Assignment2

October 2, 2024

1 Assignment 2: Spectrogram classification

Deadline: 07/10/24

Submission: Submit a PDF export of the completed notebook as well as the .ipynb file.

General: This assignment aims to practice designing and training neural networks. The task the networks solve is "predicting"/"inferring" a signal type from its spectrogram image. You will explore two neural network architectures. A starter code is provided to help with data processing and make it a bit easier.

You may modify the starter code as you see fit, including changing the signatures of functions and adding/removing helper functions. However, please ensure you adequately explain what you are doing and why.

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import collections
import scipy.io
import cv2
from google.colab.patches import cv2_imshow

import torch
import torch.nn as nn
import torch.optim as optim
```

1.1 Question 1. Data (15%)

With any machine learning problem, the first thing that we would want to do is to get an intuitive understanding of what our data looks like. Download the file `Data set` from the course page on Moodle and upload it to Google Drive. Then, mount Google Drive from your Google Colab notebook:

[2]:
```
from google.colab import drive
drive.mount('/content/gdrive')

# Find the path to the file:
path = '/content/gdrive/My Drive/assignment2' # TODO - UPDATE ME!
```

```
Mounted at /content/gdrive
```

1.1.1 Part (a) – 3%

Load the training and test data, and separate the training data into training and validation. Create the NumPy arrays `train_data`, `valid_data`, `test_data`.

1. `data`, all of which should be of shape `[N, 128, 128, 1]`. The dimensions of this NumPy array are as follows:

- `N` - the number of rows allocated to train, valid, or test
- `128` - the height of each spectrogram (i.e., the number of freq. points)
- `128` - the width of each spectrogram (i.e., the number of time samples)
- `1` - the color channels

2. `labels`, all of which should be of shape `[N,]` The dimensions of this NumPy array are as follows:

- `N` - the number of rows allocated to train, valid, or test

The pixel intensities are stored as an integer between 0 and 255. Make sure you normalize your images, namely, divide the intensities by 255 so that you have floating-point values between 0 and 1. Then, subtract 0.5 so that the elements of `train_data`, `valid_data` and `test_data` are between -0.5 and 0.5. **Note that this step actually makes a huge difference in training!**

This function might take a while to run, and it can take several minutes just to load the files from Google Drive. If you want to avoid running this code multiple times, you can save your NumPy arrays and load it later: https://docs.scipy.org/doc/numpy/reference/generated/numpy.save.html

[3]:
```
import glob
from PIL import Image
folder_path = '/content/gdrive/My Drive/assignment2/data set'
def sort_data(folder_path):
  train_path = f'{folder_path}/train_data/*.jpg'
  test_path = f'{folder_path}/test_data/*.jpg'
  train_images = {}
  test_images = {}
  for file in glob.glob(train_path):
      filename = file.split("/")[-1] # get the name of the .png file
      label = filename.split('_')[0] # get the label
      image = cv2.imread(file, cv2.IMREAD_GRAYSCALE)
      img = cv2.resize(image, dsize=(128, 128), interpolation=cv2.INTER_CUBIC)
      img = (img/255) - 0.5
      train_images[filename] = img

  for file in glob.glob(test_path):
      filename = file.split("/")[-1] # get the name of the .png file
      label = filename.split('_')[0] # get the label
      image = cv2.imread(file, cv2.IMREAD_GRAYSCALE)
      img = cv2.resize(image, dsize=(128, 128), interpolation=cv2.INTER_CUBIC)
      img = (img/255) - 0.5
```

```
      test_images[filename] = img

  sort_train_dict = dict(sorted(train_images.items()))
  sort_test_dict = dict(sorted(test_images.items()))
  train = np.array(list(sort_train_dict.values()))
  test = np.array(list(sort_test_dict.values()))
  train_label = np.array(list(sort_train_dict.keys()))
  test_labels = np.array(list(sort_test_dict.keys()))
  reindex = np.random.permutation(len(train))
  train = train[reindex]
  train_label = train_label[reindex]
  tr_label, ts_label = [], []
  for ii in range(len(train_label)):
    tr_label.append(train_label[ii].split('_')[0])
  for ii in range(len(test_labels)):
    ts_label.append(test_labels[ii].split('_')[0])
  train_label = np.array(tr_label)
  test_labels = np.array(ts_label)
  train_data, valid_data = train[int(train.shape[0] * 0.15):], train[:int(train.
↪shape[0] * 0.15)]
  train_labels, valid_labels = train_label[int(train_label.shape[0] * 0.15):],␣
↪train_label[:int(train_label.shape[0] * 0.15)]
  return train_data, train_labels, valid_data, valid_labels, test, test_labels
```

First we will run the sort function

[ ]:
```
train_data, train_labels, valid_data, valid_labels, test, test_labels =␣
↪sort_data(folder_path)
```

Load the saved npy filed from the drive

[4]:
```
dir = '/content/gdrive/My Drive/assignment2/'
train_data = np.load(dir + 'train_data.npy')
train_labels = np.load(dir + 'train_labels.npy')
valid_data = np.load(dir + 'valid_data.npy')
valid_labels = np.load(dir + 'valid_labels.npy')
test_data = np.load(dir + 'test.npy')
test_labels = np.load(dir + 'test_labels.npy')
```

1.1.2 Part (b) – 3%

We want to train a model that determines the signal type from a spectrogram. Therefore, our model will take in a spectrogram image.

Write a function generate_plots() that takes one of the data sets that you produced in part (a), and generates image plots of the different spectrograms with different classes. Your function generate_plots() plots 12 subplots of spectrogram images containing all classes.

Note: While at this stage we are working with NumPy arrays, later on, we will need to convert

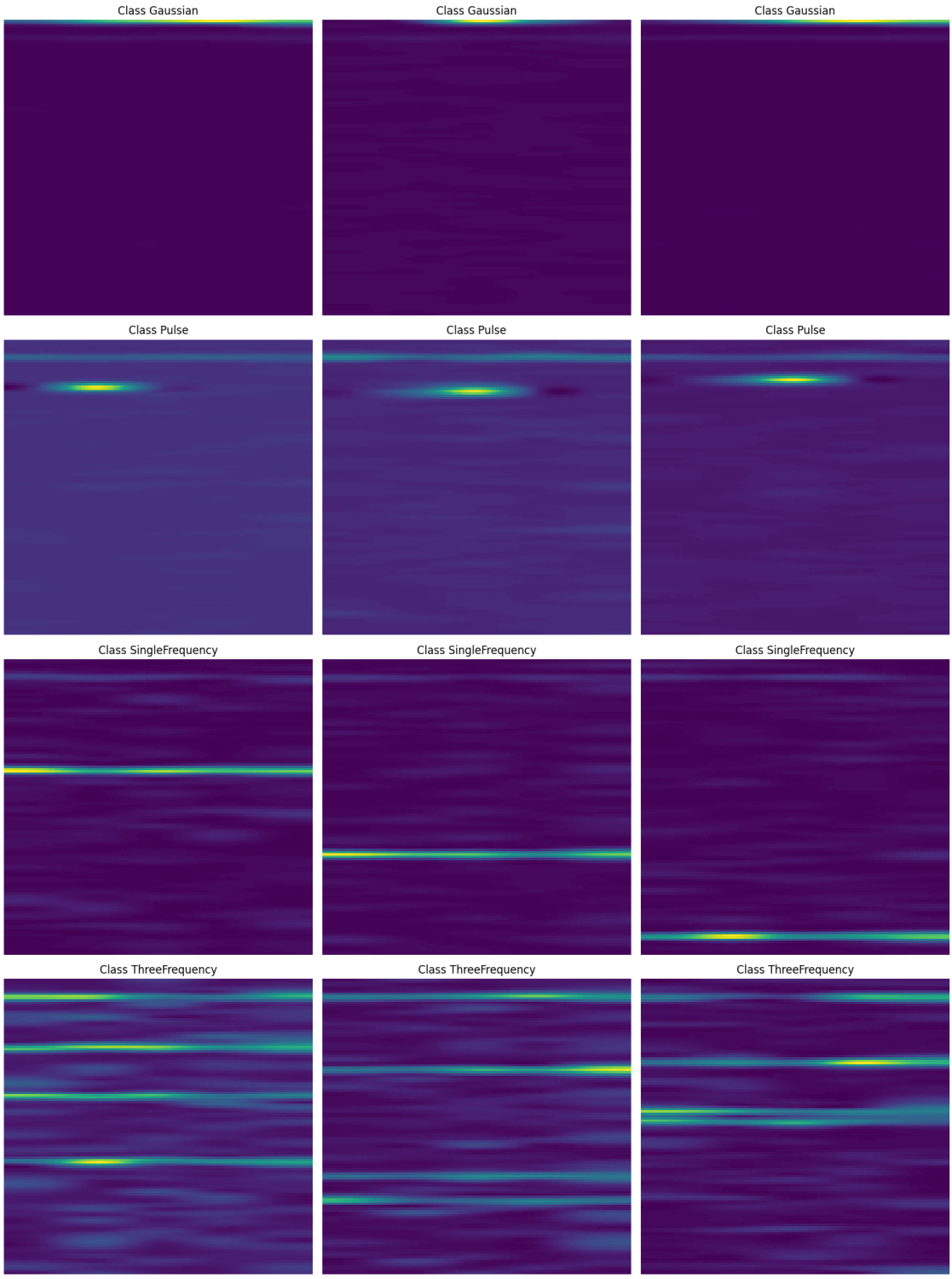
this NumPy array into a PyTorch tensor with shape [N, 128, 128].

Include the result with your PDF submission.

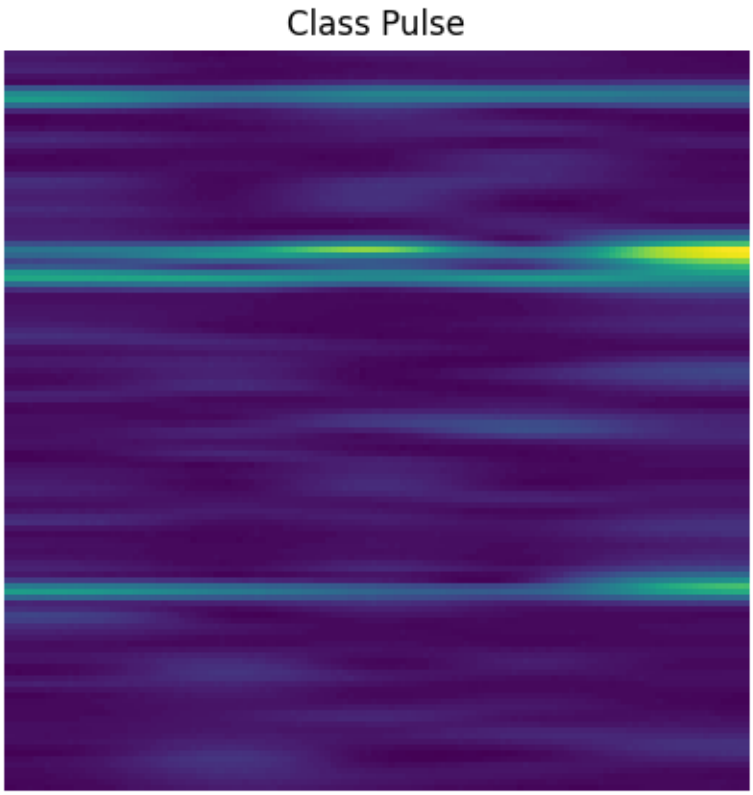
[5]:
```
# Your code goes here
def generate_plots(data):
    """
    Generates and displays spectrogram plots. If 'data' contains multiple␣
 ↪spectrograms,
    it displays up to 3 samples per class in a grid. If 'data' contains a␣
 ↪single spectrogram,
    it displays that spectrogram with its class label.

    Parameters:
    data (numpy.ndarray): The input spectrogram data.

    Returns:
    numpy.ndarray: The same data that was input.
    """

    # Check if 'data' contains multiple spectrograms by examining its dimensions
    if data.ndim > 2:
        labels = train_labels  # Assume 'train_labels' is defined globally
        unique_labels = np.unique(labels)
        num_classes = unique_labels.size

        # Prepare a dictionary to hold up to 3 samples per class
        class_samples = {}
        for label in unique_labels:
            # Find indices where the label matches
            indices = np.flatnonzero(labels == label)
            # Select up to 3 samples for the current class
            samples = data[indices[:3]]
            class_samples[label] = samples

        # Create a figure with subplots: rows = number of classes, columns = 3
        fig, axes = plt.subplots(nrows=num_classes, ncols=3, figsize=(15, 5 *␣
 ↪num_classes))

        # Ensure 'axes' is a 2D array even if there's only one class
        if num_classes == 1:
            axes = np.expand_dims(axes, 0)

        # Iterate over each class and its samples to plot them
        for i, label in enumerate(unique_labels):
            samples = class_samples[label]
            for j in range(3):
                ax = axes[i, j]
```

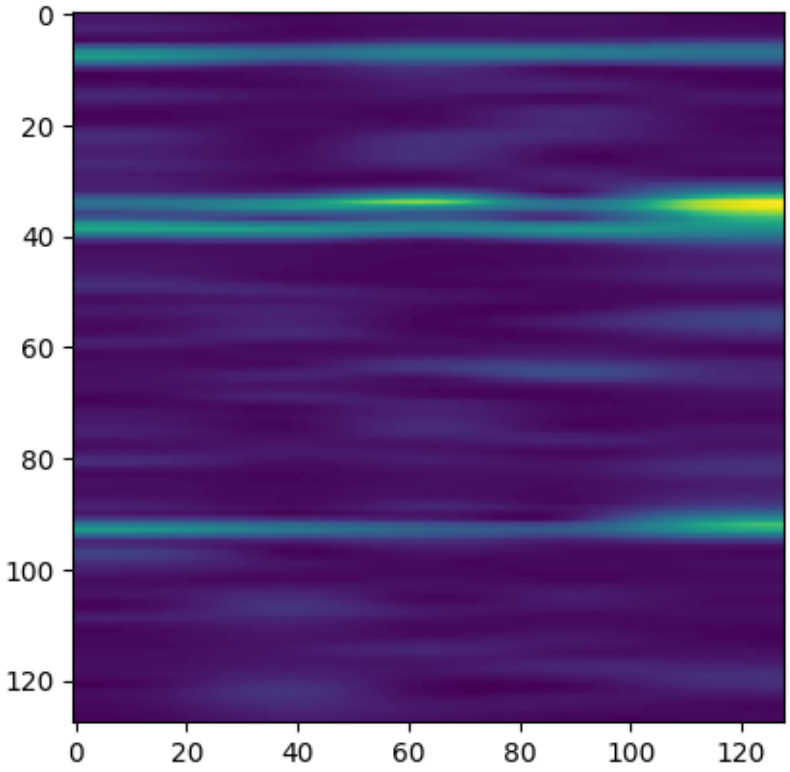
```
                if j < len(samples):
                    ax.imshow(samples[j], aspect='auto')
                    ax.set_title(f"Class {label}")
                ax.axis('off')  # Hide axis ticks and labels

        plt.tight_layout()
        plt.show()

    else:
        # If there's only one spectrogram, display it with its class label
        idx = 31 * 30  # This index should correspond to the spectrogram's label
        label = train_labels[idx]
        plt.figure(figsize=(5, 5))
        plt.imshow(data, aspect='auto')
        plt.title(f"Class {label}")
        plt.axis('off')
        plt.show()

    return data



# Run this code, include the result with your PDF submission!!
print(train_data.shape) # if this is [N, 128, 128]
print(generate_plots(train_data).shape) # should be [N, 128, 128]
idx = 20
plt.imshow(generate_plots(train_data[idx*30])) # should show spectrogram, 30 is␣
 ↪just an example.
# Please take the first 2 digits of your ID (if both of your ID starts with 0␣
 ↪change it to 1)


plt.show()
```

```
(18360, 128, 128)
```



(18360, 128, 128)


Class Pulse



1.1.3 Part (c) – 3%

Why is it important that our data set will be ***balanced***? In other words, suppose we created a data set where 99% of the images are of Gaussian spectrogram, and 1% of the images are the other classes. Why could this be a problem?

Write your explanation here:

Training a model on a dataset where one class significantly outnumbers the others can cause the model to become biased toward the majority class, resulting in poor performance on the less represented (minority) classes.

In terms of generalization, a model trained on such imbalanced data may not perform well on new, unseen data. This is because the model's bias toward the majority class can hinder its ability to accurately classify instances from other classes.

Furthermore, it's important that training data samples are independent and identically distributed (i.i.d.) to ensure that the empirical risk converges to the true risk. An imbalanced dataset disrupts this by introducing a bias toward the majority class, leading to a different distribution that can negatively affect the training process.

1.1.4 Part (d) – 3%

Our neural network will take as input spectrogram images and predict their class. Since we have four string classes we would want to convert them into numbers, where each number is assigned to

each class.

Complete the helper function `convert_class_to_number` so that the function output will be a dictionary that assigns a number to each class. Examples of how this function should operate are detailed in the code below.

You can use the defined `vocab`, `lables2num_vocab`, and `num2labels_vocab` in your code.

[8]:
```
# A list of all the labels in the data set. We will assign a unique
# identifier for each of these labels.
vocab = sorted(list(set([s for s in train_labels]))) # A mapping of index =>␣
 ↪label (string)
num2labels_vocab = dict(enumerate(vocab)) # A mapping of labels => its index
labels2num_vocab = {word:index for index, word in num2labels_vocab.items()}

def convert_class_to_number(labels):
    """
    This function takes a list of labels
    and returns a new list with the same structure, but where each label
    is replaced by its index in `num2labels_vocab`.

    Example:
    >>> convert_class_to_number([['Pulse', 'SingleFrequency', 'Pulse',␣
 ↪'Gaussian', 'ThreeFrequency'], ['ThreeFrequency', 'Pulse', 'Gaussian',␣
 ↪'SingleFrequency'])
    [[1, 2, 1, 0, 3], [3, 1, 0, 2]]
    """

    # Write your code here

    convLabs = []
    # Using list comprehension for cleaner code
    if isinstance(labels[0], list):
        # Handling the case of a list of lists
        convLabs = [[labels2num_vocab[label] for label in sublist] for sublist␣
 ↪in labels]
    else:
        # Handling the case of a flat list
        convLabs = [labels2num_vocab[label] for label in labels]

    return convLabs

convert_class_to_number([['Pulse', 'SingleFrequency', 'Pulse', 'Gaussian',␣
 ↪'ThreeFrequency'], ['ThreeFrequency', 'Pulse', 'Gaussian',␣
 ↪'SingleFrequency']])
```

[8]:
```
[[1, 2, 1, 0, 3], [3, 1, 0, 2]]
```

1.1.5 Part (e) – 3%

Since the labels in the data are comprised of 4 distinct classes, our task boils down to classification where the label space $\mathcal{S}$ is of cardinality $|\mathcal{S}| = 4$ while our input, which is comprised of spectrograms data, is treated as a vector of size 16384×1.

Implement yourself a function `create_onehot`, which takes the data in index notation and outputs it in a one-hot notation.

Start by reviewing the helper function, which is given to you:

[9]:
```
def create_onehot(data):
    """
    Convert one batch of data in the index notation into its corresponding␣
↪onehot
    notation. Remember, the function should work for st.

    input - vector with shape D (1D or 2D)
    output - vector with shape (D,4)
    """
    # Write your code here

    # Convert data to a NumPy array if it's not already
    data = np.asarray(data)
    num_classes = 4  # Assuming there are 4 classes

    if data.ndim == 1:
        # For 1D input
        num_samples = data.shape[0]
        onehot_data = np.zeros((num_samples, num_classes), dtype=int)
        onehot_data[np.arange(num_samples), data] = 1
    elif data.ndim == 2:
        # For 2D input
        num_samples, num_features = data.shape
        onehot_data = np.zeros((num_samples, num_features, num_classes),␣
↪dtype=int)
        # Using broadcasting to set the appropriate indices to 1
        onehot_data[np.arange(num_samples)[:, None], np.arange(num_features),␣
↪data] = 1
    else:
        raise ValueError("Input data must be a 1D or 2D array.")

    return onehot_data
```

1.2 Question 2. Model architecture (30%)

In this part we will look at two model architectures: a MultiLayer Perceptron (MLP) and a Convolutional Neural Network (CNN).

Since the labels are comprised of 4 distinct classes, our task boils down to classification where the label space $\mathcal{S}$ is of cardinality $|\mathcal{S}| = 4$ while our input is treated as a vector of size 128×128 (i.e., the spectrogram matrix).

We build the model in PyTorch. Since PyTorch uses automatic differentiation, we only need to write the *forward pass* of our model.

###Part (a) – Multy layer perceptron (MLP) (15%)

Please provide a detailed diagram that best describes this model's architecture. Specify the number of layers, weights, etc.

This link will help you to understand how to upload an image to the google Colab https://medium.com/analytics-vidhya/embedding-your-image-in-google-colab-markdown-3998d5ac2684

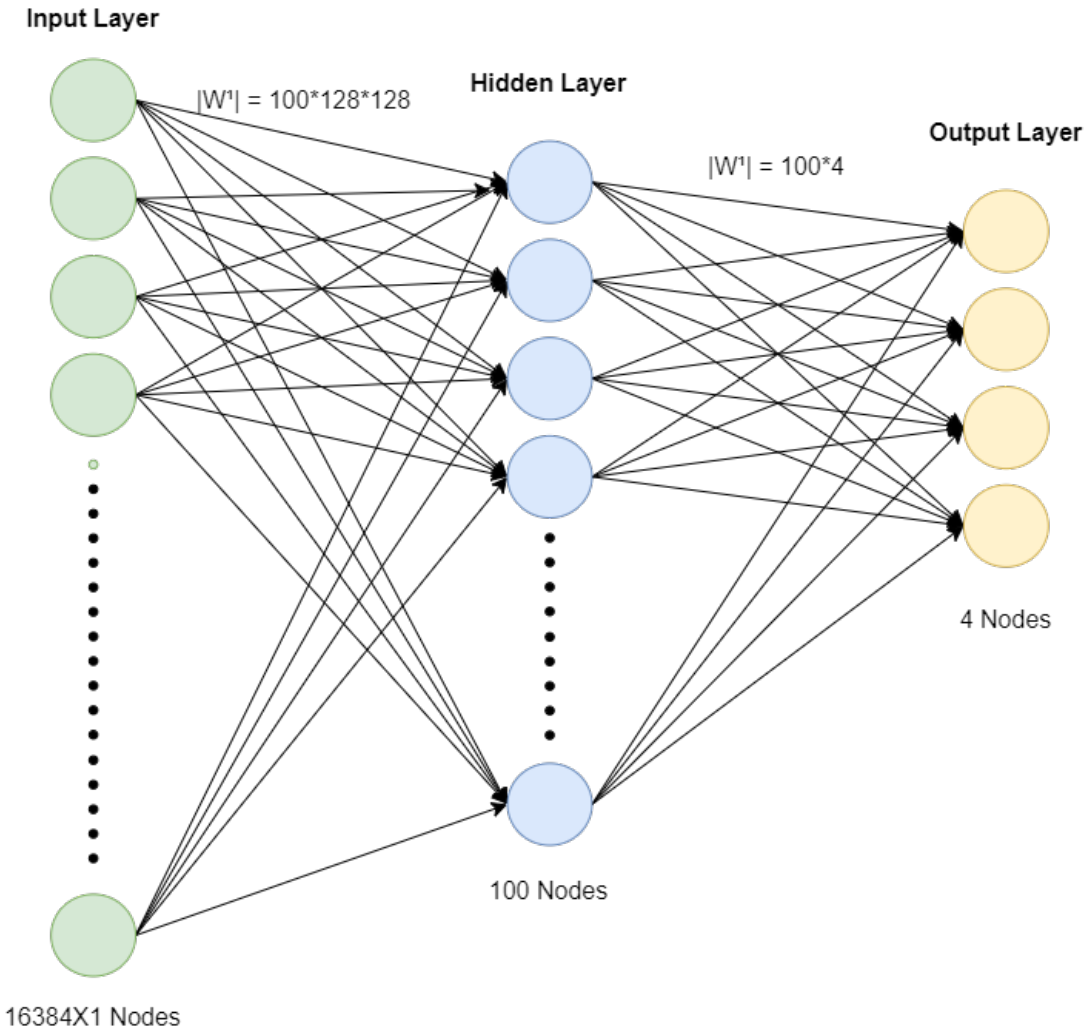
This is an example of how to change the width and height of the image scheme:

```
<img src="image path" width="400px height="200px"" />
```

[12]:
```
class PyTorchMLP(nn.Module):
    def __init__(self, num_hidden=100):
        super(PyTorchMLP, self).__init__()
        self.layer1 = nn.Linear(128*128, num_hidden)
        self.layer2 = nn.Linear(num_hidden, 4)
        self.num_hidden = num_hidden
    def forward(self, inp):
        inp = inp.reshape([-1, 128*128])
        # Note that we will be using the nn.CrossEntropyLoss(), which computes␣
 ↪the softmax operation internally, as loss criterion
        hidden = self.layer1(inp)
        output = self.layer2(hidden)
        output = torch.nn.functional.log_softmax(output, dim=1)
        return output
```

Please show the model scheme here:

[13]:
```
from IPython.display import display, Image
display(Image(filename="/content/gdrive/MyDrive/assignment2/MLP_graph.png"))
```



###Part (b) – Convolutional Neural Network (CNN) (15%)

The CNN model is given below. Please provide a detailed diagram that best describes this model's architecture. Specify the number of layers, kernel size, weights, etc.

This link will help you to understand how to upload an image to the google colab https://medium.com/analytics-vidhya/embedding-your-image-in-google-colab-markdown-3998d5ac2684

This is an example of how to change the width and height of the image scheme:

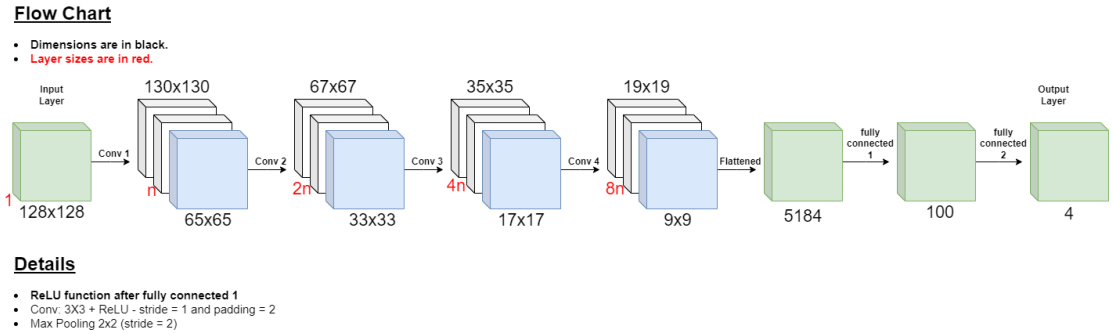
```
<img src="image path" width="400px height="200px"" />
```

[10]:
```
class CNNChannel(nn.Module):

    def __init__(self, n=8, kernel_size=3):
        super(CNNChannel, self).__init__()
```

```
        self.n = n
        self.kernel_size = kernel_size #for future calculations - added kernal␣
↪size
        self.dim=128
        for i in range(4): self.dim=int((self.dim+5-kernel_size)/2) #final dim␣
↪calculations

        self.conv1 = nn.Conv2d(in_channels=1, out_channels=n,␣
↪kernel_size=kernel_size, stride=1, padding=2)
        self.conv2 = nn.Conv2d(in_channels=n, out_channels=2*n,␣
↪kernel_size=kernel_size, stride=1, padding=2)
        self.conv3 = nn.Conv2d(in_channels=2*n, out_channels=4*n,␣
↪kernel_size=kernel_size, stride=1, padding=2)
        self.conv4 = nn.Conv2d(in_channels=4*n, out_channels=8*n,␣
↪kernel_size=kernel_size, stride=1, padding=2)
        self.fc1 = nn.Linear(self.dim*self.dim*8*n, 100) # changed by value of␣
↪n (in this case 9*9*8*8 = 5184)
        self.fc2 = nn.Linear(100, 4)

    def forward(self, xs, verbose=False):
      x = np.expand_dims(xs, axis=1)
      x = torch.Tensor(x)
      x = self.conv1(x)
      x - nn.functional.relu(x)
      x = nn.functional.max_pool2d(x, kernel_size=2, stride=2)
      x = self.conv2(x)
      x = nn.functional.relu(x)
      x = nn.functional.max_pool2d(x, kernel_size=2, stride=2)
      x = self.conv3(x)
      x = nn.functional.relu(x)
      x = nn.functional.max_pool2d(x, kernel_size=2, stride=2)
      x = self.conv4(x)
      x = nn.functional.relu(x)
      x = nn.functional.max_pool2d(x, kernel_size=2, stride=2)
      x = x.view(x.size(0), -1)
      x = self.fc1(x)
      x = nn.functional.relu(x)
      x = self.fc2(x)
      return x
```

Please show the model's scheme here:

[14]:
```
display(Image(filename="/content/gdrive/MyDrive/assignment2/CNN_graph.png"))
```



The function `estimate_accuracy` is written for you. Depending on how you set up your model and training, you may need to modify this function.

[16]:
```
def estimate_accuracy(model, data, label, batch_size=100, max_N=100000):
    """
    Estimate the accuracy of the model on the data. To reduce
    computation time, use at most `max_N` elements of `data` to
    produce the estimate.
    """
    model.eval()
    correct = 0
    N = 0
    for i in range(0, data.shape[0], batch_size):
    # get a batch of data
      xt, st = get_batch(data, label, i, i + batch_size, onehot=True) # changed␣
↪onhot to true
    # forward pass prediction
      y = model(torch.Tensor(xt))
      y = y.detach().numpy() # convert the PyTorch tensor => numpy array
      pred = np.argmax(y, axis=1)
      true = np.argmax(st, axis=1)
      #changed the for loop - for some reason it sometimes fails
      correct += np.count_nonzero(pred == true)

      N += st.shape[0]



      if N > max_N:
        break
    return correct / N
```

The following function `get_batch` will take as input the whole dataset and output a single batch

for the training. The output size of the batch is explained below.

[17]:
```
def get_batch(data, label, range_min, range_max, onehot):
    """
    Convert one batch of data into input and output
    data and return the training data (xt, st) where:
     - `xt` is an numpy array of one-hot vectors of shape [batch_size, 128, 128]
     - `st` is either
            - a numpy array of shape [batch_size, 4] if onehot is True,
            - a numpy array of shape [batch_size] containing indicies otherwise

    Preconditions:
     - `data` is a numpy array of shape [N, 128, 128] produced by a call
        to `process_data`
     - range_max > range_min
    """
    xt = data[range_min:range_max]
    st = label[range_min:range_max]
    st = convert_class_to_number(st)
    if onehot:
        st = create_onehot(st).reshape(-1, 4)
    else: # If it didnt - it dosent return numpy array and cause failure - this␣
↪should fix it
     st=np.array(st)

    return xt, st
```

1.3 Question 3. Training (34%)

Now, we will write the functions required to train the PyTorch models using the Adam optimizer and the cross entropy loss.

Our task is a multi-class classification problem. Therefore, we will use a one-hot vector to represent our target.

1.3.1 Part (a) – 15%

Complete the function `train_model`, and use it to train your PyTorch MLP and CNN models.

Plot the learning curve using the `plot_learning_curve` function provided to you, and include your plot in your PDF submission.

It is also recommended to checkpoint your model (save a copy) after every epoch.

[33]:
```
def train_model(model,
                train_data=train_data,
                train_label=train_labels,
                validation_data=valid_data,
                validation_label=valid_labels,
```

```
                batch_size=100,
                learning_rate=0.001,
                weight_decay=0,
                max_iters=1000,
                checkpoint_path=None):

    """
    Train the PyTorch model on the dataset `train_data`, reporting
    the validation accuracy on `validation_data`, for `max_iters`
    iteration.

    If you want to **checkpoint** your model weights (i.e. save the
    model weights to Google Drive), then the parameter
    `checkpoint_path` should be a string path with `{}` to be replaced
    by the iteration count:

    For example, calling

    >>> train_model(model, ...,
            checkpoint_path = '/content/gdrive/My Drive/assignment2/mlp/ckpt-{}.
↪pk')

    will save the model parameters in Google Drive every 100 iterations.
    You will have to make sure that the path exists (i.e. you'll need to create
    the folder Intro_to_Deep_Learning, mlp or cnn, etc...). Your Google Drive␣
↪will be populated with files:

    - /content/gdrive/My Drive/assignment2/mlp/ckpt-500.pk
    - /content/gdrive/My Drive/assignment2/cnn/ckpt-1000.pk
    - ...

    To load the weights at a later time, you can run:

    >>> model.load_state_dict(torch.load('/content/gdrive/My Drive/assignment2/
↪mlp/ckpt-500.pk'))

    This function returns the training loss, and the training/validation␣
↪accuracy,
    which we can use to plot the learning curve.
    """
    criterion = nn.CrossEntropyLoss()
    optimizer = optim.Adam(model.parameters(),
                           lr=learning_rate,
                           weight_decay=weight_decay)

    iters, losses = [], []
    iters_sub, train_accs, val_accs  = [], [] ,[]
```

```
    # Determine check_interval based on model type
    if isinstance(model, PyTorchMLP):
        check_interval = 100
    elif isinstance(model, CNNChannel):
        check_interval = 10
    else:
        check_interval = 100  # Default value or another appropriate number

    n = 0  # the number of iterations
    while True:
      reindex = np.random.permutation(len(train_data))
      train_data = train_data[reindex]
      train_label = train_label[reindex]

      for i in range(0, train_data.shape[0], batch_size):
          if (i + batch_size) > train_data.shape[0]:
              break
          model.train()
          # get the input and targets of a minibatch
          xt, st = get_batch(train_data, train_label, i, i + batch_size,␣
↪onehot=False)
          # convert from numpy arrays to PyTorch tensors
          xt = torch.Tensor(xt)
          st = torch.Tensor(st)

          zs = model(xt)                  # compute prediction logit
          loss =criterion(zs,st.long())  # compute the total loss
          optimizer.zero_grad()           # a clean up step for PyTorch
          loss.backward()                 # compute updates for each parameter
          optimizer.step()                # make the updates for each parameter

          # save the current training information
          iters.append(n)
          losses.append(float(loss)/batch_size)  # compute *average* loss

          if n % check_interval == 0:
              iters_sub.append(n)
              train_cost = float(loss.detach().numpy())
              train_acc = estimate_accuracy(model, train_data, train_label)
              train_accs.append(train_acc)
              val_acc = estimate_accuracy(model, validation_data,␣
↪validation_label)
              val_accs.append(val_acc)
              print("Iter %d. [Val Acc %.0f%%] [Train Acc %.0f%%, Loss %f]" % (
                    n, val_acc * 100, train_acc * 100, train_cost))
```

```
            if (checkpoint_path is not None) and n > 0:
                torch.save(model.state_dict(), checkpoint_path.format(n))

        # increment the iteration number
        n += 1

        if n > max_iters:
            return iters, losses, iters_sub, train_accs, val_accs


def plot_learning_curve(ax_loss, ax_acc, iters, losses, iters_sub, train_accs,␣
↪val_accs, label=None):
    """
    Plot the learning curve on given axes.
    """
    # Plot loss curve
    ax_loss.plot(iters, losses, label=label)
    ax_loss.set_xlabel("Iterations")
    ax_loss.set_ylabel("Loss")
    ax_loss.set_title("Loss Curve")

    # Plot training and validation accuracy curves
    ax_acc.plot(iters_sub, train_accs, label=f'Train - {label}')
    ax_acc.plot(iters_sub, val_accs, label=f'Validation - {label}')
    ax_acc.set_xlabel("Iterations")
    ax_acc.set_ylabel("Accuracy")
    ax_acc.set_title("Accuracy Curve")
```

1.3.2 Part (b) – 15%

Train your models from Questions 2(a) and 2(b). Change the values of a few hyperparameters, including the learning rate, batch size, choice of n and the kernel size in the CNN model, choice of *num_hidden* in the MLP model. You do not need to check all values for all hyperparameters. Instead, try to make significant changes to see how each change affects your scores (try to start with finding a reasonable learning rate for each network, then start changing the other parameters).

In this section, explain how you tuned your hyperparameters.

Write your explanation here: We began by identifying an appropriate learning rate for each network by observing significant improvements in the accuracy graphs.

After determining a suitable learning rate, we constructed loops for both models to experiment with different combinations of other hyperparameters specific to each model. It's important to note that we reduced the number of iterations during testing to monitor the learning curve. Once we select the optimal hyperparameters, we plan to retrain our models with more iterations.

MLP Model: We initially set the learning rate (lr) to 0.001, which showed promising progress but resulted in a noisy learning curve. To mitigate the noise, we decreased the learning rate to 0.0005; however, this slowed down the progress. Next, we increased the number of hidden nodes to

400. This adjustment led to smoother loss and accuracy curves and significantly faster convergence, albeit at the cost of increased computational time. To balance performance and efficiency, we tested lower values for the hidden nodes and experimented with different batch sizes. After running several iterations, we settled on 200 hidden nodes and a batch size of 128. This combination achieved the lowest loss among all samples and maintained high accuracy with a rapid convergence rate.

CNN Model: Similar to the MLP model, we started with a learning rate of 0.001 for the CNN, which yielded a high convergence rate but with noticeable noise. Reducing the learning rate to 0.0001 resulted in too slow convergence, so we ultimately chose a learning rate of 0.0005. We then tested various batch sizes (16, 64, 128) while keeping the kernel size and the number of channels (n) constant. We observed that smaller batch sizes produced noisier loss curves with higher values. Among the larger batch sizes, both 64 and 128 yielded similar final values, but batch size 128 was less noisy. However, the accuracy graph indicated that batch size 64 converged faster than 128, leading us to choose a batch size of 64.

After selecting the batch size and learning rate, we ran a loop to iterate over different values of kernel size and the number of channels (n). Due to computational constraints, we limited the iterations to 100. Upon reviewing the graphs, we decided on a kernel size of 5 and n equal to 8. Both the accuracy and loss metrics demonstrated good progress and convergence within these 100 iterations, suggesting further improvements with full training.

It's important to mention that we intentionally used a limited number of iterations to quickly observe the learning curves and reduce computation time. After finalizing the optimal hyperparameters, we will proceed to train our models using the required number of iterations.

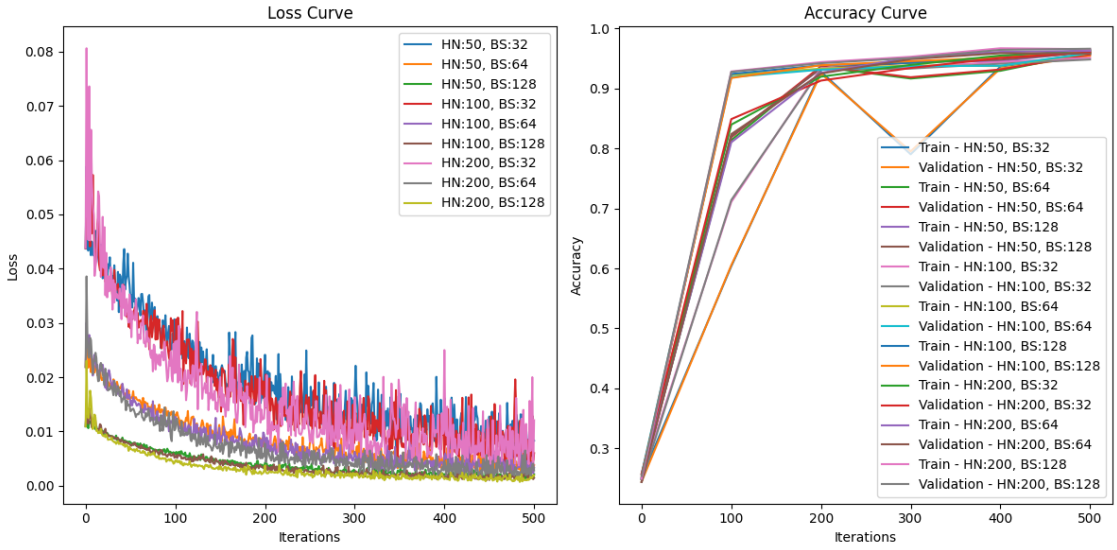
Include the training curves for the two models:

[34]:
```
learning_rate = 0.0001
hidden_nodes_lst = [50, 100, 200]  # Example values for the number of hidden␣
 ↪nodes
batch_size_lst = [32, 64, 128]     # Batch sizes

fig, (ax_loss, ax_acc) = plt.subplots(1, 2, figsize=(12, 6))

for num_hidden_nodes in hidden_nodes_lst:
    for batch_size in batch_size_lst:
        print(f"Training MLP with num_hidden_nodes={num_hidden_nodes} and␣
 ↪batch_size={batch_size}")

        pytorch_mlp = PyTorchMLP(num_hidden=num_hidden_nodes)
        learning_curve_info_mlp = train_model(
            model=pytorch_mlp,
            train_data=train_data,
            train_label=train_labels,
            validation_data=valid_data,
            validation_label=valid_labels,
```

```
            batch_size=batch_size,
            learning_rate=learning_rate,
            max_iters=500
        )

        iters, losses, iters_sub, train_accs, val_accs = learning_curve_info_mlp

        label = f'HN:{num_hidden_nodes}, BS:{batch_size}'

        plot_learning_curve(
            ax_loss, ax_acc,
            iters, losses,
            iters_sub, train_accs, val_accs,
            label=label
        )

ax_loss.legend()
ax_acc.legend()
plt.tight_layout()
plt.show()
```

```
Training MLP with num_hidden_nodes=50 and batch_size=32
Iter 0. [Val Acc 24%] [Train Acc 25%, Loss 1.398621]
Iter 100. [Val Acc 61%] [Train Acc 60%, Loss 0.827277]
Iter 200. [Val Acc 92%] [Train Acc 93%, Loss 0.772538]
Iter 300. [Val Acc 80%] [Train Acc 79%, Loss 0.524639]
Iter 400. [Val Acc 93%] [Train Acc 94%, Loss 0.516730]
Iter 500. [Val Acc 95%] [Train Acc 96%, Loss 0.266226]
Training MLP with num_hidden_nodes=50 and batch_size=64
Iter 0. [Val Acc 26%] [Train Acc 25%, Loss 1.404392]
Iter 100. [Val Acc 82%] [Train Acc 81%, Loss 0.807579]
Iter 200. [Val Acc 94%] [Train Acc 94%, Loss 0.483453]
Iter 300. [Val Acc 92%] [Train Acc 92%, Loss 0.386503]
Iter 400. [Val Acc 93%] [Train Acc 93%, Loss 0.314196]
Iter 500. [Val Acc 96%] [Train Acc 96%, Loss 0.242584]
Training MLP with num_hidden_nodes=50 and batch_size=128
Iter 0. [Val Acc 24%] [Train Acc 25%, Loss 1.399838]
Iter 100. [Val Acc 82%] [Train Acc 82%, Loss 0.749584]
Iter 200. [Val Acc 93%] [Train Acc 93%, Loss 0.545729]
Iter 300. [Val Acc 94%] [Train Acc 94%, Loss 0.356550]
Iter 400. [Val Acc 95%] [Train Acc 95%, Loss 0.266368]
Iter 500. [Val Acc 96%] [Train Acc 96%, Loss 0.272809]
Training MLP with num_hidden_nodes=100 and batch_size=32
Iter 0. [Val Acc 24%] [Train Acc 25%, Loss 1.400571]
Iter 100. [Val Acc 71%] [Train Acc 71%, Loss 0.715452]
Iter 200. [Val Acc 93%] [Train Acc 93%, Loss 0.459936]
Iter 300. [Val Acc 93%] [Train Acc 94%, Loss 0.392687]
```

```
Iter 400. [Val Acc 94%] [Train Acc 94%, Loss 0.152651]
Iter 500. [Val Acc 95%] [Train Acc 95%, Loss 0.192266]
Training MLP with num_hidden_nodes=100 and batch_size=64
Iter 0. [Val Acc 26%] [Train Acc 25%, Loss 1.397719]
Iter 100. [Val Acc 92%] [Train Acc 92%, Loss 0.814555]
Iter 200. [Val Acc 93%] [Train Acc 93%, Loss 0.502084]
Iter 300. [Val Acc 94%] [Train Acc 94%, Loss 0.400956]
Iter 400. [Val Acc 94%] [Train Acc 94%, Loss 0.346847]
Iter 500. [Val Acc 96%] [Train Acc 96%, Loss 0.211641]
Training MLP with num_hidden_nodes=100 and batch_size=128
Iter 0. [Val Acc 25%] [Train Acc 25%, Loss 1.410177]
Iter 100. [Val Acc 92%] [Train Acc 92%, Loss 0.679809]
Iter 200. [Val Acc 94%] [Train Acc 94%, Loss 0.459675]
Iter 300. [Val Acc 95%] [Train Acc 94%, Loss 0.254839]
Iter 400. [Val Acc 95%] [Train Acc 95%, Loss 0.194135]
Iter 500. [Val Acc 96%] [Train Acc 96%, Loss 0.186911]
Training MLP with num_hidden_nodes=200 and batch_size=32
Iter 0. [Val Acc 26%] [Train Acc 25%, Loss 1.402195]
Iter 100. [Val Acc 85%] [Train Acc 84%, Loss 0.850924]
Iter 200. [Val Acc 91%] [Train Acc 92%, Loss 0.515126]
Iter 300. [Val Acc 93%] [Train Acc 94%, Loss 0.522147]
Iter 400. [Val Acc 95%] [Train Acc 95%, Loss 0.800250]
Iter 500. [Val Acc 96%] [Train Acc 96%, Loss 0.386532]
Training MLP with num_hidden_nodes=200 and batch_size=64
Iter 0. [Val Acc 24%] [Train Acc 25%, Loss 1.401553]
Iter 100. [Val Acc 82%] [Train Acc 81%, Loss 0.668797]
Iter 200. [Val Acc 92%] [Train Acc 93%, Loss 0.496641]
Iter 300. [Val Acc 95%] [Train Acc 95%, Loss 0.223450]
Iter 400. [Val Acc 96%] [Train Acc 96%, Loss 0.290873]
Iter 500. [Val Acc 96%] [Train Acc 96%, Loss 0.181706]
Training MLP with num_hidden_nodes=200 and batch_size=128
Iter 0. [Val Acc 25%] [Train Acc 25%, Loss 1.400590]
Iter 100. [Val Acc 93%] [Train Acc 93%, Loss 0.597590]
Iter 200. [Val Acc 94%] [Train Acc 94%, Loss 0.480033]
Iter 300. [Val Acc 95%] [Train Acc 95%, Loss 0.288129]
Iter 400. [Val Acc 96%] [Train Acc 97%, Loss 0.155244]
Iter 500. [Val Acc 97%] [Train Acc 97%, Loss 0.242921]
```



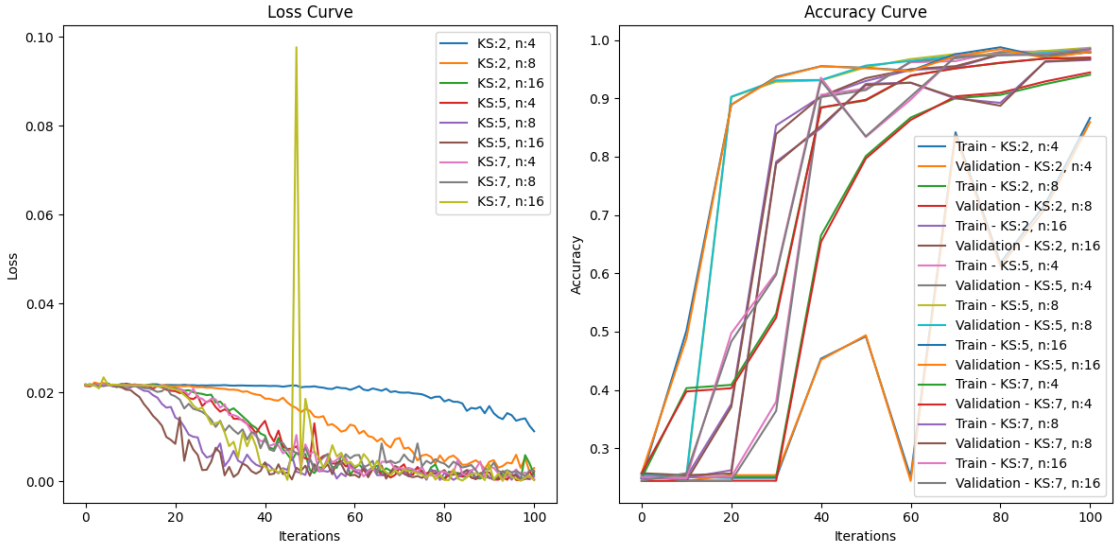
[37]:
```
learning_rate = 0.0005
batch_size = 64
kernel_sizes = [2, 5, 7]
channels = [4, 8, 16]

fig, (ax_loss, ax_acc) = plt.subplots(1, 2, figsize=(12, 6))

for kernel_size in kernel_sizes:
    for n in channels:
        print(f"Training CNN with kernel size={kernel_size} and n={n}")

        # Initialize and train the model
        model_cnn_ch = CNNChannel(n=n, kernel_size=kernel_size)
        learning_curve_info_CNN = train_model(
            model=model_cnn_ch,
            train_data=train_data,
            train_label=train_labels,
            validation_data=valid_data,
            validation_label=valid_labels,
            batch_size=batch_size,
            learning_rate=learning_rate,
            max_iters=100
        )

        # Unpack learning curve data
        iters, losses, iters_sub, train_accs, val_accs = learning_curve_info_CNN

        # Generate a label for the current hyperparameters
```

```
        label = f'KS:{kernel_size}, n:{n}'

        # Plot the learning curves using the function
        plot_learning_curve(
            ax_loss, ax_acc,
            iters, losses,
            iters_sub, train_accs, val_accs,
            label=label
        )

ax_loss.legend()
ax_acc.legend()
plt.tight_layout()
plt.show()
```

```
Training CNN with kernel size=2 and n=4
Iter 0. [Val Acc 24%] [Train Acc 25%, Loss 1.392725]
Iter 10. [Val Acc 24%] [Train Acc 25%, Loss 1.393274]
Iter 20. [Val Acc 25%] [Train Acc 25%, Loss 1.385805]
Iter 30. [Val Acc 25%] [Train Acc 25%, Loss 1.382256]
Iter 40. [Val Acc 45%] [Train Acc 45%, Loss 1.375375]
Iter 50. [Val Acc 49%] [Train Acc 49%, Loss 1.361258]
Iter 60. [Val Acc 24%] [Train Acc 25%, Loss 1.340485]
Iter 70. [Val Acc 84%] [Train Acc 84%, Loss 1.278836]
Iter 80. [Val Acc 61%] [Train Acc 62%, Loss 1.150258]
Iter 90. [Val Acc 71%] [Train Acc 72%, Loss 1.083159]
Iter 100. [Val Acc 86%] [Train Acc 87%, Loss 0.720275]
Training CNN with kernel size=2 and n=8
Iter 0. [Val Acc 24%] [Train Acc 25%, Loss 1.379974]
Iter 10. [Val Acc 24%] [Train Acc 25%, Loss 1.382723]
Iter 20. [Val Acc 24%] [Train Acc 25%, Loss 1.369261]
Iter 30. [Val Acc 24%] [Train Acc 25%, Loss 1.333922]
Iter 40. [Val Acc 65%] [Train Acc 66%, Loss 1.190770]
Iter 50. [Val Acc 80%] [Train Acc 80%, Loss 1.015863]
Iter 60. [Val Acc 86%] [Train Acc 87%, Loss 0.796715]
Iter 70. [Val Acc 90%] [Train Acc 90%, Loss 0.623686]
Iter 80. [Val Acc 91%] [Train Acc 91%, Loss 0.283417]
Iter 90. [Val Acc 93%] [Train Acc 92%, Loss 0.279882]
Iter 100. [Val Acc 94%] [Train Acc 94%, Loss 0.189322]
Training CNN with kernel size=2 and n=16
Iter 0. [Val Acc 26%] [Train Acc 25%, Loss 1.386444]
Iter 10. [Val Acc 24%] [Train Acc 25%, Loss 1.394763]
Iter 20. [Val Acc 37%] [Train Acc 38%, Loss 1.343266]
Iter 30. [Val Acc 84%] [Train Acc 85%, Loss 1.148689]
Iter 40. [Val Acc 90%] [Train Acc 90%, Loss 0.657491]
Iter 50. [Val Acc 93%] [Train Acc 93%, Loss 0.162801]
Iter 60. [Val Acc 95%] [Train Acc 95%, Loss 0.188790]
```

```
Iter 70. [Val Acc 96%] [Train Acc 95%, Loss 0.089640]
Iter 80. [Val Acc 98%] [Train Acc 98%, Loss 0.035765]
Iter 90. [Val Acc 98%] [Train Acc 98%, Loss 0.047194]
Iter 100. [Val Acc 98%] [Train Acc 98%, Loss 0.099055]
Training CNN with kernel size=5 and n=4
Iter 0. [Val Acc 25%] [Train Acc 25%, Loss 1.372792]
Iter 10. [Val Acc 24%] [Train Acc 25%, Loss 1.388950]
Iter 20. [Val Acc 24%] [Train Acc 25%, Loss 1.298639]
Iter 30. [Val Acc 36%] [Train Acc 38%, Loss 1.002961]
Iter 40. [Val Acc 90%] [Train Acc 91%, Loss 0.870444]
Iter 50. [Val Acc 91%] [Train Acc 92%, Loss 0.313621]
Iter 60. [Val Acc 96%] [Train Acc 96%, Loss 0.140700]
Iter 70. [Val Acc 97%] [Train Acc 96%, Loss 0.193319]
Iter 80. [Val Acc 98%] [Train Acc 98%, Loss 0.024324]
Iter 90. [Val Acc 97%] [Train Acc 98%, Loss 0.053237]
Iter 100. [Val Acc 98%] [Train Acc 98%, Loss 0.021971]
Training CNN with kernel size=5 and n=8
Iter 0. [Val Acc 25%] [Train Acc 25%, Loss 1.387485]
Iter 10. [Val Acc 25%] [Train Acc 25%, Loss 1.351690]
Iter 20. [Val Acc 90%] [Train Acc 90%, Loss 0.918735]
Iter 30. [Val Acc 93%] [Train Acc 93%, Loss 0.544744]
Iter 40. [Val Acc 93%] [Train Acc 93%, Loss 0.264122]
Iter 50. [Val Acc 96%] [Train Acc 95%, Loss 0.319423]
Iter 60. [Val Acc 96%] [Train Acc 97%, Loss 0.165980]
Iter 70. [Val Acc 97%] [Train Acc 98%, Loss 0.086457]
Iter 80. [Val Acc 97%] [Train Acc 98%, Loss 0.120776]
Iter 90. [Val Acc 98%] [Train Acc 98%, Loss 0.054520]
Iter 100. [Val Acc 99%] [Train Acc 99%, Loss 0.070178]
Training CNN with kernel size=5 and n=16
Iter 0. [Val Acc 26%] [Train Acc 25%, Loss 1.385570]
Iter 10. [Val Acc 49%] [Train Acc 50%, Loss 1.314646]
Iter 20. [Val Acc 89%] [Train Acc 89%, Loss 0.538790]
Iter 30. [Val Acc 94%] [Train Acc 94%, Loss 0.412555]
Iter 40. [Val Acc 95%] [Train Acc 96%, Loss 0.272588]
Iter 50. [Val Acc 95%] [Train Acc 95%, Loss 0.079252]
Iter 60. [Val Acc 95%] [Train Acc 95%, Loss 0.093308]
Iter 70. [Val Acc 97%] [Train Acc 98%, Loss 0.186897]
Iter 80. [Val Acc 98%] [Train Acc 99%, Loss 0.160919]
Iter 90. [Val Acc 97%] [Train Acc 97%, Loss 0.126125]
Iter 100. [Val Acc 98%] [Train Acc 98%, Loss 0.136474]
Training CNN with kernel size=7 and n=4
Iter 0. [Val Acc 26%] [Train Acc 25%, Loss 1.393521]
Iter 10. [Val Acc 40%] [Train Acc 40%, Loss 1.383567]
Iter 20. [Val Acc 40%] [Train Acc 41%, Loss 1.327650]
Iter 30. [Val Acc 52%] [Train Acc 53%, Loss 1.059683]
Iter 40. [Val Acc 88%] [Train Acc 88%, Loss 0.578982]
Iter 50. [Val Acc 90%] [Train Acc 90%, Loss 0.395165]
Iter 60. [Val Acc 94%] [Train Acc 94%, Loss 0.161353]
```

```
Iter 70. [Val Acc 95%] [Train Acc 95%, Loss 0.190489]
Iter 80. [Val Acc 96%] [Train Acc 96%, Loss 0.136443]
Iter 90. [Val Acc 97%] [Train Acc 97%, Loss 0.205850]
Iter 100. [Val Acc 97%] [Train Acc 97%, Loss 0.026317]
Training CNN with kernel size=7 and n=8
Iter 0. [Val Acc 26%] [Train Acc 25%, Loss 1.395234]
Iter 10. [Val Acc 25%] [Train Acc 25%, Loss 1.378872]
Iter 20. [Val Acc 26%] [Train Acc 26%, Loss 1.222944]
Iter 30. [Val Acc 79%] [Train Acc 79%, Loss 0.871023]
Iter 40. [Val Acc 85%] [Train Acc 85%, Loss 0.621166]
Iter 50. [Val Acc 92%] [Train Acc 92%, Loss 0.327272]
Iter 60. [Val Acc 93%] [Train Acc 93%, Loss 0.373646]
Iter 70. [Val Acc 90%] [Train Acc 90%, Loss 0.283271]
Iter 80. [Val Acc 89%] [Train Acc 89%, Loss 0.253796]
Iter 90. [Val Acc 96%] [Train Acc 96%, Loss 0.174897]
Iter 100. [Val Acc 97%] [Train Acc 97%, Loss 0.151268]
Training CNN with kernel size=7 and n=16
Iter 0. [Val Acc 24%] [Train Acc 25%, Loss 1.385990]
Iter 10. [Val Acc 26%] [Train Acc 25%, Loss 1.382765]
Iter 20. [Val Acc 48%] [Train Acc 50%, Loss 1.322574]
Iter 30. [Val Acc 60%] [Train Acc 60%, Loss 0.856595]
Iter 40. [Val Acc 93%] [Train Acc 94%, Loss 0.213371]
Iter 50. [Val Acc 83%] [Train Acc 83%, Loss 0.879225]
Iter 60. [Val Acc 90%] [Train Acc 90%, Loss 0.212525]
Iter 70. [Val Acc 97%] [Train Acc 97%, Loss 0.106548]
Iter 80. [Val Acc 97%] [Train Acc 97%, Loss 0.019404]
Iter 90. [Val Acc 97%] [Train Acc 98%, Loss 0.245301]
Iter 100. [Val Acc 98%] [Train Acc 99%, Loss 0.017809]
```

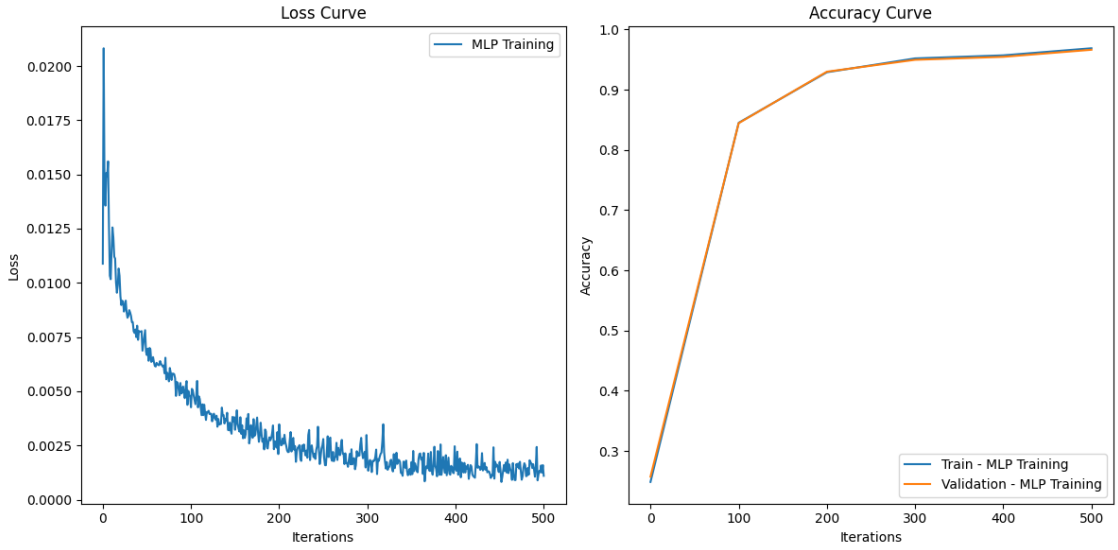


1.3.3 Part (c) – 4%

Include your training curves for the **best** models from each MLP and CNN. These are the models that you will use in Question 4.

[39]:
```
# Our best model for MLP was: num_hidden=200, batch_size=128, learning_rate=0.
 ↪0001

pytorch_mlp = PyTorchMLP(200)
print('The model we used here is MLP channel model')
learning_curve_info_MLP = train_model(
    pytorch_mlp,
    train_data=train_data,
    train_label=train_labels,
    validation_data=valid_data,
    validation_label=valid_labels,
    batch_size=128,
    learning_rate=0.0001,  # Note: Updated to match your best model's learning␣
 ↪rate
    weight_decay=0,
    max_iters=500,
    checkpoint_path='/content/gdrive/My Drive/assignment2/MLP/ckpt-{}.pk'
)

# Create the figure and axes for plotting
fig, (ax_loss, ax_acc) = plt.subplots(1, 2, figsize=(12, 6))

plot_learning_curve(
    ax_loss, ax_acc,
    *learning_curve_info_MLP,
    label='MLP Training'
)

# Add legends and display the plot
ax_loss.legend()
ax_acc.legend()
plt.tight_layout()
plt.show()
```

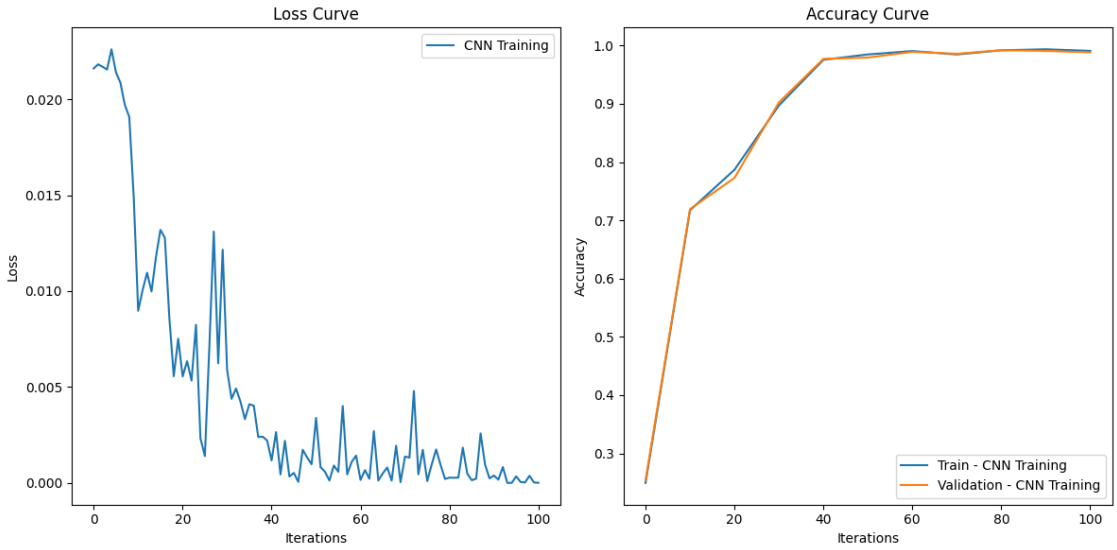
```
The model we used here is MLP channel model
Iter 0. [Val Acc 26%] [Train Acc 25%, Loss 1.393032]
Iter 100. [Val Acc 84%] [Train Acc 85%, Loss 0.545049]
Iter 200. [Val Acc 93%] [Train Acc 93%, Loss 0.444678]
Iter 300. [Val Acc 95%] [Train Acc 95%, Loss 0.233890]
Iter 400. [Val Acc 95%] [Train Acc 96%, Loss 0.183370]
Iter 500. [Val Acc 97%] [Train Acc 97%, Loss 0.140557]
```



[42]:
```
#Our best model for cCNN was: n=8, kernal_size=5, batch_size=64 learning_rate=0.
 ↪0005
model_cnn_ch = CNNChannel(n=8, kernel_size=5)
learning_curve_info_CNN = train_model(model_cnn_ch,
                                      train_data=train_data, # Add train_data
                                      train_label=train_labels, # Add train_labels
                                      validation_data=valid_data, # Add␣
 ↪validation_data
                                      validation_label=valid_labels, # Add␣
 ↪validation_labels
                                      batch_size=64,
                                      learning_rate=0.005,
                                      weight_decay=0,
                                      max_iters=100,
                                      checkpoint_path='/content/gdrive/My Drive/
 ↪assignment2/CNN/ckpt-{}.pk')
# Create the figure and axes for plotting
fig, (ax_loss, ax_acc) = plt.subplots(1, 2, figsize=(12, 6))

plot_learning_curve(
    ax_loss, ax_acc,
    *learning_curve_info_CNN,
    label='CNN Training'
)

# Add legends and display the plot
ax_loss.legend()
ax_acc.legend()
```

```
plt.tight_layout()
plt.show()
```

```
Iter 0. [Val Acc 25%] [Train Acc 25%, Loss 1.382880]
Iter 10. [Val Acc 72%] [Train Acc 72%, Loss 0.574636]
Iter 20. [Val Acc 77%] [Train Acc 79%, Loss 0.355470]
Iter 30. [Val Acc 90%] [Train Acc 90%, Loss 0.376842]
Iter 40. [Val Acc 98%] [Train Acc 98%, Loss 0.075153]
Iter 50. [Val Acc 98%] [Train Acc 98%, Loss 0.217074]
Iter 60. [Val Acc 99%] [Train Acc 99%, Loss 0.010178]
Iter 70. [Val Acc 99%] [Train Acc 98%, Loss 0.088297]
Iter 80. [Val Acc 99%] [Train Acc 99%, Loss 0.017961]
Iter 90. [Val Acc 99%] [Train Acc 99%, Loss 0.024980]
Iter 100. [Val Acc 99%] [Train Acc 99%, Loss 0.000795]
```



1.4 Question 4. Testing (21%)

1.4.1 Part (a) – 7%

Report the test accuracies of your **single best** model, separately for the test set. Do this by choosing the model architecture that produces the best validation accuracy. For instance, if your model attained the best validation accuracy in epoch 10, then the weights at epoch 10 is what you should be using to report the test accuracy.

[43]:
```
# Make sure to include the test accuracy in your report!!
# Write your code here:
model = CNNChannel(n=8, kernel_size=5)
model.load_state_dict(torch.load('/content/gdrive/My Drive/assignment2/CNN/
  ↪ckpt-100.pk')) #Best validation accuracy in this epoch
```

```
CNN_accuracy=estimate_accuracy(model,test_data,test_labels,np.
 ↪shape(test_labels)[0])
print(f"Test accuracy for CNN model is: {CNN_accuracy*100:.3f}%")
```

```
<ipython-input-43-b35d95d3d36c>:4: FutureWarning: You are using `torch.load`
with `weights_only=False` (the current default value), which uses the default
pickle module implicitly. It is possible to construct malicious pickle data
which will execute arbitrary code during unpickling (See
https://github.com/pytorch/pytorch/blob/main/SECURITY.md#untrusted-models for
more details). In a future release, the default value for `weights_only` will be
flipped to `True`. This limits the functions that could be executed during
unpickling. Arbitrary objects will no longer be allowed to be loaded via this
mode unless they are explicitly allowlisted by the user via
`torch.serialization.add_safe_globals`. We recommend you start setting
`weights_only=True` for any use case where you don't have full control of the
loaded file. Please open an issue on GitHub for any issues related to this
experimental feature.
  model.load_state_dict(torch.load('/content/gdrive/My
Drive/assignment2/CNN/ckpt-100.pk')) #Best validation accuracy in this epoch
```

```
Test accuracy for CNN model is: 99.083%
```

1.4.2 Part (b) – 7%

For each model, display one of the signal spectrograms that your model correctly classified, and one of the signal spectrograms that your model classified incorrectly.

[51]:
```
# Prepare test data
X_test = test_data
Y_test_onehot = create_onehot(convert_class_to_number(test_labels))

# Function to load a model and make predictions
def load_model_and_predict(model_class, checkpoint_path, X):
    model = model_class
    model.load_state_dict(torch.load(checkpoint_path))
    model.eval()
    with torch.no_grad():
        predictions = model(torch.Tensor(X)).numpy()
    predicted_labels = np.argmax(predictions, axis=1)
    return predicted_labels

# Function to find correct and incorrect prediction indices
def get_prediction_indices(predicted_labels, true_labels):
    correct_indices = np.where(predicted_labels == true_labels)[0]
    incorrect_indices = np.where(predicted_labels != true_labels)[0]
    return correct_indices, incorrect_indices
```

```
# Function to plot predictions
def plot_prediction(index, model_name, correctness, X, true_labels,␣
 ↪predicted_labels):
    plt.figure(figsize=(8, 6))
    plt.suptitle(f"{correctness} Prediction {model_name}", x=0.5, y=1.05,␣
 ↪fontsize=16)
    plt.title(f"True Label: {test_labels[index]}\nPredicted Label:␣
 ↪{num2labels_vocab[predicted_labels[index]]}")
    plt.imshow(X[index])
    plt.axis('off')
    plt.tight_layout()
    plt.show()

# Convert one-hot encoded labels to numeric labels
true_labels = np.argmax(Y_test_onehot, axis=1)

# Models information
models_info = [
    {
        'name': 'MLP',
        'model_class': PyTorchMLP(200),
        'checkpoint_path': '/content/gdrive/My Drive/assignment2/MLP/ckpt-500.
 ↪pk'
    },
    {
        'name': 'CNN',
        'model_class': CNNChannel(n=8, kernel_size=5),
        'checkpoint_path': '/content/gdrive/My Drive/assignment2/CNN/ckpt-100.
 ↪pk'
    }
]

# Process each model
for info in models_info:
    model_name = info['name']
    model_class = info['model_class']
    checkpoint_path = info['checkpoint_path']

    # Load model and get predictions
    predicted_labels = load_model_and_predict(model_class, checkpoint_path,␣
 ↪X_test)

    # Get indices of correct and incorrect predictions
    correct_indices, incorrect_indices =␣
 ↪get_prediction_indices(predicted_labels, true_labels)
```

```
    # Plot correct prediction
    if correct_indices.size > 0:
        plot_prediction(correct_indices[0], model_name, "Correct", test_data,␣
↪test_labels, predicted_labels)
    else:
        print(f"No correct predictions found for {model_name}.")

    # Plot incorrect prediction
    if incorrect_indices.size > 0:
        plot_prediction(incorrect_indices[0], model_name, "Incorrect",␣
↪test_data, test_labels, predicted_labels)
    else:
        print(f"No incorrect predictions found for {model_name}.")
```

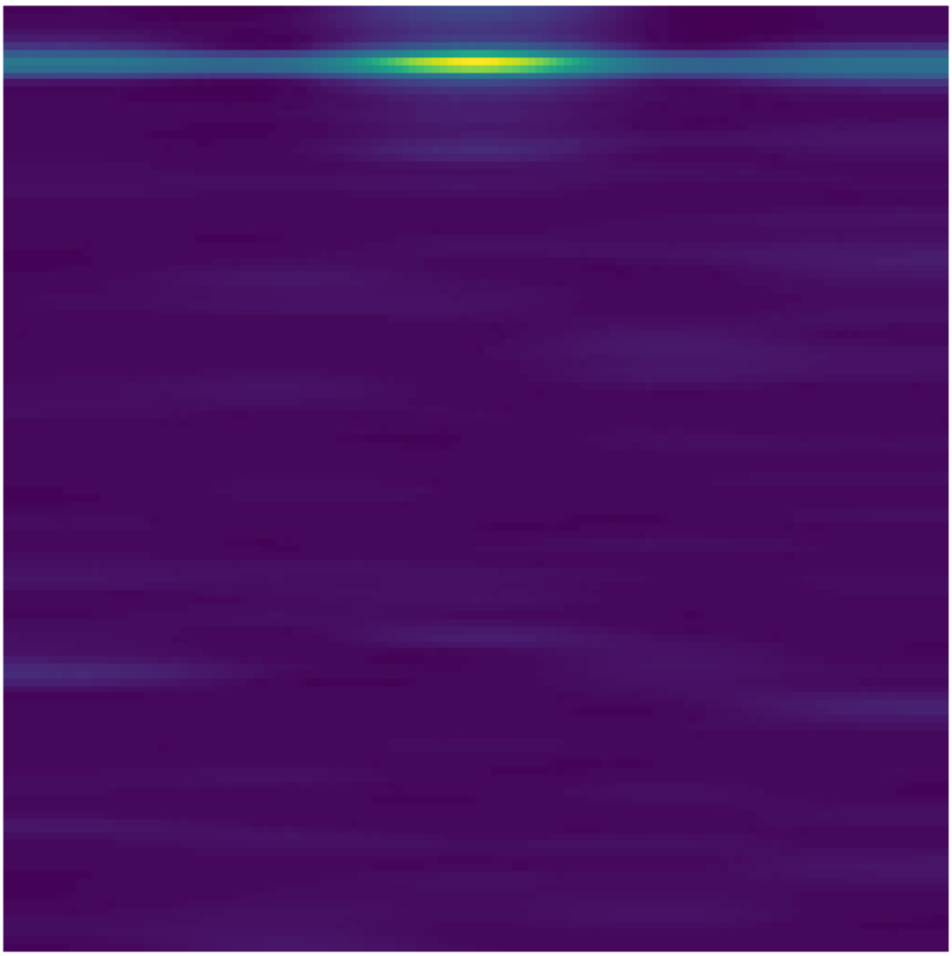
```
<ipython-input-51-ba668386014e>:8: FutureWarning: You are using `torch.load`
with `weights_only=False` (the current default value), which uses the default
pickle module implicitly. It is possible to construct malicious pickle data
which will execute arbitrary code during unpickling (See
https://github.com/pytorch/pytorch/blob/main/SECURITY.md#untrusted-models for
more details). In a future release, the default value for `weights_only` will be
flipped to `True`. This limits the functions that could be executed during
unpickling. Arbitrary objects will no longer be allowed to be loaded via this
mode unless they are explicitly allowlisted by the user via
`torch.serialization.add_safe_globals`. We recommend you start setting
`weights_only=True` for any use case where you don't have full control of the
loaded file. Please open an issue on GitHub for any issues related to this
experimental feature.
  model.load_state_dict(torch.load(checkpoint_path))
```

Correct Prediction MLP

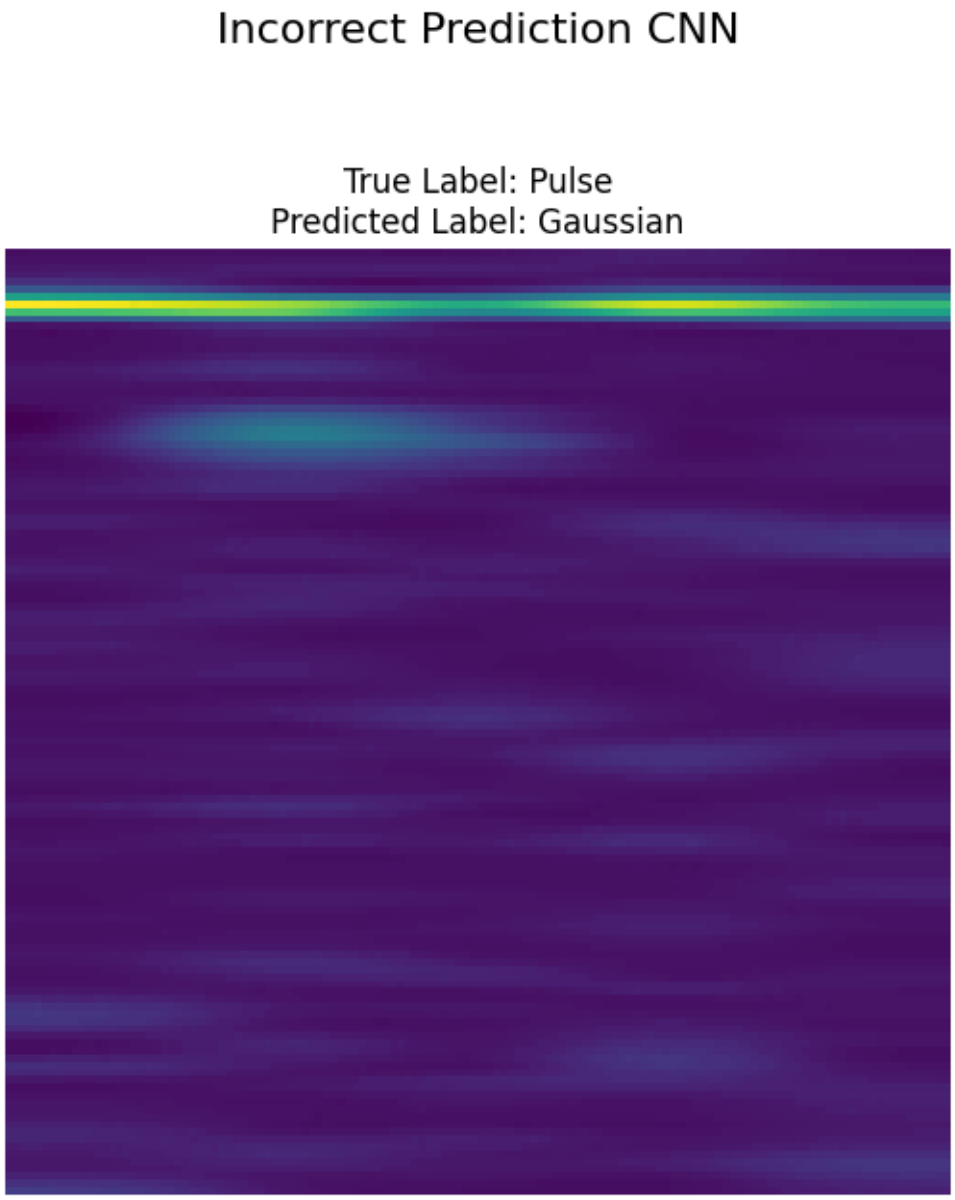
True Label: Gaussian
Predicted Label: Gaussian

Incorrect Prediction MLP

True Label: Gaussian
Predicted Label: ThreeFrequency




Correct Prediction CNN
True Label: Gaussian
Predicted Label: Gaussian



1.4.3 Part (c) – 7%

Compare the capacity, the number of layers, and performance of the two architectures, and discuss the advantages and disadvantages between these architectures.

Will one of these models perform better? Explain why.

Is the architecture choices important in machine learning?

Write your explanation here:

To begin with, the CNN Channel architecture boasts a high capacity due to its proficiency in capturing spatial hierarchies and local patterns. In contrast, the MLP architecture alters the spatial structure of images by concatenating along the height dimension. This modification makes it more challenging for the model to learn and characterize the images because our dataset consists of images where information is embedded not just in individual pixels but also in their neighboring pixels.

Secondly, when it comes to the number of layers, the MLP architecture is considerably simpler, comprising only three layers: input, hidden, and output. The CNN, however, includes convolutional layers, max pooling, a flatten function, and more. We have also learned that deeper layers generally correspond to more specific and complex features, enabling the CNN Channel to extract these detailed image features.

It's important to note that the large and diverse dataset might have contributed to the higher validation accuracy for the MLP model as well.

Additionally, it's worth mentioning that training the CNN Channel required more time and computational resources than the MLP model.

Regarding advantages and disadvantages, the CNN Channel is more effective for image-related tasks because it captures spatial hierarchies and more intricate patterns. Its drawback is the increased computation time and resource consumption. The MLP model's advantage lies in its simplicity and ease of implementation, making it well-suited for structured data. Its disadvantage is the limited capacity to handle spatial structures, as previously mentioned.

In conclusion, choosing the right architecture is crucial in machine learning because it directly impacts a model's ability to learn and generalize from the data. When making a choice, one should consider the strengths and limitations of different architectures, the nature of the data, the complexity of the task, and the available computational resources.

2 PDF export

To export a PDF of the completed notebook, you might find the following helper functions helpful. Here are some resources for additional learning.

- https://nbconvert.readthedocs.io/en/latest/
- https://nbconvert.readthedocs.io/en/latest/install.html#installing-tex

[48]:
```
!sudo apt-get install texlive-xetex texlive-fonts-recommended␣
↪texlive-plain-generic
```

```
Reading package lists… Done
Building dependency tree… Done
Reading state information… Done
The following additional packages will be installed:
  dvisvgm fonts-droid-fallback fonts-lato fonts-lmodern fonts-noto-mono
  fonts-texgyre fonts-urw-base35 libapache-pom-java libcommons-logging-java
```

```
  libcommons-parent-java libfontbox-java libfontenc1 libgs9 libgs9-common
  libidn12 libijs-0.35 libjbig2dec0 libkpathsea6 libpdfbox-java libptexenc1
  libruby3.0 libsynctex2 libteckit0 libtexlua53 libtexluajit2 libwoff1
  libzzip-0-13 lmodern poppler-data preview-latex-style rake ruby
  ruby-net-telnet ruby-rubygems ruby-webrick ruby-xmlrpc ruby3.0
  rubygems-integration t1utils teckit tex-common tex-gyre texlive-base
  texlive-binaries texlive-latex-base texlive-latex-extra
  texlive-latex-recommended texlive-pictures tipa xfonts-encodings
  xfonts-utils
Suggested packages:
  fonts-noto fonts-freefont-otf | fonts-freefont-ttf libavalon-framework-java
  libcommons-logging-java-doc libexcalibur-logkit-java liblog4j1.2-java
  poppler-utils ghostscript fonts-japanese-mincho | fonts-ipafont-mincho
  fonts-japanese-gothic | fonts-ipafont-gothic fonts-arphic-ukai
  fonts-arphic-uming fonts-nanum ri ruby-dev bundler debhelper gv
  | postscript-viewer perl-tk xpdf | pdf-viewer xzdec
  texlive-fonts-recommended-doc texlive-latex-base-doc python3-pygments
  icc-profiles libfile-which-perl libspreadsheet-parseexcel-perl
  texlive-latex-extra-doc texlive-latex-recommended-doc texlive-luatex
  texlive-pstricks dot2tex prerex texlive-pictures-doc vprerex
  default-jre-headless tipa-doc
The following NEW packages will be installed:
  dvisvgm fonts-droid-fallback fonts-lato fonts-lmodern fonts-noto-mono
  fonts-texgyre fonts-urw-base35 libapache-pom-java libcommons-logging-java
  libcommons-parent-java libfontbox-java libfontenc1 libgs9 libgs9-common
  libidn12 libijs-0.35 libjbig2dec0 libkpathsea6 libpdfbox-java libptexenc1
  libruby3.0 libsynctex2 libteckit0 libtexlua53 libtexluajit2 libwoff1
  libzzip-0-13 lmodern poppler-data preview-latex-style rake ruby
  ruby-net-telnet ruby-rubygems ruby-webrick ruby-xmlrpc ruby3.0
  rubygems-integration t1utils teckit tex-common tex-gyre texlive-base
  texlive-binaries texlive-fonts-recommended texlive-latex-base
  texlive-latex-extra texlive-latex-recommended texlive-pictures
  texlive-plain-generic texlive-xetex tipa xfonts-encodings xfonts-utils
0 upgraded, 54 newly installed, 0 to remove and 49 not upgraded.
Need to get 182 MB of archives.
After this operation, 571 MB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu jammy/main amd64 fonts-droid-fallback all
1:6.0.1r16-1.1build1 [1,805 kB]
Get:2 http://archive.ubuntu.com/ubuntu jammy/main amd64 fonts-lato all 2.0-2.1
[2,696 kB]
Get:3 http://archive.ubuntu.com/ubuntu jammy/main amd64 poppler-data all
0.4.11-1 [2,171 kB]
Get:4 http://archive.ubuntu.com/ubuntu jammy/universe amd64 tex-common all 6.17
[33.7 kB]
Get:5 http://archive.ubuntu.com/ubuntu jammy/main amd64 fonts-urw-base35 all
20200910-1 [6,367 kB]
Get:6 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 libgs9-common
all 9.55.0~dfsg1-0ubuntu5.9 [752 kB]
```

```
Get:7 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 libidn12 amd64
1.38-4ubuntu1 [60.0 kB]
Get:8 http://archive.ubuntu.com/ubuntu jammy/main amd64 libijs-0.35 amd64
0.35-15build2 [16.5 kB]
Get:9 http://archive.ubuntu.com/ubuntu jammy/main amd64 libjbig2dec0 amd64
0.19-3build2 [64.7 kB]
Get:10 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 libgs9 amd64
9.55.0~dfsg1-0ubuntu5.9 [5,033 kB]
Get:11 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 libkpathsea6
amd64 2021.20210626.59705-1ubuntu0.2 [60.4 kB]
Get:12 http://archive.ubuntu.com/ubuntu jammy/main amd64 libwoff1 amd64
1.0.2-1build4 [45.2 kB]
Get:13 http://archive.ubuntu.com/ubuntu jammy/universe amd64 dvisvgm amd64
2.13.1-1 [1,221 kB]
Get:14 http://archive.ubuntu.com/ubuntu jammy/universe amd64 fonts-lmodern all
2.004.5-6.1 [4,532 kB]
Get:15 http://archive.ubuntu.com/ubuntu jammy/main amd64 fonts-noto-mono all
20201225-1build1 [397 kB]
Get:16 http://archive.ubuntu.com/ubuntu jammy/universe amd64 fonts-texgyre all
20180621-3.1 [10.2 MB]
Get:17 http://archive.ubuntu.com/ubuntu jammy/universe amd64 libapache-pom-java
all 18-1 [4,720 B]
Get:18 http://archive.ubuntu.com/ubuntu jammy/universe amd64 libcommons-parent-
java all 43-1 [10.8 kB]
Get:19 http://archive.ubuntu.com/ubuntu jammy/universe amd64 libcommons-logging-
java all 1.2-2 [60.3 kB]
Get:20 http://archive.ubuntu.com/ubuntu jammy/main amd64 libfontenc1 amd64
1:1.1.4-1build3 [14.7 kB]
Get:21 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 libptexenc1
amd64 2021.20210626.59705-1ubuntu0.2 [39.1 kB]
Get:22 http://archive.ubuntu.com/ubuntu jammy/main amd64 rubygems-integration
all 1.18 [5,336 B]
Get:23 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 ruby3.0 amd64
3.0.2-7ubuntu2.7 [50.1 kB]
Get:24 http://archive.ubuntu.com/ubuntu jammy/main amd64 ruby-rubygems all
3.3.5-2 [228 kB]
Get:25 http://archive.ubuntu.com/ubuntu jammy/main amd64 ruby amd64 1:3.0~exp1
[5,100 B]
Get:26 http://archive.ubuntu.com/ubuntu jammy/main amd64 rake all 13.0.6-2 [61.7
kB]
Get:27 http://archive.ubuntu.com/ubuntu jammy/main amd64 ruby-net-telnet all
0.1.1-2 [12.6 kB]
Get:28 http://archive.ubuntu.com/ubuntu jammy/universe amd64 ruby-webrick all
1.7.0-3 [51.8 kB]
Get:29 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 ruby-xmlrpc all
0.3.2-1ubuntu0.1 [24.9 kB]
Get:30 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 libruby3.0
amd64 3.0.2-7ubuntu2.7 [5,113 kB]
```

```
Get:31 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 libsynctex2
amd64 2021.20210626.59705-1ubuntu0.2 [55.6 kB]
Get:32 http://archive.ubuntu.com/ubuntu jammy/universe amd64 libteckit0 amd64
2.5.11+ds1-1 [421 kB]
Get:33 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 libtexlua53
amd64 2021.20210626.59705-1ubuntu0.2 [120 kB]
Get:34 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 libtexluajit2
amd64 2021.20210626.59705-1ubuntu0.2 [267 kB]
Get:35 http://archive.ubuntu.com/ubuntu jammy/universe amd64 libzzip-0-13 amd64
0.13.72+dfsg.1-1.1 [27.0 kB]
Get:36 http://archive.ubuntu.com/ubuntu jammy/main amd64 xfonts-encodings all
1:1.0.5-0ubuntu2 [578 kB]
Get:37 http://archive.ubuntu.com/ubuntu jammy/main amd64 xfonts-utils amd64
1:7.7+6build2 [94.6 kB]
Get:38 http://archive.ubuntu.com/ubuntu jammy/universe amd64 lmodern all
2.004.5-6.1 [9,471 kB]
Get:39 http://archive.ubuntu.com/ubuntu jammy/universe amd64 preview-latex-style
all 12.2-1ubuntu1 [185 kB]
Get:40 http://archive.ubuntu.com/ubuntu jammy/main amd64 t1utils amd64
1.41-4build2 [61.3 kB]
Get:41 http://archive.ubuntu.com/ubuntu jammy/universe amd64 teckit amd64
2.5.11+ds1-1 [699 kB]
Get:42 http://archive.ubuntu.com/ubuntu jammy/universe amd64 tex-gyre all
20180621-3.1 [6,209 kB]
Get:43 http://archive.ubuntu.com/ubuntu jammy-updates/universe amd64 texlive-
binaries amd64 2021.20210626.59705-1ubuntu0.2 [9,860 kB]
Get:44 http://archive.ubuntu.com/ubuntu jammy/universe amd64 texlive-base all
2021.20220204-1 [21.0 MB]
Get:45 http://archive.ubuntu.com/ubuntu jammy/universe amd64 texlive-fonts-
recommended all 2021.20220204-1 [4,972 kB]
Get:46 http://archive.ubuntu.com/ubuntu jammy/universe amd64 texlive-latex-base
all 2021.20220204-1 [1,128 kB]
Get:47 http://archive.ubuntu.com/ubuntu jammy/universe amd64 libfontbox-java all
1:1.8.16-2 [207 kB]
Get:48 http://archive.ubuntu.com/ubuntu jammy/universe amd64 libpdfbox-java all
1:1.8.16-2 [5,199 kB]
Get:49 http://archive.ubuntu.com/ubuntu jammy/universe amd64 texlive-latex-
recommended all 2021.20220204-1 [14.4 MB]
Get:50 http://archive.ubuntu.com/ubuntu jammy/universe amd64 texlive-pictures
all 2021.20220204-1 [8,720 kB]
Get:51 http://archive.ubuntu.com/ubuntu jammy/universe amd64 texlive-latex-extra
all 2021.20220204-1 [13.9 MB]
Get:52 http://archive.ubuntu.com/ubuntu jammy/universe amd64 texlive-plain-
generic all 2021.20220204-1 [27.5 MB]
Get:53 http://archive.ubuntu.com/ubuntu jammy/universe amd64 tipa all 2:1.3-21
[2,967 kB]
Get:54 http://archive.ubuntu.com/ubuntu jammy/universe amd64 texlive-xetex all
2021.20220204-1 [12.4 MB]
```

```
Fetched 182 MB in 15s (12.0 MB/s)
debconf: unable to initialize frontend: Dialog
debconf: (No usable dialog-like program is installed, so the dialog based
frontend cannot be used. at /usr/share/perl5/Debconf/FrontEnd/Dialog.pm line 78,
<> line 54.)
debconf: falling back to frontend: Readline
debconf: unable to initialize frontend: Readline
debconf: (This frontend requires a controlling tty.)
debconf: falling back to frontend: Teletype
dpkg-preconfigure: unable to re-open stdin:
Selecting previously unselected package fonts-droid-fallback.
(Reading database … 123614 files and directories currently installed.)
Preparing to unpack …/00-fonts-droid-fallback_1%3a6.0.1r16-1.1build1_all.deb
…
Unpacking fonts-droid-fallback (1:6.0.1r16-1.1build1) …
Selecting previously unselected package fonts-lato.
Preparing to unpack …/01-fonts-lato_2.0-2.1_all.deb …
Unpacking fonts-lato (2.0-2.1) …
Selecting previously unselected package poppler-data.
Preparing to unpack …/02-poppler-data_0.4.11-1_all.deb …
Unpacking poppler-data (0.4.11-1) …
Selecting previously unselected package tex-common.
Preparing to unpack …/03-tex-common_6.17_all.deb …
Unpacking tex-common (6.17) …
Selecting previously unselected package fonts-urw-base35.
Preparing to unpack …/04-fonts-urw-base35_20200910-1_all.deb …
Unpacking fonts-urw-base35 (20200910-1) …
Selecting previously unselected package libgs9-common.
Preparing to unpack …/05-libgs9-common_9.55.0~dfsg1-0ubuntu5.9_all.deb …
Unpacking libgs9-common (9.55.0~dfsg1-0ubuntu5.9) …
Selecting previously unselected package libidn12:amd64.
Preparing to unpack …/06-libidn12_1.38-4ubuntu1_amd64.deb …
Unpacking libidn12:amd64 (1.38-4ubuntu1) …
Selecting previously unselected package libijs-0.35:amd64.
Preparing to unpack …/07-libijs-0.35_0.35-15build2_amd64.deb …
Unpacking libijs-0.35:amd64 (0.35-15build2) …
Selecting previously unselected package libjbig2dec0:amd64.
Preparing to unpack …/08-libjbig2dec0_0.19-3build2_amd64.deb …
Unpacking libjbig2dec0:amd64 (0.19-3build2) …
Selecting previously unselected package libgs9:amd64.
Preparing to unpack …/09-libgs9_9.55.0~dfsg1-0ubuntu5.9_amd64.deb …
Unpacking libgs9:amd64 (9.55.0~dfsg1-0ubuntu5.9) …
Selecting previously unselected package libkpathsea6:amd64.
Preparing to unpack …/10-libkpathsea6_2021.20210626.59705-1ubuntu0.2_amd64.deb
…
Unpacking libkpathsea6:amd64 (2021.20210626.59705-1ubuntu0.2) …
Selecting previously unselected package libwoff1:amd64.
Preparing to unpack …/11-libwoff1_1.0.2-1build4_amd64.deb …
```

```
Unpacking libwoff1:amd64 (1.0.2-1build4) …
Selecting previously unselected package dvisvgm.
Preparing to unpack …/12-dvisvgm_2.13.1-1_amd64.deb …
Unpacking dvisvgm (2.13.1-1) …
Selecting previously unselected package fonts-lmodern.
Preparing to unpack …/13-fonts-lmodern_2.004.5-6.1_all.deb …
Unpacking fonts-lmodern (2.004.5-6.1) …
Selecting previously unselected package fonts-noto-mono.
Preparing to unpack …/14-fonts-noto-mono_20201225-1build1_all.deb …
Unpacking fonts-noto-mono (20201225-1build1) …
Selecting previously unselected package fonts-texgyre.
Preparing to unpack …/15-fonts-texgyre_20180621-3.1_all.deb …
Unpacking fonts-texgyre (20180621-3.1) …
Selecting previously unselected package libapache-pom-java.
Preparing to unpack …/16-libapache-pom-java_18-1_all.deb …
Unpacking libapache-pom-java (18-1) …
Selecting previously unselected package libcommons-parent-java.
Preparing to unpack …/17-libcommons-parent-java_43-1_all.deb …
Unpacking libcommons-parent-java (43-1) …
Selecting previously unselected package libcommons-logging-java.
Preparing to unpack …/18-libcommons-logging-java_1.2-2_all.deb …
Unpacking libcommons-logging-java (1.2-2) …
Selecting previously unselected package libfontenc1:amd64.
Preparing to unpack …/19-libfontenc1_1%3a1.1.4-1build3_amd64.deb …
Unpacking libfontenc1:amd64 (1:1.1.4-1build3) …
Selecting previously unselected package libptexenc1:amd64.
Preparing to unpack …/20-libptexenc1_2021.20210626.59705-1ubuntu0.2_amd64.deb
…
Unpacking libptexenc1:amd64 (2021.20210626.59705-1ubuntu0.2) …
Selecting previously unselected package rubygems-integration.
Preparing to unpack …/21-rubygems-integration_1.18_all.deb …
Unpacking rubygems-integration (1.18) …
Selecting previously unselected package ruby3.0.
Preparing to unpack …/22-ruby3.0_3.0.2-7ubuntu2.7_amd64.deb …
Unpacking ruby3.0 (3.0.2-7ubuntu2.7) …
Selecting previously unselected package ruby-rubygems.
Preparing to unpack …/23-ruby-rubygems_3.3.5-2_all.deb …
Unpacking ruby-rubygems (3.3.5-2) …
Selecting previously unselected package ruby.
Preparing to unpack …/24-ruby_1%3a3.0~exp1_amd64.deb …
Unpacking ruby (1:3.0~exp1) …
Selecting previously unselected package rake.
Preparing to unpack …/25-rake_13.0.6-2_all.deb …
Unpacking rake (13.0.6-2) …
Selecting previously unselected package ruby-net-telnet.
Preparing to unpack …/26-ruby-net-telnet_0.1.1-2_all.deb …
Unpacking ruby-net-telnet (0.1.1-2) …
Selecting previously unselected package ruby-webrick.
```

```
Preparing to unpack …/27-ruby-webrick_1.7.0-3_all.deb …
Unpacking ruby-webrick (1.7.0-3) …
Selecting previously unselected package ruby-xmlrpc.
Preparing to unpack …/28-ruby-xmlrpc_0.3.2-1ubuntu0.1_all.deb …
Unpacking ruby-xmlrpc (0.3.2-1ubuntu0.1) …
Selecting previously unselected package libruby3.0:amd64.
Preparing to unpack …/29-libruby3.0_3.0.2-7ubuntu2.7_amd64.deb …
Unpacking libruby3.0:amd64 (3.0.2-7ubuntu2.7) …
Selecting previously unselected package libsynctex2:amd64.
Preparing to unpack …/30-libsynctex2_2021.20210626.59705-1ubuntu0.2_amd64.deb
…
Unpacking libsynctex2:amd64 (2021.20210626.59705-1ubuntu0.2) …
Selecting previously unselected package libteckit0:amd64.
Preparing to unpack …/31-libteckit0_2.5.11+ds1-1_amd64.deb …
Unpacking libteckit0:amd64 (2.5.11+ds1-1) …
Selecting previously unselected package libtexlua53:amd64.
Preparing to unpack …/32-libtexlua53_2021.20210626.59705-1ubuntu0.2_amd64.deb
…
Unpacking libtexlua53:amd64 (2021.20210626.59705-1ubuntu0.2) …
Selecting previously unselected package libtexluajit2:amd64.
Preparing to unpack
…/33-libtexluajit2_2021.20210626.59705-1ubuntu0.2_amd64.deb …
Unpacking libtexluajit2:amd64 (2021.20210626.59705-1ubuntu0.2) …
Selecting previously unselected package libzzip-0-13:amd64.
Preparing to unpack …/34-libzzip-0-13_0.13.72+dfsg.1-1.1_amd64.deb …
Unpacking libzzip-0-13:amd64 (0.13.72+dfsg.1-1.1) …
Selecting previously unselected package xfonts-encodings.
Preparing to unpack …/35-xfonts-encodings_1%3a1.0.5-0ubuntu2_all.deb …
Unpacking xfonts-encodings (1:1.0.5-0ubuntu2) …
Selecting previously unselected package xfonts-utils.
Preparing to unpack …/36-xfonts-utils_1%3a7.7+6build2_amd64.deb …
Unpacking xfonts-utils (1:7.7+6build2) …
Selecting previously unselected package lmodern.
Preparing to unpack …/37-lmodern_2.004.5-6.1_all.deb …
Unpacking lmodern (2.004.5-6.1) …
Selecting previously unselected package preview-latex-style.
Preparing to unpack …/38-preview-latex-style_12.2-1ubuntu1_all.deb …
Unpacking preview-latex-style (12.2-1ubuntu1) …
Selecting previously unselected package t1utils.
Preparing to unpack …/39-t1utils_1.41-4build2_amd64.deb …
Unpacking t1utils (1.41-4build2) …
Selecting previously unselected package teckit.
Preparing to unpack …/40-teckit_2.5.11+ds1-1_amd64.deb …
Unpacking teckit (2.5.11+ds1-1) …
Selecting previously unselected package tex-gyre.
Preparing to unpack …/41-tex-gyre_20180621-3.1_all.deb …
Unpacking tex-gyre (20180621-3.1) …
Selecting previously unselected package texlive-binaries.
```

```
Preparing to unpack …/42-texlive-
binaries_2021.20210626.59705-1ubuntu0.2_amd64.deb …
Unpacking texlive-binaries (2021.20210626.59705-1ubuntu0.2) …
Selecting previously unselected package texlive-base.
Preparing to unpack …/43-texlive-base_2021.20220204-1_all.deb …
Unpacking texlive-base (2021.20220204-1) …
Selecting previously unselected package texlive-fonts-recommended.
Preparing to unpack …/44-texlive-fonts-recommended_2021.20220204-1_all.deb …
Unpacking texlive-fonts-recommended (2021.20220204-1) …
Selecting previously unselected package texlive-latex-base.
Preparing to unpack …/45-texlive-latex-base_2021.20220204-1_all.deb …
Unpacking texlive-latex-base (2021.20220204-1) …
Selecting previously unselected package libfontbox-java.
Preparing to unpack …/46-libfontbox-java_1%3a1.8.16-2_all.deb …
Unpacking libfontbox-java (1:1.8.16-2) …
Selecting previously unselected package libpdfbox-java.
Preparing to unpack …/47-libpdfbox-java_1%3a1.8.16-2_all.deb …
Unpacking libpdfbox-java (1:1.8.16-2) …
Selecting previously unselected package texlive-latex-recommended.
Preparing to unpack …/48-texlive-latex-recommended_2021.20220204-1_all.deb …
Unpacking texlive-latex-recommended (2021.20220204-1) …
Selecting previously unselected package texlive-pictures.
Preparing to unpack …/49-texlive-pictures_2021.20220204-1_all.deb …
Unpacking texlive-pictures (2021.20220204-1) …
Selecting previously unselected package texlive-latex-extra.
Preparing to unpack …/50-texlive-latex-extra_2021.20220204-1_all.deb …
Unpacking texlive-latex-extra (2021.20220204-1) …
Selecting previously unselected package texlive-plain-generic.
Preparing to unpack …/51-texlive-plain-generic_2021.20220204-1_all.deb …
Unpacking texlive-plain-generic (2021.20220204-1) …
Selecting previously unselected package tipa.
Preparing to unpack …/52-tipa_2%3a1.3-21_all.deb …
Unpacking tipa (2:1.3-21) …
Selecting previously unselected package texlive-xetex.
Preparing to unpack …/53-texlive-xetex_2021.20220204-1_all.deb …
Unpacking texlive-xetex (2021.20220204-1) …
Setting up fonts-lato (2.0-2.1) …
Setting up fonts-noto-mono (20201225-1build1) …
Setting up libwoff1:amd64 (1.0.2-1build4) …
Setting up libtexlua53:amd64 (2021.20210626.59705-1ubuntu0.2) …
Setting up libijs-0.35:amd64 (0.35-15build2) …
Setting up libtexluajit2:amd64 (2021.20210626.59705-1ubuntu0.2) …
Setting up libfontbox-java (1:1.8.16-2) …
Setting up rubygems-integration (1.18) …
Setting up libzzip-0-13:amd64 (0.13.72+dfsg.1-1.1) …
Setting up fonts-urw-base35 (20200910-1) …
Setting up poppler-data (0.4.11-1) …
Setting up tex-common (6.17) …
```

```
debconf: unable to initialize frontend: Dialog
debconf: (No usable dialog-like program is installed, so the dialog based
frontend cannot be used. at /usr/share/perl5/Debconf/FrontEnd/Dialog.pm line
78.)
debconf: falling back to frontend: Readline
update-language: texlive-base not installed and configured, doing nothing!
Setting up libfontenc1:amd64 (1:1.1.4-1build3) …
Setting up libjbig2dec0:amd64 (0.19-3build2) …
Setting up libteckit0:amd64 (2.5.11+ds1-1) …
Setting up libapache-pom-java (18-1) …
Setting up ruby-net-telnet (0.1.1-2) …
Setting up xfonts-encodings (1:1.0.5-0ubuntu2) …
Setting up t1utils (1.41-4build2) …
Setting up libidn12:amd64 (1.38-4ubuntu1) …
Setting up fonts-texgyre (20180621-3.1) …
Setting up libkpathsea6:amd64 (2021.20210626.59705-1ubuntu0.2) …
Setting up ruby-webrick (1.7.0-3) …
Setting up fonts-lmodern (2.004.5-6.1) …
Setting up fonts-droid-fallback (1:6.0.1r16-1.1build1) …
Setting up ruby-xmlrpc (0.3.2-1ubuntu0.1) …
Setting up libsynctex2:amd64 (2021.20210626.59705-1ubuntu0.2) …
Setting up libgs9-common (9.55.0~dfsg1-0ubuntu5.9) …
Setting up teckit (2.5.11+ds1-1) …
Setting up libpdfbox-java (1:1.8.16-2) …
Setting up libgs9:amd64 (9.55.0~dfsg1-0ubuntu5.9) …
Setting up preview-latex-style (12.2-1ubuntu1) …
Setting up libcommons-parent-java (43-1) …
Setting up dvisvgm (2.13.1-1) …
Setting up libcommons-logging-java (1.2-2) …
Setting up xfonts-utils (1:7.7+6build2) …
Setting up libptexenc1:amd64 (2021.20210626.59705-1ubuntu0.2) …
Setting up texlive-binaries (2021.20210626.59705-1ubuntu0.2) …
update-alternatives: using /usr/bin/xdvi-xaw to provide /usr/bin/xdvi.bin
(xdvi.bin) in auto mode
update-alternatives: using /usr/bin/bibtex.original to provide /usr/bin/bibtex
(bibtex) in auto mode
Setting up lmodern (2.004.5-6.1) …
Setting up texlive-base (2021.20220204-1) …
/usr/bin/ucfr
/usr/bin/ucfr
/usr/bin/ucfr
/usr/bin/ucfr
mktexlsr: Updating /var/lib/texmf/ls-R-TEXLIVEDIST…
mktexlsr: Updating /var/lib/texmf/ls-R-TEXMFMAIN…
mktexlsr: Updating /var/lib/texmf/ls-R…
mktexlsr: Done.
tl-paper: setting paper size for dvips to a4:
/var/lib/texmf/dvips/config/config-paper.ps
```

```
tl-paper: setting paper size for dvipdfmx to a4:
/var/lib/texmf/dvipdfmx/dvipdfmx-paper.cfg
tl-paper: setting paper size for xdvi to a4: /var/lib/texmf/xdvi/XDvi-paper
tl-paper: setting paper size for pdftex to a4: /var/lib/texmf/tex/generic/tex-
ini-files/pdftexconfig.tex
debconf: unable to initialize frontend: Dialog
debconf: (No usable dialog-like program is installed, so the dialog based
frontend cannot be used. at /usr/share/perl5/Debconf/FrontEnd/Dialog.pm line
78.)
debconf: falling back to frontend: Readline
Setting up tex-gyre (20180621-3.1) …
Setting up texlive-plain-generic (2021.20220204-1) …
Setting up texlive-latex-base (2021.20220204-1) …
Setting up texlive-latex-recommended (2021.20220204-1) …
Setting up texlive-pictures (2021.20220204-1) …
Setting up texlive-fonts-recommended (2021.20220204-1) …
Setting up tipa (2:1.3-21) …
Setting up texlive-latex-extra (2021.20220204-1) …
Setting up texlive-xetex (2021.20220204-1) …
Setting up rake (13.0.6-2) …
Setting up libruby3.0:amd64 (3.0.2-7ubuntu2.7) …
Setting up ruby3.0 (3.0.2-7ubuntu2.7) …
Setting up ruby (1:3.0~exp1) …
Setting up ruby-rubygems (3.3.5-2) …
Processing triggers for man-db (2.10.2-1) …
Processing triggers for fontconfig (2.13.1-4.2ubuntu5) …
Processing triggers for libc-bin (2.35-0ubuntu3.4) …
/sbin/ldconfig.real: /usr/local/lib/libtbbmalloc.so.2 is not a symbolic link

/sbin/ldconfig.real: /usr/local/lib/libtbb.so.12 is not a symbolic link

/sbin/ldconfig.real: /usr/local/lib/libtbbbind_2_0.so.3 is not a symbolic link

/sbin/ldconfig.real: /usr/local/lib/libtbbbind.so.3 is not a symbolic link

/sbin/ldconfig.real: /usr/local/lib/libur_loader.so.0 is not a symbolic link

/sbin/ldconfig.real: /usr/local/lib/libtbbbind_2_5.so.3 is not a symbolic link

/sbin/ldconfig.real: /usr/local/lib/libur_adapter_opencl.so.0 is not a symbolic
link

/sbin/ldconfig.real: /usr/local/lib/libtbbmalloc_proxy.so.2 is not a symbolic
link

/sbin/ldconfig.real: /usr/local/lib/libur_adapter_level_zero.so.0 is not a
symbolic link
```

```
Processing triggers for tex-common (6.17) …
debconf: unable to initialize frontend: Dialog
debconf: (No usable dialog-like program is installed, so the dialog based
frontend cannot be used. at /usr/share/perl5/Debconf/FrontEnd/Dialog.pm line
78.)
debconf: falling back to frontend: Readline
Running updmap-sys. This may take some time… done.
Running mktexlsr /var/lib/texmf … done.
Building format(s) --all.
        This may take some time… done.
```

[49]:
```
!apt-get install pandoc
```

```
Reading package lists… Done
Building dependency tree… Done
Reading state information… Done
The following additional packages will be installed:
  libcmark-gfm-extensions0.29.0.gfm.3 libcmark-gfm0.29.0.gfm.3 pandoc-data
Suggested packages:
  texlive-luatex pandoc-citeproc context wkhtmltopdf librsvg2-bin groff ghc
nodejs php python
  libjs-mathjax libjs-katex citation-style-language-styles
The following NEW packages will be installed:
  libcmark-gfm-extensions0.29.0.gfm.3 libcmark-gfm0.29.0.gfm.3 pandoc pandoc-
data
0 upgraded, 4 newly installed, 0 to remove and 49 not upgraded.
Need to get 20.6 MB of archives.
After this operation, 156 MB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu jammy/universe amd64 libcmark-
gfm0.29.0.gfm.3 amd64 0.29.0.gfm.3-3 [115 kB]
Get:2 http://archive.ubuntu.com/ubuntu jammy/universe amd64 libcmark-gfm-
extensions0.29.0.gfm.3 amd64 0.29.0.gfm.3-3 [25.1 kB]
Get:3 http://archive.ubuntu.com/ubuntu jammy/universe amd64 pandoc-data all
2.9.2.1-3ubuntu2 [81.8 kB]
Get:4 http://archive.ubuntu.com/ubuntu jammy/universe amd64 pandoc amd64
2.9.2.1-3ubuntu2 [20.3 MB]
Fetched 20.6 MB in 4s (5,681 kB/s)
Selecting previously unselected package libcmark-gfm0.29.0.gfm.3:amd64.
(Reading database … 160443 files and directories currently installed.)
Preparing to unpack …/libcmark-gfm0.29.0.gfm.3_0.29.0.gfm.3-3_amd64.deb …
Unpacking libcmark-gfm0.29.0.gfm.3:amd64 (0.29.0.gfm.3-3) …
Selecting previously unselected package libcmark-gfm-
extensions0.29.0.gfm.3:amd64.
Preparing to unpack …/libcmark-gfm-
extensions0.29.0.gfm.3_0.29.0.gfm.3-3_amd64.deb …
Unpacking libcmark-gfm-extensions0.29.0.gfm.3:amd64 (0.29.0.gfm.3-3) …
Selecting previously unselected package pandoc-data.
Preparing to unpack …/pandoc-data_2.9.2.1-3ubuntu2_all.deb …
```

```
Unpacking pandoc-data (2.9.2.1-3ubuntu2) …
Selecting previously unselected package pandoc.
Preparing to unpack …/pandoc_2.9.2.1-3ubuntu2_amd64.deb …
Unpacking pandoc (2.9.2.1-3ubuntu2) …
Setting up libcmark-gfm0.29.0.gfm.3:amd64 (0.29.0.gfm.3-3) …
Setting up libcmark-gfm-extensions0.29.0.gfm.3:amd64 (0.29.0.gfm.3-3) …
Setting up pandoc-data (2.9.2.1-3ubuntu2) …
Setting up pandoc (2.9.2.1-3ubuntu2) …
Processing triggers for man-db (2.10.2-1) …
Processing triggers for libc-bin (2.35-0ubuntu3.4) …
/sbin/ldconfig.real: /usr/local/lib/libtbbmalloc.so.2 is not a symbolic link

/sbin/ldconfig.real: /usr/local/lib/libtbb.so.12 is not a symbolic link

/sbin/ldconfig.real: /usr/local/lib/libtbbbind_2_0.so.3 is not a symbolic link

/sbin/ldconfig.real: /usr/local/lib/libtbbbind.so.3 is not a symbolic link

/sbin/ldconfig.real: /usr/local/lib/libur_loader.so.0 is not a symbolic link

/sbin/ldconfig.real: /usr/local/lib/libtbbbind_2_5.so.3 is not a symbolic link

/sbin/ldconfig.real: /usr/local/lib/libur_adapter_opencl.so.0 is not a symbolic
link

/sbin/ldconfig.real: /usr/local/lib/libtbbmalloc_proxy.so.2 is not a symbolic
link

/sbin/ldconfig.real: /usr/local/lib/libur_adapter_level_zero.so.0 is not a
symbolic link
```

[57]:
```
!jupyter nbconvert --to pdf "/content/gdrive/My Drive/assignment2/Assignment2.
  ↪ipynb"
# TODO - UPDATE ME WITH THE TRUE PATH! and UPDATE THE FILE NAME.
```

```
[NbConvertApp] Converting notebook /content/gdrive/My
Drive/assignment2/Assignment2.ipynb to pdf
[NbConvertApp] Support files will be in Assignment2_files/
[NbConvertApp] Making directory ./Assignment2_files
[NbConvertApp] Making directory ./Assignment2_files
[NbConvertApp] Making directory ./Assignment2_files
[NbConvertApp] Making directory ./Assignment2_files
[NbConvertApp] Making directory ./Assignment2_files
[NbConvertApp] Making directory ./Assignment2_files
[NbConvertApp] Making directory ./Assignment2_files
[NbConvertApp] Making directory ./Assignment2_files
[NbConvertApp] Making directory ./Assignment2_files
```

```
[NbConvertApp] Making directory ./Assignment2_files
[NbConvertApp] Making directory ./Assignment2_files
[NbConvertApp] Making directory ./Assignment2_files
[NbConvertApp] Making directory ./Assignment2_files
[NbConvertApp] Making directory ./Assignment2_files
[NbConvertApp] Making directory ./Assignment2_files
[NbConvertApp] Writing 134791 bytes to notebook.tex
[NbConvertApp] Building PDF
[NbConvertApp] Running xelatex 3 times: ['xelatex', 'notebook.tex', '-quiet']
[NbConvertApp] Running bibtex 1 time: ['bibtex', 'notebook']
[NbConvertApp] WARNING | bibtex had problems, most likely because there were no
citations
[NbConvertApp] PDF successfully created
[NbConvertApp] Writing 1119842 bytes to /content/gdrive/My
Drive/assignment2/Assignment2.pdf
```