Cough Segmentation

remove previous data

close all; clc; clear;

add yamnet to serach path

downloadFolder = fullfile(tempdir,'YAMNetDownload');

loc = websave(downloadFolder,'https://ssd.mathworks.com/supportfiles/audio/yamnet.zip');

YAMNetLocation = tempdir;

unzip(loc,YAMNetLocation);

addpath(fullfile(YAMNetLocation,'yamnet'));

net = yamnet

disp(net.Layers);

% analyzeNetwork(net);

Create audioDatastore object of the data and split it into train and validation sets

folder\_name\_coughvid = "../../database/" + 'auto-segmentation-DL';

ads = audioDatastore(folder\_name\_coughvid, ...

'IncludeSubfolders', true, ...

'LabelSource','foldernames');

[adsTrain, adsVal, adsTest] = splitEachLabel(ads, 0.6, 0.2, 0.2);

Create audioFeatureExtractor object from audio signals

best choices:

window length = 128 [samples]

overlap coefficient between windows = 50 [%]

overlap coefficient between segments = 90 [%]

win\_info.win\_len = 128;

win\_info.overlap\_coeff\_win = 0.5;

win\_info.overlap\_coeff\_spectrograms = 0.9;

win\_info = extrct\_win\_info(win\_info);

afe = audioFeatureExtractor(...

'SampleRate', win\_info.fs, ...

'Window', win\_info.win, ...

'OverlapLength', win\_info.overlap\_len, ...

'melSpectrum', true ...

);

setExtractorParams( ...

afe, 'melSpectrum', ...

'SpectrumType', "magnitude", ...

'NumBands', win\_info.num\_bands, ...

"WindowNormalization", false ...

);

Extract features from train and validation sets

add\_true\_lbls = "yes";

[trainFeatures, trainLabels, trainSpecs] = preprocess\_seg\_yamnet(adsTrain, afe, win\_info, add\_true\_lbls);

[valFeatures, valLabels, valSpecs] = preprocess\_seg\_yamnet(adsVal, afe, win\_info, add\_true\_lbls);

% save progress

save('Features.mat', ...

'trainFeatures', 'trainLabels', 'trainSpecs', ...

'valFeatures', 'valLabels', 'valSpecs');

cough segmentation requires only 2 classes (cough / non-cough).

Read in YAMNet, convert it to a [layerGraph](docid:nnet_ref#mw_399d64ed-680b-46f0-af77-0472ba1910d8), and then replace the final [fullyConnectedLayer](docid:nnet_ref#mw_1e7fbc56-4746-4f30-8cd9-7048ce806a0d) and the final [classificationLayer](docid:nnet_ref#bu5lho8) to reflect the new task.

uniqueLabels = unique(trainLabels);

numLabels = 2;

num\_train\_pos = sum(trainLabels == "Cough");

num\_train\_neg = sum(trainLabels == "nonCough");

num\_train = num\_train\_pos + num\_train\_neg;

classWeightsUniform = 'none';

classWeightsNonUniform = [num\_train\_pos, num\_train\_neg] / num\_train;

classWeights = classWeightsNonUniform;

net = yamnet;

lgraph = layerGraph(net.Layers);

% newDenseLayer = fullyConnectedLayer(numLabels, "Name", "dense");

% lgraph = replaceLayer(lgraph, "dense", newDenseLayer);

%

% newClassificationLayer = classificationLayer( ...

% "Name", "output", ...

% "Classes", uniqueLabels, ...

% 'ClassWeights', classWeights);

% lgraph = replaceLayer(lgraph, "Sound", newClassificationLayer);

lgraph = removeLayers(lgraph, {'dense', 'softmax', 'Sound'});

layers = [

% fullyConnectedLayer(128, "Name", "fc1")

% batchNormalizationLayer("Name", "BN")

% reluLayer("Name", "activation\_27")

fullyConnectedLayer(numLabels, "Name", "fc2")

softmaxLayer("Name", "softmax")

classificationLayer( ...

"Name", "output", ...

"Classes", uniqueLabels, ...

'ClassWeights', classWeights)

];

lgraph = addLayers(lgraph, layers);

lgraph = connectLayers(lgraph,'global\_average\_pooling2d','fc2');

Define training options

miniBatchSize = 256;

MaxEpochs = 5;

InitialLearnRate = 0.0003;

L2Regularization =0.01;

validationFrequency = floor(numel(trainLabels)/miniBatchSize);

options = trainingOptions( ...

'adam', ...

'Plots', 'training-progress', ...

'Verbose', true, ...

'MaxEpochs', MaxEpochs, ...

'MiniBatchSize', miniBatchSize, ...

'Shuffle', 'every-epoch', ...

'ValidationData', {single(valFeatures),valLabels}, ...

'ValidationFrequency', validationFrequency, ...

'InitialLearnRate', InitialLearnRate, ...

'LearnRateSchedule', 'piecewise', ...

'LearnRateDropPeriod', 1, ...

'LearnRateDropFactor', 0.1, ...

'L2Regularization', L2Regularization ...

);

Train and save network

[cough\_seg\_yamnet\_model, netInfo] = trainNetwork(trainFeatures, trainLabels, lgraph, options);

Save the trained network

save cough\_seg\_yamnet\_info.mat cough\_seg\_yamnet\_model win\_info adsTrain adsVal adsTest

evaluate the performance of the network on the test set using:

* ROC & AUC
* confusion matrix
* accuracy, sensetivity, PPV, F1-score

accuracy = (TP + TN) / (TP + TN + FP + FN)

sensitivity = TP / (TP + FN)

PPV = TP / (TP + FP)

F1-score = 2 \* (PPV \* sensitivity) / (PPV + sensitivity)

% choose threshold for ROC curve

load cough\_seg\_yamnet\_info

roc\_threshold = 0.5;

extract\_scores\_seg\_yamnet(cough\_seg\_yamnet\_model, adsTest, win\_info, roc\_threshold);

plot segmentation results on validation data set

ds\_name = "voca";

folder\_name = "../../database/auto-segmentation-DL/" + ds\_name;

adsFromNewData = audioDatastore(folder\_name, ...

'IncludeSubfolders', true, ...

'LabelSource','foldernames');

ads\_predict = adsTest;

add\_true\_lbls = "yes";

add\_plots = "yes";

cough\_seg\_yamnet\_rslts = predict\_on\_new\_ads(cough\_seg\_yamnet\_model, ads\_predict, win\_info, roc\_threshold, add\_true\_lbls, add\_plots);