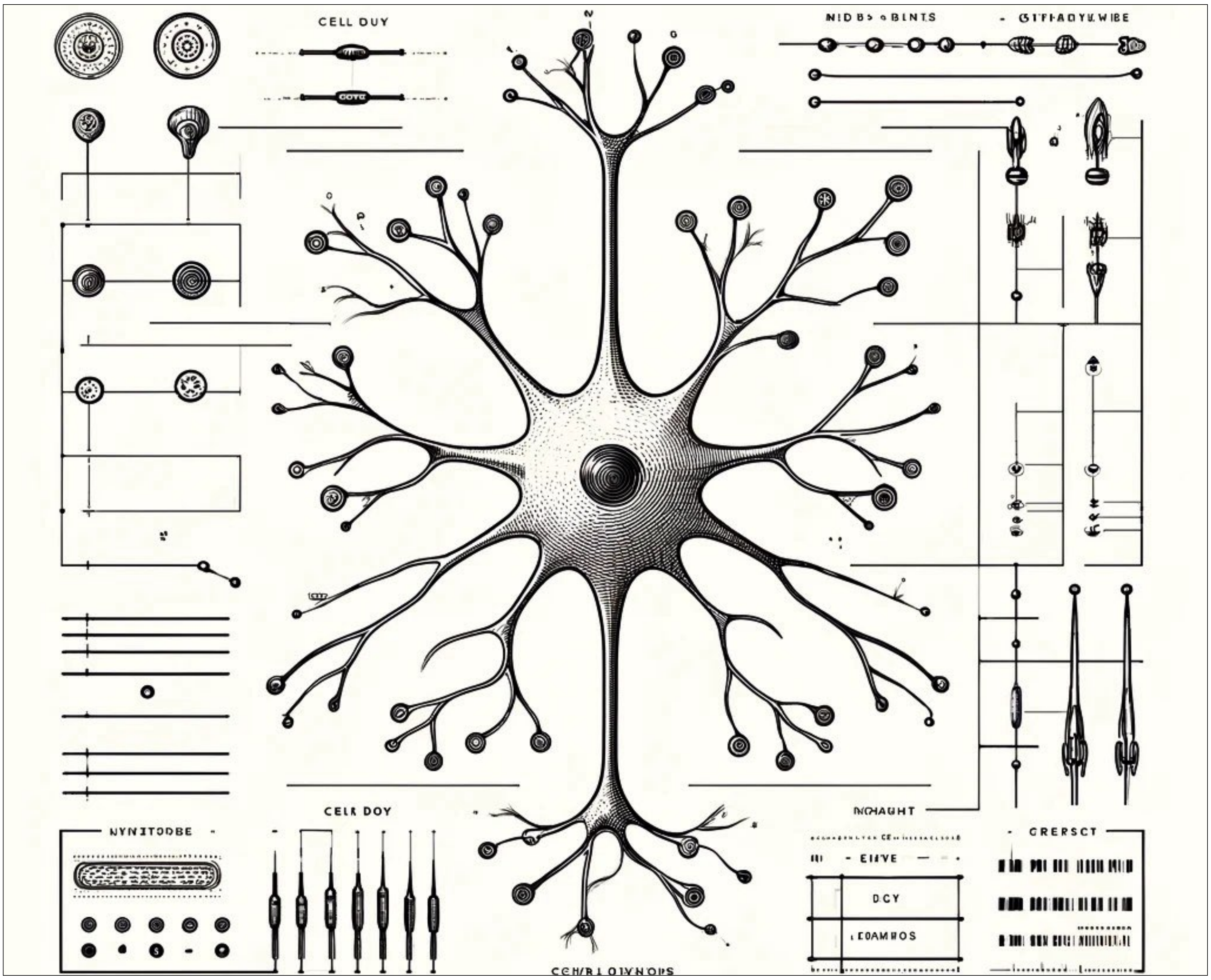
EE4305 - FUZZY/NEURAL SYSTEMS FOR INTELLIGENT ROBOTICS



Generated Using DALLE

From Feels to Keys

FUZZY SYSTEMS CRAFTING TUNES FROM NEURONS

Prepared by: Samuel EF. Tinnerholm
24 April 2024
Matriculation Number: e1329469
Email: samuel.tinnerholm@u.nus.edu

0. ABSTRACT

This paper implements a fuzzy neural network model designed to generate piano music reflective of specific emotional states input by a composer from scratch. The objective is to develop an intelligent system where composers can adjust a few numerical values representing desired emotional outputs, and consequently, the system generates music that embodies these emotions. This study explores the application of fuzzy logic, which effectively handles degrees of truth, making it suitable for modelling the complex, often non-binary nature of human emotions crucial for creating emotionally resonant music.

The methodology integrates emotional inputs using the Russell Circumplex Model, employing a combination of fuzzy logic and LSTM neural networks to manage the temporal dynamics of music composition. Despite the approach, the project did not fully achieve its objective of allowing composers to simply alter numerical values to change the music's emotional tone significantly, mostly due to the model's linear architecture.

To better meet the project's goals, alternative models with enhanced capabilities for handling user generated inputs are recommended. Future work may include a focus on optimising these aspects to realise the full potential of using AI in generating emotionally aligned music compositions.

1. INTRODUCTION

1.1 - Background information

In traditional binary logic, statements are binary: they are either true (1) or false (0). This system underpins the foundation of Boolean algebra, which has been extensively applied in digital computing and mathematical logic. However, human experiences and perceptions often do not conform to binary distinctions. For example, the question "Are you sad?" might not have a straightforward true or false answer. Human emotions are complex and can coexist in varying intensities; a person might feel predominantly sad while also experiencing a sense of relief.

Fuzzy logic provides a framework to accommodate this complexity by allowing for the representation of values between and including 0 and 1, indicating degrees of truth rather than absolute truth or falsehood. For instance, a person mourning the loss of a loved one might experience mixed emotions: 0.91 sad due to the loss, yet 0.34 joyful, reflecting a sense of relief that their loved one is no longer suffering. This illustrates how fuzzy logic can model the complex reality of human emotions.

Due to fuzzy logic's ambiguity, fuzzy logic is perfect for the uncertainty in real-world environments. For instance, in intelligent control systems for mechatronics, fuzzy logic is used to manage complex decision-making processes. These include applications in stabilising and tuning controllers for robotics systems, optimising parameter settings in real-time, and ensuring reliable operation in uncertain conditions.[1]

Moreover, fuzzy systems "{are} used in an intelligent system for detecting and eliminating potential fires in the engine and battery compartments of {…} hybrid electric vehicle{s}"[2]. Here, fuzzy logic helps to assess the risk of fire in engine compartments by processing uncertain and imprecise data, leading to more reliable safety measures.

Fuzzy logic has expanded its utility to everyday products and industrial applications, reaching areas once thought beyond its scope. Its capacity to simulate human-like reasoning in automated systems' decision-making processes makes it an essential tool for enhancing interactions between humans and machines, particularly in the domain of collaborative robots. A prime example of this application is the Swedish company ABB, which offers a diverse range of collaborative robots, known as Co-Bots, tailored to various tasks and industries, ensuring easy integration into numerous operational environments.[3]

1.2 Problem Statement

The task of generating music that reflects a composer's emotions represents a sophisticated intersection of emotion capture and musical creation using deep learning technologies. This process entails understanding the wide and nuanced spectrum of human emotions and translating these into musical compositions. The challenge is inherently complex, requiring knowledge in music theory, composition, affective science, and artificial intelligence.

AI-based systems for affective music generation consist of three main components:

1. Identifying Target Emotions: Pinpointing specific emotional states that the music needs to convey.[4]

[1] Boris, and Iuliia Zaitceva.

[2] Singh, Harpreet, et al.

[3] â€œCollaborative Robots.

[4] This is where the model fails. The model fails to convey the target emotions.

2. Generating Emotionally Expressive Music: Creating music that effectively embodies and communicates these emotions.
3. Evaluating Emotional Expressiveness: Assessing how well the produced music captures and conveys the intended emotional states.

These AI systems handle a variety of inputs, converting emotional cues from different media—such as text, images, and sensor data—into music that can evoke specific emotional responses in listeners. The methods employed range from rule-based techniques to data-driven approaches, including neural networks, and often involve hybrid strategies that blend multiple methodologies.

Despite technological advances, AI still encounters significant challenges in this domain. One primary difficulty lies in creating music that genuinely corresponds with the complex nature of human emotions. The music must not only be technically proficient but also deeply resonant on an emotional level. "Designing rules to tailor the musical feature for expressing different levels of valence has …been challenging (Miyamoto et al., 2020) and needs further exploration”[5].

1.3 Scope and Objective

This paper introduces a fuzzy neural network for piano music generation tailored to specific emotional states. While time constraints hinder the pursuit of purely aesthetic music, our focus lies in showcasing how a simple model, employing fuzzy inputs, can to some degree grasp emotions. Despite challenges such as note prediction complexity, computational resource constraints, and dataset limitations, our aim isn't solely melodic delight. Instead, the aim is to illuminate the potential of fuzzy logic, illustrating its capacity to interpret the composer's mood through generated music.

[5] Agres, Kat R., and Adyasha Dash.

3. METHODOLOGY

3.1 System Design - See Figure 1

Input Layer

The model processes 8 features with the input dimensionality specified as (100, 262). The '100' in this context signifies the model's consideration of data pertaining to the previous 100 notes or velocities, represented as a time series to provide a rich historical context for enhanced predictive accuracy.

The '262' component is itemised as follows: Four of the inputs represent different emotional states, categorised as Q1, Q2, Q3, and Q4 (see Figure 2 for depiction), encoded in a manner that captures the fuzzy characteristics of emotions. Inputs for 'step' (the interval since the last note) and 'duration' (how long the note is held) are included, aggregating to six foundational inputs.

The remaining 256 dimensions are attributed to the utilisation of the 'pretty_midi' library. This library, designed for interpreting MIDI files, offers a tonal range of 128 distinct pitches (notes) and an equivalent range for dynamics (velocities). Both notes and velocities are subjected to one-hot encoding to permit the model to discern intricate musical expressions. Thus, the complete input size is established as 128 (notes) + 128 (velocities) + 6 (emotional states and timing features), arrayed over 100 sequential instances.

LSTM and Attention Layers

An LSTM layer, delineated by a shape of (None, 100, 128), is directly succeeded by an Attention layer with an identical output shape, indicating a reflective mirroring of size and capacity. This LSTM-Attention coupling is sequentially replicated.

Following this sequence is a Bidirectional LSTM layer, denoted with an output shape of (None, 128), indicative of its bidirectional processing capability, enhancing the model's temporal understanding.

Branching

The model bifurcates into two distinct paths: one for categorical outputs and another for continuous outcomes. The categorical path is responsible for predicting notes and velocities, both of which undergo one-hot encoding. The continuous path manages the prediction of 'step' and 'duration', formatted as continuous variables.

Categorical Branch

In the categorical arm, a Dense layer equipped with 512 units feeds into a Dropout layer, a strategic inclusion to mitigate overfitting risks. This pathway eventually diverges into two distinct output features: pitch and velocity, each characterised by a Dense layer with 128 units. These units apply a softmax activation function, catering to the multi-class nature of the one-hot encoded targets.

Regression Branch

The regression branch parallels the categorical with an analogous initial structure—a Dense layer followed by a Dropout layer. It culminates in two separate outputs: one for 'step' and another for 'duration'. Each is connected to a Dense layer with a single unit that employs a ReLU activation function, framing them within regression paradigms.

3.2 Iterative Design

The exploration of neural network architectures for the project was extensive, but remained linear, focusing primarily on the inclusion of Long Short-Term Memory (LSTM) layers and attention mechanisms. These components were omnipresent across the various iterations due to their proven efficacy in handling sequential data. LSTM layers are particularly adept at remembering information over extended sequences, a crucial feature for processing time-series data or tasks like NLP, where context spans many inputs. The attention mechanism further refines this by allowing the model to focus selectively on parts of the input sequence that are most relevant to the current output, thus enhancing the model's ability to make connections over long distances within the input data.

The decision to incorporate these elements consistently across models was informed by their compatibility with the nature of our dataset and the tasks at hand. However, while crafting the architecture, it was observed that scaling up the complexity – by increasing the number of neurons and layers – did not correspond to a proportional increase in performance. This was particularly evident when considering the size of our dataset, which was relatively small at 2.1 MB.

Models with gargantuan architectures, akin to the originally designed 1GB model (same as Figure 1, but more neurons per layer), often necessitate massive datasets to learn effectively without overfitting. With a limited dataset, such a colossal network runs the risk of memorising the training data rather than generalising to new, unseen data. This phenomenon is exacerbated by the diminishing returns in model performance past a certain complexity threshold, especially when computational resources are a bottleneck.

The architecture presented, while ambitious, faced practical challenges when it came to training. Empirical recommendations from literature often advocate for extensive training, spanning hundreds[6] to thousands[7] of epochs, to fully exploit the potential of such complex networks. However, given the computational constraints and the dataset's size, pursuing this approach was infeasible. It not only demanded an exorbitant amount of computational time but also risked overfitting due to the disproportionate size of the model relative to the dataset.

After training the gargantuan 1GB model for 48h, and making close to zero progress with the generated notes sounding out of tune and dissonant reflecting my own feelings at the time (since the due date was approaching) I pivoted fast. A new model was chosen about 100MB, but this again proved too ambitious. This is when a tiny model was created, as modelled in Figure 1, and described in 3.1. This new model only took about 2 minutes per epoch, given we train on 250 out of the available MIDI files, making the dataset even smaller.

In retrospect, the pivot to a more simplified model was a strategic response to these limitations. By tailoring the model's complexity to align with the available data and computing resources, the revised goal of showing how a fuzzy model can ‘understand’ emotion, instead of generating pleasant music aimed to demonstrate the applicability of fuzzy logic in discerning emotional states from data. This shows the importance of model optimisation, not just in the dimension of performance but also in resource efficiency and suitability to the task's scale.

[6] Rey, Arturo.

[7] Abbou, Raphael.

3.3 Fuzzy Logic Input

When users input their emotional state, they provide a score from 0 to 1 for each quadrant. These quadrant scores correspond to emotional states and are placed on the Russell Circumplex Model[8], which is visualised in the Figure 3. The model is divided in four, each quadrant reflecting a range of emotions: Q1 for positive and high arousal emotions (top right), Q2 for negative and high arousal emotions (top left), Q3 for negative and low arousal emotions (bottom left), and Q4 for positive and low arousal emotions (bottom right).

The "membership functions," which are part of the fuzzy logic used by the ML model, are shaped by two dimensions: Arousal and Valence, as labeled on the axes of Figure 3. A high Arousal score may intersect with emotions like "Excited" or "Angry," which are high on the vertical axis. The horizontal axis of Valence differentiates between positive emotions (towards the right) and negative emotions (towards the left). This system allows the model to interpret nuanced emotional inputs for processing.

Flexible Fuzzy Inputs

This model excels with its flexible data preprocessing, which enables it to easily integrate varied inputs such as video feeds capturing facial expressions.

The Fuzzy Logic could be further developed to include:

1. **Emotional Tone Analysis**: Leveraging vocal nuances to gauge emotional contexts.
2. **Gesture Recognition**: Using physical gestures to influence the music's dynamics and tempo.
3. **Environmental Sensors**: Modifying music based on the ambiance.

Integrating robotics, the model could automate musical performances on physical instruments, adapting in real-time to various inputs. In settings such as restaurants or bars, it can assess the room's mood through video, or sound and adjust the music accordingly. Using simple membership functions, the model can processes fuzzy input data to produce music that reflects the atmosphere of a room, thereby improving the ambiance and enhancing the overall experience for patrons.

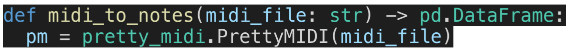
3.4 Data Processing

Preprocessing:

The dataset utilised in this research is sourced from EMOPIA[9] and primarily comprises midi files. MIDI (Musical Instrument Digital Interface) files are a standard file format used to encode the information about a sequence of musical notes and control signals for musical instruments. These files don't contain actual audio data but rather include instructions that tell music hardware or software how to reproduce a composition.

In addition to musical sequences, the dataset features emotional labels aligned with four quadrants—Q1, Q2, Q3, and Q4—which are integral for fuzzy logic applications within our model.

The MIDI files were processed like in code excerpt 1[10].



```
def midi_to_notes(midi_file: str) -> pd.DataFrame:
  pm = pretty_midi.PrettyMIDI(midi_file)
```

[8] Russell, J. A.

[9] Hung, Hsiao-Tzu, et al.

[10] â€œGenerate Music with an RNN

```
  instrument = pm.instruments[0]
  notes = collections.defaultdict(list)

  # Sort the notes by start time
  sorted_notes = sorted(instrument.notes, key=lambda note: note.start)
  prev_start = sorted_notes[0].start

  for note in sorted_notes:
    start = note.start
    end = note.end
    notes['pitch'].append(note.pitch)
    notes['velocity'].append(note.velocity)
    notes['start'].append(start)
    notes['end'].append(end)
    notes['step'].append(start - prev_start)
    notes['duration'].append(end - start)
    prev_start = start
  return pd.DataFrame({name: np.array(value) for name, value in notes.items()})
```

(GitHub in Appendix)

This results in a table like so:

pitch	velocity	start	end	step	duration
43	48	0.148438	0.729167	0.000000	0.58072
55	39	0.164062	0.729167	0.015625	0.565104

The normalised quartiles corresponding to the song are then appended into the table:

pitch	velocity	start	end	step	duration	Q1	Q2	Q3	Q4
43	48	0.148438	0.729167	0.000000	0.58072	-0.469478	-0.519611	0.964776	0.161622
55	39	0.164062	0.729167	0.015625	0.565104	-0.469478	-0.519611	0.964776	0.161622

Normalisation of the Fuzzy Inputs may not be necessary but was done out of habit, and to be truly fuzzy should not be normalised.

A time series is introduced for the LSTM, the pitch and velocity are one hot encoded, and labels are introduced. The final table (shortened, and excluding time series) looks like this:

pitch	velocity	start	end	step	duration	Q1	Q2	Q3	Q4	pitch label	velocity label	step label	duration label
0,0...1,0	0,0...1,0	0.148438	0.729167	0.000000	0.58072	-0.469478	-0.519611	0.964776	0.161622	0,0...1,0	0,0...1,0	0.015625	0.565104

The model is then ready to fit to the data. Input shape takes the form of (100, 262), as discussed in 3.1, with four outputs, pitch, velocity, step, and duration.

Post Processing:

The output has to be prepared to be played as a midi file, so the anti-function of code excerpt 1 is taken as seen in code excerpt 2:

```
def notes_to_midi(notes, output_file='predicted_music.mid'):
    pm = pretty_midi.PrettyMIDI()
    instrument = pretty_midi.Instrument(program=pretty_midi.instrument_name_to_program("Acoustic Grand Piano"))

    for index, note_info in notes.iterrows():
        note = pretty_midi.Note(
            velocity=int(note_info['velocity']),
            pitch=int(note_info['pitch']),
            start=note_info['start'],
            end=note_info['end']
        )
        instrument.notes.append(note)
    pm.instruments.append(instrument)
    pm.write(output_file)
    print(f'MIDI file saved as {output_file}')
    return pm
```

3.5 Training

The training data comprised sequences extracted from a curated subset of 250 MIDI files from the EMOPIA dataset[11]. This subset was chosen to balance the computational load and the diversity of data needed for effective model training.

A pivotal aspect of the model's training involved adjusting the loss functions for each output variable—pitch, velocity, step, and duration. To rectify initial underperformance in pitch prediction, the loss weights were calibrated, placing a heavier emphasis on the pitch component. The specific weight settings—1.0 for pitch, 0.005 for velocity, 0.0001 for step, and 0.001 for duration—were determined through iterative testing:

```
loss_weights={
          'pitch': 1.0,
          'velocity': 0.005,
          'step': 0.0001,
          'duration':0.001,
      }
```

These values may seem extreme, but turns out predicting pitch is actually 10_000x harder than predicting step (given this data, and the architecture used (Figure 1)). This might be because velocity, step, and duration are more uniform than pitch. This approach prioritised pitch accuracy while still allowing the model to learn other attributes like note velocity and duration.

During training, the model used a combination of custom and standard loss functions. For the step and duration predictions, a custom loss function (mse_with_positive_pressure) was utilized to introduce a penalty for negative predictions, thus improving the model's forecasting accuracy for these continuous outputs[12].

3.6 Evaluation of Final Model

The evaluation process we employ is both direct and unconventional. To conduct this assessment, we instruct the model to generate a large dataset consisting of 2_500 notes, specifically focusing on each dominant quadrant identified in our study. Once this synthetic data is generated, we undertake a detailed comparison against the actual notes from the corresponding quadrant in the training dataset. This method helps in assessing how well the model can replicate the typical characteristics of notes in each quadrant.

To enhance transparency and provide a clearer visual understanding of the results, we utilise multiple graphical representations. We generate a series of plots and charts using both Matplotlib and Seaborn libraries.

Some of the plots may not be easy to interpret at first glance, such as Figure 6. To clarify, Figure 6 does not depict stacked bars. Instead, it places the shorter bar in front of the taller one, ensuring that both can be seen. This approach is chosen over reducing the opacity, which can make the chart hard to read, especially when it contains many bars as in Figure 4.

[11] Hung, Hsiao-Tzu, et al.

[12] â€œGenerate Music with an RNN

4. RESULTS

4.1 Training Data

In section 3.6, we delve into the process of crafting specific figures, which relate to our training data, and these are visually represented in Figure 4. Upon analysis, distinct differences emerge among the quadrants. For instance, Q1 and Q2 consistently exhibit higher volume levels compared to Quadrants Q3 and 4 Q4. This divergence could stem from the tendency for songs inducing heightened arousal to feature louder dynamics, while those associated with lower arousal tend to be quieter.

Moreover, there's a noticeable trend regarding the duration of songs in these quadrants. Q1 and Q2 generally feature shorter durations, relatively speaking, in contrast to the longer durations observed in Q3 and Q4. This discrepancy may suggest that songs inducing higher arousal tend to be faster-paced, thus resulting in shorter durations on average.

4.2 Generated Data

In examining the generated music from the model, a critical observation emerges: previous notes exert a more significant influence on the generated output than the current emotional input (the variable we want the composer to change, and which should change the feel of the outputted music). This phenomenon can be elucidated by dissecting the network architecture provided and considering the inherent characteristics of the LSTM (Long Short-Term Memory) layers.

The architecture, as discussed in 3.1 and shown in Figure 1, illustrates the utilisation of bidirectional LSTM layers—specifically, bidirectional_63 and bidirectional_64. LSTM layers are adept at capturing temporal dependencies within sequences, which means they give substantial weight to the context provided by preceding notes. While this is beneficial for maintaining musical coherence over time, it can inadvertently cause the LSTM to favour historical data over the current emotional cues, potentially diminishing the impact of new emotional inputs on the subsequent generation.

Furthermore, the attention mechanism, represented by Attention_Base, enhances the LSTM's focus on specific parts of the input sequence. While this ideally aids in emphasising critical emotional nuances, the attention model may still be biased towards the patterns established by previously encountered notes. This bias towards historical notes, and emotional states, can overshadow the current emotional input's immediate influence on the generation process, as the network learns to predict the next note based on the intricate interplay of past sequences rather than the current emotional state.

Temperature in Music Generation

Temperature is a hyper parameter used in controlling the randomness of predictions in models that generate sequences. In the context of music generation, it affects how surprising or predictable the music will sound. A low temperature makes the model more conservative; it's more likely to repeat the same note with the same velocity, duration, and step, resulting in a less diverse and potentially monotonous piece. Conversely, a high temperature increases randomness, which might lead to a composition that sounds erratic or lacks coherence.

In the prediction code (see appendix), temperature scales the logits before applying the softmax function to obtain a probability distribution. The softmax_with_temperature function modifies the concentration of the probability distribution—lower temperatures sharpen the distribution (more peaky), leading to less variability in output, and higher temperatures flatten it (less peaky), leading to more variability.

Temperature's Role in Emotional Expression

The temperature parameter was introduced to strike a balance between variability and predictability in generated music. By carefully tuning this parameter, we can ensure that the music has enough novelty without deviating too much from a coherent musical line. However, this balance is delicate; as indicated, the low temperature setting, typically around 0.025 ± 0.015, constrains the model to repeat similar patterns, which might be contributing to the overshadowing of the emotional inputs.

Potential Impact on Emotional Distribution

If the temperature is too low, the model's predictions become deterministic and less influenced by the current emotional inputs. It tends to repeat patterns that were successful in the past, in this case, the previous notes. This conservative approach could hinder the model's responsiveness to changes in the desired emotional state, leading to a generation that fails to capture the dynamic emotional shifts intended by the user.

4.3 Direct Comparison

Comparing Figure 4 and Figure 5, it is evident that the generated notes predominantly fall within the actual note range across all emotional quadrants (Q1, Q2, Q3, Q4). Actual notes are the 100 notes the dataset makes predictions from, and the generated is the predicted results.

Pitch and Velocity Distributions

From the histograms (Figures 6-9), we observe a multimodal distribution in both pitch and velocity, which suggests that the model generates a variety of note pitches and velocities rather than favouring a narrow range. The distribution of pitches appears to peak around certain values, indicating preferred tonalities or octaves that the model generates more frequently. Similarly, the velocity histograms show the dynamics with which notes are played, peaking at certain values which might correspond to musical articulations or expressiveness dictated by the model.

However, there are notable differences in the generated and actual (reference) distributions. For example, certain quartiles show a heavier tail in the generated pitch distribution compared to the actual, which implies that the model occasionally favours pitches outside of the commonly used range. The velocity histograms reflect a similar trend, with the generated distributions displaying slightly different skews and spreads, suggesting variations in the dynamic range that the model learns to express.

Step and Duration Distributions

Step and duration histograms illustrate the temporal aspects of note sequences. The step parameter, which determines the time interval between consecutive notes, shows a concentration at lower values in the generated data, indicating a tendency for notes to occur close together. The duration parameter, indicative of how long a note is held, also shows a concentration of values, with the generated histograms often peaking at different values compared to the actual data. This suggests that the model may have a bias towards generating notes of certain lengths, which could impact the rhythmic feel and the overall flow of the generated music.

In this section each dataset will be compared to 2_500 generated notes:

Q1 - High Valance, High Arousal

Q1 corresponds to positive, high-energy emotions like excitement and delight. This quadrant would typically feature music with higher pitches, signalling happiness or brightness, and faster tempos indicating energy. Faster steps and shorter durations might contribute to an energetic rhythm, although excessively short notes could detract from the sense of melody and harmony that characterises joyful music. The plot suggests that the model's generated pitches for Q1 are well-distributed but with an overemphasis on mid-range pitches, which might dilute the expression of delight or happiness. The velocity's close match suggests good dynamic variation, which is important for conveying enthusiasm. However, the lack of longer durations may mean the model is less capable of producing sustained, harmonically rich passages that are often associated with joyful music.

Q2 - Low Valance, High Arousal

Q2 is associated with negative yet high-energy emotions, such as tension or distress. Music in this quadrant would likely have a broad range of pitches, including lower ones to convey the negative valence and varying velocities to express the high arousal. The duration and step features would be mixed, reflecting both the agitation of high arousal and the depth of negative emotions. The plot for Q2 shows a close pitch alignment, suggesting the model captures the emotional context well, while the velocity indicates an effective range for conveying dynamic shifts. However, the preference for shorter steps might oversimplify the complex rhythms usually present in tense or angry music. Similarly, the absence of longer note durations may not fully express the lingering tension characteristic of such emotions.

Q3 - Low Valance, Low Arousal

Q3's quadrant represents emotions that are both negative and low-energy, like sadness or depression. Such music typically features slower tempos, longer note durations for a more legato feel, lower pitches, and softer velocities. The generated pitch distribution for Q3 is narrower than actual, possibly missing the mark in expressing the full emotional weight of sadness. The overextended velocity range may create dynamics that are too varied for the typically subdued expression of low-energy, negative emotions. The generated music also tends towards shorter steps, which could inadvertently add a sense of restlessness that contradicts the expected languid pace of such music.

Q4 - High Valance, High Arousal

Q4 encompasses positive, low-energy emotions like contentment and relaxation. This quadrant would likely feature music with lower velocities reflecting the calmer dynamics, higher pitches indicative of positive emotions, and longer note durations to give a sense of ease and flow. In the histograms for Q4, the generated pitches are somewhat aligned with the actual data but with a significant deviation in the 50-60 range, showing a distribution that can convey a cheerful or serene atmosphere to some extent. However, the generated velocity histogram indicates a discrepancy, particularly in the mid-range velocities, which suggests that the model might not capture the nuances of dynamic variation as effectively as needed for relaxed positivity. Furthermore, the generated step histogram shows a higher proportion of very short steps, which might lead to a choppier rhythm than is ideal for conveying low arousal. Finally, the duration histogram displays a good approximation in the range of shorter durations, but it significantly lacks the longer durations that would typically be used to create the languid, flowing qualities associated with Q4 emotions. This could result in a less cohesive and smooth musical texture, potentially reducing the sense of calmness and relaxation.

What does it mean?

Model Complexity:

The model is generating features that do not accurately reflect the subtleties of the expected emotional states, it could imply that the model's architecture is too simplistic to capture the complexity of musical emotions. Music is a richly nuanced medium where emotions are conveyed through a delicate balance of harmony, rhythm, dynamics, and tempo. A model with insufficient complexity may not be able to model these intricate relationships effectively.

Training Epochs and Overfitting:

The fact that the model was only trained for a couple of hundred epochs, whereas 1000+ if recommended[13], suggests that the model may not have fully converged to an optimal state. Training for more epochs could allow the model to learn more nuanced patterns in the data and better generalise to produce music that closely aligns with the desired emotional states.

However, more epochs do not always equate to better performance. There's a risk of overfitting, where the model becomes too tailored to the training data and loses its ability to generalise to new inputs.

[13] Abbou, Raphael.

5. ADVANTAGES AND DISADVANTAGES

5.1 Advantages of the Proposed System

Highlight unique benefits and innovative aspects.

The primary advantage of our system lies in its handling of emotional inputs through fuzzy logic. Operators can select from a spectrum of emotions, which the system processes using a data pipeline to predict and generate music that reflects these emotions. This approach leverages the Russell Circumplex Model to categorise emotions into coherent sets that are intuitively understandable by humans. By accepting vague and complex emotional inputs and converting them into distinct musical outputs, the system showcases the strength of fuzzy logic in managing and interpreting nuanced human emotions. The adaptability of fuzzy logic allows the model to deal effectively with the inherent ambiguity of emotional states, ensuring that the outputs are both meaningful and contextually appropriate. Examples of somewhat diverse 'musical' outputs generated by the model can be found in the appendix GitHub link.

5.2 Limitations and Challenges

On EMOPIA:

The dataset employed in this investigation is considerably undersized, occupying just slightly more than 2.1 megabytes of digital storage. This volume is distinctly inadequate when measured against the vast data requirements typical in contemporary data-driven research environments. The constrained size of the dataset severely limits the model's capacity to assimilate and learn from a broad array of patterns, which is essential for robust machine learning applications. Such limitations in data diversity are a significant detriment to the model's ability to generalise beyond its training inputs, thereby restricting its effectiveness in varied real-world scenarios.

Additionally, the dataset demonstrates a noticeable cultural homogeneity, as it has been exclusively annotated by four individuals residing in Taiwan, all of whom share a similar cultural background. This uniformity in the data annotation process introduces a monocultural bias, skewing the dataset's representation and potentially influencing the model's output. The lack of varied cultural perspectives in the dataset can severely impact the model's utility across different cultural contexts, reducing its effectiveness and applicability in global or culturally diverse environments. This cultural bias not only limits the scope of the model's applicability but also raises concerns about its fairness and accuracy when deployed in settings that differ from the cultural context of the dataset's annotators.

On computations:

From a computational standpoint, the model encountered notable constraints that significantly impacted its performance and efficacy. Initially conceived to operate with a computational budget encompassing several gigabytes, stringent resource limitations necessitated a considerable reduction in the model's size to mere megabytes.

This downscaling of the model's operational scale may have led to a reduction in its ability to process and learn from complex data patterns effectively. A more expansive architecture, capable of leveraging the full spectrum of the initially intended computational resources, might have captured a wider array of nuances in the data, thereby enhancing the model's learning accuracy and output quality. The reduction in model complexity due to the constrained computational capacity likely resulted in an underperformance that a more robust system could potentially overcome.

Moreover, it is plausible that the suboptimal training outcomes could partly be attributed to limitations in the expertise available for managing large-scale model training. Effective training of high-capacity models not only requires sufficient hardware but also a sophisticated understanding of model architecture optimisation, data handling, and algorithm efficiency. The potential lack of advanced skills in these areas might have contributed to the inability to fully exploit the computational resources, even when such resources were adequate. This scenario underscores the importance of proficiency in high-performance computing techniques and model management to fully harness the capabilities of large neural networks, thereby mitigating performance issues even within significant computational constraints.

On transfer learning:

The primary and most critical limitation in the development of the model lies in the exclusion of transfer learning methodologies from its architectural framework. Transfer learning, a pivotal technique in machine learning, involves the adaptation of models that have been pre-trained on a substantial amount of data to new, often related tasks. This approach could have harnessed models that are already proficient in music generation, thus providing a robust baseline from which to undertake further model refinement. This refinement would specifically focus on the nuanced representation of emotional states in musical compositions, allowing for a more targeted and effective tuning process.

Initiating the model with a pre-trained base would have conferred several advantages, notably in accelerating the training phase and enhancing the model's overall performance. Pre-trained models come equipped with learned features that can drastically reduce the learning curve for new tasks, which in this case involves the generation of emotionally nuanced music. The adoption of such models would likely result in a more efficient training process and potentially superior final outputs.

Moreover, the integration of transfer learning principles would have equipped the model with advanced capabilities for handling complex and abstract inputs, such as those associated with fuzzy emotional concepts. This capability would greatly enhance the model's proficiency in producing music that not only aligns with but also evokes specific emotional states, thereby enriching the emotional depth and resonance of the musical outputs.

On Architecture:

Forecasting the subsequent note in a dataset poses significant challenges. While a sophisticated neural network, incorporating LSTM and Attention layers, might suffice for this prediction task, introducing emotional elements into the output complicates matters. When attempting to achieve the objective of creating a flexible input that can substantially alter the network's output, integrating time series emotions proves challenging. Despite efforts, this goal wasn't achieved, yet valuable insights into architectural considerations were gained. This will be fully discussed in 6.1

6. CONCLUSION

6.1 Future System Design

Upon reviewing the model's architecture and its performance, it's evident that the current system captures the nuances of emotional content in music, though not as robustly as anticipated. The original vision entailed a dynamic generation of music that allowed a composer to tweak a four input parameters and significantly alter the piece's emotional tone. To an extent, this capability exists within the model; however, as evidenced by the comparison graphs (Figures 6-9), it is apparent that preceding musical notes have a disproportionate influence over the generated music compared to the intended emotional input, as discussed in section 4.3.

This disproportionate influence suggests a fundamental flaw in the system's design, indicating that the model should have been not only substantially larger in scale but should also undergo specific architectural modifications for improved performance.

One of the primary issues is the model's tendency to prioritise past emotions over the current emotional state desired by the user. This behaviour likely stems from the training methodology, which focuses on one emotion at a time without introducing variability during the training phase, leading to a model that struggles with transitioning between emotions, but continuously generates okay results for one dominating emotion.

A potential solution involves introducing a novel input stream that disregards previous emotional states, considering solely the current emotion specified by the user. This input, labeled 'Emotional_Input' in Figure 10, would be introduced to the model downstream of the LSTM and attention layers. These layers are designed to handle time-series data, and while previous emotions should influence the prediction, they should not overshadow the current emotional input, hence this downstream suggestion.

Moreover, to enhance the model's focus on emotional elements in music, the initial input layer could be bifurcated into multiple branches. One proposed bifurcation would separate velocity, pitch, step, and duration from emotional content, thereby allowing the model to dedicate more resources to understanding and generating the emotional aspects of music.

Alternatively, for the generation of entirely new music pieces with distinct moods, the model could be designed to completely disregard past emotional states, ensuring that new compositions are not influenced by preceding moods but maintain coherence based on the pitch and velocity of previous notes.

This idea of branching is visualised in Figure 11, showcasing a more intricate architecture. In such a model, certain aspects like pitch and emotional content might necessitate more complex configurations due to their inherent complexity, while other elements like duration and step could be modelled more straightforwardly (as attempted by the model_weights- a change in architecture would be significantly more efficient).

The most promising alteration, however, could be the strategic placement of the desired emotional output as an input later in the model's layers downstream of the LSTM and Attention layers as discussed in an earlier paragraph. This change would potentially enable the model to give precedence to the emotional tone intended by the user, thereby aligning the generated music more closely with the user's creative vision.

These proposed changes reflect an evolutionary step in the model's development, expanding its capabilities to capture and generate music that resonates with the intended emotional nuances, creating a tool that is both powerful and sensitive to the artistic direction of the composer.

6.2 Reflective Analysis and Lessons Learned

Reflecting on the journey of this project, several key lessons emerge, each shedding light on the intricate balance between theoretical aspirations and the pragmatic realities of developing intelligent systems.

Complexity and Simplicity in Model Design

One of the most profound insights gained from this project centers on the complexity of the models employed. Initially, the ambition was to leverage advanced neural network architectures, including LSTM and attention mechanisms, to capture the nuanced interplay of emotions in music generation. However, the reality of limited dataset size and computational resources necessitated a pivot towards simpler, more manageable models. This experience underscores the critical importance of aligning model complexity with available data and computing power. Too often, there's a temptation to employ state-of-the-art models without adequate consideration for their practicality in specific use cases. This state-of-the-art think plagued this paper , and if realised earlier the final product could have been far improved.

The Challenge of Training Time

Training even a relatively simple model proved to be a daunting task, primarily due to the computational demands and the time required for meaningful results. The initial attempts with larger models were significantly hampered by time constraints, resulting in just a few epochs before a new model was employed. It took many iterations of this process to eventually adopt a smaller, more focused model. This shift highlighted the a balance between training duration and model efficacy.

Selecting the Right Architecture

Selecting the right architecture for a specific task is crucial, a lesson that became evident throughout the duration of this project. Initially, a complex model intended to capture the subtle nuances of emotional expression in music. The architecture, while linear with some branching, predominantly managed inputs as a time series of notes combined with emotional data. However, this linearesque model encountered significant challenges.

The primary issue stemmed from the handling of time series data for emotions, which proved to be misaligned with the project's goals. Our objective was to enable a composer to modify the musical output dramatically through straightforward inputs, essentially allowing them to alter the emotional tone of the music with minimal effort. Unfortunately, the chosen architecture was ill-suited for this level of dynamic input manipulation. It was too rigid and linear to accommodate the fluid, often instantaneous shifts in emotional input that was aimed for.

A more flexible and responsive architecture might have allowed for the direct and impactful translation of simple inputs into varied emotional outputs, aligning better with our goals. Such an architecture would ideally handle inputs in a way that directly influences the generation process, allowing for immediate and significant changes to the music’s emotional character based on the composer's input. This capability was not realised in the current project but remains a potential avenue for future exploration with the right architectural adjustments.

Hardships and Adaptability

Throughout the project, several hardships were encountered, from technical limitations to unexpected setbacks in model performance. Each challenge required adaptability and resilience. For instance, when faced with the looming deadline and a model that wasn't performing as expected, the decision to switch to a simpler model was crucial. This adaptability not only salvaged the project but also provided a clear path forward under tight constraints.

Importance of Data and Preprocessing

A significant takeaway from this endeavour is the crucial role of data quality and preprocessing. The limitations of the EMOPIA dataset, particularly its size and cultural homogeneity, posed substantial challenges. This experience highlights the need for a robust preprocessing pipeline and the acquisition of diverse and extensive datasets to train more generalised and effective models.

Reflecting on Emotional Depth and AI

This project has significantly deepened my understanding of the complexities involved in replicating the depth of human emotions through artificial intelligence, particularly within the context of music generation. Emotional expression in music embodies nuanced transitions and overlays—such as the interplay between joy and melancholy (like experienced when moving abroad, you're excited to go somewhere new but sad to leave what you have built) or the concurrent feelings of excitement and anxiety (like I will probable experience when I am close to graduation (2nd year now), due to the difficulties of the job market but eagerness to get out there) —which present a substantial challenge to capture accurately using AI techniques, including advanced implementations of fuzzy logic.

Fuzzy logic, inherently adept at managing ambiguous and imprecise data by assigning degrees of truth rather than binary conditions, was anticipated to offer considerable advantages in this project. It supports environments where decisions are not clear-cut, which is emblematic of the process of interpreting human emotions. In this endeavour, the Russell 1980 Circumplex Model of emotions was employed, which categorises emotions along two axes: arousal and valence. This model facilitated the structuring of emotional inputs into our system, allowing fuzzy logic to process complex emotional states in a more organised and interpretable manner.

However, the practical outcomes highlighted some critical limitations of our approach. While fuzzy logic theoretically could handle the complexities of emotional nuances effectively, the specific architecture used in this project did not do justice to the capabilities of fuzzy logic. The architecture, primarily designed to accommodate straightforward predictive modelling with clear input-output relationships, struggled with the ambiguity and subtlety required for processing the multi-dimensional emotional data as per the Russell model. This misalignment resulted in poorer than expected results, where the generated music often failed to reflect the intended emotional depth or complexity.

The implementation revealed that even a robust framework like fuzzy logic requires a compatible architectural approach to fully harness its potential. The need for an architecture that can more flexibly accommodate the fuzzy parameters and effectively translate them into the musical output became evident. This mismatch between the theoretical advantages of fuzzy logic and the practical limitations of the chosen architecture reveals a critical aspect of AI research: the importance of aligning the model architecture with the specific nuances of the task at hand.

Reflecting on this experience, it is clear that while fuzzy logic presents a promising method for enhancing AI's ability to mimic human emotional depth, the selection of an appropriate system architecture is crucial. Future research could explore alternative architectures that might better leverage the strengths of fuzzy logic, potentially leading to more successful implementations in emotionally-aware AI systems. This project thus not only highlights the current limitations of AI in creative expressions like music but also sets the stage for future explorations and innovations that could eventually enable AI systems to truly resonate with and amplify the complex spectrum of human emotions.

REFERENCES

Abbou, Raphael. *DeepClassic: Music Generation with Neural Neural Networks*, Stanford, web.stanford.edu/class/archive/cs/cs224n/cs224n.1204/reports/custom/report11.pdf. Accessed 25 Apr. 2024.

Agres, Kat R., and Adyasha Dash. â€œAI-Based Affective Music Generation Systems: A Review of Methods, and Challenges.â€ *Ar5iv*, National University of Singapore, ar5iv.labs.arxiv.org/html/2301.06890. Accessed 22 Apr. 2022.

â€œCollaborative Robots.â€ *Robotics*, new.abb.com/products/robotics/robots/collaborative-robots. Accessed 22 Apr. 2024.

â€œGenerate Music with an RNN : Tensorflow Core.â€ *TensorFlow*, www.tensorflow.org/tutorials/audio/music_generation. Accessed 22 Apr. 2024.

Hung, Hsiao-Tzu, et al. â€œEMOPIA: A Multi-Modal Pop Piano Dataset For Emotion Recognition and Emotion-Based Music Generation.â€ *Emopia*, 2021, annahung31.github.io/EMOPIA/.

Boris, and Iuliia Zaitceva. Methods of Intelligent Control in Mechatronics and Robotic Engineering: A Survey, Boris, and Iuliia Zaitceva. â€œMethods of Intelligent Control in Mechatronics and Robotic Engineering: A Survey.â€ *MDPI*, 5 Aug. 2022, www.mdpi.com/2079-9292/11/15/2443.

(PDF) a Circumplex Model of Affect, Jan. 2020, www.researchgate.net/publication/235361517_A_Circumplex_Model_of_Affect.

Rey, Arturo. â€œHow to Generate Music Using Machine Learning.â€ *Medium*, Medium, 11 Dec. 2022, arturorey.medium.com/how-to-generate-music-using-machine-learning-72360ba4a085.

Russell, J. A. â€œA Circumplex Model of Affect.â€ *American Psychological Association*, Journal of Personality and Social Psychology, 1980, psycnet.apa.org/record/1981-25062-001.

Singh, Harpreet, et al. â€œReal-Life Applications of Fuzzy Logic.â€ *Advances in Fuzzy Systems*, Hindawi, 26 June 2013, www.hindawi.com/journals/afs/2013/581879/.

APPENDIX

Github Repo:

https://github.com/realfishsam/fuzzy_proj

Figures:

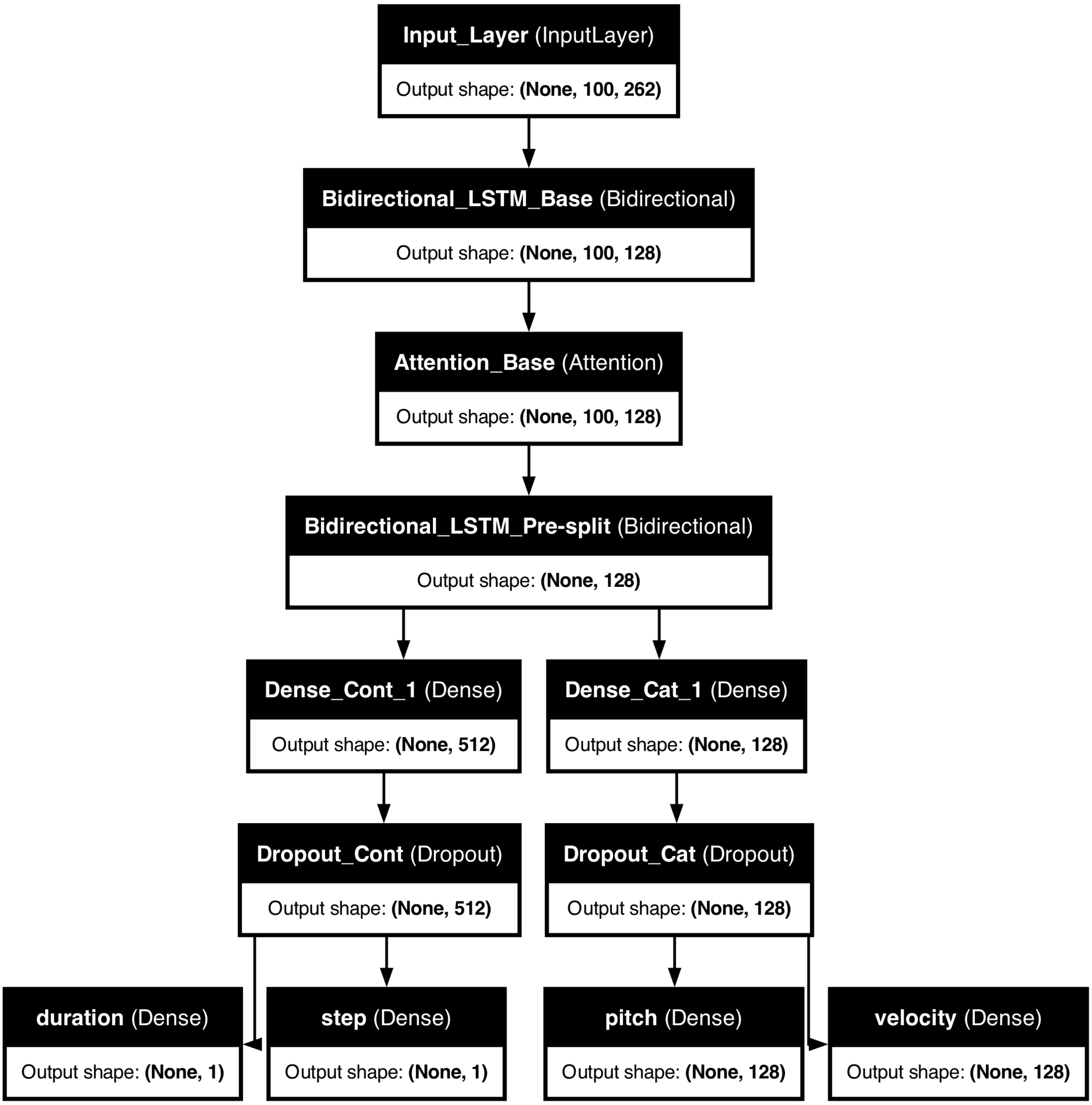

Input_Layer (InputLayer)
Output shape: (None, 100, 262)
Bidirectional_LSTM_Base (Bidirectional)
Output shape: (None, 100, 128)
Attention_Base (Attention)
Output shape: (None, 100, 128)
Bidirectional_LSTM_Pre-split (Bidirectional)
Output shape: (None, 128)
Dense_Cont_1 (Dense)
Output shape: (None, 512)
Dense_Cat_1 (Dense)
Output shape: (None, 128)
Dropout_Cont (Dropout)
Output shape: (None, 512)
Dropout_Cat (Dropout)
Output shape: (None, 128)
duration (Dense)
Output shape: (None, 1)
step (Dense)
Output shape: (None, 1)
pitch (Dense)
Output shape: (None, 128)
velocity (Dense)
Output shape: (None, 128)

Figure 1. Actual Model Architecture

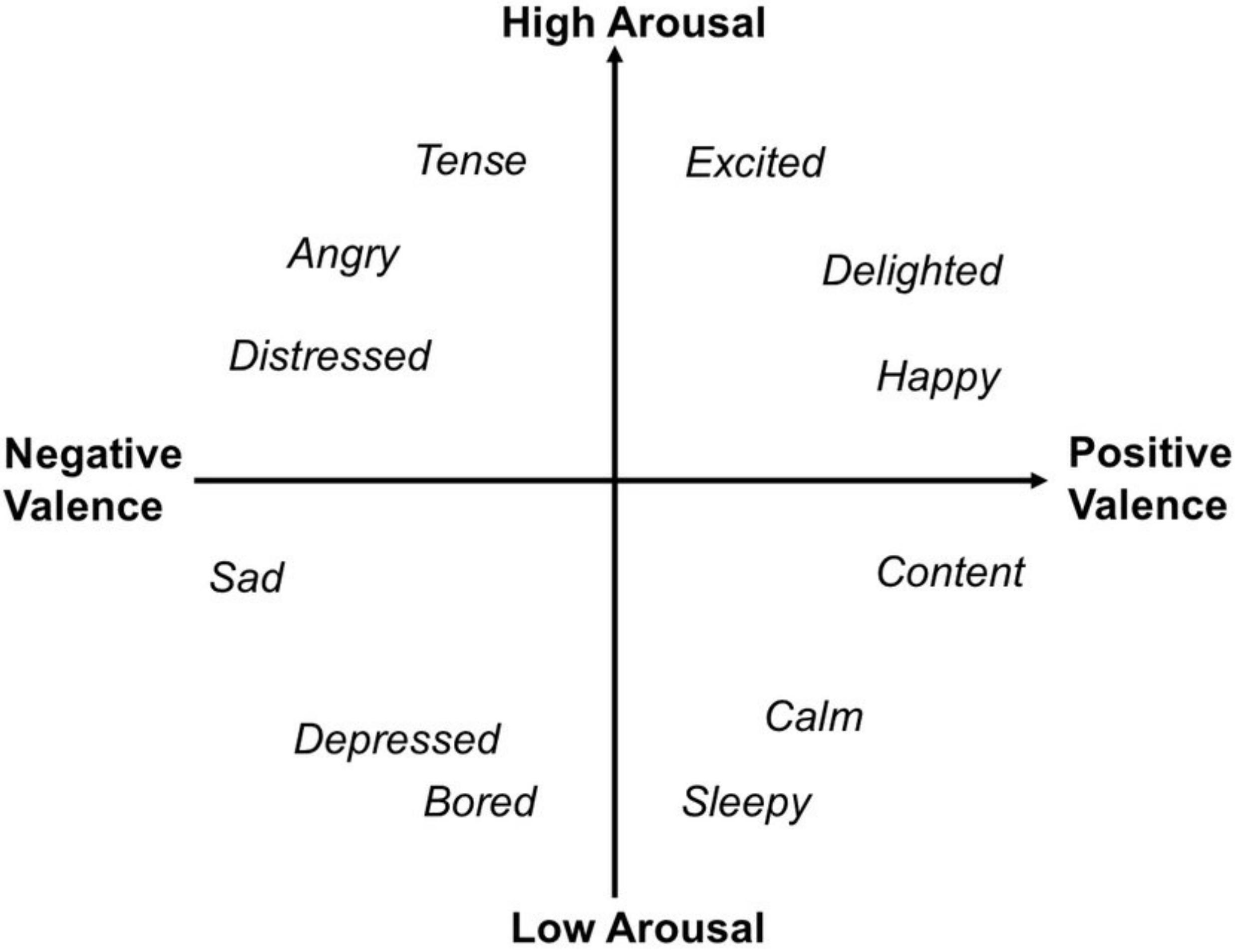


Figure 2. A circumplex model of affect. (Paper: Russell, J. A., img: *(PDF) a Circumplex Model of Affect*

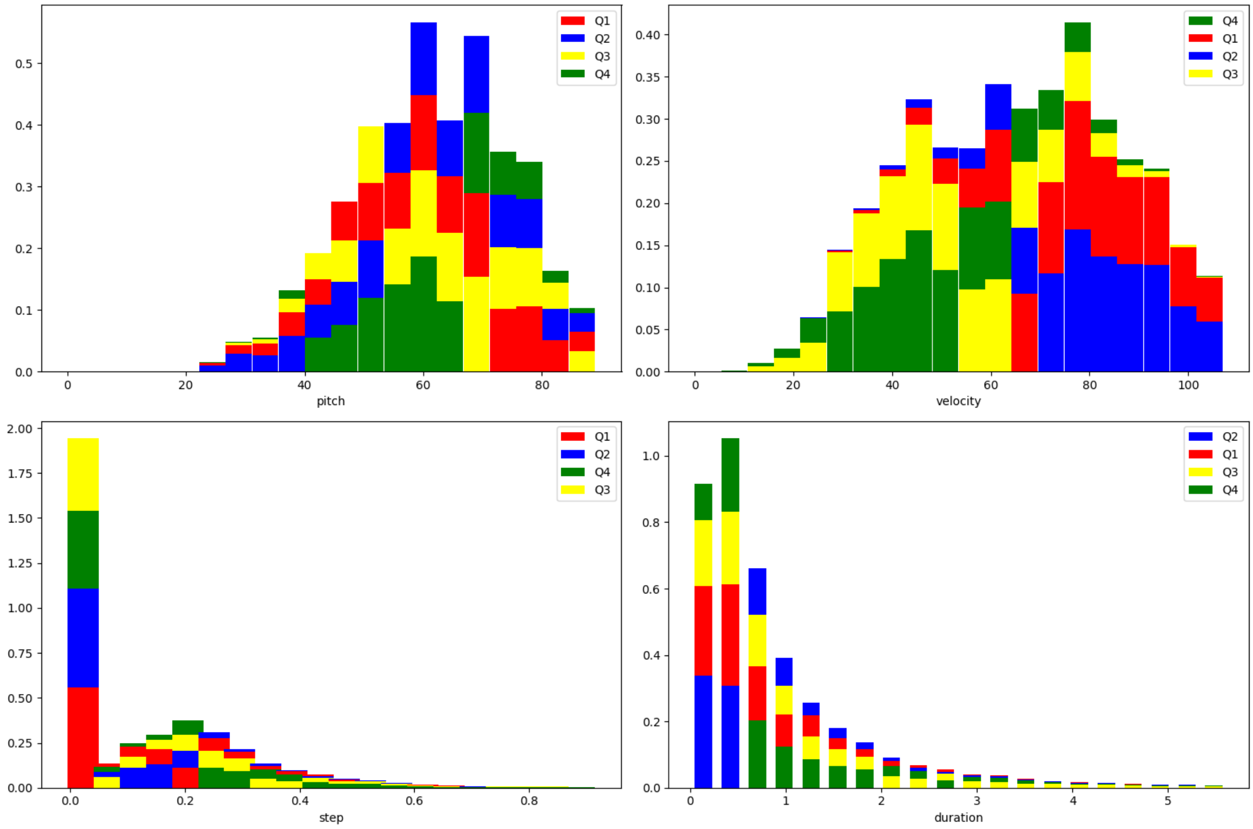


Figure 4. Actual Distributions

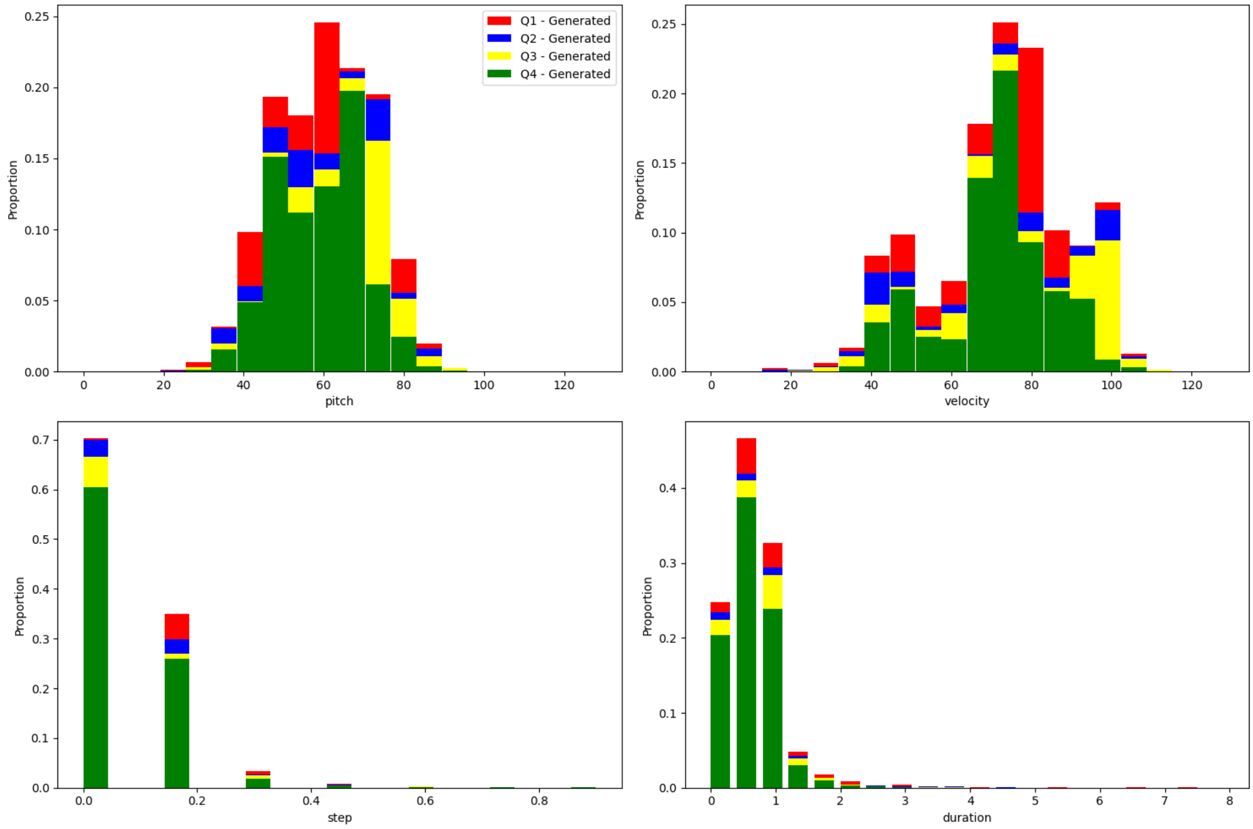


Figure 5. Predicted Distributions

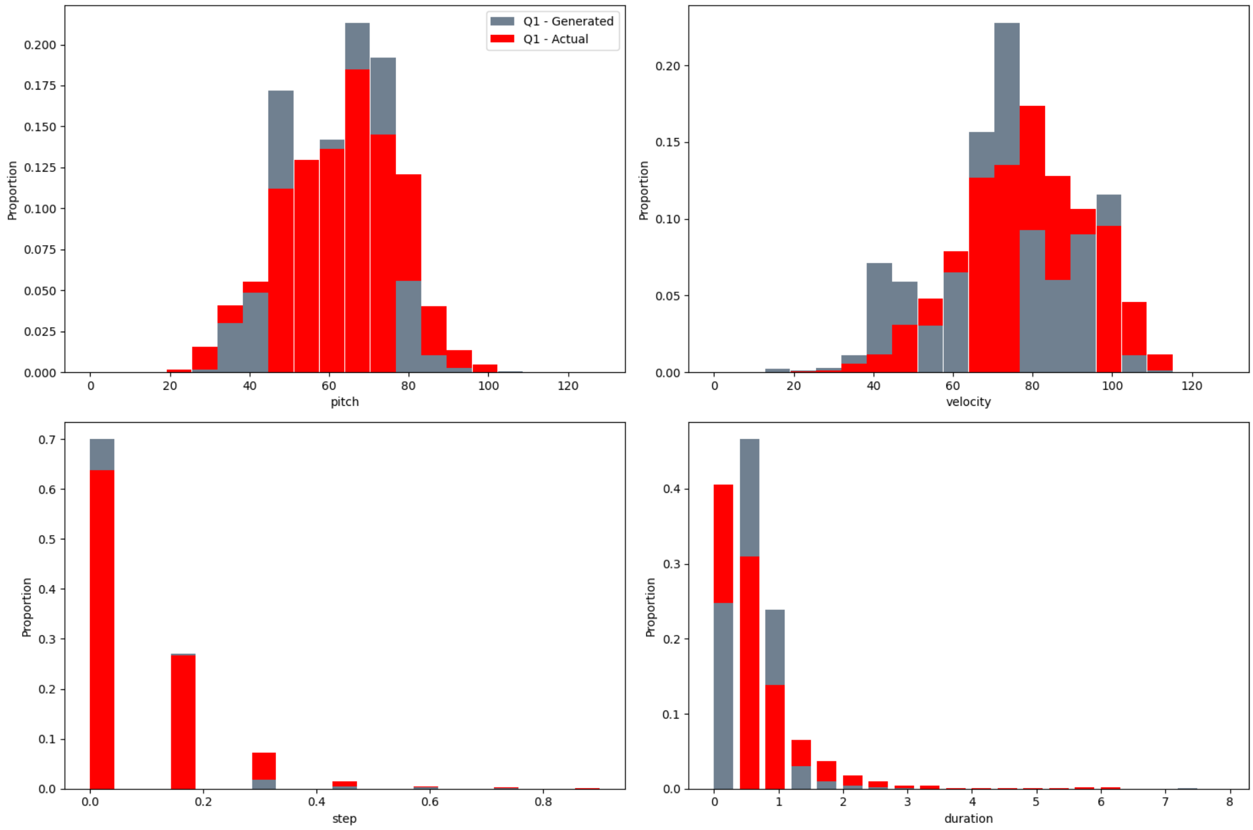


Figure 6. Q1 Actual vs Predicted

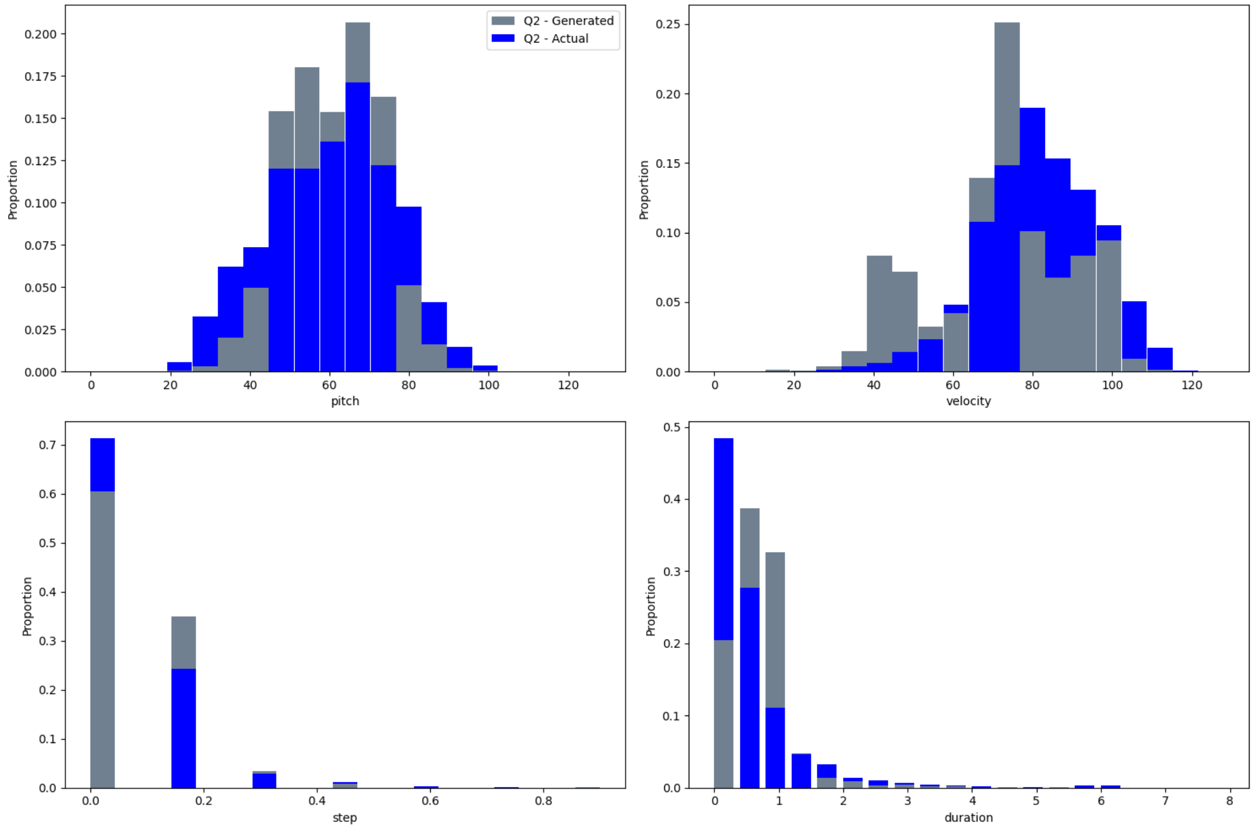


Figure 7. Q2 Actual vs Predicted

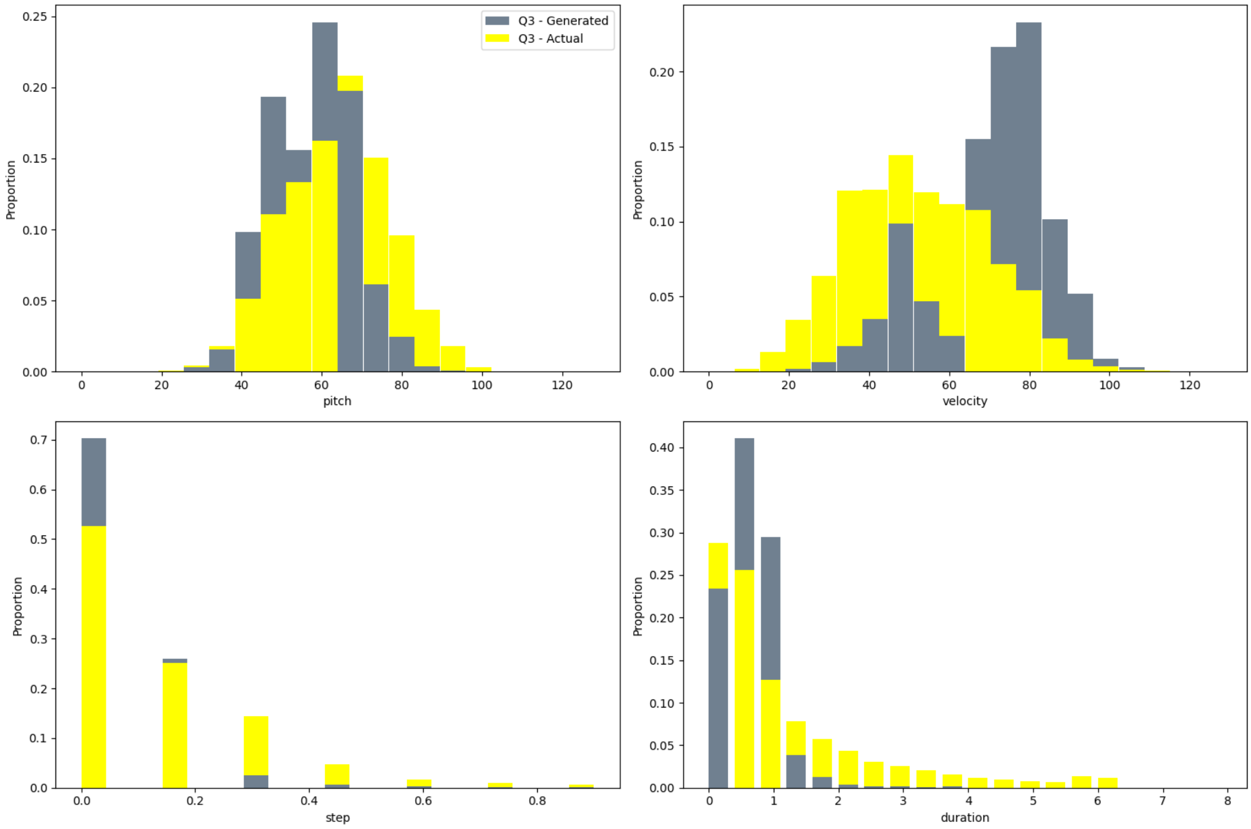


Figure 8. Q3 Actual vs Predicted

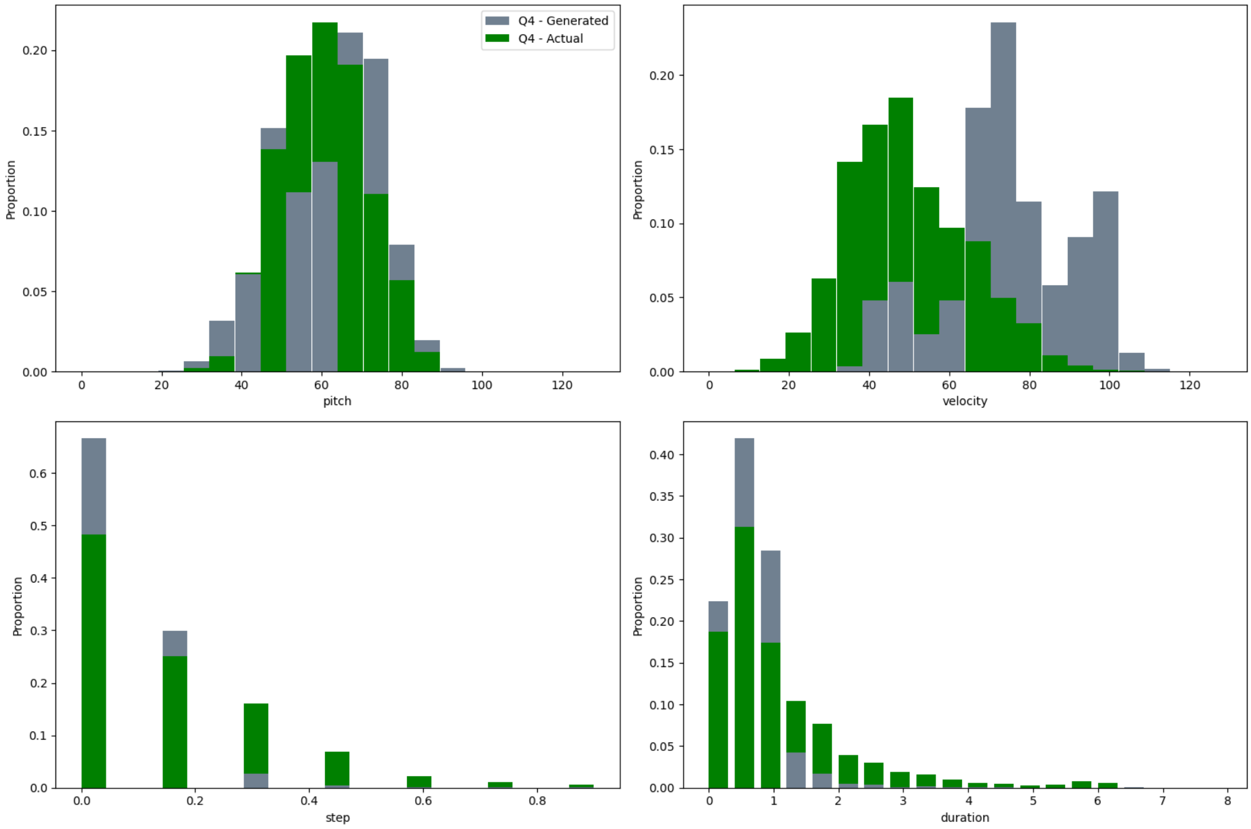


Figure 9. Q4 Actual vs Predicted

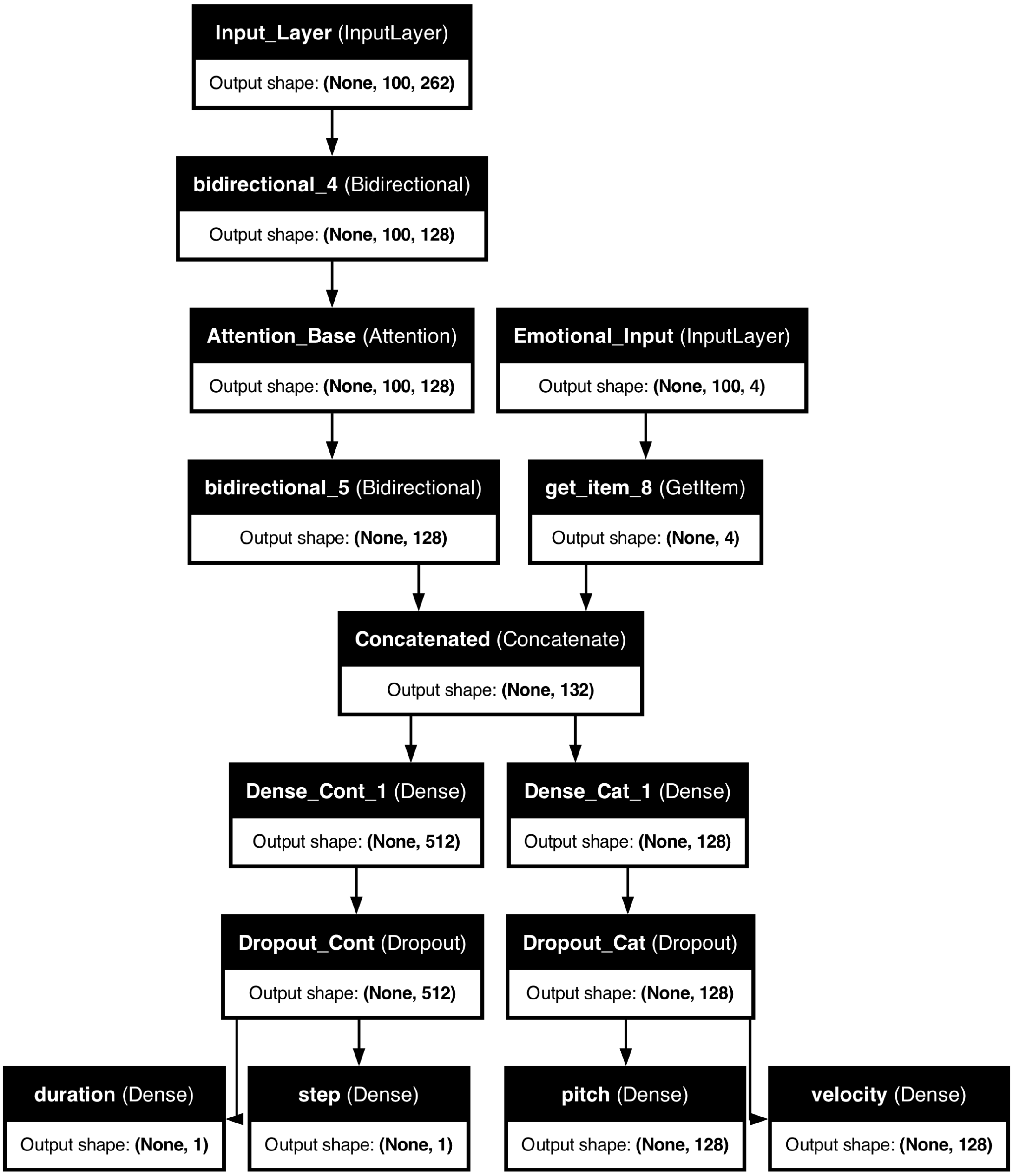

Input_Layer (InputLayer)
Output shape: (None, 100, 262)
bidirectional_4 (Bidirectional)
Output shape: (None, 100, 128)
Attention_Base (Attention)
Output shape: (None, 100, 128)
Emotional_Input (InputLayer)
Output shape: (None, 100, 4)
bidirectional_5 (Bidirectional)
Output shape: (None, 128)
get_item_8 (GetItem)
Output shape: (None, 4)
Concatenated (Concatenate)
Output shape: (None, 132)
Dense_Cont_1 (Dense)
Output shape: (None, 512)
Dense_Cat_1 (Dense)
Output shape: (None, 128)
Dropout_Cont (Dropout)
Output shape: (None, 512)
Dropout_Cat (Dropout)
Output shape: (None, 128)
duration (Dense)
Output shape: (None, 1)
step (Dense)
Output shape: (None, 1)
pitch (Dense)
Output shape: (None, 128)
velocity (Dense)
Output shape: (None, 128)

Figure 10. Model With Downstream 'Emotional_Input'

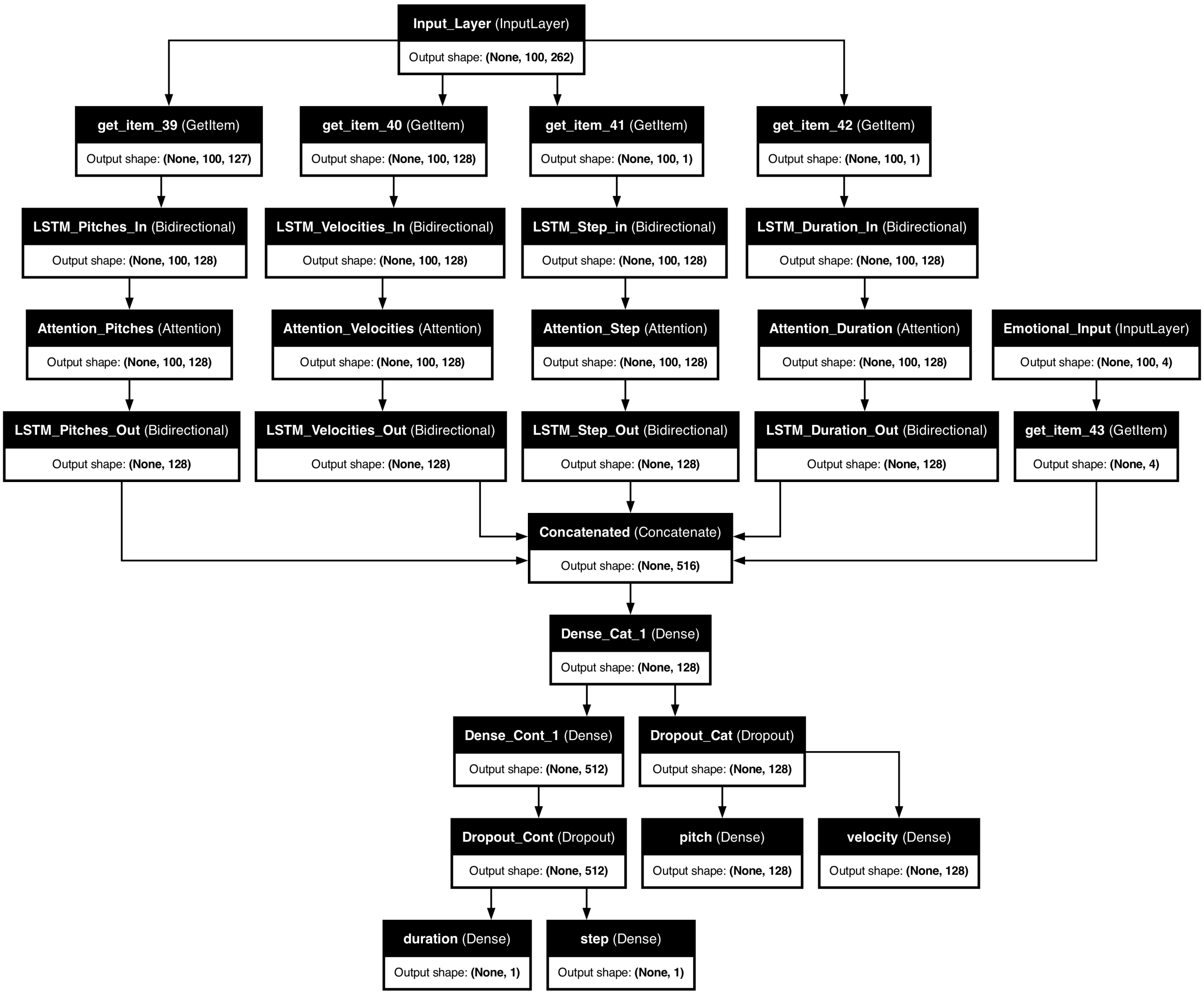

Input_Layer (InputLayer)
Output shape: (None, 100, 262)
get_item_39 (GetItem)
Output shape: (None, 100, 127)
get_item_40 (GetItem)
Output shape: (None, 100, 128)
get_item_41 (GetItem)
Output shape: (None, 100, 1)
get_item_42 (GetItem)
Output shape: (None, 100, 1)
LSTM_Pitches_In (Bidirectional)
Output shape: (None, 100, 128)
LSTM_Velocities_In (Bidirectional)
Output shape: (None, 100, 128)
LSTM_Step_in (Bidirectional)
Output shape: (None, 100, 128)
LSTM_Duration_In (Bidirectional)
Output shape: (None, 100, 128)
Attention_Pitches (Attention)
Output shape: (None, 100, 128)
Attention_Velocities (Attention)
Output shape: (None, 100, 128)
Attention_Step (Attention)
Output shape: (None, 100, 128)
Attention_Duration (Attention)
Output shape: (None, 100, 128)
Emotional_Input (InputLayer)
Output shape: (None, 100, 4)
LSTM_Pitches_Out (Bidirectional)
Output shape: (None, 128)
LSTM_Velocities_Out (Bidirectional)
Output shape: (None, 128)
LSTM_Step_Out (Bidirectional)
Output shape: (None, 128)
LSTM_Duration_Out (Bidirectional)
Output shape: (None, 128)
get_item_43 (GetItem)
Output shape: (None, 4)
Concatenated (Concatenate)
Output shape: (None, 516)
Dense_Cat_1 (Dense)
Output shape: (None, 128)
Dense_Cont_1 (Dense)
Output shape: (None, 512)
Dropout_Cat (Dropout)
Output shape: (None, 128)
Dropout_Cont (Dropout)
Output shape: (None, 512)
pitch (Dense)
Output shape: (None, 128)
velocity (Dense)
Output shape: (None, 128)
duration (Dense)
Output shape: (None, 1)
step (Dense)
Output shape: (None, 1)

Figure 11. Model With Extreme Branching