

# AI CLUB PROJECT MEMBER APPLICATION

**ANISH VEERAKUMAR**

**ME23B156**

## 1 Questionnaire Section

### A. Managerial Questionnaire

**1. Essentials: State your motivation to become an AI club project member. What do you feel makes you a decent fit for this job? Justify by stating your skills/strengths and previous experience (if any). (Explain why you're choosing AI/ML not just for a PoR but out of interest.)**

- I stumbled onto AI/ML during my first semester in insti when the apps for AI Club DCs rolled out. I had no prior knowledge at that time and just applied out of enth. I watched multiple YouTube videos and also learned from my peers. The DC app turned out to be my entry point into the field of AI/ML.
- Being part of AI Rahman would provide me with the chance to continue learning, directly from experienced project leads. By working alongside a team, I will also be in touch with likeminded AI enthusiasts.
- Working on a real-life project will also give me a good amount of exposure and a learning curve that will help me pursue AI on a higher level.
- Experience :
  - AI Club DC :
    - Introduced to basic concepts of AI/ML
    - Took part in convolve and proceeded to the second round.
    - Was involved in a mini project that involved training an AI model to translate French sentences to English using concepts such as se2seq and bahadanu attention.
  - Taken the course ID2090: Scientific Computing .
  - Shaastra Volunteer and E-Cell Assosiate Manager :
    - Worked together in a team with a diverse group of people to complete tasks for my vertical.
- As a candidate for the Project Manager role in AI Rahman, I bring a unique blend of experience in both music and artificial intelligence, which I believe positions me ideally for this position. Moreover, I am deeply committed to continuous learning and development, particularly in the field of AI. Joining AI Rahman presents an exceptional opportunity for me to learn and enhance my skills and expertise in AI while contributing meaningfully to the project's success.

### 2. Commitments/PoRs:

**a. How much time do you think you can commit to AI Club weekly?**

- I am willing to put in 4 hours per week towards AI Club. However, I understand that the hours required will be higher as we approach events such as Open House. I am willing to put in the extra hours whenever required.

**b. What other PoRs/activities do you plan to take up next year? In case of clashes, how will you prioritize your role as a project member?**

- I will be applying for the post of Shaastra Spons&PR coordinator next year. The peak of the PoR can be expected around December and the first week of January.
- As I head closer to the Shaastra period(January First week), I will have to spend more time with them to complete my responsibilities. I will be available with the AI Club for the rest of the year.
- There should be very little to no Shaastra work around the Open House time(March).

## **Section B: Common Technical Questionnaire: Quantile Regression**

**Notebook:**

[https://colab.research.google.com/drive/1TuS896lZ\\_wJi3IHf\\_8HY70onmcNRsPKm?usp=sharing](https://colab.research.google.com/drive/1TuS896lZ_wJi3IHf_8HY70onmcNRsPKm?usp=sharing)

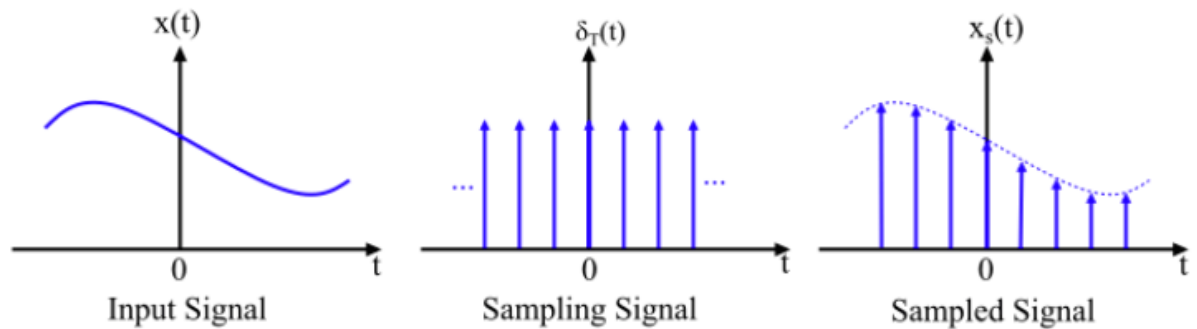
**Bonus: Play around with the value of  $\tau$  to find what value achieves convergence quicker.**

- On plotting a graph between the quantile values and the loss at the end of the 3<sup>rd</sup> forward propagation, I noticed that the loss steadily decreased as I increased the quantile.
- However after a point (quantile = 0.5), the loss starts to increase before starting to saturate.
- This is because when the quantile is at 0.5, it doesn't give any special preference to the outlier values.
- This means the model is effectively capturing the central tendency of the data.

## **Section C: Project Specific Questionnaire**

**a.)What is the sampling of signals? Mention the frequencies at which a music audio is sampled and why. Using the concept of bit depth mention the relationship between audio quality, dynamic range, and sampling frequency.**

- Sampling is the process of breaking down continuous signals into discrete values. It is used to convert the analog form of a signal into digital form.
- The time gap between two discrete values is called as the sampling rate( $T_s$ ), its inverse is called as sampling frequency( $F_s$ ).
- For an accurate representation of the analog signal in discrete form, the sampling frequency must exceed twice the frequency of the input analog signal. Failing to meet this criterion may result in the loss of amplitude values for specific time intervals within the analog signal.



- The most common sampling frequencies used in digital audio are 44.1 kHz and 48 kHz. Human hearing generally ranges from about 20 Hz to 20 kHz. Therefore, to accurately capture the entire audible spectrum, a sampling frequency of at least 40 kHz (twice the upper limit of human hearing) is required. That's why the standard CD audio format uses a sampling frequency of 44.1 kHz.
- Bit depth is the number of bits of information present in each sample of a digital audio signal.
- Dynamic Range represents the difference between the quietest and loudest sounds that can be captured or reproduced without distortion.
- When you use higher bit depths, you get better sound quality because you can capture more detail in the volume of each sound. With more bits, there are more levels to record how loud or quiet the sound is accurately. This helps reduce unwanted noise and makes the conversion from analog to digital sound more accurate.
- Higher bit depth directly translates to a wider dynamic range, allowing for more detailed and realistic audio.
- Higher bit depth and sampling frequency generally lead to better audio quality
- For most music and audio applications, 44.1 kHz sampling rate and 16-bit depth are considered standard and offer good quality

**b. Consider a signal  $x(t)=10 \sin(20000t - 30)$ . Sample the signal at a sampling frequency of 1kHz and obtain the discrete time equivalent  $x[n]$  (for  $0 \leq n \leq 5$ ), where  $x[n]=x(nT_s)$  [ $T_s$  is the sampling time] . Calculate the discrete time fourier transform for the sampled signal  $x[n]$ .**

$$x(t) = 10 \sin(20000t - 30)$$

$$x[n] = x(nT_s) \quad T_s = 10^{-3} \text{ sec}$$

$$x[n] = 10 \sin(20t - 30)$$

$$\begin{aligned}
 \text{for } n=0: & \quad x[0] = 10 \sin(-30) = -5 \\
 \text{for } n=1: & \quad x[1] = 10 \sin(-10) = -1.736 \\
 \text{for } n=2: & \quad x[2] = 10 \sin(10) = 1.736 \\
 \text{for } n=3: & \quad x[3] = 10 \sin(30) = 5 \\
 \text{for } n=4: & \quad x[4] = 10 \sin(50) = 7.66 \\
 \text{for } n=5: & \quad x[5] = 10 \sin(70) = 9.396
 \end{aligned}$$

$$\begin{aligned}
 X(e^{j\omega}) &= \sum_{n=-\infty}^{\infty} x[n] e^{-j\omega n} \\
 X(e^{j\omega}) &= -5 - 1.736 e^{-j\omega} + 1.736 e^{-2j\omega} + 5 e^{-3j\omega} \\
 &\quad + 7.66 e^{-4j\omega} + 9.396 e^{-5j\omega}
 \end{aligned}$$

c. Write a shortnote on how fourier transforms can be used to analyze music audio signals. (hint: check Short Time Fourier Transforms and Mel Spectrograms).

- Fourier transforms play an important role in the analysis of music audio signals, providing insights into various aspects of the sound, such as frequency content, timbre, and temporal characteristics. Two common techniques used in music signal analysis are the Short-Time Fourier Transform (STFT) and Mel Spectrograms.
- **Short-Time Fourier Transform (STFT):**
  - Breaks down music signals into short segments.
  - Computes the frequency content for each segment.
  - Shows how the frequency makeup of the music changes over time.
  - Useful for tasks like identifying instruments, tracking changes in sound, and creating audio effects.
- **Mel Spectrograms:**
  - Use a nonlinear frequency scale called the Mel scale.
  - Aligns with how humans perceive sound.
  - Provides a representation of music that matches human hearing better.

- Helps analyze tonal characteristics, distinguish between instruments, and classify music genres effectively.

**d. What is MIDI representation in music? Mention its advantages over digital audio. What are limitations of MIDI representations?**

- MIDI(Musical Instrument Digital Interface) is a music transmission and storage standard that doesn't contain the musical note directly, rather it contains 'instructions' for the device to play them.
- These instructions include note-on and note-off commands, velocity (how hard a note is struck), pitch, modulation, and other parameters.

**Advantages:**

- MIDI files are significantly smaller in size compared to digital audio files because they contain instructions rather than actual audio data. This makes them ideal for transmitting over networks or storing large collections of music.
- MIDI data is easily editable. You can change notes, timing, instrument sounds, and other parameters without affecting the original recording.
- MIDI allows for easy interchangeability of instruments, giving composers and producers the freedom to experiment with different sounds. This feature will particularly be useful for us in this project.
- MIDI can control a wide range of instruments and sound modules, allowing for great creative possibilities.
- MIDI enables real-time control over various musical parameters during performance or recording.

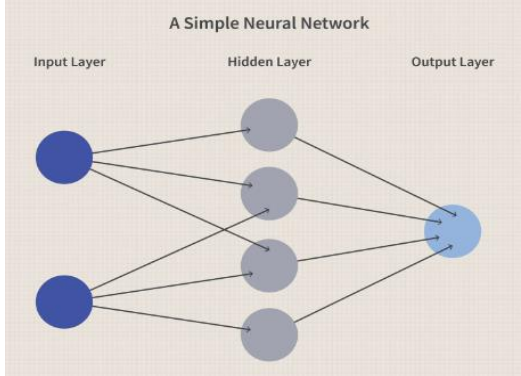
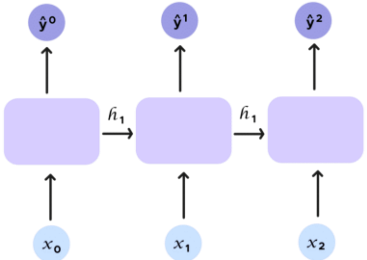
**Limitations:**

- MIDI files don't contain actual audio data, so the sound quality is dependent on the playback device or software synthesizer used. The quality can vary significantly based on the capabilities of the device.
- MIDI primarily deals with note information and cannot capture subtle details that an artist uses.
- MIDI is primarily designed for electronic instruments and may not accurately represent the nuances of acoustic instruments.

## Sequence 2 Sequence Models

- a. How do RNNs differ from traditional ANNs in capturing long term dependencies? Why do you think this is important for music generation?**

Traditional ANNs	Recurrent Neural Network
ANNs process each input independently without considering the sequence.	RNNs are specifically designed to handle sequential data by retaining memory of past inputs through recurrent connections.

ANNs typically consist of input, hidden, and output layers, with feedforward connections between them.	RNNs feature recurrent connections that create loops within the network, allowing them to persist information over time steps.
ANNs lack explicit memory and do not retain information about previous inputs, making them suitable for tasks where past context is not crucial.	RNNs possess memory capabilities that enable them to capture temporal dependencies and contextual information across sequential data, making them ideal for tasks like language modeling, time series prediction, and speech recognition.
Training ANNs is generally straightforward, as each input is processed independently, and there are no feedback loops to consider.	Training ANNs is generally straightforward, as each input is processed independently, and there are no feedback loops to consider.
ANNs are widely used in various machine learning tasks, including classification, regression, and pattern recognition.	RNNs excel in tasks requiring an understanding of sequences and temporal dependencies, such as natural language processing, speech recognition and sentiment analysis.
ANNs encompass various architectures, including Multilayer Perceptrons (MLPs), Convolutional Neural Networks (CNNs), and Generative Adversarial Networks (GANs).	RNNs come in different variants, such as vanilla RNNs, Long Short-Term Memory (LSTM) networks, and Gated Recurrent Units (GRUs), each offering unique advantages for specific applications.
 <p>A Simple Neural Network</p> <p>Input Layer Hidden Layer Output Layer</p>	

Now, why are RNNs important for music generation?

- One of the key strengths of RNNs, compared to traditional ANNs, is their ability to handle sequential data with 'context'.
- This is particularly valuable in music generation because music is, at its core, an arrangement of notes, chords, and rhythmic elements.
- Unlike ANNs, which treat each input independently, RNNs possess an internal memory that allows them to analyze and retain information from previous notes in a sequence.
- This "memory" is what enables RNNs to grasp the harmonic progressions, melodic contours, and rhythmic patterns that govern musical structure.

**b)Go through the reference links given below and provide a summary of encoder, decoder blocks . Give an intuitive explanation on attention mechanism.**

**i)Encoder :**

- Neural Networks do not recognise and understand English words like we do.
- Words are transformed into matrices using methods such as one-hot encoding, enabling neural networks to work with text data.
- These matrices are then passed through RNNs, such as LSTMs or GRUs, which help in capturing sequential patterns within the text.
- The final hidden state of the RNN, known as the context vector, contains the essence of the input sequence.
- This context vector plays a vital role in subsequent steps, such as decoding or generating output.

**ii)Decoder**

- After obtaining the context vector from the encoder, the decoder plays an important role in generating meaningful output based on this information.
- It takes the context vector as input and begins to produce the desired output sequence, which could be a translation, a response etc.
- Like the encoder, the decoder also employs recurrent neural networks (RNNs), such as LSTMs or GRUs, to handle sequential data.
- During the decoding process, the RNN generates one token at a time, utilizing the context vector and previously generated tokens as guidance.
- This iterative generation continues until the decoder produces the entire output sequence, effectively completing the task at hand.
- Overall, the decoder serves as the counterpart to the encoder, working in tandem to transform input data into meaningful output in machine learning applications, particularly in natural language processing tasks.

**iii)Attention**

- While dealing with large inputs in neural networks, such as those processed by LSTMs or GRUs, it's common for important information to be overlooked.
- Attention mechanisms are introduced to address this issue by allowing the model to focus on relevant parts of the input sequence.
- These mechanisms assign weights to intermediate hidden states based on their relevance to the current context.
- Methods such as cosine similarity are used to find the relation between the Current query and the other queries , in order to find the weights of each hidden state.
- These weights are then normalized using the softmax function to ensure proper weighting.
- Finally, the context vector is formed by taking a weighted sum of the intermediate hidden states, where the weights reflect their importance in the current context.

**c)(Brownie Question) Nowadays, Graphics Processing Units have taken over the world of Machine Learning. With their extreme parallelization capabilities, they have revolutionized computation, reduced training time by a significant amount. As seen in the**

**above sequence-to-sequence models, there is not much parallelization involved. Each word is processed only after the previous word, the computations are mostly sequential. Can you think of any way to leverage the power of GPUs here (i.e. make computations parallel)?**

- CPUs are like the brains of computers, handling various tasks step by step. GPUs, on the other hand, have many smaller cores that work together to tackle simpler tasks all at once. This makes GPUs great for tasks like AI and machine learning because they can handle lots of similar calculations simultaneously.
- Imagine searching for a word in a document. A CPU might do this one word at a time, while a GPU could scan through rows of words at once, checking for the target word in parallel.
- Some tasks, like calculating the Fibonacci sequence, need to be done step by step. CPUs are better for these because they're good at handling complex, linear tasks that can't easily be split up.
- GPUs shine when it comes to tasks like adding or multiplying matrices because each calculation in the matrix can be done independently. This means the GPU can tackle a lot of similar operations at once, speeding up the process.
- When training models like sequence-to-sequence, we can group multiple input sequences together and send them to the encoder all at once. This takes advantage of the GPU's ability to handle parallel tasks, making things much faster.
- For models with attention mechanisms, like multi-head attention, each attention head can work independently, thanks to the GPU's parallel processing. This speeds up computation.
- Similarly, when decoding sequences, we can process multiple time steps simultaneously by batching sequences together, making the most of the GPU's parallel capabilities.
- During training, we can use data parallelism to spread the workload across multiple GPUs. This means each GPU handles a portion of the training data, making training faster overall.

## **Coding Questionnaire**

Link :

[https://colab.research.google.com/drive/1krlyk3vMNAY3QmpwObTlEXHw0Mn92\\_xK?usp=sharing](https://colab.research.google.com/drive/1krlyk3vMNAY3QmpwObTlEXHw0Mn92_xK?usp=sharing)