215s 626ms/step - loss: 0.4644 - accuracy: 0.8512 - val_loss: 2.1360 - val_accuracy: 0.5519
Epoch 2/15
343/343 [=====] - 218s 635ms/step - loss: 0.4552 - accuracy: 0.8553 - val_loss: 2.0953 - val_accuracy: 0.5510
Epoch 3/15
343/343 [=====] - 218s 637ms/step - loss: 0.4144 - accuracy: 0.8648 - val_loss: 2.2857 - val_accuracy: 0.5414
Epoch 4/15
343/343 [=====] - 218s 637ms/step - loss: 0.4179 - accuracy: 0.8715 - val_loss: 2.2940 - val_accuracy: 0.5436
Epoch 5/15
343/343 [=====] - 219s 638ms/step - loss: 0.3528 - accuracy: 0.8859 - val_loss: 2.3283 - val_accuracy: 0.5501
Epoch 6/15
343/343 [=====] - 217s 634ms/step - loss: 0.3623 - accuracy: 0.8855 - val_loss: 2.3610 - val_accuracy: 0.5446
Epoch 7/15
343/343 [=====] - 218s 637ms/step - loss: 0.3188 - accuracy: 0.8991 - val_loss: 2.4941 - val_accuracy: 0.5531
Epoch 8/15
343/343 [=====] - 218s 636ms/step - loss: 0.3320 - accuracy: 0.8933 - val_loss: 2.7976 - val_accuracy: 0.5398
Epoch 9/15
343/343 [=====] - 218s 637ms/step - loss: 0.3456 - accuracy: 0.8941 - val_loss: 2.5550 - val_accuracy: 0.5385
Epoch 10/15
343/343 [=====] - 218s 636ms/step - loss: 0.3253 - accuracy: 0.8980 - val_loss: 2.5229 - val_accuracy: 0.5461
Epoch 11/15
343/343 [=====] - 218s 635ms/step - loss: 0.2729 - accuracy: 0.9119 - val_loss: 2.6797 - val_accuracy: 0.5488
Epoch 12/15
343/343 [=====] - 219s 639ms/step - loss: 0.2635 - accuracy: 0.9161 - val_loss: 2.9016 - val_accuracy: 0.5386
```

GUI screenshots



Summary

The model making was relatively easy and was no big deal; our only challenges were audio conversion, the datasets, and the GUI; other than that, everything was all right.

Speaking about the model, in conclusion, we would like to say that the model could have been implemented better if it were not for the opus file format of the second dataset; due to less time, we switched to a more specific dataset in order to show a demonstration of the model and its working and our knowledge and work to be assessed in this situation. Although fewer data, the working and the coding would have been the same if we had more enormous data.

Project Presentation Link:

https://youtu.be/jMOGc_WMSTE

End of Semester Final Deliverables CheckList

The following documents must be included in the final submission by the due date.

Deliverables Check List	Submitted Yes/No/NA
Main Document	
<ul style="list-style-type: none">Cover Sheet	Yes
<ul style="list-style-type: none">Check List	Yes
<ul style="list-style-type: none">Project Report (Video link attached)	Yes
Appendices	
<ul style="list-style-type: none">Source code and executable	Yes
<ul style="list-style-type: none">Who did what?	Yes
<ul style="list-style-type: none">Presentation slides	Yes

Who did what?

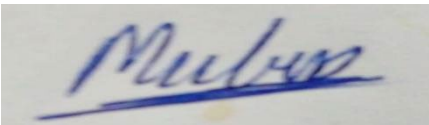
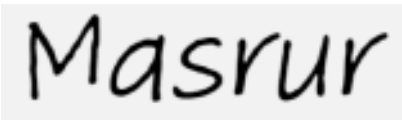
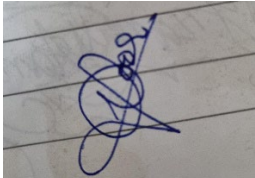
As you are to be individually assessed, it is necessary to ensure your marker understands your contribution. This document demonstrates who was responsible for each piece or contribution to each piece of work in your project. The following is a template to present and **must be signed by all team members**.

Project Title	
Mohammad Mobin	I collected the first dataset. I have done the coding for the second dataset and took the liberty to convert audio to image, found a good code source, and implemented it within our dataset. At first, I wrote down all codes from my theoretical knowledge and forwarded it to Masrur, and we both collectively debugged the code after meeting each other physically, and I finalised the conversion of audio to image. I started the layout of the report and vaguely wrote essential things.
Masrur Rahman Zahin	Built and trained the second model based on transfer learning and helped debug the first CNN model. Helped clean up and process the dataset for training and prediction; the training was done on my computer. Finalised and proofread the report.
Noor Ul Ain Khurshid	I have done dataset collection for the second dataset, researched the GUI implementation, presented my GUI idea, its interface, and functioning system to my team finished it, and assisted Masrur in the coding and discussion in the transfer learning model. I assisted their different sites and helped input during our discussions. I helped in additional theory knowledge for assisting Masrur in report writing. It was my idea to suggest a third dataset. Also, I did mini progress checks initially since I wanted to learn better from my teammates.

Group reflection

Project Title	
Mohammad Mobin	I learned good team management and the importance of setting our deadlines to meet requirements. At times, I wanted to procrastinate, but since I made up our deadlines, I could not come up to disrespect the management and followed accordingly to get the work done. I also realised that my pc was way too slow to deal with ai codes, so I wrote down codes and debugged together on Masrur's laptop since that way we both could work our heads on solving problems together and faster
Masrur Rahman Zahin	Working with an audio dataset was a new challenge that I had not faced before. It was pretty tricky to work with this type of data as my computer struggled a lot to keep up with the sheer amount of processing required to load it all. We had some issues converting the audio files into an image, but once that was done, it was pretty straightforward working with an image dataset as I have built multiple multi-class images classifying ai models before. Even though we were given enough time to complete the project, collecting, learning and processing the dataset took most of our time as we could not make any progress without a proper dataset. I feel we could have done better if we had a few more days to work on this project, as I could have optimised our model a lot better. The main takeaway from this unit this semester was all the concepts I have learned because of the oral defence, as it immensely helped me get a better understanding of what AI indeed is and how it all works.
Noor Khurshid	This was my first semester working with AI, and it was indeed a learning experience for me to work with two teammates who had sufficient knowledge about the concepts, and I got to learn a lot from them. I do feel like we could have performed much better if we had found a better dataset sooner and if this assignment wasn't done remotely. We could have coordinated better. Mubin and Masrur lived together to get this assignment done, so I feel like my contribution wasn't as much as theirs. I did enjoy learning AI and getting to know AI on a deeper level. I learnt python this semester and its implementation for the model's GUI, which was the most exciting part of this assignment for me.

I declare this is an accurate description of team contributions of the team members

Team Member Name	Signature	Date
Mohammad Mobin		24.05.2022
Masrur Rahman Zahin		24.05.2022
Noor Ul Ain Khurshid		24.05.2022

References

[1] IBM Cloud Education, "What is Supervised Learning? | IBM," Ibm, Aug. 19, 2020.
<https://www.ibm.com/cloud/learn/supervised-learning>.

[2] Tech Target, "What is convolutional neural network? - Definition from WhatIs.com,"
Techtarget, Apr. 25, 2018.
<https://www.techtarget.com/searchenterpriseai/definition/convolutional-neural-network>.

Theory: Cbse artificial intelligence for class 10 book

Converting audio to image:
<https://towardsdatascience.com/seeing-is-believing-converting-audio-data-into-images-5ed1a2ca6647>

Understanding melspectrogram:
<https://librosa.org/doc/main/generated/librosa.feature.melspectrogram.html>

Display the spectrogram:
<https://stackoverflow.com/questions/52432731/store-the-spectrogram-as-image-in-python>

Saving the spectrogram:
<https://localcoder.org/store-the-spectrogram-as-image-in-python>