

CAS Applied Data Science University of Bern

Module 6 Deep Learning – Music vs Speech Classification

Stefano Brandinu & Marc Bratschi

Content

1. Objectives & Data
2. Data preparation, preprocessing and split
3. Goal
4. Model building
 - Only Convolutional Layers
 - With Pooling
5. Final model and evaluation
6. Conclusion

Jupyter notebook on Github: [Link](#)

Objective & Data

Objective

- > This project tries to implement a deep learning model that classifies audio samples in music or speech.

Data

- > 128 audio files of 30 seconds each
 - 64 speech files
 - 64 music files

Data preparation, preprocessing and split

Split in train and test set

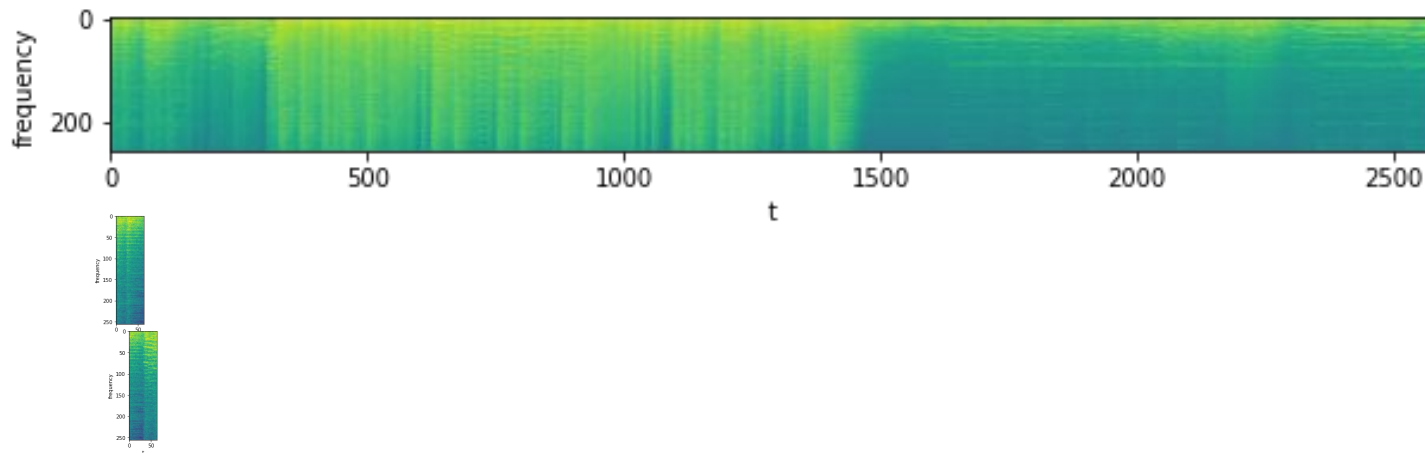
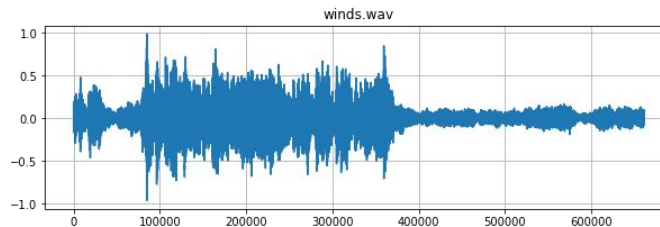
- > Get directory of each audio file
- > All audios from a category (i.e. speech or music) are split into a training (80%) and validation (20%) set

Data preprocessing

- > Load audio files for music and speech datasets separately for training and validation
- > Applying discrete Fourier Transformation
- > Transform and cut into smaller slices (79 per audio with overlapping)
- > Put together samples of speech and music
- > Add labels
- > Results in:
 - Training set: 8'058
 - Validation set: 2'054

Data preparation, preprocessing and split

Example of the **frequency** of a music file (winds.wav) including a **fourier transformation** and the first **slice**.

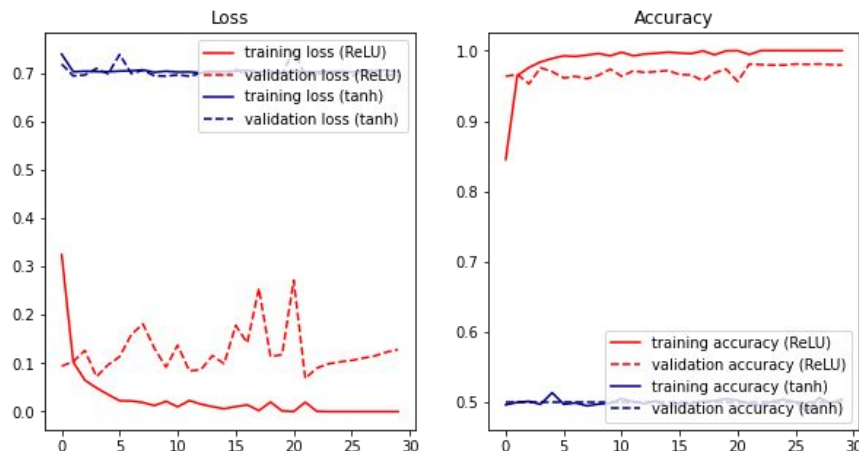


Model building

Model architecture

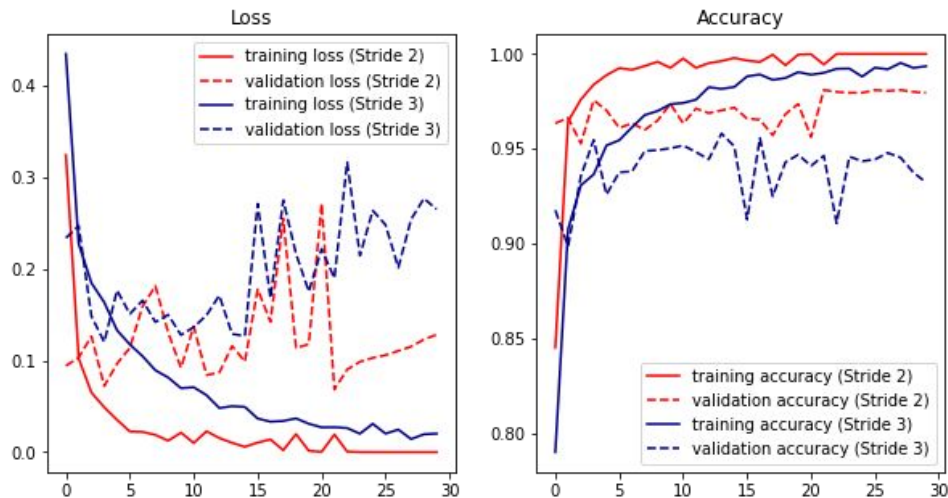
- > With and without Pooling Layers
 - Epochs = 30
 - Batch Size = 10
- > Testing different activations functions for hidden layers
 - ReLU and tanh
- > Chose model based on accuracy
- > Try different hyperparameters
 - Stride
 - Kernels
- > Test Learnings rates
 - 0.0005
 - 0.001

Model building - only Convolutional Layers



- tanh activation function does not seem to work
- tanh activation function has some drawbacks
- One of the main limitations is the “vanishing gradient” problem.
- This makes it difficult for the model to learn and converge.

Model building - only Convolutional Layers



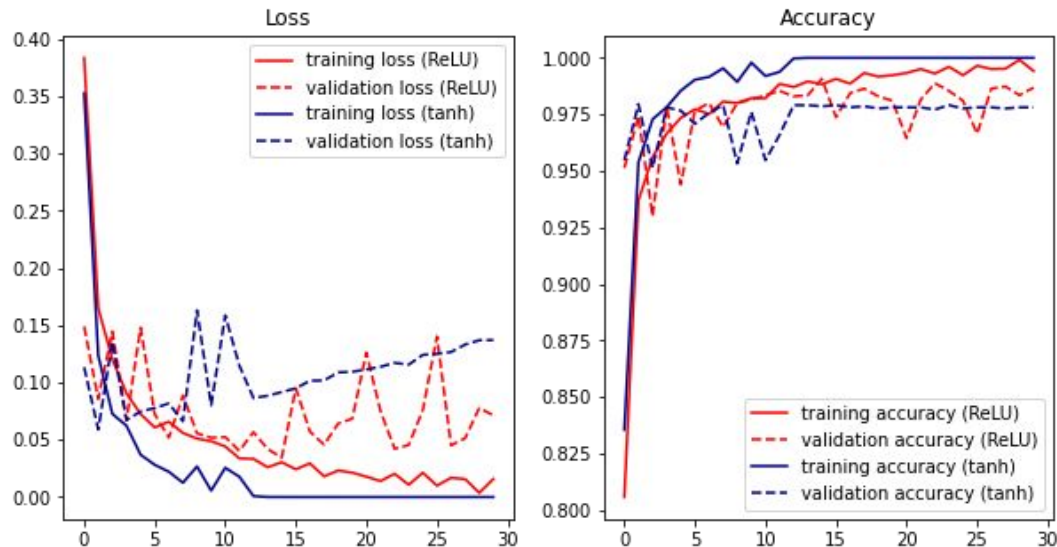
- Using a smaller stride value will result in a larger output volume which can increase the computational cost of the network.
- Using a larger stride can also result in a loss of information as some areas of the input may not be considered in the output.
- According to the plot implementing stride 2 leads to lower validation loss and better validation accuracy

Model building - **only Convolutional Layers**

Findings

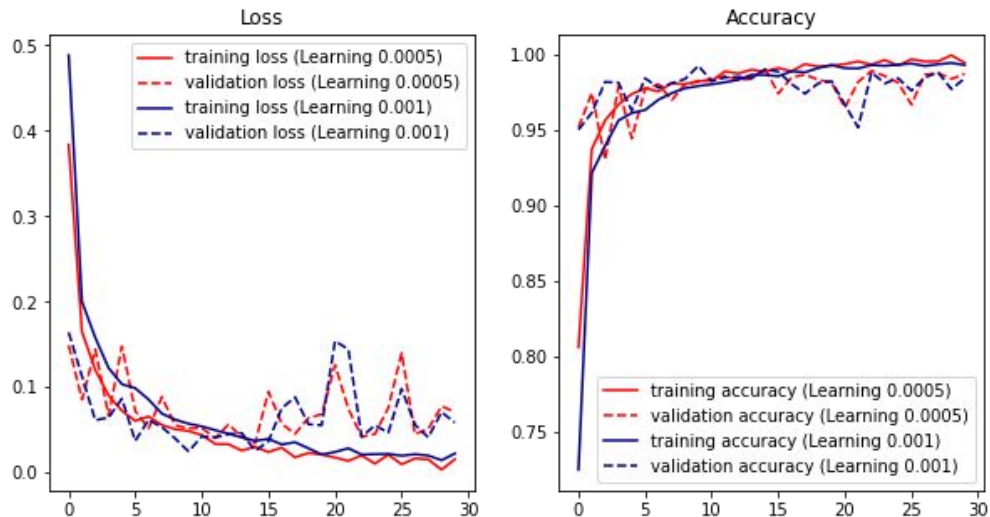
- tanh activation function does not seem to work
- Better implement ReLU activation function
- Stride of 2 leads to lower validation loss and better validation accuracy than stride of 3

Model building - with Pooling Layers



- ReLU activation function shows lower validation loss and better validation accuracy than tanh activation function

Model building - with Pooling Layers



- Testing model 1 with different learning rates (0.0005 and 0.001)
- No visible difference with different learning rates

Model building - with Pooling Layers

- Testing different value for kernel size, stride and learning rate does not show strong differences in validation set.
- For the final model the following parameters will be implemented:
 - Activation function ReLU (Rectified Linear Unit)
 - Padding “same”
 - Kernel size = 2x2
 - Strides = 2
 - Higher strides does not work well with many layers
 - Learning rate = 0.0005

Model Comparison

Model with pooling layer

Model: "sequential_14"

Layer (type)	Output Shape	Param #
conv2d_56 (Conv2D)	(None, 128, 32, 32)	160
max_pooling2d_28 (MaxPooling2D)	(None, 64, 16, 32)	0
conv2d_57 (Conv2D)	(None, 64, 16, 64)	8256
max_pooling2d_29 (MaxPooling2D)	(None, 32, 8, 64)	0
conv2d_58 (Conv2D)	(None, 32, 8, 128)	32896
max_pooling2d_30 (MaxPooling2D)	(None, 16, 4, 128)	0
conv2d_59 (Conv2D)	(None, 16, 4, 256)	131328
max_pooling2d_31 (MaxPooling2D)	(None, 8, 2, 256)	0
flatten_14 (Flatten)	(None, 4096)	0
dense_28 (Dense)	(None, 512)	2097664
dense_29 (Dense)	(None, 2)	1026

=====
Total params: 2,271,330
Trainable params: 2,271,330
Non-trainable params: 0

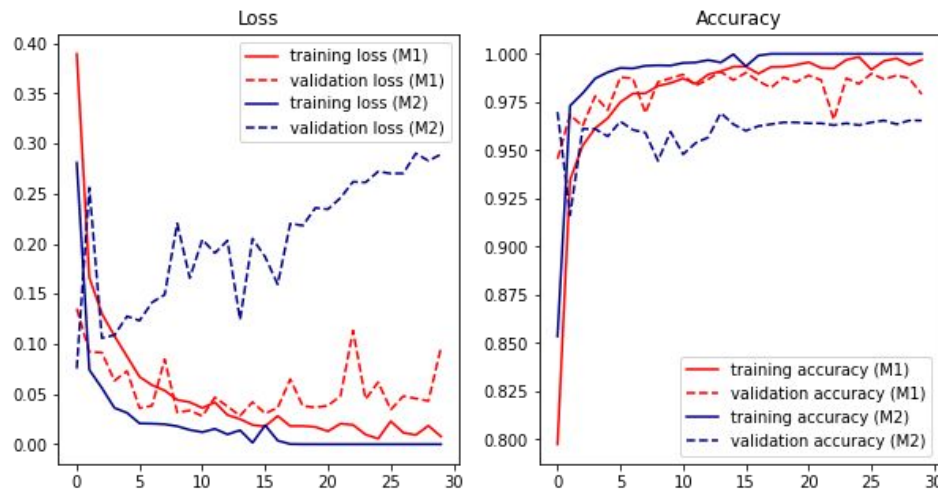
Model only convolutional layer

Model: "sequential_15"

Layer (type)	Output Shape	Param #
conv2d_60 (Conv2D)	(None, 128, 32, 32)	320
conv2d_61 (Conv2D)	(None, 64, 16, 64)	18496
conv2d_62 (Conv2D)	(None, 32, 8, 128)	73856
conv2d_63 (Conv2D)	(None, 16, 4, 256)	295168
flatten_15 (Flatten)	(None, 16384)	0
dense_30 (Dense)	(None, 512)	8389120
dense_31 (Dense)	(None, 2)	1026

=====
Total params: 8,777,986
Trainable params: 8,777,986
Non-trainable params: 0

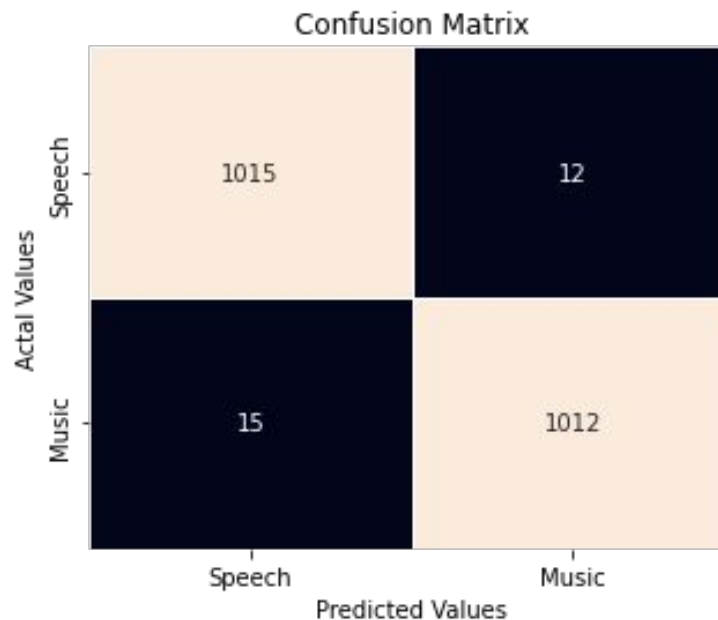
Comparison Model 1 & Model 2



- The comparison of the model with pooling layers and the model only with convolutional layers shows lower validation loss and better training accuracy with pooling layer.
- Apparently this is mainly due to the pooling layer which improves the performance with less trainable parameters

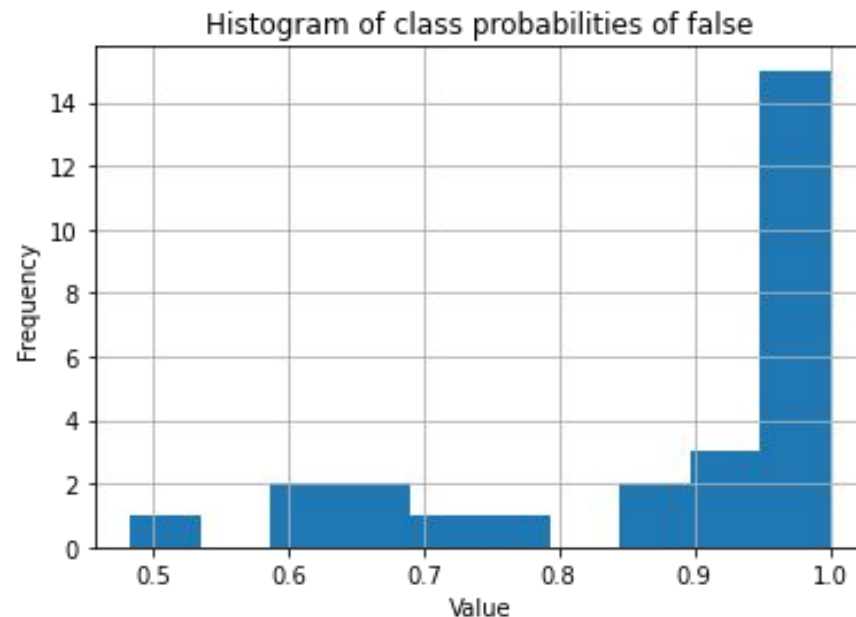
Evaluation - Confusion Matrix

- > Validation accuracy: 98%
- > Even distribution of wrong predicted classes

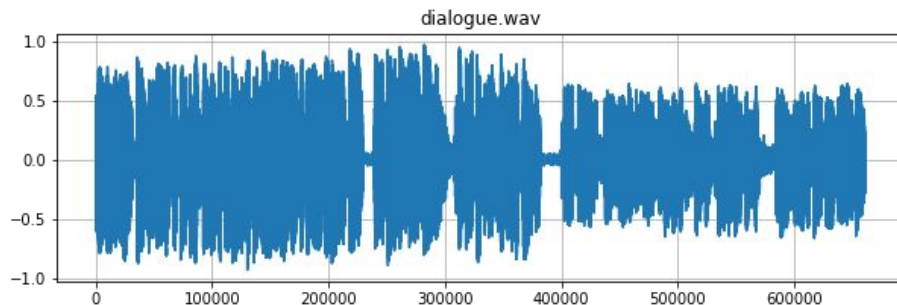
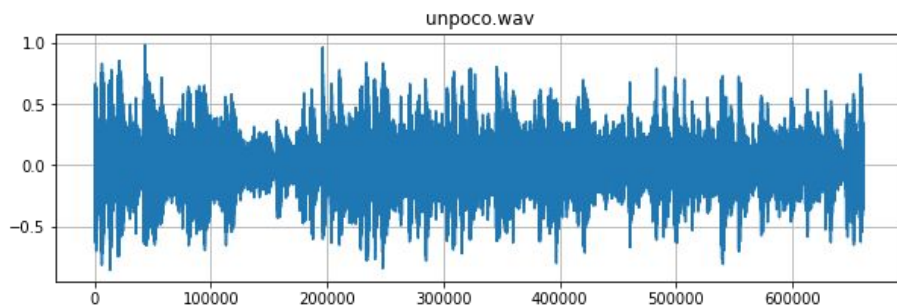
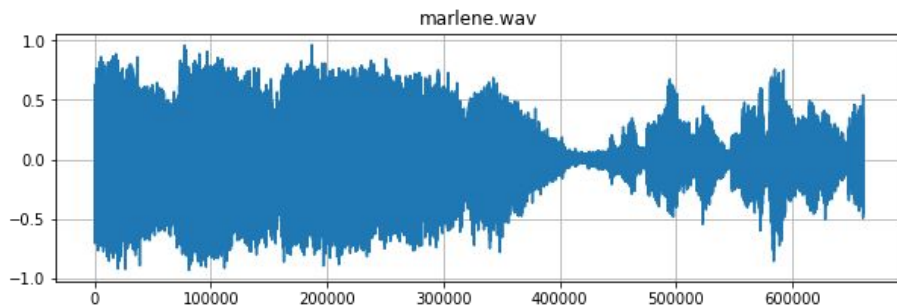


Evaluation - Probabilities and Audio Files

- > Fairly high probabilities by wrong ones
- > Remember: 79 slices per audio
 - Not whole song, only segment is wrongly predicted
- > Names of wrong predicted audio files
 - ncherry.wav': 4,
 - gravity.wav': 8,
 - voices.wav': 1,
 - nj105.wav': 2,
 - dialogue1.wav': 2,
 - nj105a.wav': 2,
 - dialogue.wav': 6,
 - my_voice.wav': 2

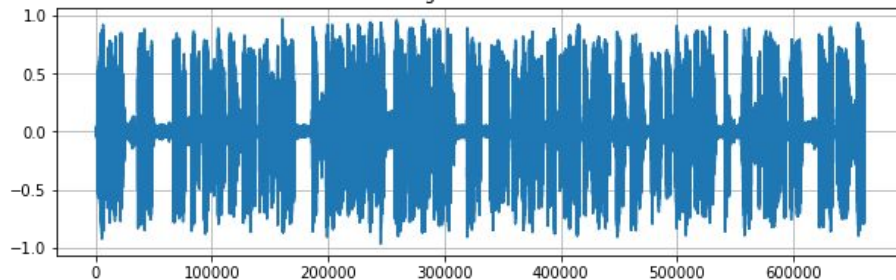


Evaluation - Speech recognized as Music

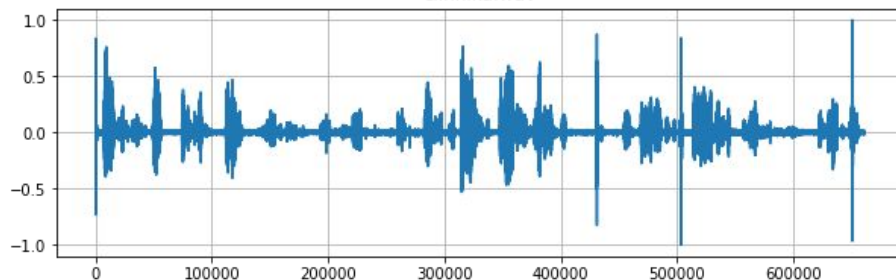


Evaluation - Music recognized as Speech

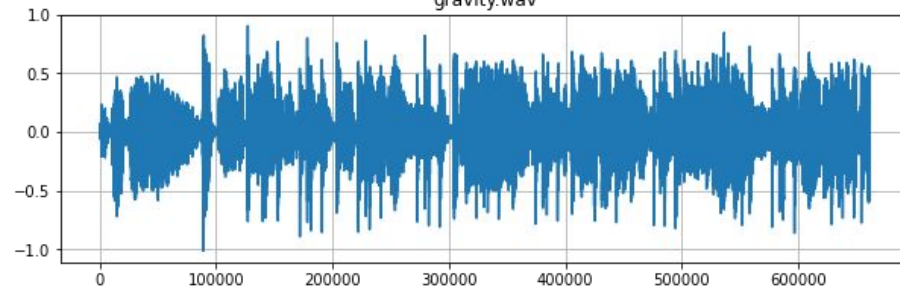
vegetables1.wav



ellhnika.wav



gravity.wav



Conclusion

- > Model works very good with moderate computational effort
- > Pooling layers improved the performance with less trainable parameters
- > Variation of model parameters would be better done by grid search