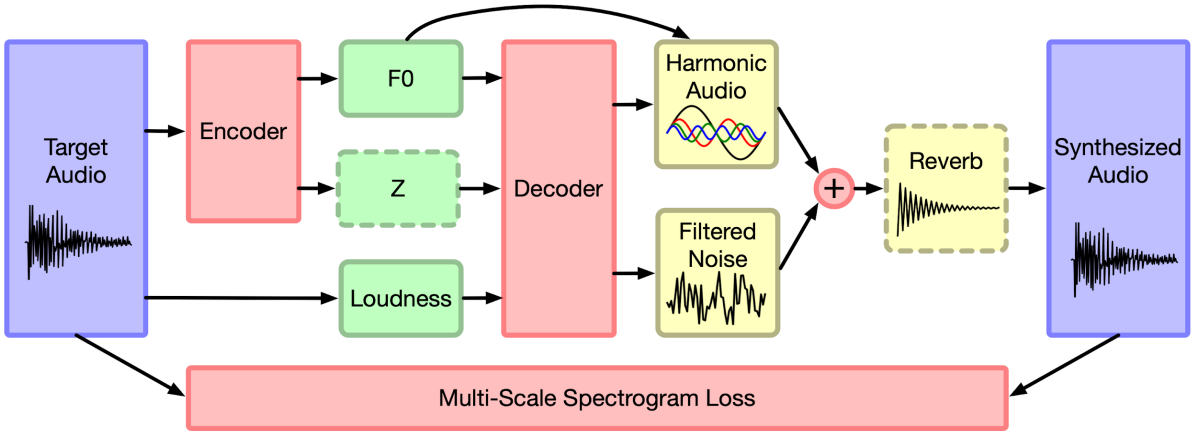



# DDSP paper and codes summary

[DDSP paper and codes summary | WonderLand]

Questions/difficulties:

1. Sample rate v.s frame rate
2. Audio dataset(10-20 minutes long per audio)
3. Computational power.
4. Loading the model trained with z-encoder.
5. Is it enough to use fundamental frequency and loudness?

The brain stem faithfully represents the three main features of speech and music: timing (onsets/offsets and envelope of the response), pitch (encoding of the fundamental frequency), and timbre (harmonics)

http://europepmc.org/backend/ptpmcrender.fcgi?accid=PMC3989107&blobtype=pdf

# T3 - Making DDSP truly generative

GMM:
https://drive.google.com/file/d/1iOqQKZoDNR3M2U6nHb6O2lqxen30JzSb/view?usp=sharing

To retrieve a complete generative mechanism and sample new sounds, one has to define a probability distribution over “z”. One simple approach for this may consist in fitting a generative model over latent variables “z” provided by the pre-trained encoder network.

T3.1 - Build a dataset of “z” obtained from the trained encoder (from T2.3) and your dataset (from T2.1);

T3.2 - Get familiar with GMMs (if necessary) and fit a GMM on the dataset from T3.1;

T3.3 - Evaluate the generation quality of the DDSP synthesizer using the GMM from T3.2 and **predefined pitch and loudness profiles.**

Expected outcome of T3: A fully generative model based on DDSP which can generate musical sounds without any input audio signal.

# T2 - DDSP implementation

Training DDSP in Speech
***Google Colab**:*
Training an DDSP model
https://colab.research.google.com/drive/1xv86RxOZivrAzipIZ3tKjUbtMhKCJW6e?usp=sharing

https://github.com/magenta/ddsp/blob/master/ddsp/colab/tutorials/3_training.ipynb (简化版 training)

**T2.1** - As DDSP requires only a few minutes of audio for training, we let you collect and process sounds of a monophonic and harmonic instrument of your choice.

**T2.2** - Validate data preprocessing (pitch and loudness estimation);
Check loudness (voluntary,done)

Z-encoder (for what experiments is this Z used for)
https://github.com/magenta/ddsp/blob/master/ddsp/training/encoders.py

**Try to train a model complete(with z)**
**T2.3** - Train different variations of the DDSP synthesizer (harmonic only, harmonic + noise, etc.); compare decoder()
Decoder: https://github.com/magenta/ddsp/blob/master/ddsp/training/decoders.py

Tone Transfer via Differentiable Signal Processing
https://wandb.ai/kbganeshan/tone-transfer/reports/Tone-Transfer-via-Differentiable-Signal-Processing--VmlldzozNjY5ODg

**T2.4** - Evaluate the audio reconstruction quality.
PEMO-Q (PErception MOdel-based Quality estimation):
https://ieeexplore.ieee.org/document/1709880

PEAQ

Expected outcome of T2: One or several DDSP synthesizers trained on a dataset that you built.

1. 语音信号处理（博客，详细总结）
   [语音信号处理- 随笔分类- 凌逆战]

2. 加窗的作用：

**信号加窗**

1、矩形窗

$$w(n) = \begin{cases} 1 & 0 \le n \le L-1 \\ 0 & \text{其他} \end{cases}$$

2、汉明窗(Hamming)

$$w(n) = \begin{cases} \frac{1}{2}(1 - cos(\frac{2\pi n}{L-1})) & 0 \le n \le L-1 \\ 0 & \text{其他} \end{cases}$$

3、海宁窗(Hanning)

$$w(n) = \begin{cases} 0.54 - 0.46cos(\frac{2\pi n}{L-1}) & 0 \le n \le L-1 \\ 0 & \text{其他} \end{cases}$$

通常对信号截断、分帧需要加窗，因为截断都有频域能量泄露，而窗函数可以减少截断带来的影响。

窗函数在scipy.signal信号处理工具箱中，如hanning窗：

语音是随时间而变化的信号，但是语音在一小段时间内(10~30ms)语音信号近似看做平稳，即语音信号具有短时平稳特性。如此一来，可以把语音信号分为一些短段(分帧)来进行处理。语音信号的分帧是采用可移动的有限长度窗口进行加权的方法来实现的，一般每一秒的帧数为33`100帧，分帧时采用交叠分段的方法，使得帧与帧之间平滑过渡，保持其连续性，前一帧和后一帧的交叠部分称为帧移，帧移与帧长的比值一般取为0~1/2。
矩形窗的主瓣宽度小于汉明窗，具有较高的频谱分辨率，但是矩形窗的旁瓣峰值较大，因此其频谱泄漏比较严重。相比，汉明窗的主瓣宽度较宽，约大于矩形窗的一倍，但是它的旁瓣衰减较大，具有更平滑的低通特性，能够在较高的程度上反映短时信号的频率特性。

3. 频谱图，语谱图，语音波形图

### 语音波形图

**波形图**表示语音信号的响度随时间变化的规律，**横坐标表示时间，纵坐标表示声音响度**，我们可以从时域波形图中观察语音信号随时间变化的过程以及语音能量的起伏

### 频谱图

**频谱图**表示语音信号的功率随频率变化的规律，信号频率与能量的关系用频谱表示，频谱图的**横轴为频率**，变化为采样率的一半（奈奎斯特采样定理），**纵轴为频率的强度（功率）**，以分贝（dB）为单位

### 语谱图

**横坐标是时间，纵坐标是频率，坐标点值为语音数据能量，能量值的大小是通过颜色来表示的**，颜色越深表示该点的能量越强。一条条横方向的条纹，称为“声纹”。

### 频谱图和语谱图的区别：

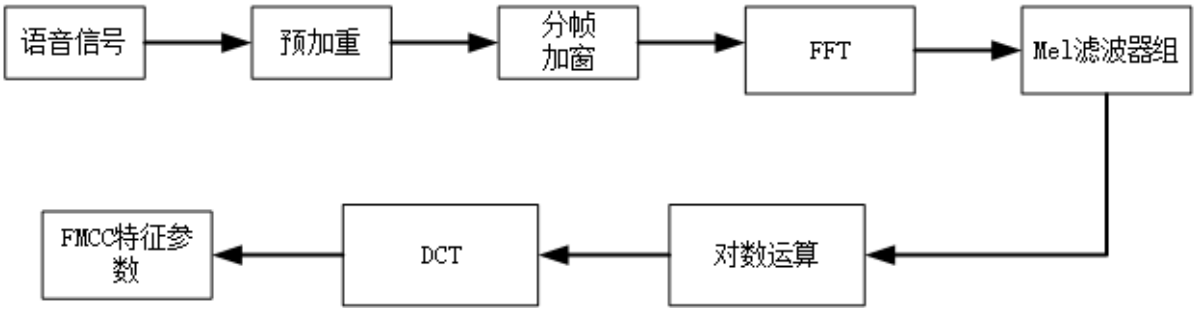
- A.频谱图(spectrum)来表示某一瞬间的波形图中的频率分布，而语谱图则是研究一段时间内语音的变化，特别是频率的变化
- C.频谱图表达了声压或振幅和频率的关系，一个持续的元音可有一个稳定的频谱图，最低频率峰值即为基频，而高频的峰值为谐音或泛音 (harmonics)
- D.根据滤波器的带宽可将语谱图分为两类。带宽为300Hz的宽带语谱图可以显示细致的时间结构，但谐波结构不太清楚。带宽为45Hz的窄带语谱图使时间的结构模糊，但是频率的信息较好
- E.在频谱图包络中可以有一些较宽的峰值，称为共振峰(formant)；在较宽的共振峰带中可以看到个别的谐波频率

4. 语音信号的上、下采样和重采样：
   音频信号重采样知识_aa98865646的博客
   音频重采样分为上采样和下采样，即插值和抽取。
   在实现有理数级重采样时，则是将上采样和下采样做结合（例如48kHz 转 44.1kHz时，将44.1kHz近似为44kHz，将48kHz下采样到4kHz，再上采样至44kHz来实现）。
   由数字信号处理中，时域信号和频域信号的时-频对偶特性可知：时域的抽取，对应频域的延拓;时域的插值，对应频域的压缩。如果对信号的频率成分不做限制的话，频域的延拓可能会引发频谱混叠；频域的压缩来引起频谱镜像相应。
   因此在下采样前，要经过滤波器滤波来防止混迭，即抗混迭(antialiasing filter)滤波 上采样后也要经过滤波处理，即抗镜像(anti-image filter)滤波

5. **MFCC,梅尔倒频谱**
   梅爾倒頻譜

   梅尔倒频谱系数通常可以用于作为语音识别系统中的特征质观察，例如：可以自动辨认一个人透过电话说的数字。梅尔倒频谱系数通常也可以作为声纹识别（Speaker Recognition），也就是、用来识别某段语音频号的发话者是谁的技术。
   语音信号的梅尔频率倒谱系数(MFCC)的原理讲解及python实现- 凌逆战

   滤波器组（***Filter banks***）




6. **Hop size**
   Essentia & 背景知识- 知乎
   sample rate 采样率 = 对这个点所在位置测量的频率。
   bit rate 比特率 = 采样率 * 量化精度 * 声道数，是指单位时间内处理的数据量。
   buffer size = window size = 每次分析步骤所需的sample数。通常是1024或2048。

   ***hop size*** = 两个相邻窗口之间错开的sample数，越小，则说明时序解析度越高，计算成本也越高。通常为buffer size的一半或四分之一。

   frame size = 帧长，媒体帧的长度。
   fps = 帧率。一个帧可能包含多个采样。音频基本都是这样，视频帧则一般一帧一采样。因此fps这个概念通常用于视频和游戏领域。
   bit depth = 位深度，每次采样sample里包含的信息的bit数。
   channels = 声道数，双声道文件大小是单声道两倍。

   > In STFT, you must decide how frequently to perform FFT computations on the signal. If your FFT size is 512 samples, and you have a hop size of 512 samples, you are sliding the analysis frame along the signal with no overlap, nor any space between analyses. If your hop size is 256 samples, you are using 50% overlap. The hop size can be small (high overlap) if you want to very faithfully recreate the sound using an IFFT, or very large if you're only concerned about the spectrum's or spectral features' values every now and then.

7. RNN
   循环神经网络

# T1 - Preliminary work

**T1.1** - The DDSP library comes with a conference paper, a blog, a demo web page, and several Google Colab notebooks for demos and tutorials. You will have to study this material to understand what DDSP is, what possibilities it offers for sound generation and what are the limitations.

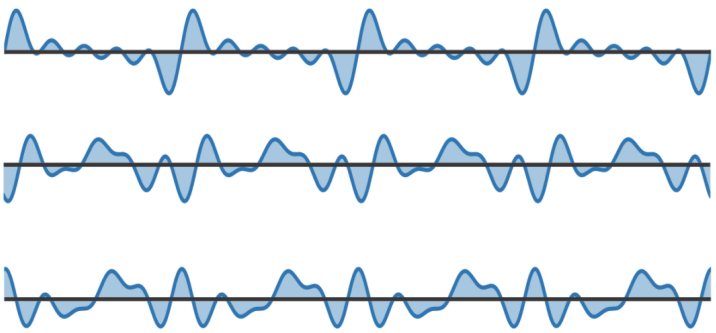
1. **What is DDSP?**
   Differentiable + DSP.

2. **Possibilities offered by DDSP for sound generation?**
   Interpretable(F0,z,loudness),
   Timbre transfer: tuning the voice(into a violin, etc)

3. **limitations?**

- Phase Invariance(优点还是缺点？)：
  The maximum likelihood loss of autoregressive waveform models is imperfect because a waveform's shape does not perfectly correspond to perception.

The maximum likelihood loss of autoregressive waveform models is imperfect because a waveform's shape does not perfectly correspond to perception. For example, the three waveforms below sound identical (a relative phase offset of the harmonics) but would present different losses to an autoregressive model.




The DDSP Autoencoder model uses a CREPE model to extract fundamental frequency. The supervised variant uses pretrained weights that are fixed during training, while the unsupervised variant learns the weights jointly with the rest of the network. The unsupervised model learns to generate the correct frequency, but with less timbral accuracy due to the increased difficulty of the task.

**T1.2** - DDSP official implementation is based on Tensorflow. You can find on Github other implementations based on PyTorch. If necessary, you will have to get familiar with one of these deep learning libraries, following tutorials of your choice.

**Expected outcome of** T1:
**A presentation** about the general principles of DDSP and a **Jupyter notebook** showing the basics of Tensorflow or PyTorch (depending on the chosen library).

Books:

- AI courses in Rennes（Autoencoders 一章）:
https://centralesupelec.edunao.com/course/view.php?id=2204
- Dive into deep learning(英文版第一到五章，重点是MLP):
https://d2l.ai/

1. **DDSP:**
   DDSP: Differentiable Digital Signal Processing
   Differentiable Digital Signal Procressing (DDSP) enables direct integration of classic signal processing elements with end-to-end learning, utilizing strong inductive biases without sacrificing the expressive power of neural networks. This approach enables high-fidelity audio synthesis without the need for large autoregressive models or adversarial losses, and permits interpretable manipulation of each separate model component.

2. **CREPE: A CONVOLUTIONAL REPRESENTATION FOR PITCH ESTIMATION**

Paper: https://arxiv.org/pdf/1802.06182.pdf
**Abstract**: The task of estimating the fundamental frequency of a monophonic sound recording, also known as pitch tracking, is fundamental to audio processing with multiple applications in speech processing and music information retrieval. To date, the best performing techniques, such as the pYIN algorithm, are based on a combination of DSP pipelines and heuristics. While such techniques perform very well on average, there remain many cases in which they fail to correctly estimate the pitch. In this paper, we propose a data-driven pitch tracking algorithm, CREPE, which is based on a deep convolutional neural network that operates directly on the time-domain waveform. We show that the proposed model produces state-of-the-art results, performing equally or better than pYIN. Furthermore, we evaluate the model's generalizability in terms of noise robustness. A pretrained version of CREPE is made freely available as an open-source Python module for easy application.

3. **End-to-End training:**

https://towardsdatascience.com/e2e-the-every-purpose-ml-method-5d4f20dafee4

The traditional approach design for a spoken language understanding system is a pipeline structure with several different components, exemplified by the following sequence:

> ***Audio (input) -> feature extraction -> phoneme detection -> word composition -> text transcript (output).***

A clear limitation of this pipelined architecture is that each module has to be optimized separately under different criteria. The E2E approach consists in replacing the aforementioned chain for a single Neural Network, allowing the use of a single optimization criterion for enhancing the system:

> ***Audio (input) — — — (NN) — → transcript (output)***

Mike Lewis et al. introduce an E2E learning approach for natural language negotiations [2]. The resulting system is a dialogue agent based on a single Neural Network able to negotiate to achieve an agreement. This was done by training the NN using data from a large dataset of human-human negotiation records containing a variety of different negotiation tactics.

4. **A-weighting 提取噪声(I-encoder)**
   https://en.wikipedia.org/wiki/A-weighting

5. **MLP, multi-layer perceptron**
   https://zh.wikipedia.org/wiki/%E5%A4%9A%E5%B1%82%E6%84%9F%E7%9F%A5%E5%99%A8

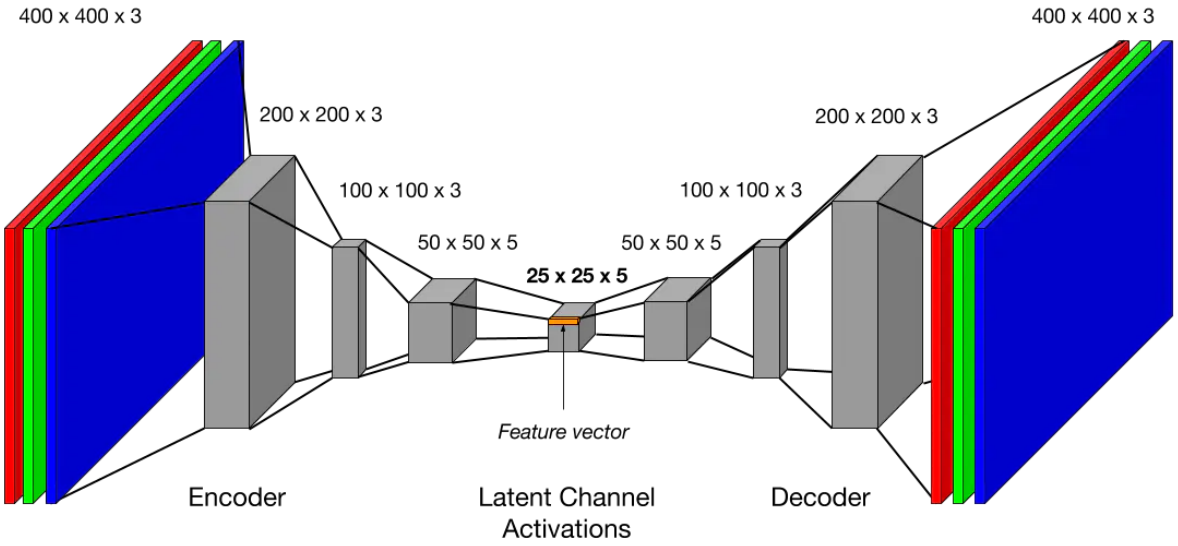
   多层感知器(Multilayer Perceptron,缩写MLP）是一种前向结构的人工神经网络，映射一组输入向量到一组输出向量。MLP可以被看作是一个有向图，由多个的节点层所组成，每一层都全连接到下一层。除了输入节点，每个节点都是一个带有非线性激活函数的神经元（或称处理单元）.

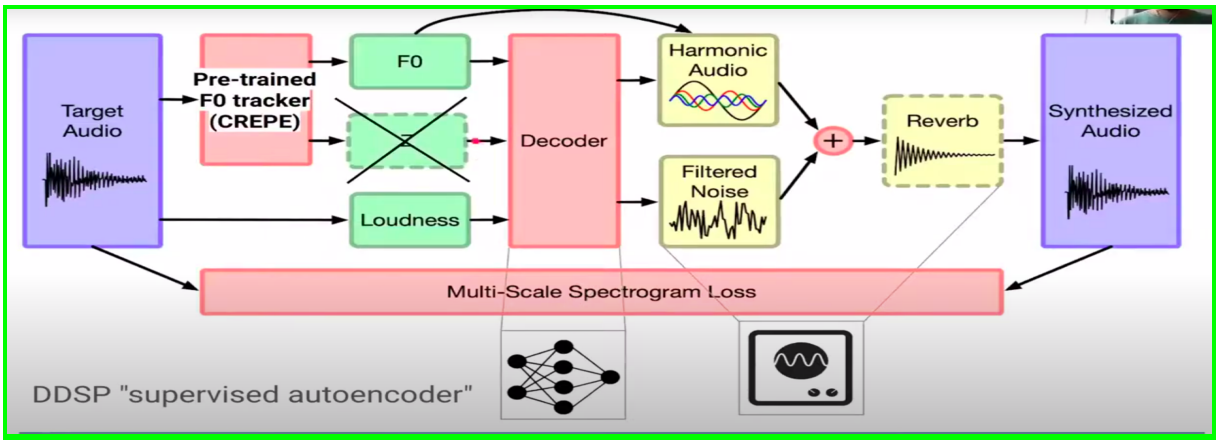
6. **GRU: gated recurrent unit**
   用门控机制控制输入、记忆等信息而在当前时间步做出预测.
   Gated recurrent unit
   **Gated recurrent unit**s (**GRU**s) are a gating mechanism in recurrent neural networks, introduced in 2014 by Kyunghyun Cho et al.[1] The GRU is like a long short-term memory (LSTM) with a forget gate,[2] but has fewer parameters than LSTM, as it lacks an output gate.[3] GRU's performance on certain tasks of polyphonic music modeling, speech signal

modeling and natural language processing was found to be similar to that of LSTM.[4][5] GRUs have been shown to exhibit better performance on certain smaller and less frequent datasets

门控循环单元： RNN的改进版
[经典必读：门控循环单元（GRU）的基本概念与原理]
LSTM（long short-term memory）通过门控机制使循环神经网络不仅能记忆过去的信息，同时还能选择性地忘记一些不重要的信息而对长期语境等关系进行建模，而 GRU 基于这样的想法在保留长期序列信息下减少梯度消失问题。

7. **Autoencoder(自编码器)**




自编码监督学习
source:https://www.youtube.com/watch?v=cYkSWqMHCPk

8. **Basic knowledge about acoustics**
Room reverb : usually implicitly modeled by neural synthesis algorithms
(Room Reverb - SoundBridge)
Room acoustic 室内声学 Room acoustics - Wikipedia

9. 对瞬时频率升取样（upsampling）：Bilinear interpolation双线性插值 - 维基百科，自由的百科全书 (wikipedia.org) 足矣
   但是 the amplitudes and harmonic distributions of the additive synthesizer required smoothing to prevent artifacts：
   HOW：a smoothed amplitude envelope by adding overlapping Hamming windows（tf.signal.hamming_window | TensorFlow Core v2.4.1）

10. **Violin solo resource**：Free Sheet Music, Royalty Free & Public Domain Music | Musopen 且将获取的violin音频转为标准的dataset亦有线上工具

# Daily report:

- Day 1:
  Read the paper of DDSP_ICLR and had a general understanding of the steps of DDSP achieving.
  Learned the theory of MLP and followed the jupyter notebooks in “Dive into deep learning”.

- Day 2
  Deciding to use Tensorflow or PyTorch library?(官方库是用tensorflow写的，但老师只会PyTorch。。) 两者区别 https://zhuanlan.zhihu.com/p/37102973
  Tf library: https://github.com/magenta/ddsp（非常完善并且有人维护更新，代码复杂，有点看不懂)
  PyTorch library: https://github.com/sweetcocoa/ddsp-pytorch
  https://github.com/acids-ircam/ddsp_pytorch
  The timeline of each task.(T1,T2,
).
  **Projet 排课SG8: 13**个半天， **ST7: 20**个半天**.**

- Day 3
  T1: a **Jupyter notebook** showing the basics of Tensorflow or PyTorch (depending on the chosen library). More details about this notebook task?
https://colab.research.google.com/drive/1XzonkFgEYrcHE29GiqWdwnGtML3mbYBk?usp=sharing
- Day 4
  Finishing T1