Assignment 5
COVID-19 detection from coughs

SLPDL 2022

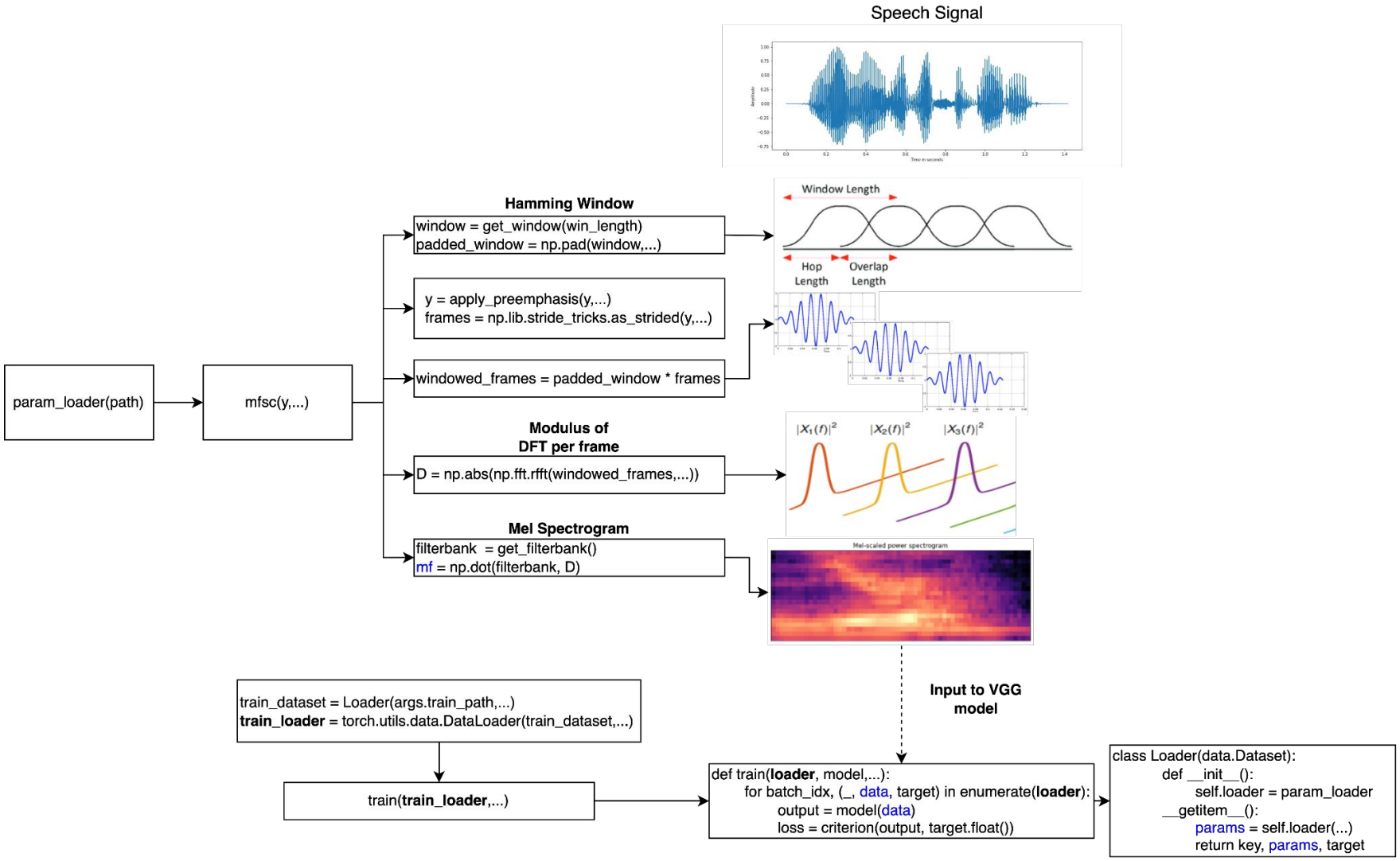
Javier Ferrando Monsonís
javier.ferrando.monsonis@upc.edu

VGG

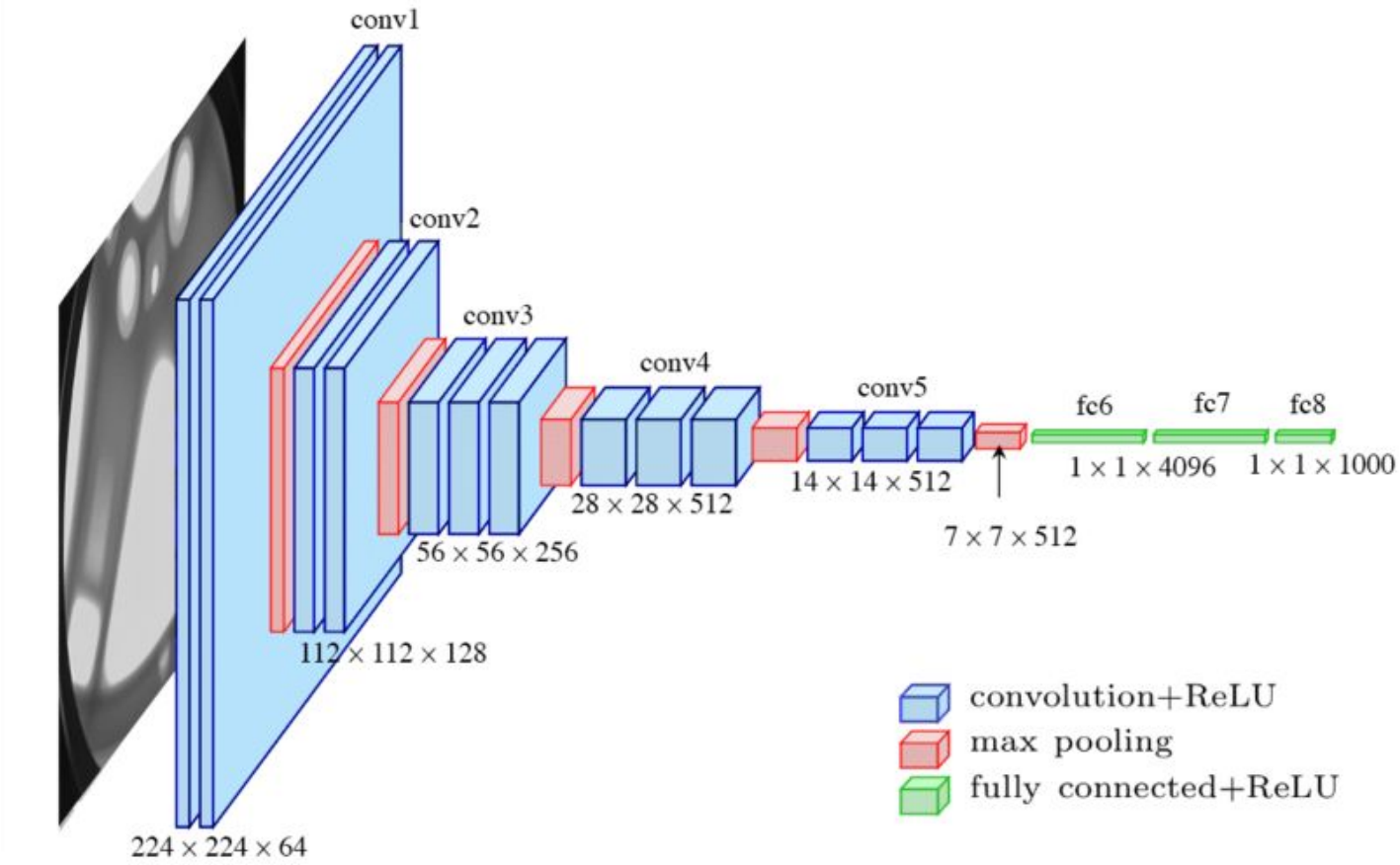
```
args = SimpleNamespace(
    # general options
    train_path = '../input/covid2/train',         # train data folder
    valid_path = '../input/covid2/valid',         # valid data folder
    test_path = '../input/covid2/test',           # test data folder
    batch_size = 32,                              # training and valid batch size
    test_batch_size = 32,                         # batch size for testing
    arc = 'VGG13',                                # VGG11, VGG13, VGG16, VGG19
    epochs = 100,                                 # maximum number of epochs to train
    lr = 0.0001,                                  # learning rate
    momentum = 0.9,                               # SGD momentum, for SGD only
    optimizer = 'adam',                           # optimization method: sgd | adam
    seed = 1234,                                  # random seed
    log_interval = 5,                             # how many batches to wait before logging traini
ng status
    patience = 5,                                 # how many epochs of no loss improvement should
we wait before stop training
    checkpoint = '.',                             # checkpoints directory
    train = True,                                 # train before testing
    cuda = True,                                  # use gpu

    # feature extraction options
    window_size = .04,                            # window size for the stft
    window_stride = .02,                          # window stride for the stft
    window_type = 'hamming',                      # window type for the stft
    normalize = True,                             # use spect normalization
    num_workers = 2,                              # how many subprocesses to use for data loading
)
```

VGG


Speech Signal
Hamming Window
window = get_window(win_length)
padded_window = np.pad(window,...)
Window Length
Hop Length
Overlap Length
y = apply_preemphasis(y,...)
frames = np.lib.stride_tricks.as_strided(y,...)
windowed_frames = padded_window * frames
param_loader(path)
mfsc(y,...)
Modulus of DFT per frame
D = np.abs(np.fft.rfft(windowed_frames,...))
$|X_1(f)|^2$
$|X_2(f)|^2$
$|X_3(f)|^2$
Mel Spectrogram
filterbank = get_filterbank()
mf = np.dot(filterbank, D)
Mel-scaled power spectrogram
Input to VGG model
train_dataset = Loader(args.train_path,...)
train_loader = torch.utils.data.DataLoader(train_dataset,...)
train(train_loader,...)
def train(loader, model,...):
for batch_idx, (_, data, target) in enumerate(loader):
output = model(data)
loss = criterion(output, target.float())
class Loader(data.Dataset):
def __init__():
self.loader = param_loader
__getitem__():
params = self.loader(...)
return key, params, target

VGG

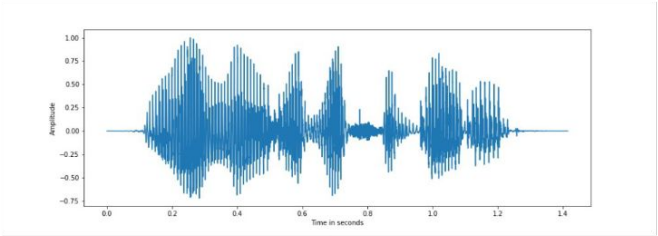

conv1
conv2
conv3
conv4
conv5
fc6
fc7
fc8
224 × 224 × 64
112 × 112 × 128
56 × 56 × 256
28 × 28 × 512
14 × 14 × 512
7 × 7 × 512
1 × 1 × 4096
1 × 1 × 1000
convolution+ReLU
max pooling
fully connected+ReLU

HuBERT

```
args = SimpleNamespace(
    # general options
    train_path = '../input/covid2/train',         # train data folder
    valid_path = '../input/covid2/valid',         # valid data folder
    test_path = '../input/covid2/test',           # test data folder
    batch_size = 22,                              # training and valid batch size
    test_batch_size = 22,                         # batch size for testing
    epochs = 50,                                  # maximum number of epochs to train
    lr = 0.0002,                                  # learning rate
    momentum = 0.9,                               # SGD momentum, for SGD only
    optimizer = 'adam',                           # optimization method: sgd | adam
    seed = seed,                                  # random seed
    log_interval = 5,                             # how many batches to wait before logging traini
ng status
    patience = 5,                                 # how many epochs of no loss improvement should
we wait before stop training
    checkpoint = '.',                             # checkpoints directory
    train = True,                                 # train before testing
    cuda = True,                                  # use gpu
    num_workers = 2,                              # how many subprocesses to use for data loading
    adapter_hidden_size = 64
)
```

HuBERT

Speech Signal



param_loader(path)

```
processor = Wav2Vec2FeatureExtractor.from_pretrained(MODEL)
```

Input to HuBERT model

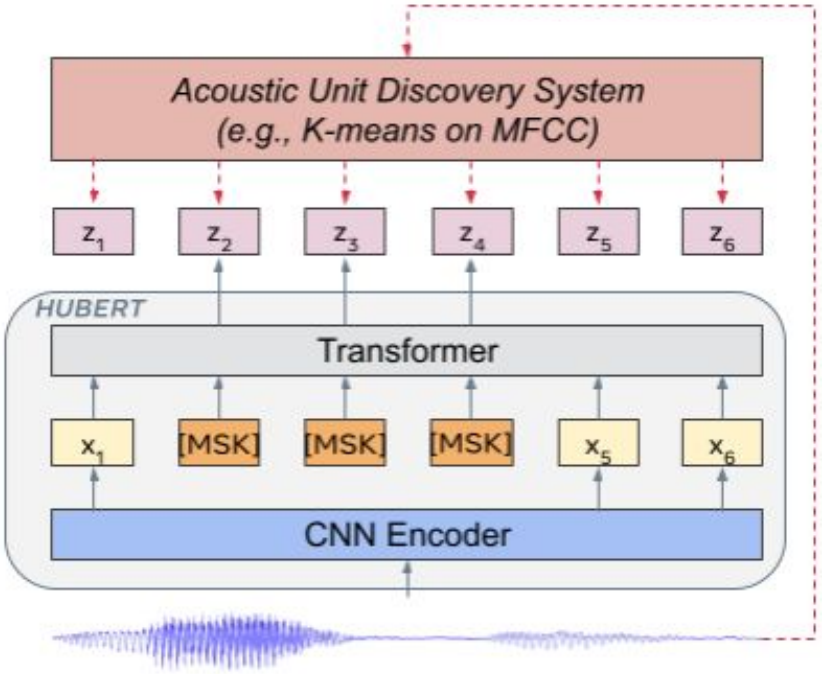
```
train_dataset = Loader(args.train_path,...)
train_loader = torch.utils.data.DataLoader(train_dataset,...)
```

```
train(train_loader,...)
```

```
def train(loader, model,...):
    for batch_idx, (_, data, target) in enumerate(loader):
        output = model(data)
        loss = criterion(output, target.float())
```

```
class Loader(data.Dataset):
    def __init__():
        self.loader = param_loader
    __getitem__():
        params = self.loader(...)
        return key, params, target
```

HuBERT



https://arxiv.org/pdf/2106.07447.pdf

Tasks

- Exploratory Data Analysis
- Experiments with VGG approach (mel-spectrogram, pooling, model size,...)
- Experiments with HuBERT approach (fine-tuning strategy, hyperparameters,...)
- Weighted average of layer's hidden states with trained parameters https://huggingface.co/transformers/v4.11.3/_modules/transformers/models/hubert/modeling_hubert.html#HubertForSequenceClassification
- Improving cross-validation strategy for dealing with few amount of data