Proposal: The optimization of Wav2Vec 2.0 transformer model and transfer learning in the classification and diagnosis of time-series heart sounds

SAT 5114 Small Project 4

Tony Geglio

1. Introduction to the project

Cardiac sound signal classification is an important area of research in cardiology that aims to accurately identify abnormal heart sounds, which can be indicative of various cardiac disorders. Accurate classification of cardiac sound signals is essential for making timely and accurate diagnoses, and for developing effective treatment plans. However, current methods for analyzing and classifying cardiac sound signals have several limitations and challenges.

One potential solution to these challenges is the use of artificial intelligence (AI), machine learning (ML), and deep learning (DL) techniques, which have shown promise in improving the accuracy and reliability of sound analysis for cardiac sound signal classification. The limitations and challenges of current methods for cardiac sound signal classification include the subjectivity of human interpretation, which can lead to variability in diagnoses, and the lack of standardization in terminology and classification schemes. In addition, traditional signal processing techniques may not be effective for capturing subtle differences in sound patterns that are indicative of cardiac disorders.

Problems to Solve

Currently, patients with cardiovascular disease have tools for monitoring their heart, however early diagnosis of heart problems is still difficult. The proposal here is: through very inexpensive microphone recorders connected to a human body and communicating data with a smartphone could provide continuous analysis to people with heart problems, and send alerts, rather than requiring the patient to perform a measurement and then react. This project goal is to test machine learning approaches that could be implemented in heart sound classification tools.

Technical Issues

1. Model Size - Data processing must be fast on a patient device or smartphone app. Large models may have too many parameters to do inference quickly

2. Data amount - Data storage space may be a technical issue given that this device hypothetically records the heart continuously.
3. Generalization - The model may have a hard time generalizing in the real world because every situation is unique, and it is unlikely that the training data captured every possibility.

Clinical Issues

1. Interpretability - If the transfer learning model is too complex, it may be difficult to explain and interpret. Doctors may have a hard time trusting and replacing more traditional methods with one that uses deep learning if the model is not explainable enough.

Ethical Issues

1. Data Ownership - One solution to data storage is to upload and analyze data online. This brings up ethical issues of data ownership.
2. Patient Tracking - It would also be useful to track and record the patient's location for life-saving situations, but this also has many ethical concerns.
3. Responsible Party - Who is responsible if the method fails to perform a proper diagnosis or early warning if the patient and doctor are relying on it as a first step of seeking help.

Suggestions

This method should not replace other traditional methods for patients with serious heart issues, which may include blood pressure monitors and regular checkups. This device would be more of an additional tool the patient and doctor can use for monitoring, but it should not be relied upon for saving the patient's life. Also, federated learning techniques should be used to incorporate new private data into the model for retraining as the patient data can contribute to making the model more robust.

Similar Work

Here are two publications on the current state of the art in cardiac sound signal classification published between 2020 and 2023

1. One relevant review paper is titled “Heart sound classification using signal processing and machine learning algorithms" (2022) by Zeinali et. al[1]. In this study, they explored extracting signal features such as amplitude, dominant frequencies, and the discrete wavelet transforms, and information theory features. The authors explored using the support vector machines classifier (SVC), gradient boosting classifier (GBC), and random forest classifier (RFC). The accuracy ranged from 75 - 87% with gradient boosting performing the best.
2. One common method for statistical machine learning classification of audio data is explained in multiple speaker identification studies[2] [3]. Their method is to extract features, in this case, extracting the Mel Frequency Cepstral Coefficients (MFCC) from audio signals.

Implementation Plan and Data Description

Previously, I explored statistical machine learning methods to classify heart sounds. Now, I am proposing a transfer learning approach, using Wav2Vec 2.0[4]. Wav2vec is a deep transformer network originally designed for natural language processing. This classification challenge uses data from a past competition for classifying heart sounds[5]. There are two heart sound data sets: Set A with 4-classes, and Set B with 3-classes. The challenge was to submit a model to correctly classify an unlabeled set. I do not have the labels for the unlabeled sets, therefore I will use the training sets for validation and testing of the Wav2Vec 2.0 transformer.

Data

Data was collected for a 2011 challenge proposed by Bentley et al. The challenge included 2 data sets: **data set A**) with heart sounds from the general public via the iStethoscope Pro iPhone app; and, **data set B**) with heart sounds from a clinic trial in hospitals using the digital stethoscope DigiScope. Combined, there are a total of 585 samples, each being a short clip in .wav format ranging anywhere from 3 to 30 seconds. The class balance is relatively balanced in Set A while Set B is unbalanced with many more "normal" samples. Set B are generally around 3 seconds, and Set A contains longer recordings.

Set A has a total of 124 recordings

- 4 categories for Set A:
    a) Normal (31)
    b) Murmur (34)
    c) Extra Heart Sound (19)
    d) Artifact (40)

Set B has a total of 464 recordings

- 3 classes contained in Set B:
    a) Normal (320)
    b) Murmur (95)
    c) Extrasystole (46)

Past Performances

In the plot below, I have highlighted the class precision and sensitivity (recall) because I will be reporting those metrics in my classification report.

		ISEP/IPP Portugal J48 / MLP	CS UCL	SLAC Stanford
Challenge 1 A	Total error	4 219 736.5	3 394 378.8	**1 243 640.7**
Challenge 1 B	Total error	**72 242.8**	75 569.8	76 444.4
Challenge 2 A	Precision of Normal	0.25 / 0.35	**0.46**	
	Precision of Murmur	0.47 / **0.67**	0.31	
	Precision of ExtraS	**0.27** / 0.18	0.11	
	Precision of Artifact	0.71 / **0.92**	0.58	
	Artifact Sensitivity	0.63 / **0.69**	0.44	
	Artifact Specificity	0.39 / **0.44**	**0.44**	
	Youden Idx Artifact	0.01 / **0.13**	-0.09	
	F-score	**0.20** / **0.20**	0.14	
	Total Precision	1.71 / **2.12**	1.47	
Challenge 2 B	Precision of Normal	0.72 / 0.70	**0.77**	
	Precision of Murmur	0.32 / 0.30	**0.37**	
	Precision of ExtraS	0.33 / **0.67**	0.17	
	Heart prb Sensitivity	0.22 / 0.19	**0.51**	
	Heart prb Specificity	0.82 / **0.84**	0.59	
	Youden Idx Hrt prb	**0.04** / 0.02	0.01	
	Discriminant Power	0.05 / 0.04	**0.09**	
	Total Precision	1.37 / **1.67**	1.31	

Performance Results

Wav2Vec 2.0 Transformer Results with no data augmentation

Set A

```
***** Running Evaluation *****
  Num examples = 19
  Batch size = 32
  TOTAL TIME: 142.68
              precision    recall  f1-score   support

    artifact       0.71      0.83      0.77         6
    extrahls       0.00      0.00      0.00         3
      murmur       0.80      0.80      0.80         5
      normal       0.57      0.80      0.67         5

    accuracy                           0.68        19
   macro avg       0.52      0.61      0.56        19
weighted avg       0.59      0.68      0.63        19
```

Set B

```
***** Running Prediction *****
  Num examples = 70
  Batch size = 32
  TOTAL TIME: 8.25
              precision    recall  f1-score   support

  extrastole       0.00      0.00      0.00         7
      murmur       0.67      0.13      0.22        15
      normal       0.72      0.98      0.83        48

    accuracy                           0.70        70
   macro avg       0.46      0.37      0.35        70
weighted avg       0.64      0.70      0.62        70
```

Wav2Vec 2.0 Transformer Results with Augmentation and Oversampling

Set A with Transfer Learning

	precision	recall	f1-score	support
artifact	0.99	0.99	0.99	852
extrahls	0.76	0.83	0.80	96
murmur	0.91	0.92	0.92	337
normal	0.84	0.79	0.81	226
----------------	-----------	--------	----------	---------
accuracy			0.94	1511
macro avg	0.88	0.88	0.88	1511
weighted avg	0.94	0.94	0.94	1511

	artifact	extrahls	murmur	normal
artifact	847	2	2	1
extrahls	2	80	1	13
murmur	0	5	311	21
normal	4	18	26	178

Set B with Transfer Learning

	precision	recall	f1-score	support
extrastole	0.43	0.39	0.41	111
murmur	0.71	0.69	0.70	162
normal	0.80	0.82	0.81	500
----------------	-----------	--------	----------	---------
accuracy			0.73	773
macro avg	0.65	0.63	0.64	773
weighted avg	0.73	0.73	0.73	773

	extrastole	murmur	normal
extrastole	43	7	61
murmur	6	112	44
normal	50	38	412

Statistical Classifier (MLP) Results with Augmentation and Oversampling

Set A with MLP

MLPClassifier(activation='logistic', alpha=0.01, hidden_layer_sizes=(40, 20), max_iter=500)

[x] performance for MLP classifier

	precision	recall	f1-score	support
artifact	1.00	1.00	1.00	767
extrahls	0.91	0.99	0.95	86
murmur	0.99	0.96	0.97	303
normal	0.96	0.96	0.96	204
----------------	----------	--------	----------	---------
accuracy			0.98	1360
macro avg	0.96	0.98	0.97	1360
weighted avg	0.98	0.98	0.98	1360

	artifact	extrahls	murmur	normal
artifact	767	0	0	0
extrahls	0	85	0	1
murmur	0	3	292	8
normal	0	5	4	195

TOTAL TIME: 1155.32

Set B with MLP

MLPClassifier(activation='logistic', alpha=0.01, hidden_layer_sizes=(40, 20), max_iter=500)

[x] performance for MLP classifier

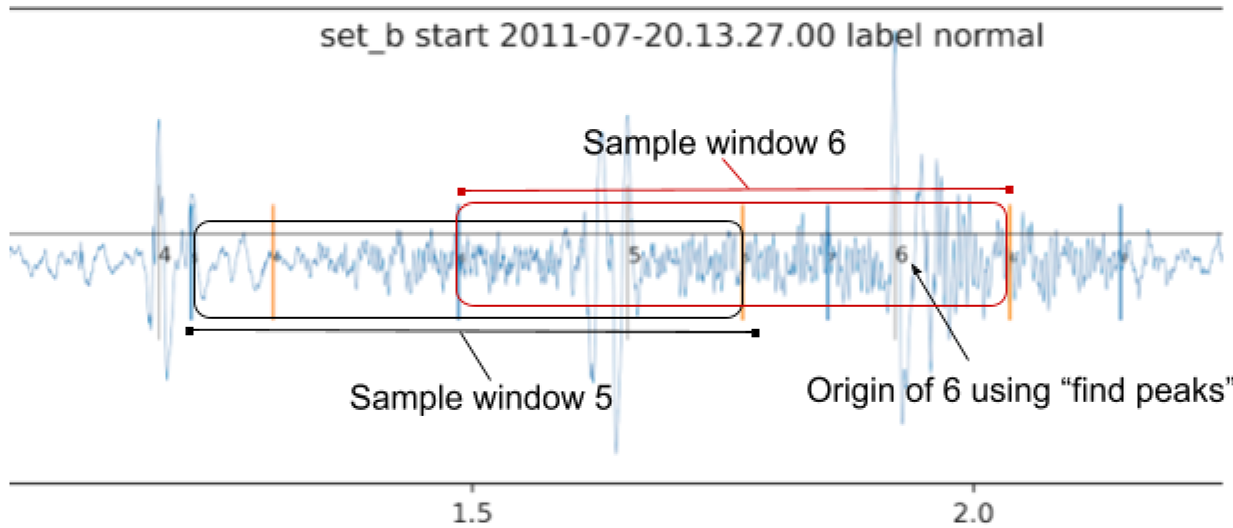
	precision	recall	f1-score	support
extrastole	0.51	0.65	0.57	100
murmur	0.61	0.75	0.67	146
normal	0.87	0.76	0.81	450
----------------	----------	--------	----------	---------
accuracy			0.74	696
macro avg	0.66	0.72	0.68	696
weighted avg	0.77	0.74	0.75	696

	extrastole	murmur	normal
extrastole	65	13	22
murmur	10	109	27
normal	53	57	340

TOTAL TIME: 244.53

Data Augmentation and oversampling Method

Data Augmentation is performed in the python file called “WavPreprocess-Hearbeat.py”. The augmentation method generates 0.55 second samples from the recordings. Each sample’s “origin” was located using the numpy “find peaks” function. Note there is sample overlap which creates one form of oversampling. I also oversampled using built-in functions for hugging face datasets called “interleave” which generates a class balance. See the figure visualizing the samples 5 and 6 taken from a set B recording. A total of ten beats were detected in this file, so in this case, a 3.5 second recording generates 10 samples totaling 5.5 seconds of data. The augmentation outputs an audio (.wav) file for each sample.



Datasets Creation

After Augmentation, and storing the samples locally, the dataset can be created. Here I use the "DatasetDict" functionality from Hugging Face Dataset (Datasets). Datasets combines functionality from both Tensorflow and Pytorch. My program displays the following information after generating the dataset:

```
Creating Dataset set_a
10073  labels counted
10073  audio files counted
Saving test dataset separately to: /work/ajgeglio/Tap_Data/Other_data/test_dataset
DatasetDict({
    train: Dataset({
        features: ['audio', 'label'],
        num_rows: 7051
    })
    validation: Dataset({
        features: ['audio', 'label'],
        num_rows: 1662
    })
    test: Dataset({
        features: ['audio', 'label'],
        num_rows: 1360
    })
})

DATASET CREATION TIME: 2.39
Filter:  85%|████████████████████████████████████████
| 6000/7051 [02:21<00Filter:
99%|█████████████████████████████████████████████████
█ | 7000/7051 [02:24<00Filter:
100%|████████████████████████████████████████████████
███| 7051/7051 [02:25<00
```

```
DatasetDict({
    train: Dataset({
        features: ['audio', 'label'],
        num_rows: 15904
    })
    validation: Dataset({
        features: ['audio', 'label'],
        num_rows: 1662
    })
    test: Dataset({
        features: ['audio', 'label'],
        num_rows: 1360
    })
})
OVERSAMPLING TIME: 675.50
dict_values(['artifact', 'extrahls', 'murmur', 'normal'])
############ WAVEFORM ##################
Sample rate: 16000
Real sample time(s): 0.54425
waveform shapes: original--> (8708,)
#########################################
TOTAL TIME: 701.04

Creating Dataset set_b
5148  labels counted
5148  audio files counted
Saving test dataset separately to: /work/ajgeglio/Tap_Data/Other_data/test_dataset
DatasetDict({
    train: Dataset({
        features: ['audio', 'label'],
        num_rows: 3603
    })
    validation: Dataset({
        features: ['audio', 'label'],
        num_rows: 849
    })
    test: Dataset({
        features: ['audio', 'label'],
        num_rows: 696
    })
})
DATASET CREATION TIME: 0.46
DatasetDict({
    train: Dataset({
        features: ['audio', 'label'],
        num_rows: 6990
    })
    validation: Dataset({
        features: ['audio', 'label'],
        num_rows: 849
    })
    test: Dataset({
        features: ['audio', 'label'],
        num_rows: 696
    })
})
OVERSAMPLING TIME: 207.74
dict_values(['extrastole', 'murmur', 'normal'])
############ WAVEFORM ##################
Sample rate: 16000
Real sample time(s): 0.55
waveform shapes: original--> (8800,)
#########################################
TOTAL TIME: 210.24
(Wav2vec) ajgeglio@cheetah:~/OtherProjects$ python Wav2Vec-Heartbeat.py --set set_a --create_dataset --oversample
Creating Dataset set_a
10073  labels counted
10073  audio files counted
Saving test dataset separately to: /work/ajgeglio/Tap_Data/Other_data/test_dataset
DatasetDict({
    train: Dataset({
        features: ['audio', 'label'],
        num_rows: 7051
    })
    validation: Dataset({
        features: ['audio', 'label'],
        num_rows: 1662
    })
    test: Dataset({
        features: ['audio', 'label'],
        num_rows: 1360
```

```
    })
})
DATASET CREATION TIME: 2.39
Filter:  85%|██████████████████████████████████████████████
| 6000/7051 [02:21<00Filter:
99%|█████████████████████████████████████████████████████
█ | 7000/7051 [02:24<00Filter:
100%|████████████████████████████████████████████████████
███| 7051/7051 [02:25<00
DatasetDict({
    train: Dataset({
        features: ['audio', 'label'],
        num_rows: 15904
    })
    validation: Dataset({
        features: ['audio', 'label'],
        num_rows: 1662
    })
    test: Dataset({
        features: ['audio', 'label'],
        num_rows: 1360
    })
})
OVERSAMPLING TIME: 675.50
dict_values(['artifact', 'extrahls', 'murmur', 'normal'])
```

Transfer Learning Set-up

Wav2Vec 2.0 Model

Next a custom python file loads the Wav2Vec 2.0 transformer model architecture, weights, feature extractor, and training configurations.

```python
from huggingface_hub import notebook_login
import evaluate
from datasets import Audio, Dataset, load_from_disk, DatasetDict,  interleave_datasets
# from transformers import AdamW, get_linear_schedule_with_warmup
from transformers import Wav2Vec2Model, Wav2Vec2Config
from transformers import AutoFeatureExtractor
from transformers import AutoModelForAudioClassification, TrainingArguments, Trainer
```

Importing hugging face datasets, and the Wav2Vec2 model and configuration files from the transformers library

```
1 (initializing a BertForSequenceClassification model from a BertForSequenceClassification model).
Some weights of Wav2Vec2ForSequenceClassification were not initialized from the model checkpoint at facebook/wav2vec2-base and are newly initialized: ['projector.
weight', 'classifier.weight', 'classifier.bias', 'projector.bias']
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.
Wav2Vec2ForSequenceClassification(
  (wav2vec2): Wav2Vec2Model(
    (feature_extractor): Wav2Vec2FeatureExtractor(
      (conv_layers): ModuleList(
        (0): Wav2Vec2GroupNormConvLayer(
          (conv): Conv1d(1, 512, kernel_size=(10,), stride=(5,), bias=False)
          (layer_norm): GroupNorm(512, 512, eps=1e-05, affine=True)
        )
        (1): Wav2Vec2NoLayerNormConvLayer(
          (conv): Conv1d(512, 512, kernel_size=(3,), stride=(2,), bias=False)
        )
        (2): Wav2Vec2NoLayerNormConvLayer(
          (conv): Conv1d(512, 512, kernel_size=(3,), stride=(2,), bias=False)
        )
        (3): Wav2Vec2NoLayerNormConvLayer(
          (conv): Conv1d(512, 512, kernel_size=(3,), stride=(2,), bias=False)
        )
        (4): Wav2Vec2NoLayerNormConvLayer(
          (conv): Conv1d(512, 512, kernel_size=(3,), stride=(2,), bias=False)
        )
        (5): Wav2Vec2NoLayerNormConvLayer(
          (conv): Conv1d(512, 512, kernel_size=(2,), stride=(2,), bias=False)
        )
        (6): Wav2Vec2NoLayerNormConvLayer(
          (conv): Conv1d(512, 512, kernel_size=(2,), stride=(2,), bias=False)
        )
      )
    )
    (feature_projection): Wav2Vec2FeatureProjection(
      (layer_norm): LayerNorm((512,), eps=1e-05, elementwise_affine=True)
      (projection): Linear(in_features=512, out_features=768, bias=True)
      (dropout): Dropout(p=0.1, inplace=False)
    )
    (encoder): Wav2Vec2Encoder(
```

Wav2Vec 2.0 begins with a convnet with 6 layers followed by layer_norm, and dropout

```
            (output_dropout): Dropout(p=0.1, inplace=False)
          )
          (final_layer_norm): LayerNorm((768,), eps=1e-05, elementwise_affine=True)
        )
        (11): Wav2Vec2EncoderLayer(
          (attention): Wav2Vec2Attention(
            (k_proj): Linear(in_features=768, out_features=768, bias=True)
            (v_proj): Linear(in_features=768, out_features=768, bias=True)
            (q_proj): Linear(in_features=768, out_features=768, bias=True)
            (out_proj): Linear(in_features=768, out_features=768, bias=True)
          )
          (dropout): Dropout(p=0.1, inplace=False)
          (layer_norm): LayerNorm((768,), eps=1e-05, elementwise_affine=True)
          (feed_forward): Wav2Vec2FeedForward(
            (intermediate_dropout): Dropout(p=0.0, inplace=False)
            (intermediate_dense): Linear(in_features=768, out_features=3072, bias=True)
            (output_dense): Linear(in_features=3072, out_features=768, bias=True)
            (output_dropout): Dropout(p=0.1, inplace=False)
          )
          (final_layer_norm): LayerNorm((768,), eps=1e-05, elementwise_affine=True)
        )
      )
    )
  )
  (projector): Linear(in_features=768, out_features=256, bias=True)
  (classifier): Linear(in_features=256, out_features=3, bias=True)
)
```

Our 3-class classification layer. All parameters are trainable

11 Attention Encoder layers in the transformer

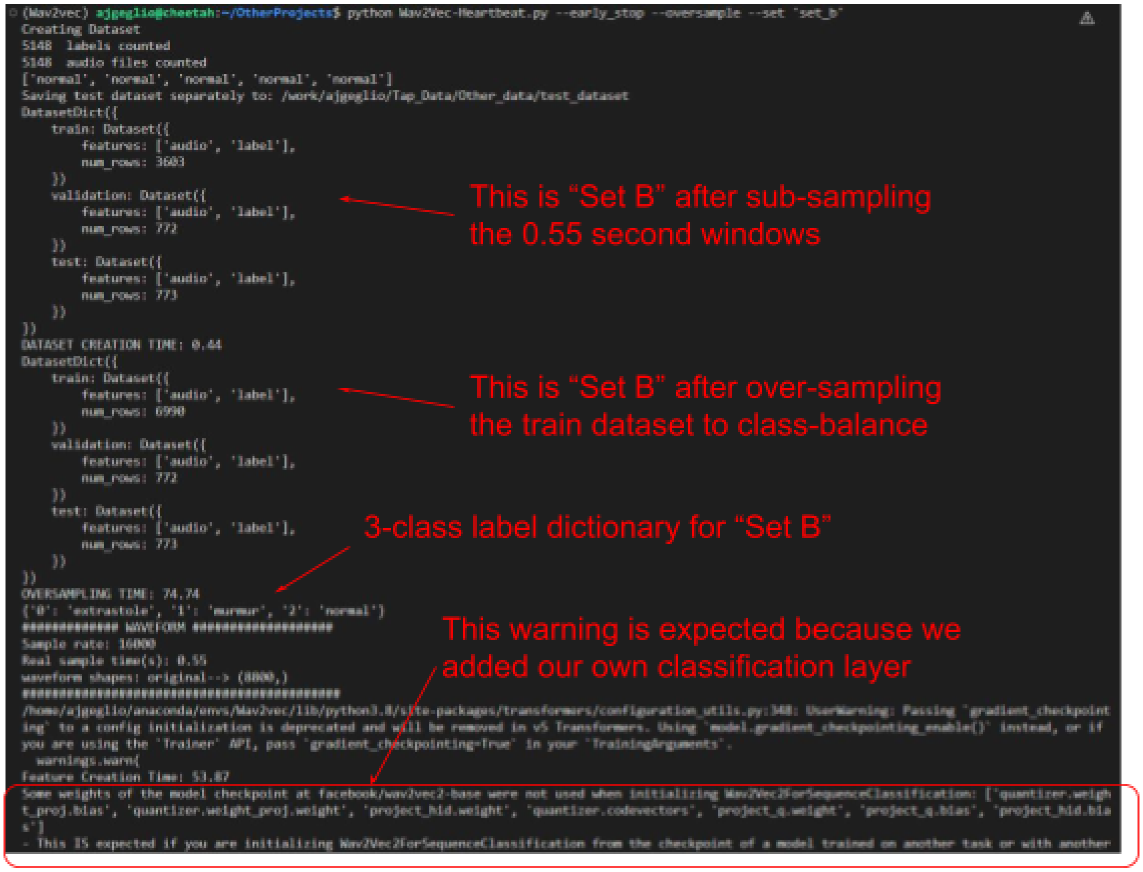
```
######################## Training the model ####################################

Now that our data is ready, we can download the pretrained model and fine-tune it.
For classification we use the AutoModelForAudioClassification class.
Like with the feature extractor, the from_pretrained method will download and cache the model for us.
As the label ids and the number of labels are dataset dependent, we pass num_labels, label2id,
and id2label alongside the model_checkpoint here:
'''
model_checkpoint = "facebook/wav2vec2-base"
batch_size = 32
model = AutoModelForAudioClassification.from_pretrained(
    model_checkpoint,
    num_labels=num_labels,
    label2id=label2id,
    id2label=id2label,
    )

'''
In some cases, you might be interested in keeping the weights of the pre-trained encoder frozen
and optimizing only the weights of the head layers. To do so, simply set the requires_grad attribute
to False on the encoder parameters, which can be accessed with the base_model submodule on any
task-specific model in the library:
'''
# for param in model.base_model.parameters():
#     param.requires_grad = False
```

Model Definition

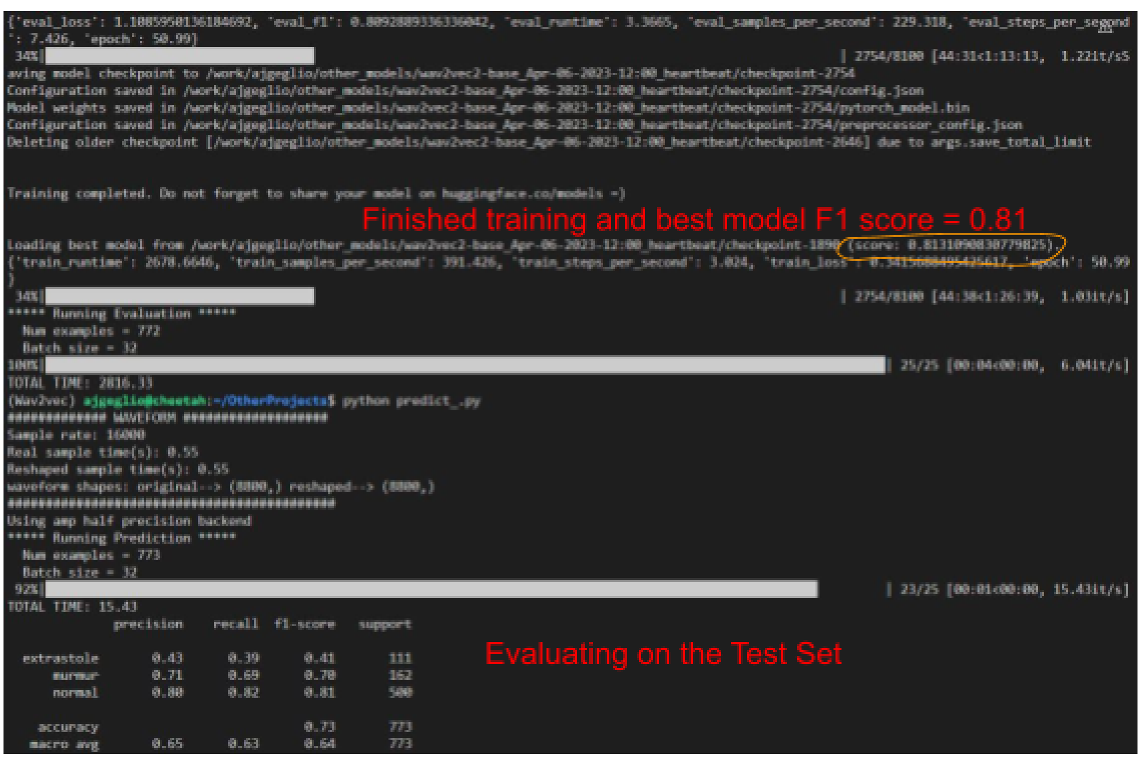
This is where I load the model with the wav2vec2-base checkpoint

All layers in the transformer are trainable unless I un-comment this code

Model Output



```
(Wav2vec) ajgeglio@cheetah:~/OtherProjects$ python Wav2Vec-Heartbeat.py --early_stop --oversample --set 'set_b'
Creating Dataset
5148  labels counted
5148  audio files counted
['normal', 'normal', 'normal', 'normal', 'normal']
Saving test dataset separately to: /work/ajgeglio/Tap_Data/Other_data/test_dataset
DatasetDict({
    train: Dataset({
        features: ['audio', 'label'],
        num_rows: 3603
    })
    validation: Dataset({
        features: ['audio', 'label'],
        num_rows: 772
    })
    test: Dataset({
        features: ['audio', 'label'],
        num_rows: 773
    })
})
DATASET CREATION TIME: 0.44
DatasetDict({
    train: Dataset({
        features: ['audio', 'label'],
        num_rows: 6990
    })
    validation: Dataset({
        features: ['audio', 'label'],
        num_rows: 772
    })
    test: Dataset({
        features: ['audio', 'label'],
        num_rows: 773
    })
})
OVERSAMPLING TIME: 74.74
{'0': 'extrastole', '1': 'murmur', '2': 'normal'}
############ WAVEFORM ##################
Sample rate: 16000
Real sample time(s): 0.55
waveform shapes: original--> (8800,)
##########################################
/home/ajgeglio/anaconda/envs/Wav2vec/lib/python3.8/site-packages/transformers/configuration_utils.py:368: UserWarning: Passing `gradient_checkpointing` to a config initialization is deprecated and will be removed in v5 Transformers. Using `model.gradient_checkpointing_enable()` instead, or if you are using the `Trainer` API, pass `gradient_checkpointing=True` in your `TrainingArguments`.
  warnings.warn(
Feature Creation Time: 53.87
Some weights of the model checkpoint at facebook/wav2vec2-base were not used when initializing Wav2Vec2ForSequenceClassification: ['quantizer.weight_proj.bias', 'quantizer.weight_proj.weight', 'project_hid.weight', 'quantizer.codevectors', 'project_q.weight', 'project_q.bias', 'project_hid.bias']
- This IS expected if you are initializing Wav2Vec2ForSequenceClassification from the checkpoint of a model trained on another task or with another
```

```
{'eval_loss': 1.1085950136184692, 'eval_f1': 0.8092889336336042, 'eval_runtime': 3.3665, 'eval_samples_per_second': 229.318, 'eval_steps_per_second': 7.426, 'epoch': 50.99}
 34%|████████            | 2754/8100 [44:31<1:13:13,  1.22it/s
aving model checkpoint to /work/ajgeglio/other_models/wav2vec2-base_Apr-06-2023-12:00_heartbeat/checkpoint-2754
Configuration saved in /work/ajgeglio/other_models/wav2vec2-base_Apr-06-2023-12:00_heartbeat/checkpoint-2754/config.json
Model weights saved in /work/ajgeglio/other_models/wav2vec2-base_Apr-06-2023-12:00_heartbeat/checkpoint-2754/pytorch_model.bin
Configuration saved in /work/ajgeglio/other_models/wav2vec2-base_Apr-06-2023-12:00_heartbeat/checkpoint-2754/preprocessor_config.json
Deleting older checkpoint [/work/ajgeglio/other_models/wav2vec2-base_Apr-06-2023-12:00_heartbeat/checkpoint-2646] due to args.save_total_limit


Training completed. Do not forget to share your model on huggingface.co/models =)

Loading best model from /work/ajgeglio/other_models/wav2vec2-base_Apr-06-2023-12:00_heartbeat/checkpoint-1890 (score: 0.8131090830779825).
{'train_runtime': 2678.6646, 'train_samples_per_second': 391.426, 'train_steps_per_second': 3.024, 'train_loss': 0.3415688495425617, 'epoch': 50.99
}
 34%|████████            | 2754/8100 [44:38<1:26:39,  1.03it/s]
***** Running Evaluation *****
  Num examples = 772
  Batch size = 32
100%|████████████████████| 25/25 [00:04<00:00,  6.04it/s]
TOTAL TIME: 2816.33
(Wav2vec) ajgeglio@cheetah:~/OtherProjects$ python predict_.py
############ WAVEFORM ##################
Sample rate: 16000
Real sample time(s): 0.55
Reshaped sample time(s): 0.55
waveform shapes: original--> (8800,) reshaped--> (8800,)
##########################################
Using amp half precision backend
***** Running Prediction *****
  Num examples = 773
  Batch size = 32
 92%|██████████████████  | 23/25 [00:01<00:00, 15.43it/s]
TOTAL TIME: 15.43
              precision    recall  f1-score   support

  extrastole       0.43      0.39      0.41       111
      murmur       0.71      0.69      0.70       162
      normal       0.80      0.82      0.81       500

    accuracy                           0.73       773
   macro avg       0.65      0.63      0.64       773
```

Why might Transfer learning work well here?

Wav2Vec 2.0 is a very deep transformer network designed for audio classification. With transfer learning, we have access to the complex model that is open source and has been developed by many contributors. Secondly, the pretrained weights on the facebook/wav2vec2-base are the wav2vec 2.0 model parameters trained on hundreds of hours of audio data. By using transfer learning with these weights, it will reduce the amount of data necessary to optimize the model with our additional head classifier layer. Additionally, the model comes with a feature extractor on top of the network. The feature extractor was designed to amplify the important audio features in samples prior to training. As you can see in the example above, we scored better than the past results for the “murmur” and “normal” classes in just 44 minutes of training.

Statistical ML Methods and Results

Statistical classification can be done on the MFCC features by representing the signal features in a tabular format. I will test out bagging and boosting methods, such as Random Forest and Adaboost, as well as a Support Vector Machine and Multi-Layer Perceptron and compare their performance on the MFCC data. Later, I may explore deep learning on the raw signals because 1D-CNN on audio has shown good performance in other studies and requires less processing time.

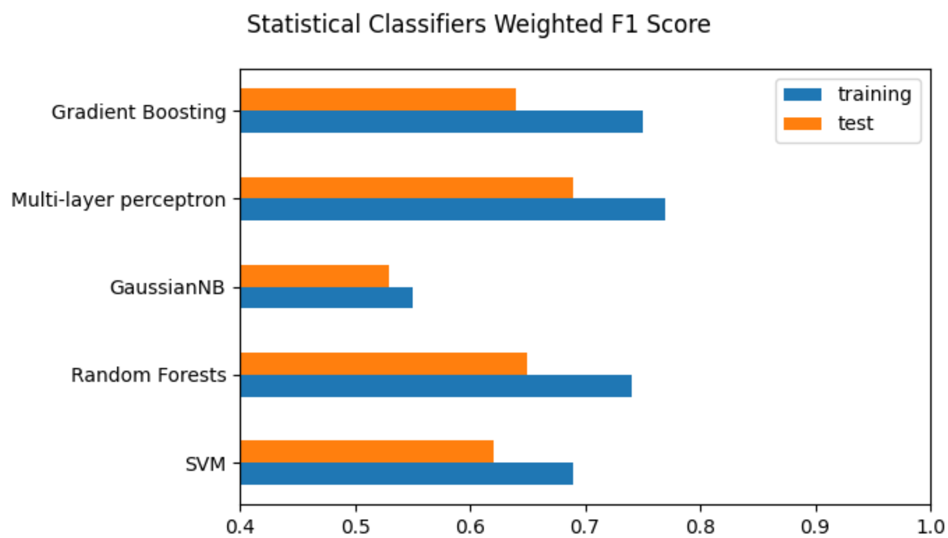
Method Modification

Here I explore a 5-class classification problem after combining the two data sets. After the combination, the data contained 5 unique classes:

a. Normal
b. Murmur
c. Extra Heart Sound
d. Artifact
e. Extrasystole

We use a train-test-split of 75/25, gridsearchCV for parameter tuning, 5-fold cross-validation for resampling, and the weighted average F1-score is used to optimize the models.

Statistical ML Results



Random Forests

Best Model

RandomForestClassifier(max_depth=4, n_estimators=15, random_state=42)
{'criterion': 'gini', 'max_depth': 4, 'max_features': 'sqrt', 'min_samples_split': 2, 'n_estimators': 15}

Testing

accuracy:0.714
recall:0.714
Precision:0.716

Classification Report

	precision	recall	f1-score	support
artifact	1.00	0.90	0.95	10
extrahls	0.44	0.80	0.57	5
extrastole	0.00	0.00	0.00	12
murmur	1.00	0.25	0.40	32
normal	0.69	0.95	0.80	88
accuracy			0.71	147
macro avg	0.63	0.58	0.54	147

| weighted avg | 0.72 | 0.71 | **0.65** | 147 |

Multi-Layer Perceptron

Best Model

MLPClassifier(activation='logistic', alpha=0.1, hidden_layer_sizes=(40, 20),
max_iter=500)
{'activation': 'logistic', 'alpha': 0.1, 'hidden_layer_sizes': (40, 20)}

Testing

accuracy:0.728
recall:0.728
precision:0.672

Classification Report

	precision	recall	f1-score	support
artifact	1.00	0.90	0.95	10
extrahls	0.50	1.00	0.67	5
extrastole	0.00	0.00	0.00	12
murmur	0.64	0.50	0.56	32
normal	0.75	0.88	0.81	88
accuracy			0.73	147
macro avg	0.58	0.66	0.60	147
weighted avg	0.67	0.73	**0.69**	147

Support Vector Machine

SVC(C=0.1, gamma=0.01, kernel='linear', max_iter=10000, probability=True)
{'C': 0.1, 'gamma': 0.01, 'kernel': 'linear'}

Testing

accuracy:0.673
recall:0.673
precision:0.625

Classification Report

	precision	recall	f1-score	support
artifact	1.00	0.70	0.82	10
extrahls	0.29	0.40	0.33	5

extrastole	0.00	0.00	0.00	12
murmur	0.64	0.28	0.39	32
normal	0.68	0.92	0.78	88
accuracy			0.67	147
macro avg	0.52	0.46	0.47	147
weighted avg	0.63	0.67	**0.62**	147

Gradient Boosting Classifier

GradientBoostingClassifier(max_depth=1, n_estimators=50, random_state=42)
{'learning_rate': 0.1, 'max_depth': 1, 'n_estimators': 50}

Testing

accuracy:0.701
recall:0.701
precision:0.658
Classification Report

	precision	recall	f1-score	support
artifact	1.00	0.70	0.82	10
extrahls	0.57	0.80	0.67	5
extrastole	0.00	0.00	0.00	12
murmur	0.73	0.25	0.37	32
normal	0.69	0.95	0.80	88
accuracy			0.70	147
macro avg	0.60	0.54	0.53	147
weighted avg	0.66	0.70	**0.64**	147

Naive Bayes

GaussianNB()
{'priors': None}

Testing

accuracy:0.483
recall:0.483
precision:0.715
Classification Report

	precision	recall	f1-score	support
artifact	0.73	0.80	0.76	10
extrahls	0.31	1.00	0.48	5
extrastole	0.13	0.58	0.21	12
murmur	0.86	0.38	0.52	32

normal	0.76	0.44	0.56	88
accuracy			0.48	147
macro avg	0.56	0.64	0.51	147
weighted avg	0.71	0.48	**0.53**	147

Team Members:

Currently I am working independently.

Works Cited

[1] Y. Zeinali and S. T. A. Niaki, “Heart sound classification using signal processing and machine learning algorithms,” *Mach. Learn. Appl.*, vol. 7, p. 100206, Mar. 2022, doi: 10.1016/j.mlwa.2021.100206.

[2] S. Nakagawa, L. Wang, and S. Ohtsuka, “Speaker Identification and Verification by Combining MFCC and Phase Information,” *IEEE Trans. Audio Speech Lang. Process.*, vol. 20, no. 4, pp. 1085–1095, May 2012, doi: 10.1109/TASL.2011.2172422.

[3] M. Hasan, M. Jamil, G. Rabbani, and Md. S. Rahman, “Speaker Identification Using Mel Frequency Cepstral Coefficients,” *Proc. 3rd Int. Conf. Electr. Comput. Eng. ICECE 2004*, Dec. 2004.

[4] A. Baevski, H. Zhou, A. Mohamed, and M. Auli, “wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations.” 2020.

[5] “Classifying Heart Sounds Challenge.” http://www.peterjbentley.com/heartchallenge/ (accessed Feb. 21, 2023).