Ground rules for the team project:

- Your project's topic should be relevant to the class; you should negotiate this with the instructor by email or slack or during an office hour.
- Teams should have four members. Exceptions might be made for smaller teams (e.g., if the number of students enrolled is not divisible by four).
- Your team will give three 10-minute in-class presentations. These presentations need to be self-contained but concise (time will be strictly enforced).
    - January 23/25 (week 4): motivate your project (who cares?), present your plan (what is each team member's milestone each week?), and tell us how we'll know whether you succeeded at the end of the quarter, in six weeks.
    - February 13/15 (week 7): show us where you are in your plan, discuss what you've learned, and tell us what to expect at the end of the quarter. If there are changes to the plan, discuss them. Don't assume anyone remembers all the details from your first presentation!
    - March 5/7 (week 10): final project presentation. Again, make this presentation self contained (don't assume we remember your earlier presentations). Why was this project worth pursuing? What was your goal? Did you achieve it?
- The goal of the project is learning, for your team and for the rest of the class. This includes:
    - Finding the important bits of information that aren't explained in the papers.
    - Discovering problems with existing datasets and codebases.
    - Making tough decisions about using poorly documented research code vs. implementing things yourself.
    - Learning how to better estimate how long tasks will take for you and your teammates to complete.

Advice

- Focus on scoping a project you can implement and rigorously evaluate in the time you have. Don't aim for a publication; that virtually never happens in a class like this! Instead, think of this project as a warmup to publishable research you might do later.
- The more detail you spell out early on, the easier it will be to execute. What experiments do you want to run? What datasets will you use? What evaluation metrics will you apply? What codebases will you work with? What do you plan to implement? As you work through these details with your team, don't despair. Instead, prioritize: if you start feeling like the project idea you were excited about is too much for your team to finish in one quarter, try to think about an easier first step that would still be meaningful.
- Remember to include time to prepare presentations and the final report in your plan!
- When deciding among topics, make evaluation a top priority. A strong project has a clear evaluation plan. It's okay to perform terribly on your evaluation; it's not okay to evaluate terribly.
- Many great research projects start by trying to reproduce (some of) an already published paper's experiments. Feel free to peruse Noah's CSE 517 project instructions and template and craft your project as (1) reproducing experiments from a paper you find exciting and (2) extending those experiments in an interesting direction. Note that a

failed reproduction does not mean your project is a failure! If you were rigorous in your attempt and your evaluation, the failed attempt is valuable information for future research.

- Open-source projects can also be a great jumping off point for projects. If there's an existing library you want to improve, focus on (1) crafting an argument about the value of the improvements, (2) scoping exactly what you will build, and (3) defining the tests you will execute to demonstrate that what you've contributed has value. For example, you may propose to implement algorithm/model from paper X in codebase Y and reproduce specific experiments from X using your implementation.
- Finding a mentor is often incredibly helpful for open-ended class projects. This can be informal (e.g., emails to the authors of a paper you're working off of). We can try to help make connections, but remember that the course staff aren't yet well connected in the AI-for-music community.
- Datasets and evaluation methodologies can be extremely important contributions to a field. They usually take a lot of time and experimentation, often with human informants, experts, and/or users, to do well. If you want to go in this direction, you'll need to be extremely focused and know in advance exactly who's helping you. Expect to work hard to convince the instructor to agree to this kind of project, even though he's incredibly sympathetic to this kind of research. **The instructor is probably going to say "no" to projects involving human judgments of fully automatically generated music.**

Grades

Project grades are shared by all members of the team.

- Each presentation is worth 15 points (total 45). The rubric:
  - ☐ Motivate the project clearly. What are you trying to do and why is it worth doing?
  - ☐ Success criteria: what will a successful project look like? For the final presentation, explain whether you met the criteria. For machine learning problems, the evaluation dataset is arguably even more important than the training dataset.
  - ☐ Plan: for the first presentation, give a detailed plan (who will do what, each week?). For the second, show what progress has been made and clearly explain changes to your plan. For all three presentations, clearly explain what implementation, evaluation, and/or analysis work you've completed and what remains.
  - ☐ Lessons learned: what important or useful information can you share with the rest of the class?
- The final report, due March 11, is worth 10 points. Make sure to acknowledge anyone who gave you significant help with the project, whether they are enrolled in the class or not. Your team will have the opportunity to nominate a small number of students from the class (outside your team) who were especially helpful, to receive some bonus credit.

Some project ideas from course staff (not necessarily fleshed out/reasonable projects, you'll need to do some work to make sure that these are feasible, and evaluable.)

- **[Separation]** Sound Demixing Challenge 2023 just released two challenges: Music Demixing and Cinematic Sound Demixing. The goal of the challenge is to perform music source separation (first challenge) or separate movie audio into dialogue, sound effects, and music (second challenge) [Link]
- **[Separation]** Multi-microphone music source separation in performance. In musical performance, you usually have multiple musicians playing different instruments at different locations. Standard single-channel music source separation can divide the input mixture into multiple tracks corresponding to different instruments. Problems may arise in the case that you want to separate music into multiple tracks of the same instrument.
- **[Generation]** In audio/music compression (see SoundStream, EnCodec, DAC, FunCodec), a typical pipeline is to convert audio input into a very compact quantized representation usually with some hierarchical structures. This pipeline is working very well for speech representation; however, not many works have been studied for the music case. This project aims to look at the same idea for music representation, with a possible downstream task of upsampling the music representation (i.e., using coarse-level representation to improve fine-level representation).
- **[Generation]** Many of recent works on audio/music generation like MusicLM or AudioCraft consist of a single autoregressive language model that operates over discrete music representations. Your project can be to extensively (objectively not subjectively) evaluate these models.
- **[Representation Learning]** DSP (digital signal processing) is very essential in the listening experience, ranging from EQs, noise cancellation, and digital sound effects. Recently, neural-based DSP (i.e., DDSP paper) attempts to make DSP block differentiable. This project can look into how you may use neural-based DSP for the task of your choices, including but not limited to compression and enhancement.
- **[Beat and Downbeat Detection]** Beat and downbeat detection are useful for many other tasks, including alignment, transcription, tempo detection, performance analysis, etc. One of the most popular libraries for this is madmom, a signal processing library for music, which includes beat trackers and downbeat trackers. However, the models trained for these tasks are “by no means trained on classical music” (documentation). This project could expand on this library to train beat and downbeat detectors that *are* trained on classical music, specifically classical piano (the instrument/genre used in a lot of other tasks).
- **[Performance Analysis]** Especially with classical music, there can be many recordings for a single score. These recordings will often be of different people, potentially different instruments, different choices, maybe different arrangements, different following of the musical “roadmap”, etc. This project is designed to compare different performances of the same piece of music and see where they are the same, and where they are different. There might be a dataset for this and already a way to evaluate it, there might not be. If not, this probably isn’t a good project, so look into the evaluation potential before committing!
- **[Transcription]** Can you extract the chords from a piece of music? Auditory music? Symbolic music? Different than extracting the pitches and lengths this is more could you denote where the piece is based around a specific chord. For auditory this might look

like a spectrogram or waveform annotated with chords at specific times, and for symbolic, this might look like a piece of sheet music with chords above the staves. Again look into if there is a way to evaluate this before committing.

- **[Transcription]** There are many different transcription models for music, and a lot of them are focused on a specific genre of music, or a specific instrument. Could you take a transcription model trained on classical music and fine-tune it for jazz or vice versa? Consider looking at two transcription models trained on different genres/types of music, and fine-tuning them to work on the genre of the other, and then evaluating compared to the “out of the box” version on each genre.
- **MIREX tasks** MIREX (Music Information Retrieval Evaluation eXchange) is a collective group of tasks for years from 2005-2021 that have given baselines, datasets, and projects to work on. Examples are query by humming, lyric transcription, drum transcription, beat tracking, key detection, music detection, and many more. Your project can be to implement one of these tasks, and compare to the given results (some years/projects have more results than others. Make sure that if your task given is one that doesn’t have given results (as with many of the 2021 tasks) or you pick an older task that you find a recent, reasonable baseline, or you have a well defined evaluation metric. Some of the above projects may have evaluation metrics in here as well.