UNIVERSITY OF CAMBRIDGE

Mobile Health Practical 2

Introduction to Deep Learning

Drs. Jing Han & Georgios Rizos

Goal of the Practical

- Introducing the audio dataset
- Spectrograms
- Machine learning setup (loading data, partitioning)
- Supervised learning and evaluation (train-validation-test loop)
- A bit on semi-supervised learning

Colab Notebook for the Practical

You can open the notebook by:

- Visiting the relevant link on Moodle
- Scanning the QR code on the right
- Visiting https://shorturl.at/kOP27

Colab Notebook for the Practical

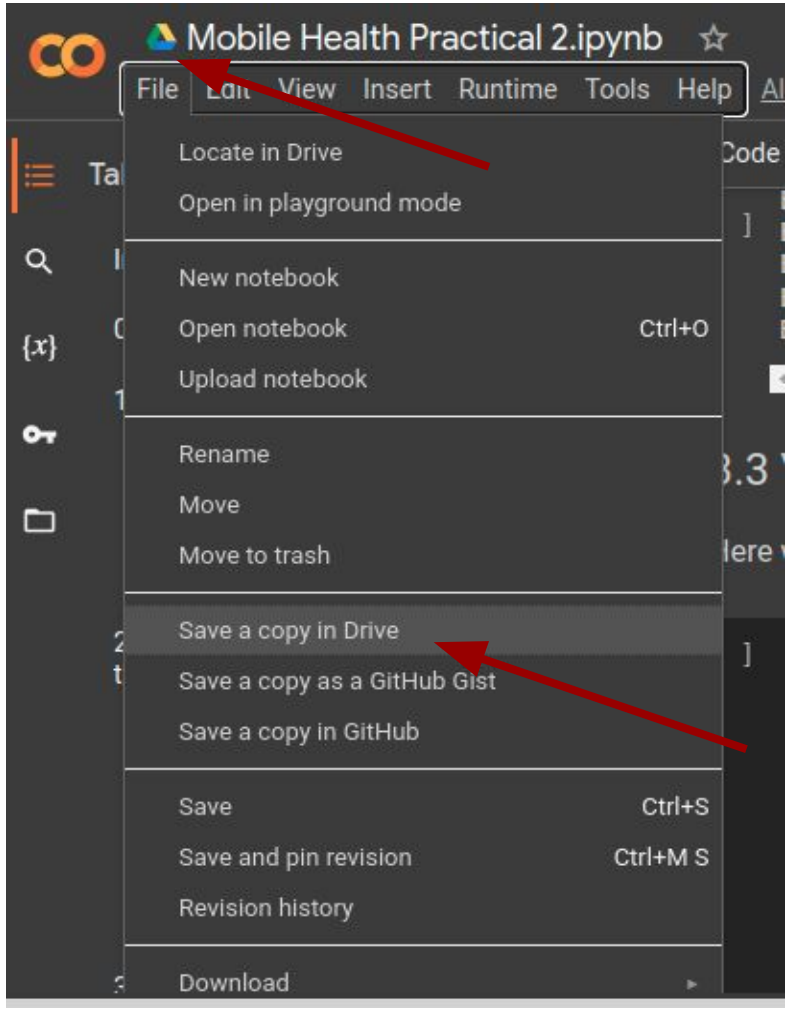
Then, please save a copy of this notebook into your Drive:

- File > Save a Copy in Drive

For assignment 2, you will also need to:

- Edit > Notebook settings, select GPU

Note that there are usage limits.



Introducing the audio dataset

Respiratory Sound Database from ICBHI 2017 challenge

- Raw data explored, labelled, and segmented already from challenge
- 600 segmented breathing cycles (wav format, 44.1 kHz sampling rate)
- 77 patients (a subset of original)
- Annotations in file name: abnormal (1) vs normal (0) respiratory sound
  - Abnormal denotes presence of crackles and/or wheezes

e.g., 226_1b1_Pl_sc_LittC2SE_segment_1_1_0.wav. “_1” indicates presence of crackle and “_0” indicates absence of wheeze in this recording.

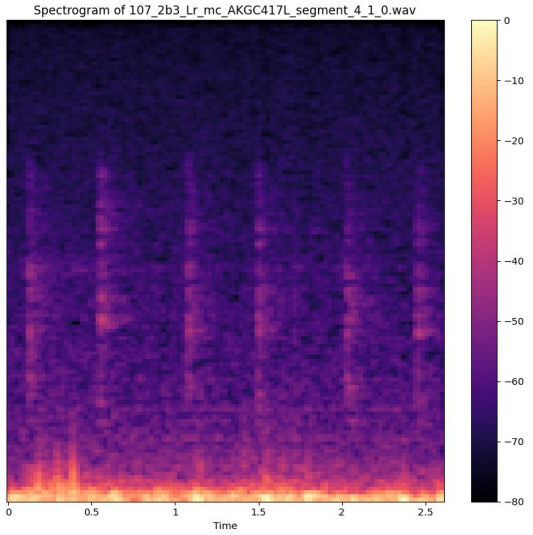
Introducing the audio dataset

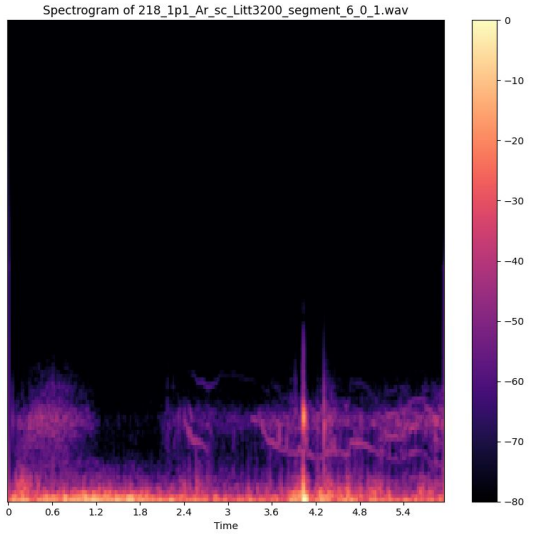
Task can be performed

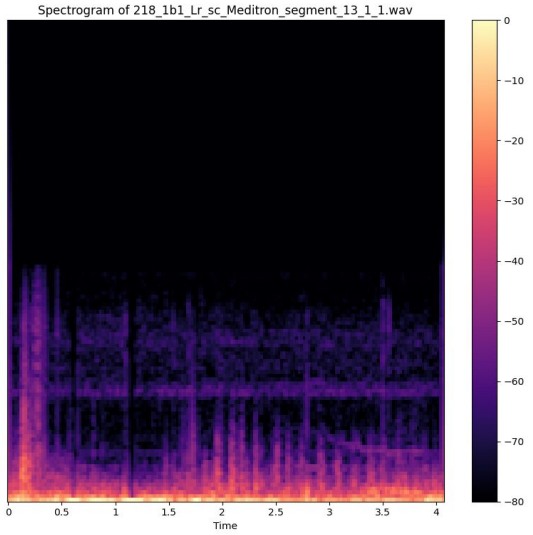
- Binary classification:

  0 for normal breath (“_0_0”)

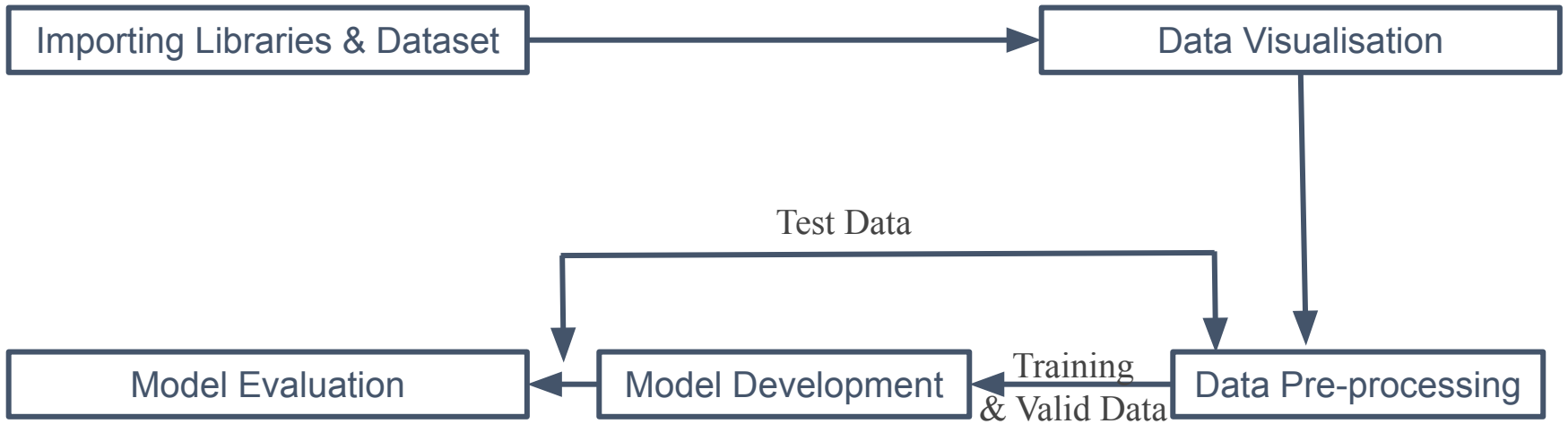
  1 for abnormal breath (including “_1_0”, “_0_1”, “_1_1”)



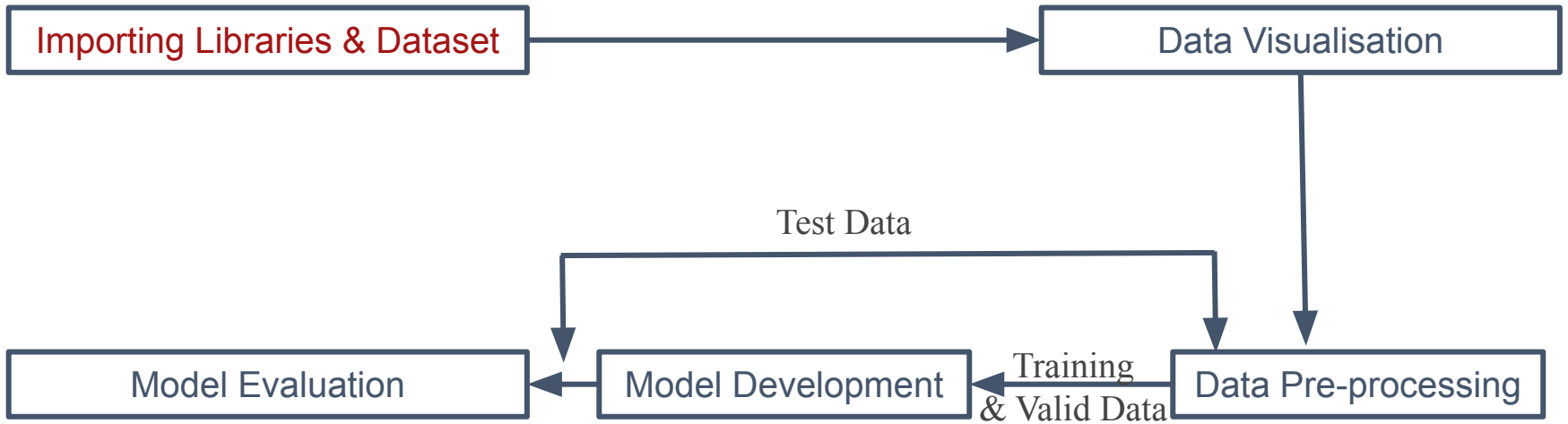




Machine Learning Pipeline



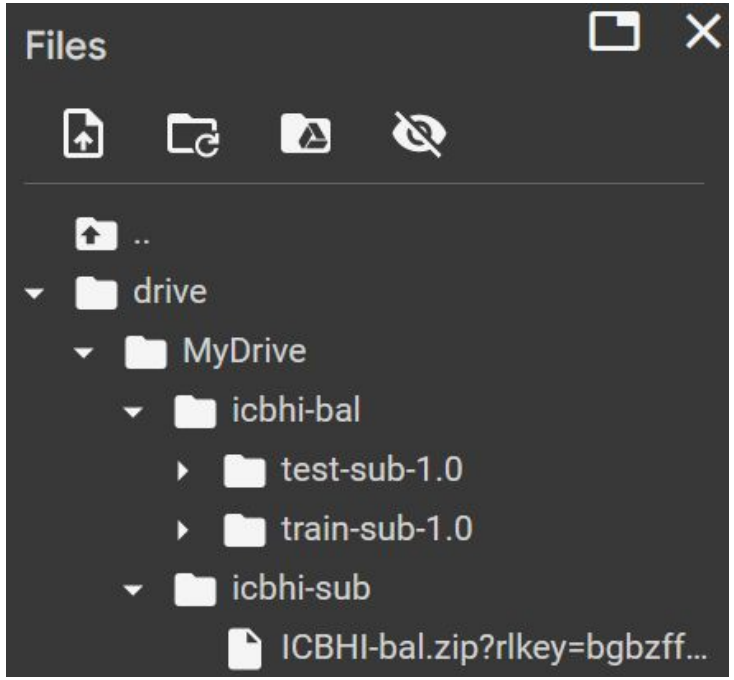
Machine Learning Pipeline



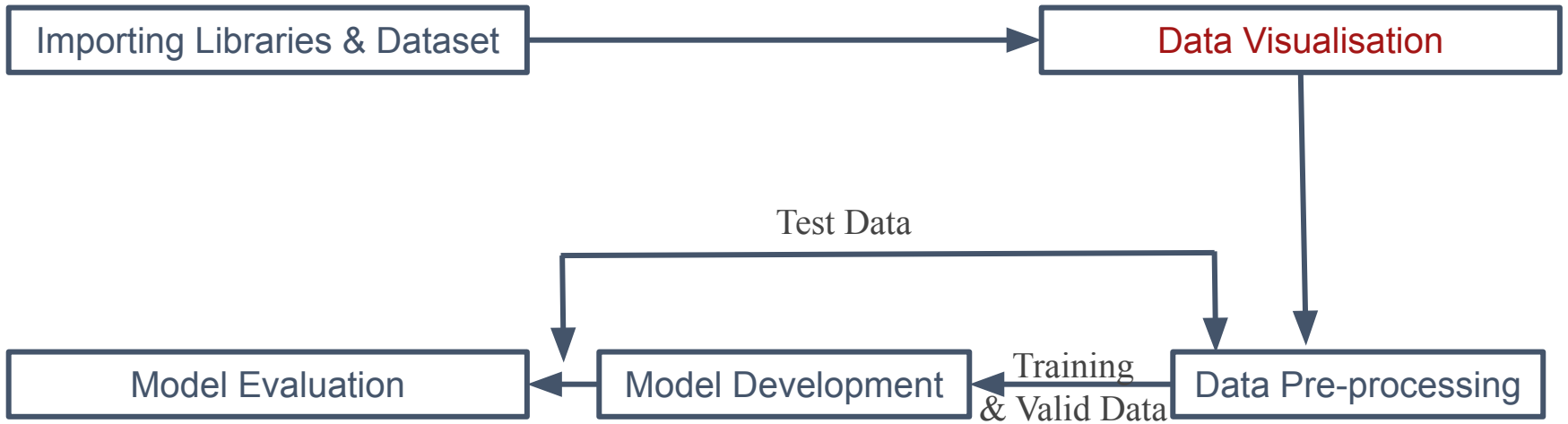
Machine Learning Pipeline

Loading dataset on colab programmatically

```
## Mount Google Drive to colab. You can then view your Google Drive contents via the Files tab on the left.
from google.colab import files
from google.colab import drive
drive.mount('/content/drive')

## Uncomment below and programmatically download a ZIP of the dataset. (Only need to run this block once; it can then be commented out)
wget -P /content/drive/MyDrive/icbhi-sub
https://www.dropbox.com/scl/fi/qjqv9ytlxodubyim37sbd/ICBHI-bal.zip?rlkey=bgbzffy60ynw6vbq4xn1vwq18&dl=0

## Uncomment below and unzip the dataset. (Only need to run this block once; it can then be commented out)
unzip /content/drive/MyDrive/icbhi-sub/ICBHI-bal.zip* -d /content/drive/MyDrive/icbhi-bal/

sound_dir_loc = np.array(gb.glob("/content/drive/MyDrive/icbhi-bal/*/*.wav"))
```

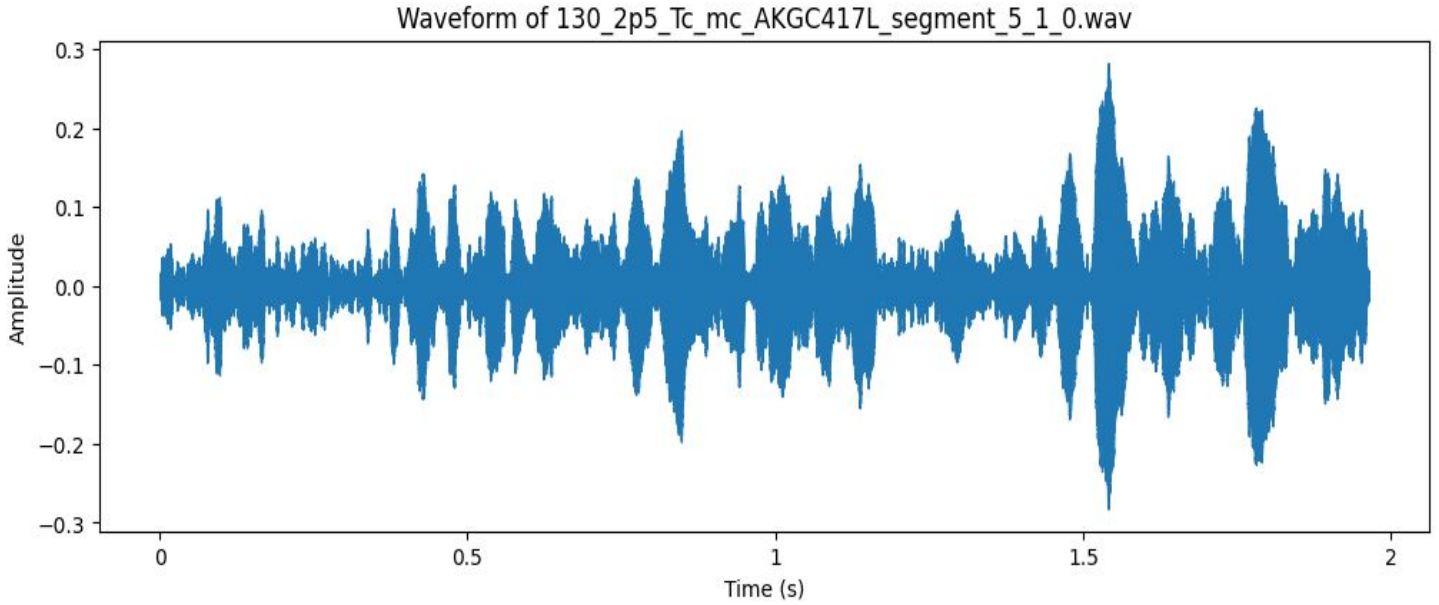


Machine Learning Pipeline



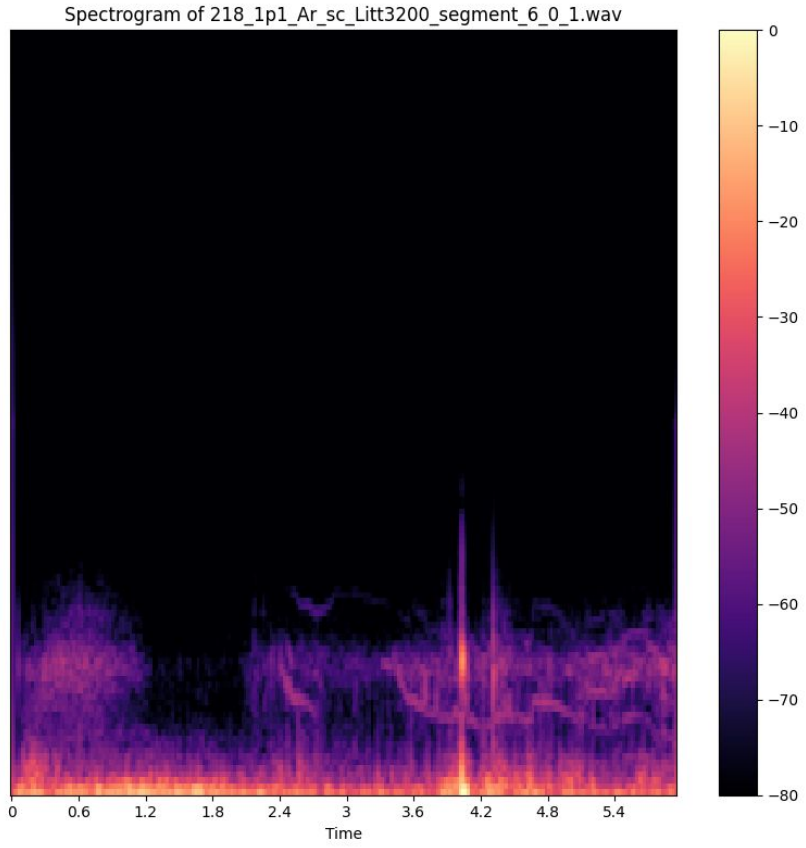
Data visualisation: audio waveform

```
def plot_waveform(audio_file_path):

    y, sr = librosa.load(audio_file_path)

    plt.figure(figsize=(12, 4))
    librosa.display.waveshow(y, sr=sr)
    plt.title('Waveform of '+str(audio_file_path.split('/')[-1]))
    plt.xlabel('Time (s)')
    plt.ylabel('Amplitude')
    plt.show()
```

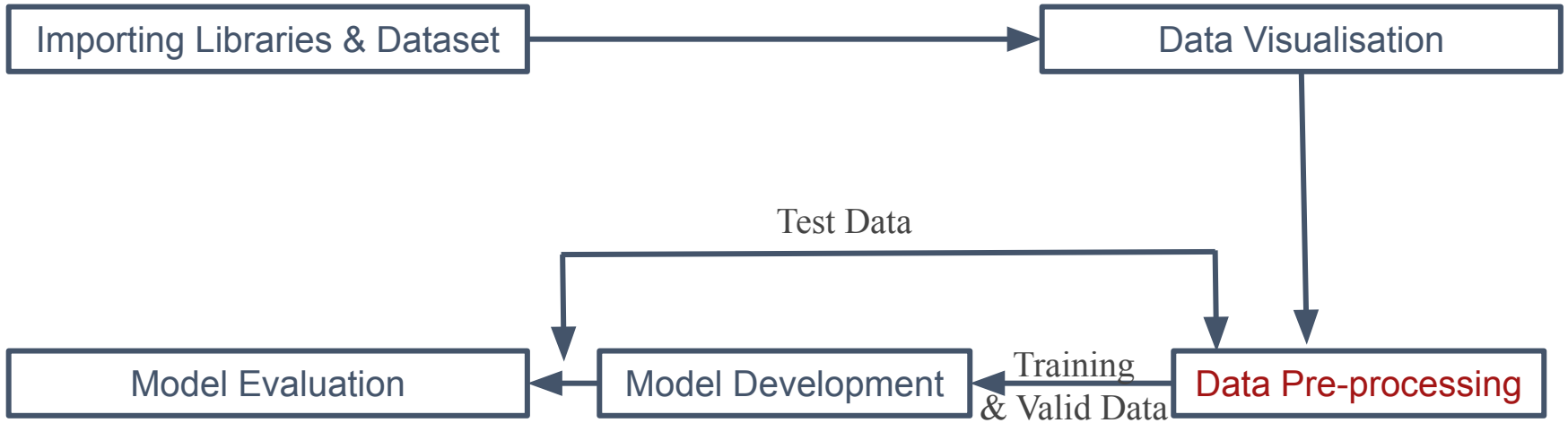


Data visualisation: mel-spectrogram

```
def build_spectogram(file_path):
    plt.interactive(False)
    plt.figure(figsize=(12, 4))
    file_audio_series,sr = librosa.load(file_path)
    spec_image = plt.figure(figsize=[8,8])
    spectrogram = librosa.feature.melspectrogram(y=file_audio_series, sr=sr)
    librosa.display.specshow(librosa.power_to_db(spectrogram, ref=np.max), x_axis='time')
    plt.colorbar()
    plt.title('Spectrogram of '+str(file_path.split('/')[-1]))
    plt.tight_layout()
    plt.show()
```



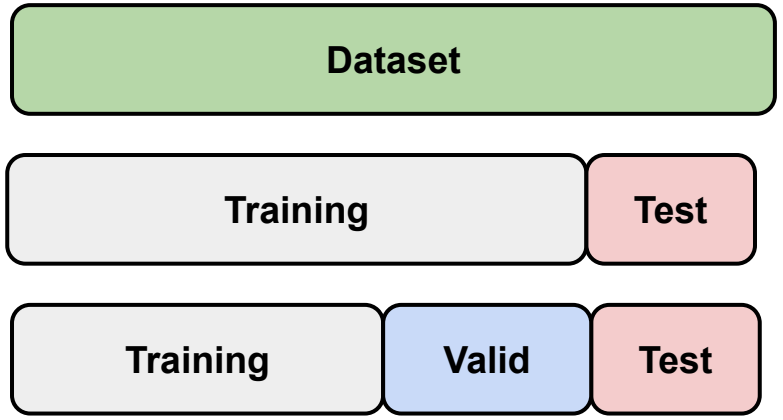
Machine Learning Pipeline



- Data Splitting
- Feature Extraction
- DataLoader

Data Partitioning

- **Training set:**
  - Data used to fit the model
- **Validation set:**
  - Data used to evaluate the model while tuning model's hyperparameters
- **Test set:**
  - Data used to evaluate success of the final model



How to partition our dataset into the subsets for optimal evaluation?

- user-independent training & test split
- train valid ratio set as 9:1

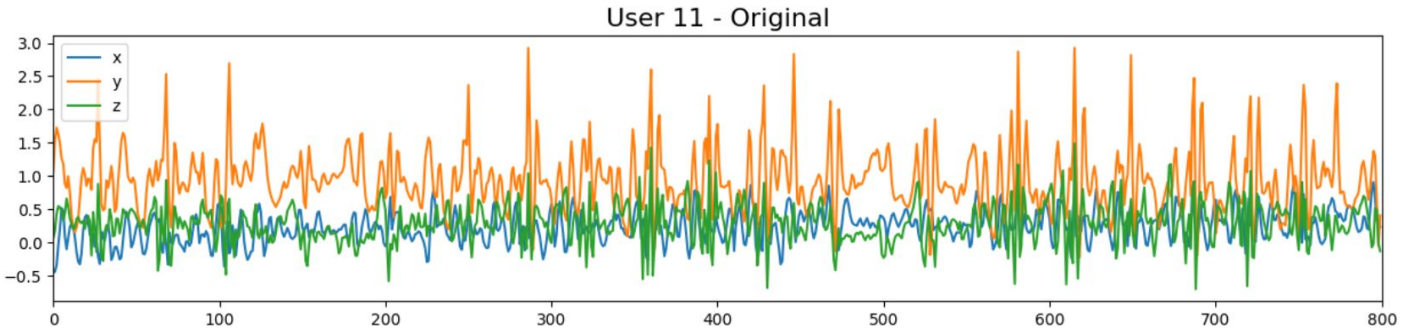
Feature Extraction and Dataset Preparation

```
class AudioDataset(Dataset):
    def __init__(self, file_list, transform=None):
        self.file_list = file_list
        self.transform = transform

    def __len__(self):
        return len(self.file_list)

    def __getitem__(self, idx):
        audio_file = self.file_list[idx]
        # load the audiofile
        waveform, sample_rate = torchaudio.load(audio_file, normalize=True)
        # compute melspectrograms
        specgram = torchaudio.transforms.MelSpectrogram()(waveform)

        if self.transform:
            specgram = self.transform(specgram)

        # assign binary labels based on file path
        label = 1 if "pos" in self.file_list[idx] else 0  # 1 for pos, 0 for neg

        return specgram, label

transform = transforms.Compose([
    transforms.Resize((128, 64)),  # Resize the spectrogram to a specific size
    # transforms.ToTensor(),
])

# Instances of AudioDataset are created for train/valid/test
train_dataset = AudioDataset(train_file_list, transform)
val_dataset = AudioDataset(valid_file_list, transform)
test_dataset = AudioDataset(test_file_list, transform)
```

Digression – IMU data loading

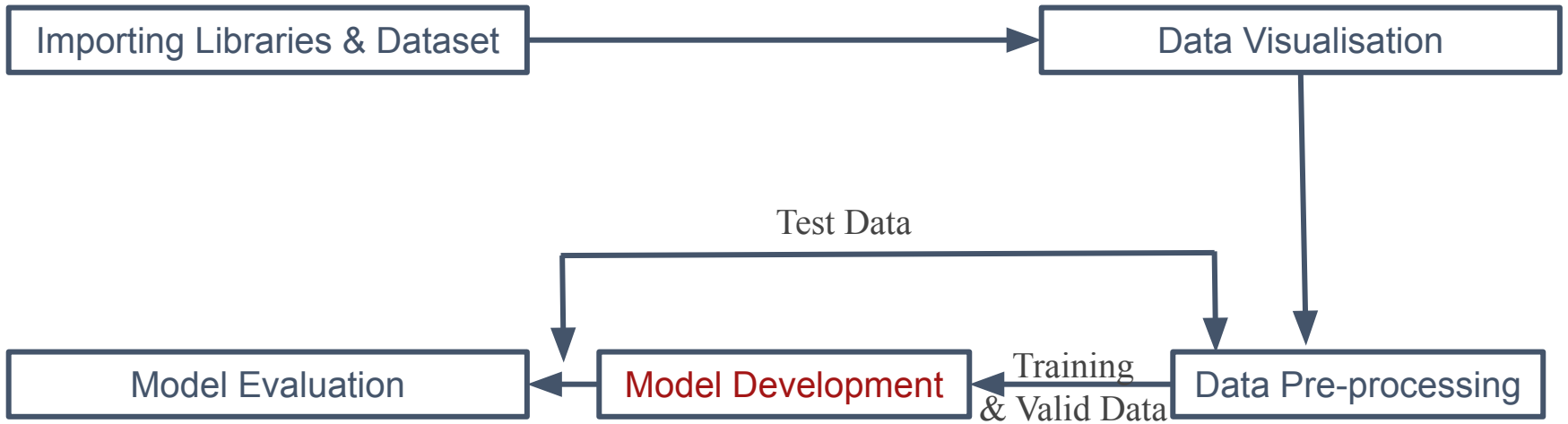
In assignment 2 you will work with IMU data, not audio

- As with every modality, there are multiple ways one can preprocess the data
- E.g., Spectrogram approach also possible and useful in IMU case
- An alternative would be to process the raw time-series by a stack of CNN and/or RNN layers

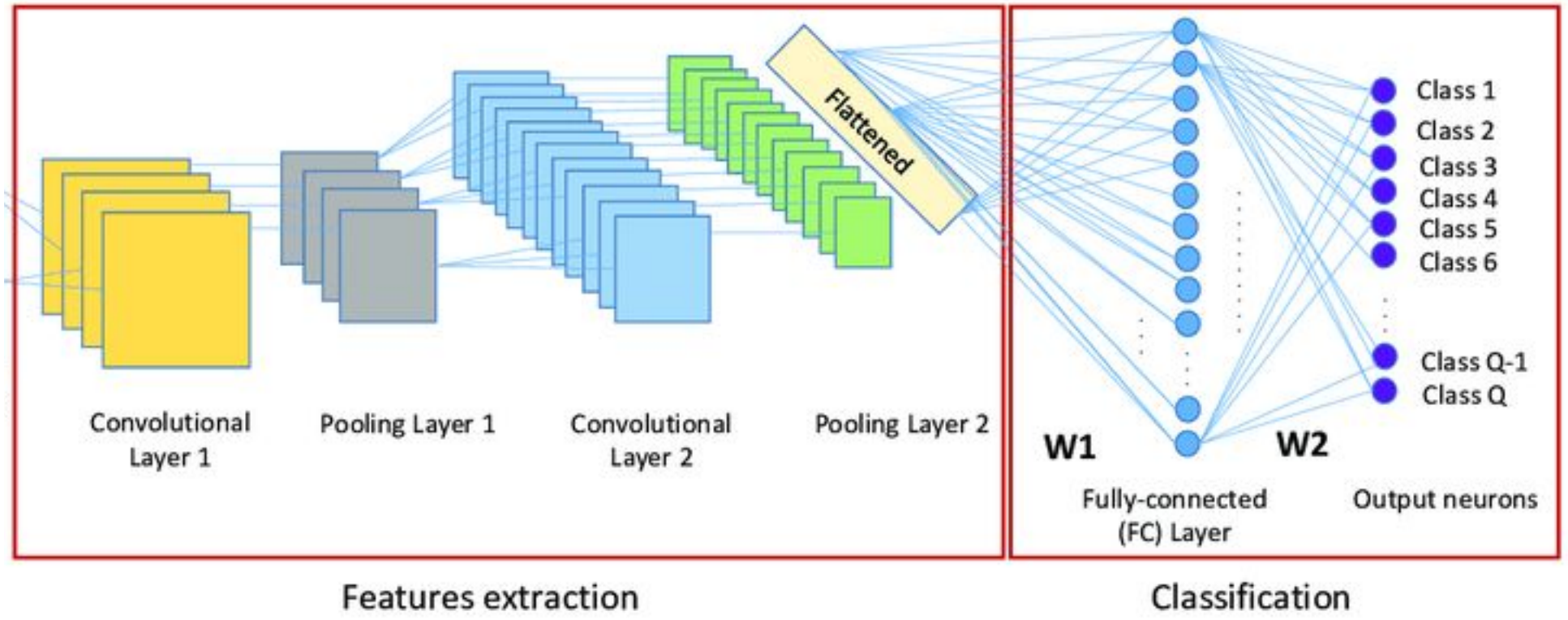


- Code will be provided in assignment 2 for reading and normalisation

Machine Learning Pipeline



Build a Model - simple CNN



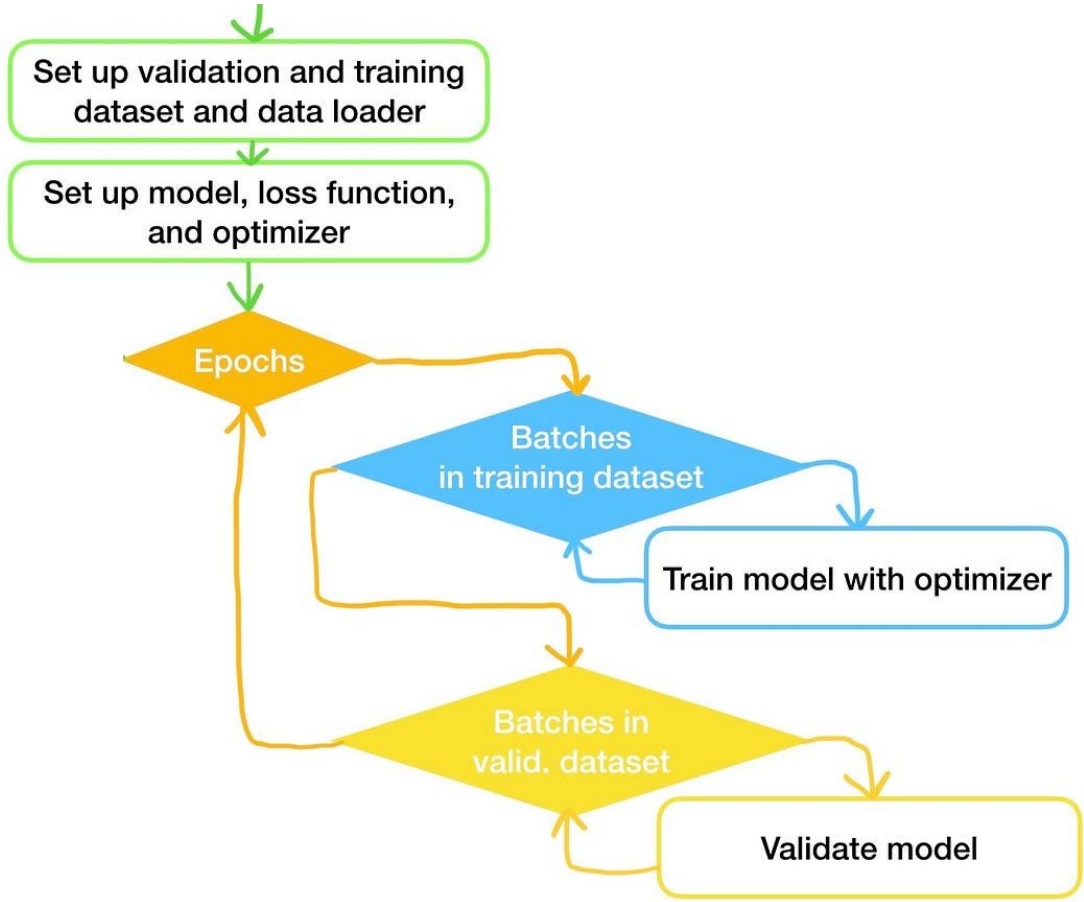
Build a Model - simple CNN

```
### Define a simple 2-layer CNN model
class SimpleCNN(nn.Module):
    def __init__(self):
        super(SimpleCNN, self).__init__()
        self.conv1 = nn.Conv2d(1, 16, kernel_size=3,
stride=1, padding=1)
        self.relu1 = nn.ReLU()
        self.pool1 = nn.MaxPool2d(kernel_size=2,
stride=2)

        self.conv2 = nn.Conv2d(16, 32, kernel_size=3,
stride=1, padding=1)
        self.relu2 = nn.ReLU()
        self.pool2 = nn.MaxPool2d(kernel_size=2,
stride=2)

        self.flatten = nn.Flatten()
        self.fc1 = nn.Linear(32 * 16 * 32, 128)
        self.relu3 = nn.ReLU()
        self.fc2 = nn.Linear(128, 2)

    def forward(self, x):
        # YOUR CODE GOES HERE
        return x
```

```
### hyperparameters
learning_rate = 0.001
epochs = 16

# Initialize the model, loss function, and optimizer
model = SimpleCNN()
criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.parameters(),
lr=learning_rate)
```

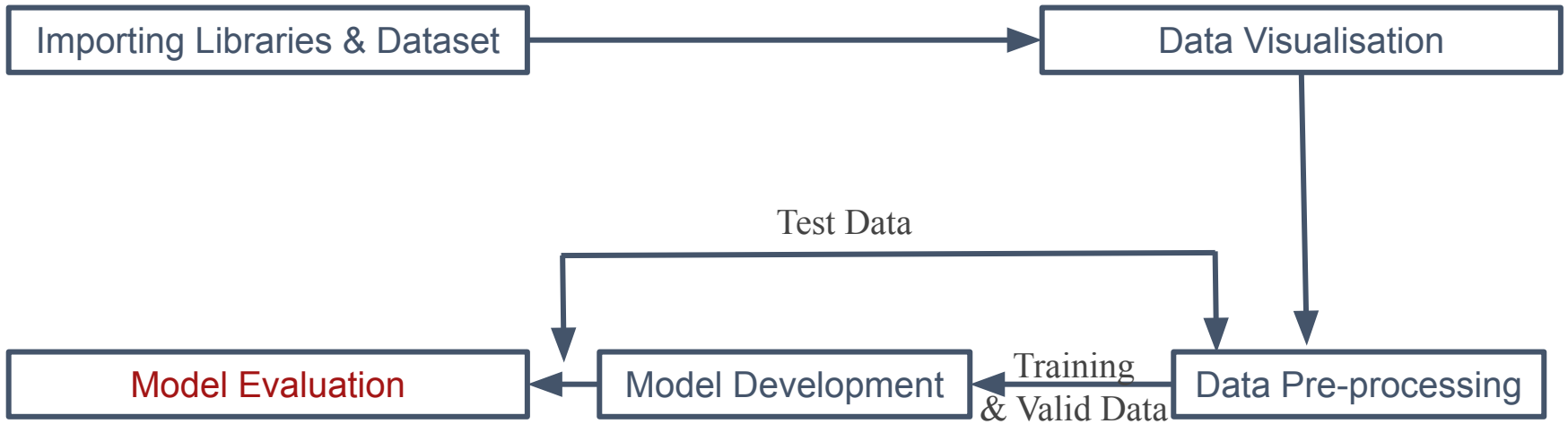
Train a Model

```
### Training loop
for epoch in range(epochs):
    model.train()
    # Training steps
    for inputs, labels in train_loader:
        optimizer.zero_grad()
        outputs = model(inputs)  # compute forward pass
        loss = criterion(outputs, labels)  # calculate loss
        loss.backward()  # backpropagate gradients
        optimizer.step()  # updating model parameters
        train_loss += loss.item()
        _, predicted = outputs. max(1)

    # Validation steps
    model. eval ()

    with torch.no_grad():  ### no backpropagation for valid set
        for inputs, labels in val_loader:
            outputs = model(inputs)
            loss = criterion(outputs, labels)
            val_loss += loss.item()

            _, predicted = outputs. max(1)
            total += labels.size( 0)
            correct += predicted.eq(labels). sum().item()
    # Save the model with the best accuracy on the validation set

# Save the best model to a file
if best_model_state is not None:
    torch.save(best_model_state, 'best_model_on_dev.pth' )
    print('Best model saved!' )
```



Task: visualise loss & accuracy per epoch

Machine Learning Pipeline



- Accuracy on Test Data
- Other performance measures, e.g., confusion matrix & classification report

- 2 Conv. Layers + Max. Pooling
- 2 FC Layers
- Train/validation loop
- Visualise Loss & accuracy per epoch

- Data Splitting
- Feature Extraction
- DataLoader

Evaluate the Model

```
# Loads the best model weights and evaluate on the test set
model = SimpleCNN()
model.load_state_dict(torch.load('best_model_on_dev.pth'))
model.eval()

all_predictions = []
all_labels = []

with torch.no_grad():
    for inputs, labels in test_loader:
        outputs = model(inputs)
        predictions = torch.argmax(outputs, dim=1).tolist()
        all_predictions.extend(predictions)
        all_labels.extend(labels.tolist())

accuracy = accuracy_score(all_labels, all_predictions)
print(f"Test Accuracy: {accuracy * 100:.2f}%")
```

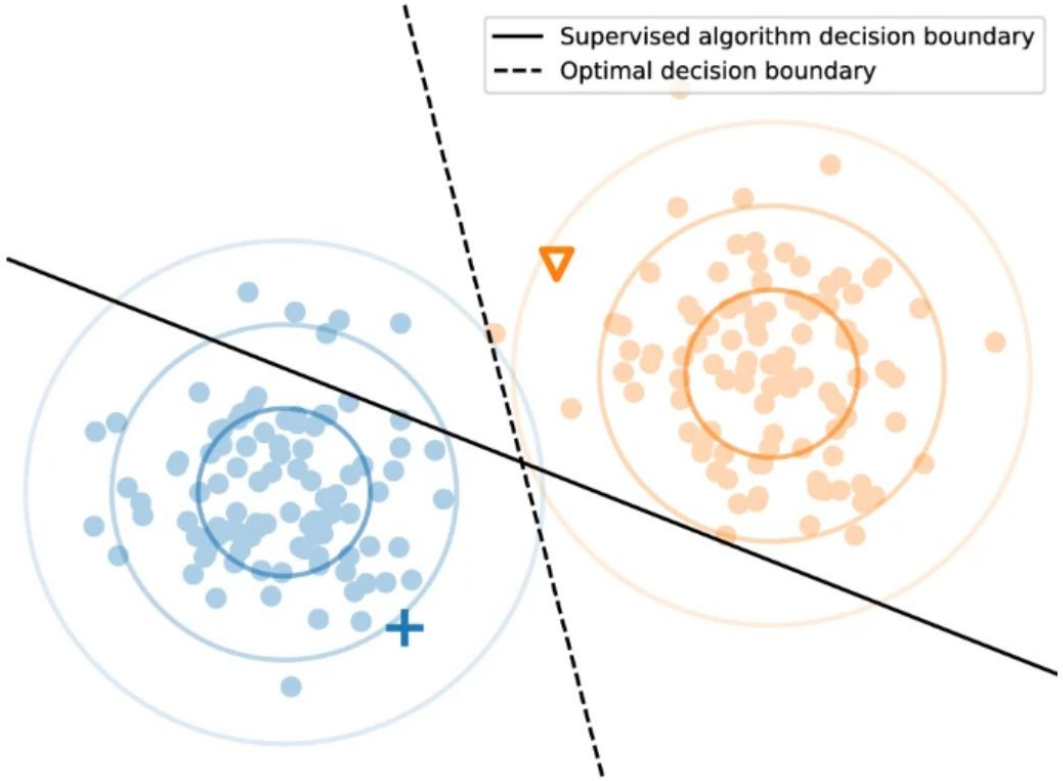
Task: compute confusion matrix and create classification_report on test data

Extension: CNN + RNN combination?

Task: one can add an RNN layer on top of the CNN layers

- Need to add an RNN (e.g., GRU) layer in the __init__() function of the Model
- Then, to alter the forward() function as well
  - An RNN layer processes a batch of (sequence_length x features)-sized tensor.
  - From the CNN we have a batch of (channels x frequency x sequence_length)
  - What are the features? How do we reformat our data?

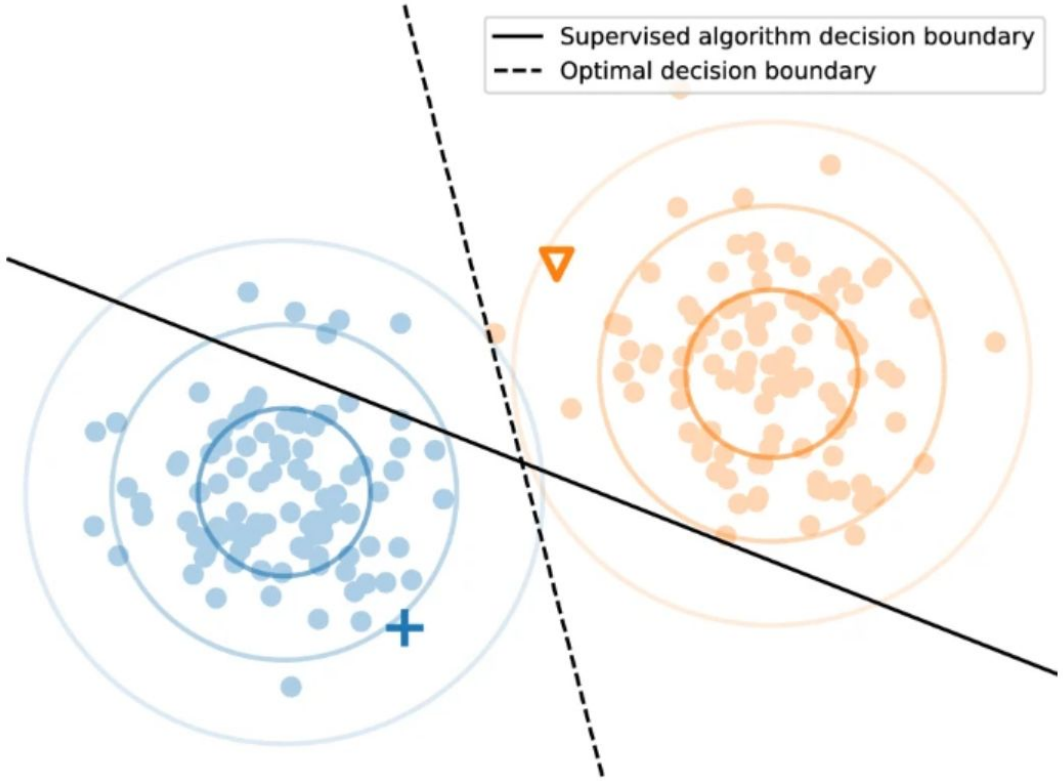
Semi-supervised Learning: smoothness/cluster hypothesis



Van Engelen, J. E., & Hoos, H. H. (2020). A survey on semi-supervised learning. *Machine learning*, *109*(2), 373-440.

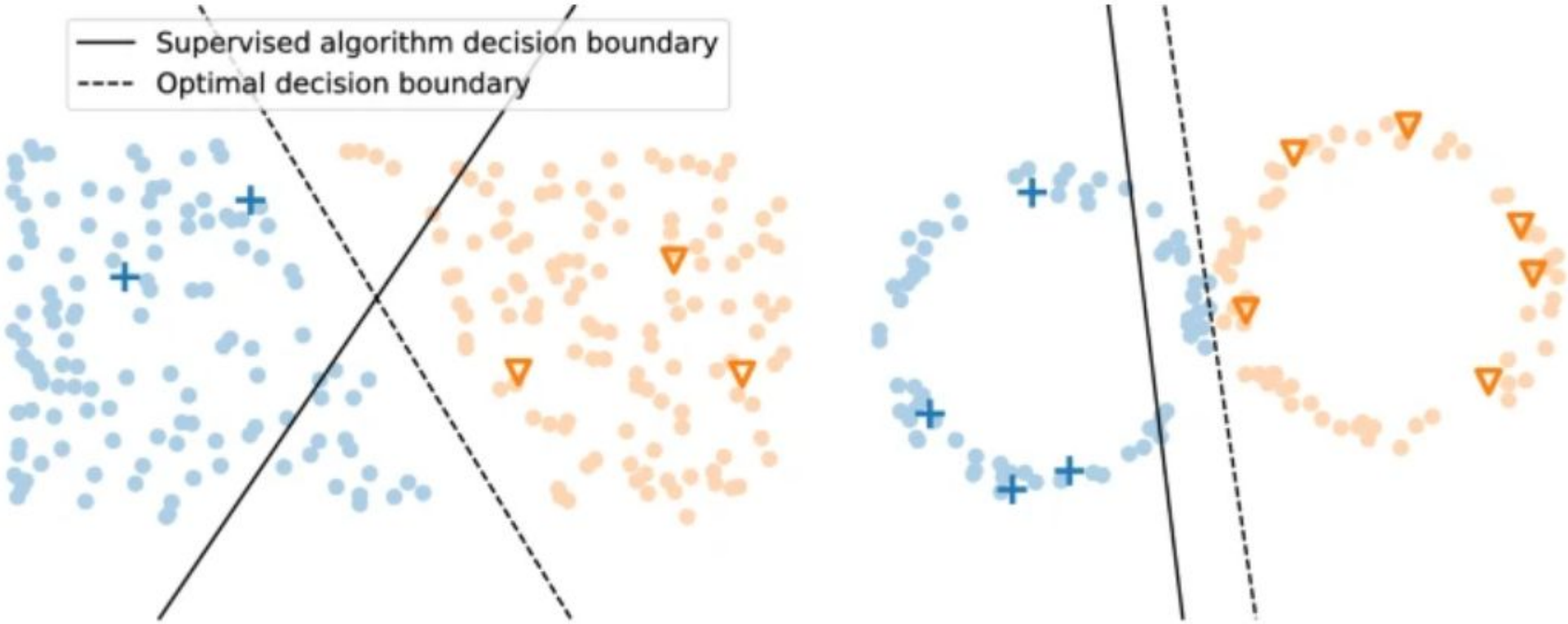
Semi-supervised Learning: smoothness/cluster hypothesis

- A few labelled data
- We don’t want a naive discrimination
- Consider the generating functions
  - (and the unlabelled data)



Van Engelen, J. E., & Hoos, H. H. (2020). A survey on semi-supervised learning. *Machine learning*, *109*(2), 373-440.

Semi-supervised Learning: smoothness/cluster hypothesis



Van Engelen, J. E., & Hoos, H. H. (2020). A survey on semi-supervised learning. *Machine learning*, *109*(2), 373-440.

Semi-supervised Learning

- $D_L = \{X_L, Y_L\}$ – labelled data set
- $D_U = \{X_U\}$ – unlabelled data set

$$\min_{\theta} \underbrace{\sum_{x \in X_L, y \in Y_L} \mathcal{L}_s(x, y, \theta)}_{\text{supervised loss}} + \alpha \underbrace{\sum_{x \in X_U} \mathcal{L}_u(x, \theta)}_{\text{unsupervised loss}} + \beta \underbrace{\sum_{x \in X} \mathcal{R}(x, \theta)}_{\text{regularization}},$$

Semi-supervised Learning

- $D_L = \{X_L, Y_L\}$ – labelled data set
- $D_U = \{X_U\}$ – unlabelled data set

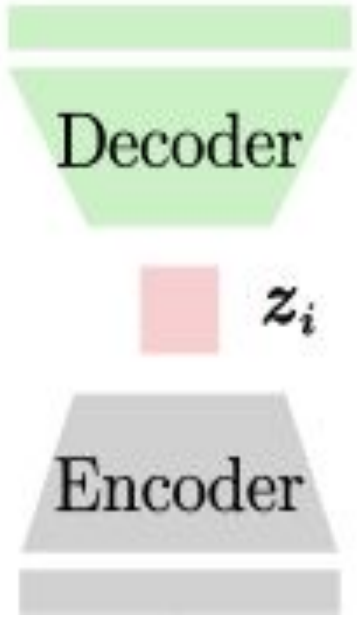
$$\min_{\theta} \underbrace{\sum_{x \in X_L, y \in Y_L} \mathcal{L}_s(x, y, \theta)}_{\text{supervised loss}} + \alpha \underbrace{\sum_{x \in X_U} \mathcal{L}_u(x, \theta)}_{\text{unsupervised loss}} + \beta \underbrace{\sum_{x \in X} \mathcal{R}(x, \theta)}_{\text{regularization}},$$

There's the new part!

- The most common idea is to get the model to label the unlabelled sample as it would have neighbouring samples

Semi-supervised Learning: reconstruction-based

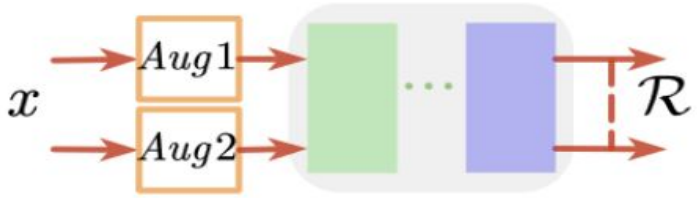
- Apart from supervised learning, pass all your data through an autoencoder
- Minimise reconstruction loss
- Can be applied on all types of data: vision, audio, graph.

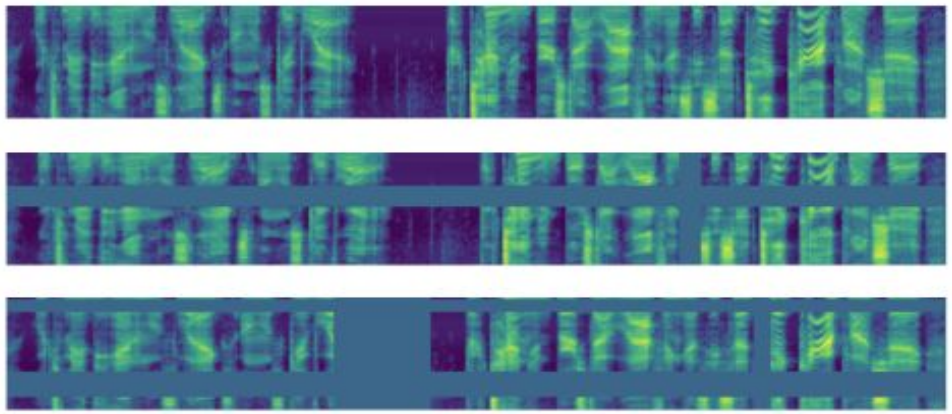


- Yang, X., Song, Z., King, I., & Xu, Z. (2022). A survey on deep semi-supervised learning. *IEEE Transactions on Knowledge and Data Engineering*

Semi-supervised Learning: consistency regularisation

- Π-model

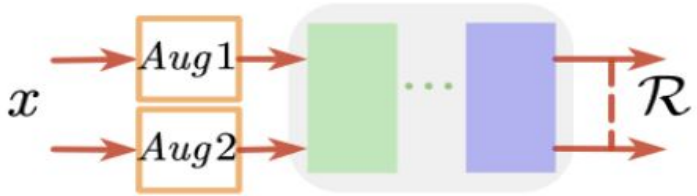


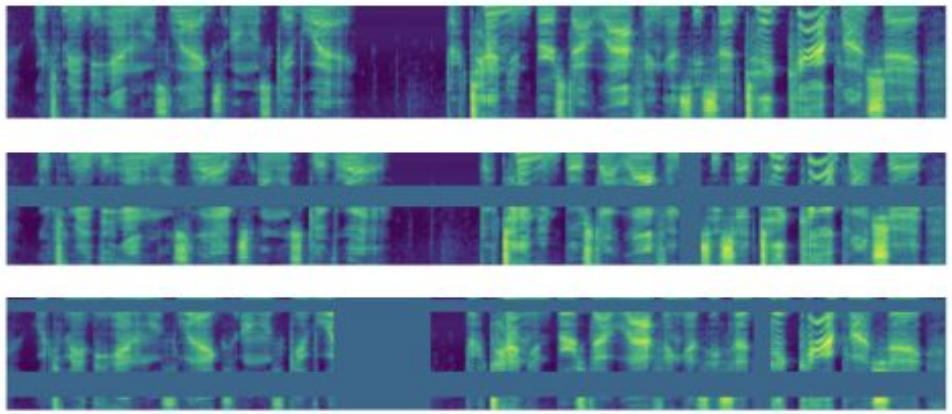


- Yang, X., Song, Z., King, I., & Xu, Z. (2022). A survey on deep semi-supervised learning. *IEEE Transactions on Knowledge and Data Engineering*
- *Park, D. S., Chan, W., Zhang, Y., Chiu, C. C., Zoph, B., Cubuk, E. D., & Le, Q. V. (2019). SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition. Interspeech 2019.*

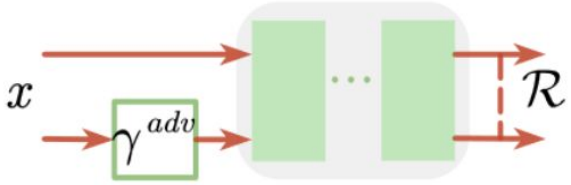
Semi-supervised Learning: consistency regularisation

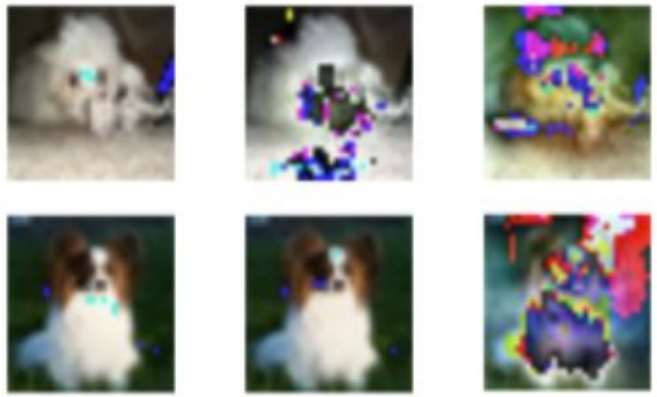
- Π-model





- Virtual Adversarial Training

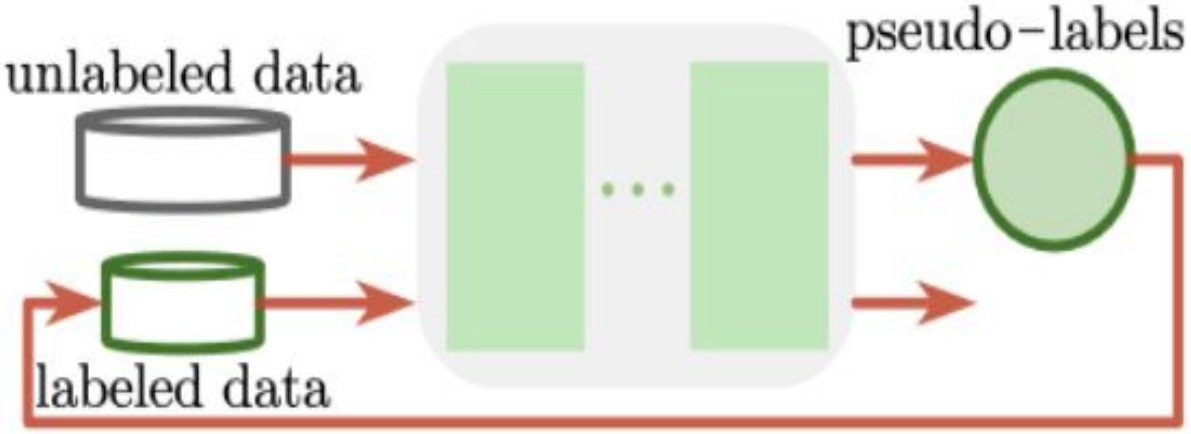




- Yang, X., Song, Z., King, I., & Xu, Z. (2022). A survey on deep semi-supervised learning. *IEEE Transactions on Knowledge and Data Engineering*
- *Park, D. S., Chan, W., Zhang, Y., Chiu, C. C., Zoph, B., Cubuk, E. D., & Le, Q. V. (2019). SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition. Interspeech 2019.*
- *Miyato, T., Maeda, S. I., Koyama, M., & Ishii, S. (2018). Virtual adversarial training: a regularization method for supervised and semi-supervised learning. IEEE transactions on pattern analysis and machine intelligence, 41(8), 1979-1993.*

Semi-supervised Learning: Pseudolabelling

- We can assume current predictions of model are good enough to be used as labels



- Challenges arise: how confident are we that the pseudo-labels are correct?

- Yang, X., Song, Z., King, I., & Xu, Z. (2022). A survey on deep semi-supervised learning. *IEEE Transactions on Knowledge and Data Engineering*

Assignment 2

- Released on Moodle, due on **15/03/2024 at noon**
- Weighting: **60**% of the course grade
- Part II: Colab notebook and reflection report of 1,200 words
- Part III/MPhil/CDT: Colab notebook and a reflection report of 1,800 words
- Please use the **open help forum** for any questions
- Office hours:
  - 05/03/2024 at 15:00-16:00
  - 12/03/2024 at 14:00-15:00

Time to work on colab – any questions?