Deep Learning for Sound-to-Light Automation in Stage Lighting Applications

Manuel Hils

Master's thesis

Deep Learning for Sound-to-Light Automation in Stage Lighting Applications

Manuel Hils

31. October 2023

Chair of Data Processing
Technische Universität München

Manuel Hils. *Deep Learning for Sound-to-Light Automation in Stage Lighting Applications.* Master's thesis, Technische Universität München, Munich, Germany, 2023.

Supervised by Prof. Dr.-Ing. Klaus Diepold and Stefan Röhrl; submitted on 31. October 2023 to the Department of Electrical and Computer Engineering of the Technische Universität München.

Chair of Data Processing, Technische Universität München, 80290 München, Germany, `http://www.ldv.ei.tum.de/`.

Abstract

Stage lighting applications usually require an operator who programs the dependency between sound and light. The industry is working on an improved sound-to-light automation whereas the scientific research in the field is limited. This thesis analyzed how sound-to-light automation can benefit from deep learning.

Therefore, three different data sets were created: A synthetic data set where a light is pulsing with the beat, a trace of a basic real-world light show and an extraction of control signals from a video of a professional light show.

The proposed model consists of a convolutional recurrent neural network (CRNN) that receives a log-Mel spectrogram together with its positive difference as input features. The loss function takes the absolute values of hue and brightness into account, as well as their change compared to the previous sample. Four additional metrics support the evaluation of the performance.

The results showed that learning a direct dependency between low level music features and the lighting output via a CRNN is a promising approach for sound-to-light automation. The first two data sets turned out to be more valuable than the third for this task. While the model learned the general structure of the first data set, especially the second revealed limitations that require further research. The relation between sound and light was not sufficient in the third data set.

Contents

Acronyms

CNN	convolutional neural network
CRNN	convolutional recurrent neural network
DMX512	digital multiplex
FFmpeg	fast forward moving picture experts group
HSV	hue saturation value
LSTM	long short-term memory
RGB	red green blue
RNN	recurrent neural network
STFT	short-time Fourier transformation

1. Introduction

While music typically attracts the most attention at concerts, stage lighting also highly influences the overall experience. It can emphasize the emotions transported by the music. Although machine learning algorithms conquer one creative domain after the other these days, the ideas behind today's stage lighting concepts still depend on the operator in most cases.

Existing light boards and software solutions usually take music into account for jumping through a sequence of scenes or adjusting the speed of effects to the beat. Professional systems also offer a variety of music features that can be used for controlling further parameters of the light show. However, the general idea about which colors, movements, intensities and effects fit to the music, usually comes from the operator in case of diversified shows.

This is a challenge on small events where no dedicated light technician is available: Mobile DJs who want to concentrate on the music or private party organizers might need to rely on basic preprogrammed sequences of their light board. The industry understood this shortcoming and reacted with **increased automation** in the last years. More advanced preprogrammed light shows that take further music features into account enhance the correlation between music and light. They also limit the impression of repeating light sequences these preprogrammed solutions are usually prone to. The software *SoundSwitch* is one of the forerunners that go a step further. It analyzes a song and creates a light show in alignment to the given music. However, it is only possible to precalculate the show for a whole song in advance. Therefore, this feature cannot be used live.

The startups *Limbic Media* and *AI Light Show* recently developed products to overcome this limitation and advertise **artificial intelligence based solutions** that create light shows live. Besides the lack of independent tests until now, little is known about how they work internally.

Sound-to-Light Automation is also subject to **scientific research**. For example, Hsiao et al. [15] use machine learning approaches to extract emotions from the music and choose fitting light parameters afterwards. Nevertheless, the impression is that the resulting functions cannot compete with the complexity of the commercial products mentioned before or with human invented shows.

The **goal of this thesis** is to contribute to reducing the gap between the capabilities of commercial products and scientific research and to analyze a new approach for sound-to-light automation. Therefore, the target is to find out whether deep learning based approaches are able to create a light show directly from the music input, without the intermediate step of emotion estimation. The program shall be designed in a way that it can also run in a live scenario. In order to limit complexity, the output is limited to the color and brightness of one light.

The following research questions are covered within the thesis:

- **RQ 1:** How is it possible to create training data for the different phases of the project?
- **RQ 2:** Which model is suitable for predicting the light output based on the music input?
- **RQ 3:** How can the quality of stage lighting be measured?
  - What kind of loss function can be used for training a model?
  - How can the outcome of different models be compared?

2. Related Work

The number of sources that describe sound-to-light automation is rather limited. Therefore, this chapter also covers related disciplines from the field of music information retrieval. The target is to find existing approaches that can be ported to sound-to-light automation. At the beginning, section 2.1 briefly summarizes the data processing fundamentals the following sections refer to.

2.1. Data Processing

The data processing overview is divided into two deep learning architectures, a procedure for extracting audio features and two representations for colors.

2.1.1. Long Short-Term Memory

Time series forecasting has a long history with the development of classic methods like the autoregressive integrated moving average model for example in the 1970s [29].

Several neural network architectures target time series forecasting, too. Recurrent neural networks (RNNs) belong to this group and use feedback loops to keep an internal state. That way, the model memorizes information from previous time steps and can make predictions depending on the current input and the information from the past [11]. A popular variant of RNNs is long short-term memory (LSTM). Hochreiter and Schmidhuber initially proposed this concept for solving the problem of vanishing or exploding loss signals of previous RNNs in 1997 [14].

Architecture

The architecture of LSTMs described in this section is based on the representation by Zhang et al. [47].

Similar to a standard recurrent neural network, a feedback loop returns the previous output of the model back into the model. The previous output is the hidden state in figure 2.1 of the LSTM unit and it becomes concatenated with the new input values. For simplicity, the result of this concatenation is called "input" in the following. In addition to the hidden state, the unit contains an internal state which is also keeping information across time steps.

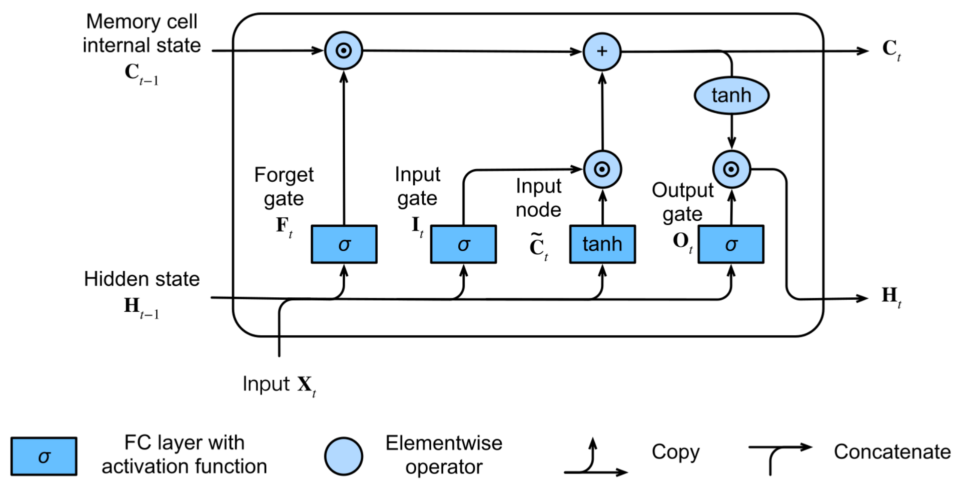


Figure 2.1: Representation of LSTM architecture by Zhang et al. [47]

One important improvement of an LSTM compared to standard RNNs embodies the control of the information flow via three different gates. The input gate determines to which extent the internal state memorizes new information. This means it evaluates how relevant the current input is for future time steps. The forget gate defines which portion of the internal state is forgotten and the output gate states how much the internal state shall influence the output of the LSTM.

All three gates contain fully connected layers that are optimized during the training process, followed by a sigmoid activation function. Due to the activation function, the output of the gates ranges from zero to one.

There is one further fully connected layer with trained weights, but this time with a tanh activation function: The input node depends on the input and contains the new information the internal state shall memorize (if the input gate allows it). The new output consists of the tanh function applied on the internal state multiplied with the output gate.

The hidden size describes the number of features in the hidden state and is a hyperparameter that influences the model's complexity.

Training Process

The time-dependent relation between input and outputs of RNNs requires adapted training processes. One common approach for that purpose is backpropagation through time. The idea can be imagined as unfolding an RNN: The behavior of an RNN after two time steps equals the one of a feed-forward neural networks that contains two instances of this RNN without the feedback loop [38]. This form allows standard backpropagation again.

However, the computational costs of the unrolling process increase for longer sequences and the accumulating numerical error can impede the optimization process [22]. Williams and Peng proposed therefore the truncated backpropagation through time where the backpropagation is limited to a fixed number of time steps [44].

The optimal sequence length that is taken into account for the backpropagation differs between applications. Khodabakhsh et al. [18] showed for example that modifications like the number of features in a multivariate experiment can also influence the best choice.

2.1.2. Convolutional Neural Networks

The introduction of convolutional neural networks (CNNs) improved various applications determinative, such as image classification, object detection or image segmentation in the field of computer vision [21].

Using standard feed-forward neural networks for image recognition would lead to a large amount of trainable parameters because they connect every input pixel of the image via a separate weight with the following layers [1]. From a functional point of view, a major drawback is the sensitivity regarding shifted objects within the image. When the model learned to detect a certain object in a picture, the static connection between each input pixel and the upper layers remembers also the location of the object and would not recognize the object anymore at a different position in the image [47].

The target of convolutional layers is to improve this behavior with filters that perform a convolution across the image. Typically, a CNN consists of several convolutional layers followed by a dropout layer and one or more fully connected layers at the end. While the first convolutional layers extract low-level features like edges, the following layers detect more complex objects based on these low-level features [43].

This is just a short overview for a better understanding of the thesis. A more detailed view on CNNs can be found for example in an article from C.-C. Jay Kuo [20].

2.1.3. Log-Mel Spectrogram

A common method for feature extraction in the field of music information retrieval is the log-Mel spectrogram [28, 45]. It is based on a short-time Fourier transformation (STFT) that yields the frequency distribution of the audio signal over time. The output of the STFT is summarized via frequency subbands [2]. The size of each frequency subband is adopted to the hearing sensation of humans.

In the 1930s, Stevens et al. [39] analyzed how the human ear perceives different frequencies based on experiments. They discovered that the same distance between two frequencies feels different depending on the height of the frequencies.

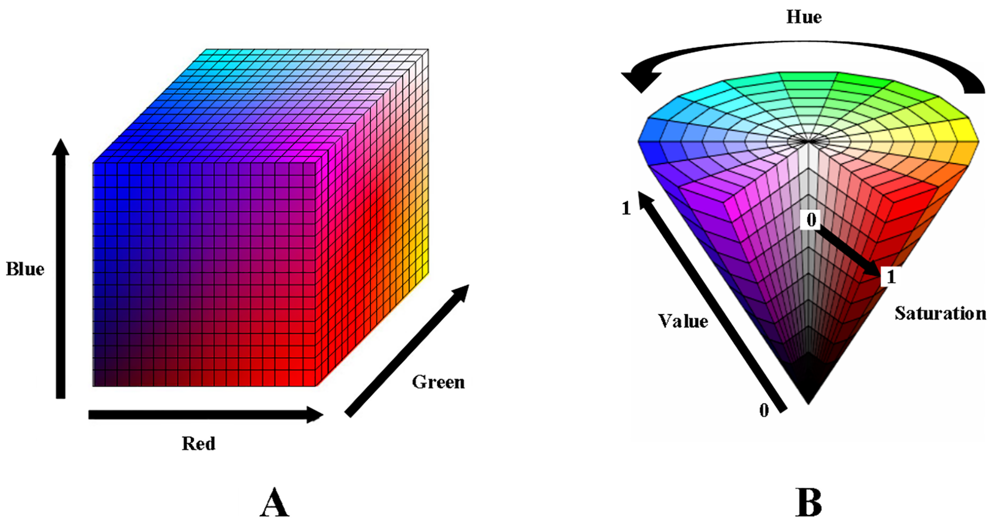


Figure 2.2: Comparison of RGB (A) and HSV color space (B) by Chen et al. [4]

For example, a distance of 100 Hz feels larger when the two tones are 400 Hz and 500 Hz compared to 4 kHz and 4.1 kHz. The target of Stevens et al. was to find a scale for frequencies so that the distance between two tones on this scale matches the distance human listeners sense. The result is called the *Mel scale* and was originally formulated as a table. O'Shaughnessy [10] proposed the approximation

$$\mathrm{Mel}(f) = 2595 \log_{10}\left(1 + \frac{f}{700}\right) \mathrm{mel} \tag{2.1}$$

for converting a frequency f to the Mel scale.

In a log-Mel spectrogram, the frequency subbands have an equal size on the Mel scale and the obtained amplitudes are displayed in decibel. Figure 3.7 shows an example of such a spectrogram.

2.1.4. Color Spaces

There is a variety of color spaces proposed in the literature in order to describe the appearance of light. Relevant for this thesis are the RGB (**r**ed, **g**reen, **b**lue) and the HSV (**h**ue, **s**aturation, **v**alue) model that are visualized in figure 2.2.

The RGB model follows the biological processing of colors in the human eye and defines the light as a combination of red, green and blue and many of today's applications such as digital cameras or screens employ this color space [23].

Smith proposed the HSV model as a color model designed to be closer to how humans perceive light [37]: It expresses the light with the dimensions hue, saturation and value. Value is a measure of brightness in this case. In the following chapters this is referred to as "brightness" because "value" can be ambiguous with other values. A conversion from the RGB color space to the HSV representation is possible via the definition [34]:

$$R, G, B \in [0, 1];\; MAX = \max(R, G, B);\; MIN = \min(R, G, B)$$

$$H = \begin{cases} 60^\circ * (0 + \frac{G-B}{MAX-MIN}, & \text{if } MAX = R \\ 60^\circ * (2 + \frac{B-R}{MAX-MIN}, & \text{if } MAX = G \\ 60^\circ * (4 + \frac{R-G}{MAX-MIN}, & \text{if } MAX = B \\ 0, & \text{if } R = G = B \end{cases} \tag{2.2}$$

$$S = \begin{cases} 0, & \text{if } R = G = B \\ \frac{MAX-MIN}{MAX}, & \text{else} \end{cases} \tag{2.3}$$

$$V = MAX \tag{2.4}$$

2.2. Stage Lighting

Sound-to-light automation is subject to scientific research as well as a development target for commercial products. However, the number of available publications in this research area is very limited compared to other machine learning applications. The following sections provide an overview of the current state.

2.2.1. Scientific Research

Hwangbo et al. [16] created a basic sound-to-light system that shares similarities with light organs from the 1970s: The colors green, blue, white and red are assigned in increasing order to a frequency band of 300 Hz each. The brightness depends on the amount of change in the amplitude of the music.

In contrast, Nouvel et al. calculate the power of the audio signal in the frequency band below 100 Hz in order to distinguish the intensity of the lights [32]. Above a certain noise threshold, there is a linear relation between the power inside the frequency band and the brightness. The hue is selected either randomly or via a remote control in this proposal.

Nguyen et al. [24] developed a model that estimates the emotion of a segment of the song as an intermediate step. The emotion is described in the Thayer emotion plane which is a two-dimensional representation that expresses arousal and

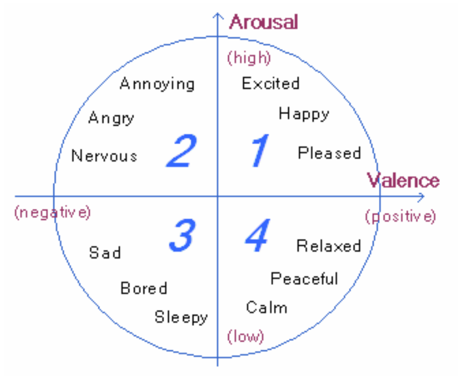


Figure 2.3: Visualization of the Thayer emotion plane by Yang et al. [46]

valence [40] as displayed in figure 2.3. The model calculates the location inside the Thayer plane with a fixed formula with five different music features as input. The second step is to assign fitting lighting effects to each emotion. Therefore, the Thayer plane is divided into 12 different emotions such as excited, relaxed or bored. The mapping of the 12 emotions to the light output is static. This leads to a constant color as long as the emotion is in the same area and the beat of the song is not considered. In addition, the quality of the emotion recognition with a linear function of five music features needs to be verified.

Hsiao et al. [15] used support vector machines for the emotion estimation instead. Although the idea of emotions as an intermediate step is the same as in the work of Nguyen et al., Hsiao et al. extended the program with further components: Hue and saturation are calculated for a segment by support vector regression with both the emotion and the genre as input. The genre detection consists of a similarity comparison with songs from a data base. In addition, the brightness is calculated with the average intensity of a song segment and the current onset signal. The dynamic change of the onset creates a flashing effect in accordance with the music.

The segment based algorithm requires a segment detection that works with an energy envelope: When the energy envelope crosses dynamic boundaries, this is a potential segment boundary. However, there are further steps for selecting a subset of the potential segment boundaries and for fine-tuning the exact time of the transition between the segments.

For achieving training data, five professional light technicians labelled example song extracts with hue values. The relation to the music emotions are difficult to distinguish in the resulting color maps (see figure 2.4): While some genres as metal, reggae or country received an almost constant hue, a variety of colors are assigned to other genres like rock and pop across the Thayler plane.

As a consequence of the segment based approach, the algorithm works only

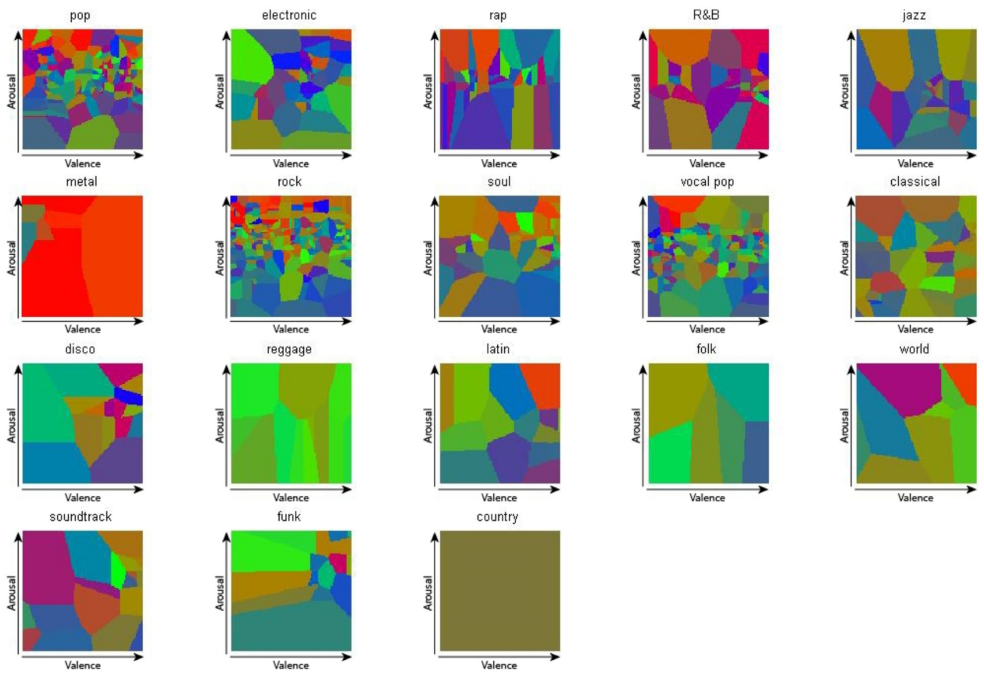


Figure 2.4: Results of Hsiao et al. [15]: Selected hue depending on genre and emotion

precomputed in this form: In a live scenario, the average intensity that is needed for the brightness would only be known at the end of the segment. The same holds for the emotion as input for the hue and saturation selection.

2.2.2. Commercial Products

There is a **tradition** of developing products that align music and light since centuries. In 1877 for example, Bainbridge Bishop constructed a lighting device as an extension for an organ [35]. Initially driven by sunlight, the device used levers and shutters in order to visualize the played music on a small screen [35].

Except from the common motivation, today's commercial products share few similarities with the machines from the 19th century. The light control board *grandMA3* from *MA Lighting* stands exemplary for products that are typical in **professional environments**. According to its user manual, the board calculates audio features, such as the beat or the intensity of different frequency bands. Those values can not only be used for controlling the transitions between lighting scenes, but also for setting any parameter in the program [9]. Since this mapping between audio feature and lighting parameter has to be programmed by a technician for a specific application, this can be categorized as a semiautomatic sound-to-light solution.

Entry level light boards offer less features which reduces the amount of necessary preparation effort on the one hand. On the other hand, it leads to a frequent repetition of lighting effects and a poorer relationship between sound and light. The *DMX LED EASY Operator Deluxe* from *Eurolite*, an example for this category, brings 12 preprogrammed sequences where the transition between the steps can be aligned to the beat [8].

In the last years, companies tried to combine an increased complexity of the light shows together with automation. The software *SoundSwitch* for example introduced the function *AutoScripting*. *AutoScripting* splits a song into different segments such as intro or chorus and prepares a lightshow in accordance to the music automatically [6]. However, the whole song has to be analyzed beforehand. A backup solution for a live setup is the function *AutoLoop* which offers preprogrammed sequences that are synchronized with the beat [6]. However, the choice of the preprogrammed sequence is up to the operator which reduces the level of automation for this function.

The startups *Limbic Media* and *AI Light Show* both announced that they started shipping of **artificial intelligence based light controllers** in 2023. Usually, the vendors of commercial products do not explain details of the functionality, but *Limbic Media* filed a patent that gives at least insights into the potential structure of the program [25]. The patent mentions frequency and envelope analysis of the audio signal as input features and a database of previously processed sound signals in order to find a base for the current light control. In addition, it states that neural networks can be incorporated for the detection of music genres, patterns or speech styles.

2.3. Music Information Retrieval

Despite the lack of deep learning based papers described in section 2.2.1, there are related music information retrieval disciplines that employ deep learning and provide insights for this thesis: The connection between stage lighting and **music emotion recognition / music classification** can be seen in the work of Hsiao et al. [15] which relies on the estimated emotions and genres as an intermediate step for stage lighting. At the same time, creating light shows has also a strong correlation to **beat tracking**, since a common strategy of the commercially available solutions (described in section 2.2.2) is to synchronize light changes with the beat.

Therefore, the design choices of deep learning models out of these two disciplines are analyzed as a foundation for the approach presented in section 3.4.

2.3.1. Feature Extraction

A common procedure within the analyzed papers is to rely on log-Mel spectrograms as input parameters for the model [3, 5, 12, 26, 41]. Hizlisoy et al. [13] include a variety of further features (e.g. timbre, tonality, harmonic change detection function, centroid etc.), but this is an exception within the selection.

However, there is still a variation of how the spectrograms are calculated in detail. The window length reaches from 23.2 ms until 92.9 ms while 46.4 ms is a typical choice. Cheng et al. [5] and Böck et al. [3] use the three different window lengths in combination (23.2, 46.4 and 92.9 ms) which increases the input threefold. The hop length differs from 10 ms to 46.4 ms. While the frequency range typically covers the range from 30 Hz - 17 kHz, the number of frequency bins varies between 20 and 136 in the considered papers. This amount influences the number of input features for the model. It has to be noted that the small values of 20 and 36 are only chosen by papers that rely on three different window lengths which increased the number of features this way.

After obtaining the log-Mel spectrograms, similar steps were proposed in the literature in order to create further features. While the network *BeatNet*, proposed by Heydari et al. [12], uses the first order difference of the spectrogram as additional input features, others employ only the positive values of the first order difference [5, 41]. Böck and Schedl [3] compute a median of the spectrogram over some sampling points, before taking the positive values of the first order difference of this median spectrogram. The results reveal an improved behavior when taking only the positive values compared to taking both positive and negative values or leaving out the first order difference completely.

2.3.2. Model architecture

Within the analyzed papers, convolutional recurrent neural networks (CRNN) are a typical design choice for beat tracking systems as well as music classification and emotion recognition. A CRNN is a network that combines convolutional and recurrent layers. Vogl et al. [41] proved that alternative approaches with RNNs or CNNs can also provide good results depending on the specific task and the data set.

3. Methods

Divided into six sections, this chapter first describes the interfaces to interact with the model, before it explains the choice of a color space. Section 3.3 gives insights about the creation of three different data sets while section 3.4 explains the model itself. Section 3.5 states not only the metrics that are used for the final performance measurement, but also the loss function that is the base for the training process in section 3.6.

3.1. Interfaces

3.1.1. Audio Input

There are two different use cases concerning the audio input: During development, prerecorded audio files are imported via the Python package *Librosa* [27]. It supports a variety of audio formats as MP3 or WAVE for example. In a live scenario, a microphone gathers the input data. The module *python-sounddevice* retrieves the input from the computer's microphone in this case.

3.1.2. Light Output

Displaying the output in form of a video enables the development of the model independent from lighting hardware. Therefore, a Python wrapper [19] for *FFmpeg* creates an MP4 file based on the numpy matrices given out from the model. However, an alternative output form is necessary to interact with real hardware in a live scenario.

There are two common protocols for controlling fixtures: DMX512 (**d**igital **m**ultiple**x**) and *Art-Net*. DMX512 is a serial protocol developed in the 1980's and allows 40 data frames per seconds with each frame containing 512 channels [31]. Every channel can have a value between 0 and 255 and carries the information for controlling a specific parameter of a light, e.g. the brightness, the intensity of a certain color or an angle it should turn to.

The development of more sophisticated light shows in the following decades led to a demand of an increased number of channels. A solution for this is *Art-Net* which is a protocol for sending large amount of DMX512 data over Ethernet [31]. However, DMX512 is a sufficient protocol choice for this thesis because the target is to control only a single light. Therefore, a USB to DMX512 adapter together with the Python program *PyDMX* [36] facilitates the control of fixtures in a live scenario as an alternative to the video output.

3.2. Color Space

A broad variety of fixtures which are capable of mixing colors works based on the standard RGB color model for additive color mixture. Often, four channels control those lights: The intensities for the colors red, green and blue and a dimmer value. For the compatibility with those lights, the final **output** of the program is according to the RGB color model. The dimmer channel is sent out with a constant value of 255 which corresponds to the highest possible brightness. Reducing the brightness is nevertheless achieved by reducing the red, green and blue values together. A trace of the *eurolite DMX LED easy operator deluxe*, an entry level control board for lights, shows that it works it in the same way. So, the assumption is that a high number of commercially available fixtures understands such a control pattern.

Although the model sends out commands according to the RGB color space as the final output, it works **internally** based on the HSV model due to its more intuitive interpretation. However, the saturation equals the maximum value all the time in order to limit complexity. Although white light or less saturated other colors also play a role in stage lighting applications, the absence seems to be a minor drawback. The personal impression is that most of the time colorful lights dominate the light shows in clubs or at concerts.

Hue and value remain as the outputs the model has to predict. A postprocessing step converts it to the RGB color model afterwards.

3.3. Data Sets

A variety of data sets is publicly available for many machine learning tasks and can be used for training new models. In the area of stage lighting, something similar could not be found yet. So the first step is to create data sets for this particular task. Three different approaches allow a stepwise verification and improvement of the model.

1. A **synthetic data set** where a red light pulses with the beat provides the opportunity to setup the model with deterministic training data, just one relevant output and simplified performance metrics. The majority of the thesis works with this data set.

2. A trace of a **basic real-world light show** with a sawtooth profile of the brightness and slightly changing colors displays how well the model architecture developed with the synthetic data set can also cope with a bit more complex data sets.

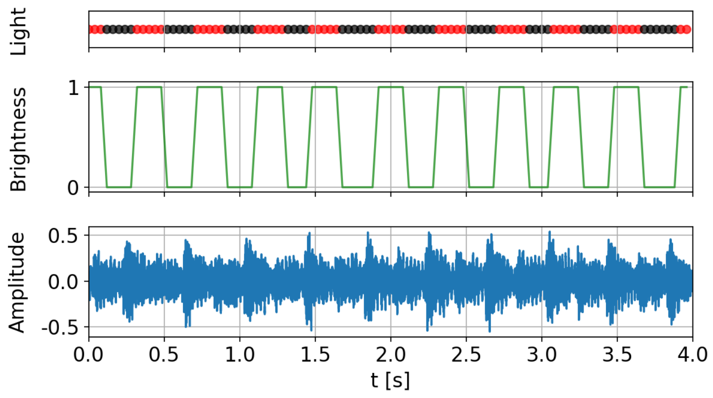


Figure 3.1: Example of Data Set 1 output

3. The third data set is obtained from a video of a **professional light show**. The target of this data set is to find out whether also a model with practical relevance can be trained with the same approach.

3.3.1. Data Set 1 – Synthetic Light Control

The built-in beat detection of *Librosa* forms the foundation of the synthetic data set. It estimates beat events based on the onset strength as proposed by Ellis [7]. Each detected beat event sets the brightness to the maximum value for 200 ms before it returns to zero. The hue is permanently zero which results in a red light flashing according to the beat.

Figure 3.1 shows an example extract of the brightness signal together with the corresponding music input. On the top of the figure, the translation of hue and brightness to the RGB model visualizes the visible light. As in the whole thesis, the saturation stays 1 for this conversion.

Music Selection

Preliminary tests showed difficulties with the reproducibility of the results. Therefore, the *Spotify* playlist *Mainstage* with 50 songs replaced the initial music selection of 15 songs from different genres. The playlist *Mainstage* contains mainly songs that can be classified as electronic dance music. As a consequence, the results might not hold for other genres, especially not for those with less strong beats. On the other hand, this makes the selection a good starting point for the first setup of the model in simpler conditions.

Nevertheless, the difficulty of detecting the beat varies between the songs. While the base drum is clearly visible in the amplitude of songs like 'When You're Lonely' from VIZE and Emma Steinbakken, this is more difficult for a calm segment of

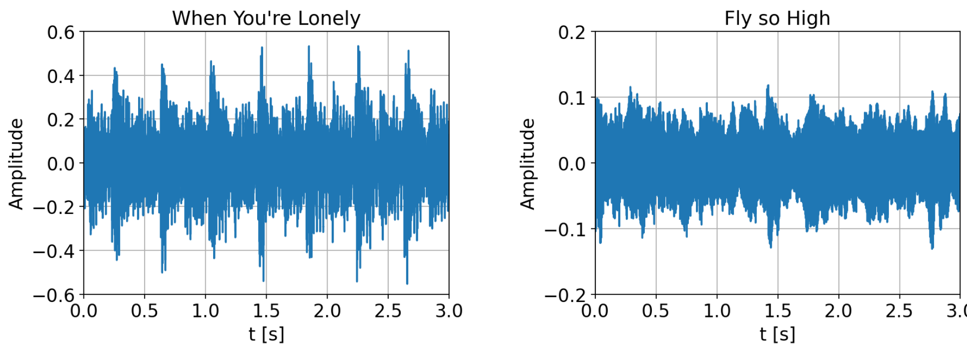


Figure 3.2: Comparison of song extracts with different difficulty for beat detection: 'When You're Lonely' from VIZE and Emma Steinbakken and 'Fly so High' from Zombic, Rocco and Steve 80

the song 'Fly so High' from Zombic, Rocco and Steve 80 as displayed in figure 3.2. Hence, these two extracts will be used as examples when visualizing the predictions of the model.

The chances of generalization increase by choosing music with varying tempo. If all songs in the data set had a similar tempo, the model might learn the temporal sequence of outputs without respect to the music. Within the selected songs, the tempo varies between 122 and $170\,\text{bpm}$. Figure 3.11 visualizes the distribution of the tempo while table A.1 in the annex contains the complete list of songs together with their tempo. All songs are cut after $2{:}00\,\text{min}$ in order to obtain the same length.

3.3.2. Data Set 2 – Basic Real-World Example

Source of Control Signals

A trace of a basic real world light control serves as an intermediate step between the data sets described in the sections 3.3.1 and 3.3.3. The function *Autoloop* of *SoundSwitch* acts as the source of the lighting data. Although a variety of predefined *Autoloops* with e.g. different color schemes are available and they can also be customized, the choice here is to use only the option "Pulse Blue". It pulses the brightness with a saw tooth profile according to the beat while the color is varying from blue to turquoise and purple. Due to the static choice of just one *Autoloop*, the result is still a basic light control without practical relevance. Nevertheless, the new brightness profile and the added colors add more complexity compared to the synthetic data set from section 3.3.1.

The audio input consists of the same songs as in Data Set 1 (see table A.1).

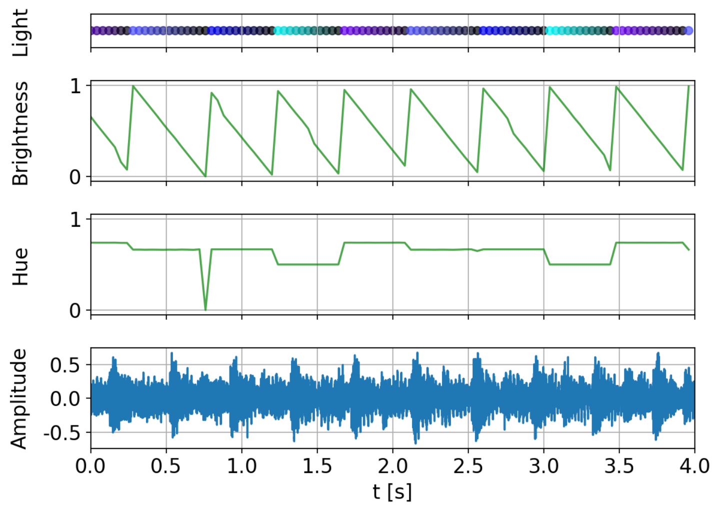


Figure 3.3: Example of Data Set 2 output

DMX512 Logger

Since *SoundSwitch* does not provide an integrated function for exporting traces, a DMX512 logger was built for that task. An *Arduino Leonardo R3* and a *CQRobot DMX Shield MAX485* act as the hardware base for the logger. From software point of view, the *Arduino* receives the DMX signals with the library *DMXSerial*. It forwards the received data via a serial connection to a PC where a Python script parses it. The Python package *sounddevice* records the audio input in parallel so that the lightshow is obtained in synchronisation with the music input. An example of the outcome can be seen in figure 3.3.

3.3.3. Data Set 3 – Video of Professional Light Show

Parsing the Video

For parsing a video of a light show in order to obtain more advanced training data, each frame of the video is split into four segments. Every segment is then condensed to a single output light with a hue and a brightness, even though the segment of the input image might contain a variety of different lights.

The brightness of the condensed light

$$v_{con} = \min\left\{10\,\frac{N_{pixels} \mid v_{pixel} > v_{min}}{N_{pixels}}, 1\right\} \tag{3.1}$$

depends on the proportion of pixels who's brightness v_{pixel} is above the boundary v_{min} of 40%. The boundary and the factor 10 inside 3.1 are chosen after trying

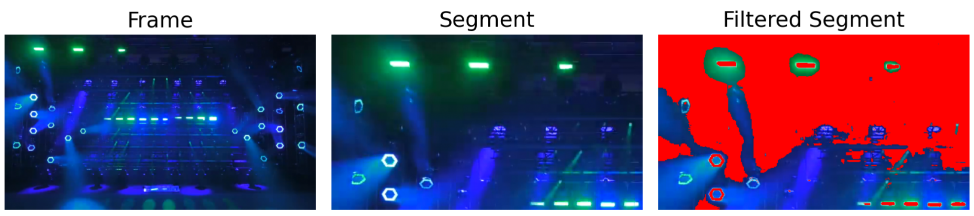


Figure 3.4: Splitting and filtering of a video frame

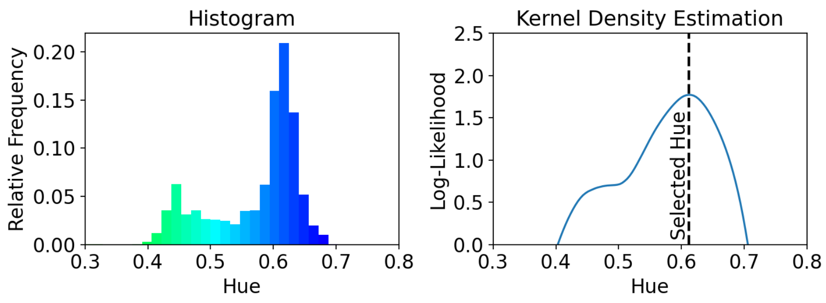


Figure 3.5: Hue selection for filtered segment of figure 3.4

different values. The minimum function ensures that the brightness is limited to the maximum range of the color model.

Before picking a hue h_{con} of the condensed light, a filter reduces the pixels of the frame segment to those with a brightness and saturation of more than 40%. This eliminates the influence of dark areas around the lights as well as the impact of overexposed pixels when a light is shining directly into the camera. The removed parts are colored red in figure 3.4.

The dominant color of the remaining pixels in the segment is retrieved by a density estimation with a linear kernel and a bandwidth of 0.1 that is computed over the color distribution. The maximum turning point of the resulting curve is selected as hue h_{con} (figure 3.5).

Although this method generates information about four different lights, only the upper left is employed until now. Figure 3.6 displays an example excerpt.

Video Selection

YouTube provides an almost unlimited access to potential training data. Starting with specific stage lighting videos, the data set could be extended with records from live concerts or music videos in theory.

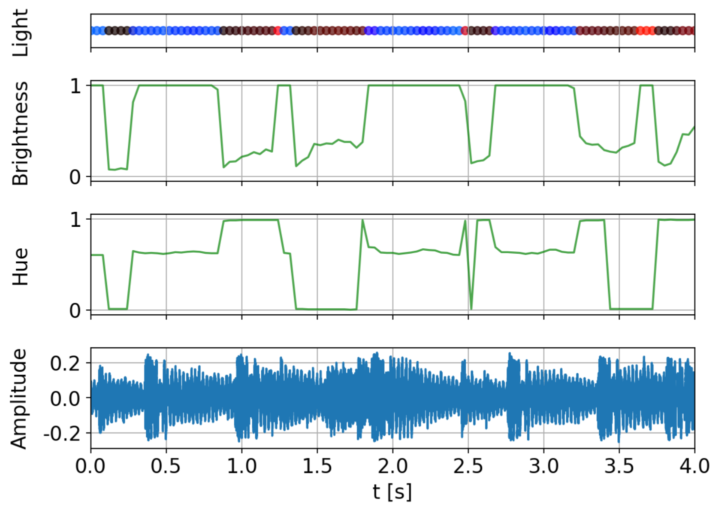


Figure 3.6: Example of Data Set 3 output for an extract of the song 'Good Times Roll' from Big Gigantic and GRiZ

The base for this data set is a video[1] uploaded by *GLP*, a manufacturer of lighting equipment. It is a medley composed of eleven songs listed in table A.2 and has a length of $8\,\mathrm{min}$ in total. Since the model predicted too constant output values during first tests with this data set, another variant that is shortened to a subset of five songs with higher dynamic was added. The selection of the songs is also given in table A.2. Contrary to the other two data sets, this set contains songs from different genres.

3.4. Model

Since the target is to analyze whether deep learning can improve stage lighting automation, the models for beat detection systems and music emotion recognition described in section 2.3 are a promising starting point. Although both disciplines are related to stage lighting, the emphasis within this thesis lies on the affinity to beat detection because the data sets 1 and 2 do not have a correlation with the emotions of the underlying music.

In addition, the beat is better visible in diagrams and the quality how well light and beat fit together is expected to be less subjective than emotions. This simplifies the description in a thesis without audio or video support. Therefore, the proposed solution is oriented closer towards the beat detection papers out of section 2.3.

[1]GLPimpression. "GLP Prolight+Sound 2017 Show". www.youtube.com/watch?v=HAFVJAKpzFk (visited on 10/30/2023).

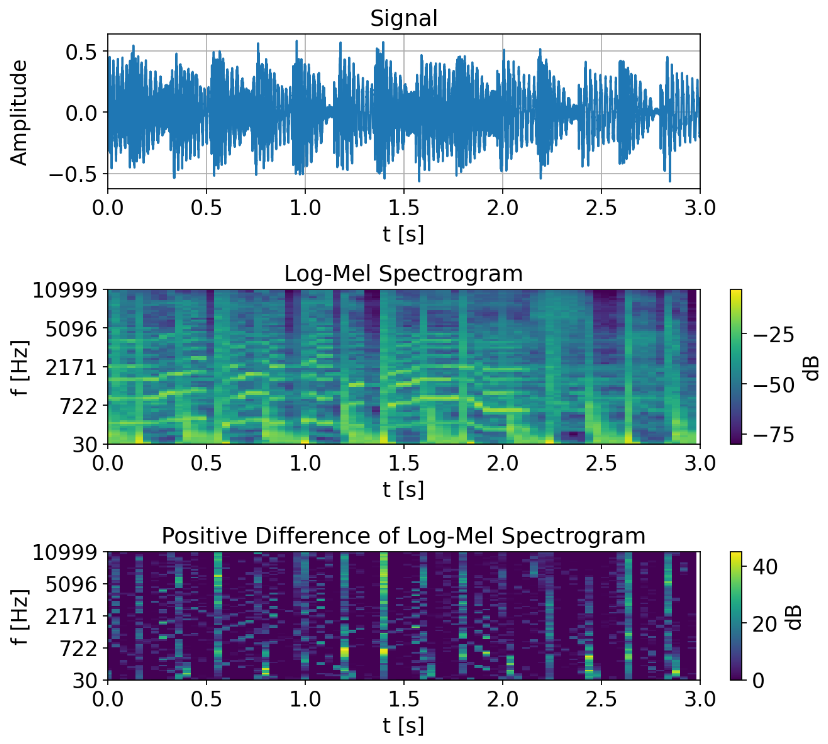


Figure 3.7: Excerpt of the audio input together with the corresponding log-Mel spectrogram and its positive difference

3.4.1. Feature Extraction

Following the state of the art for deep learning based beat detection systems, the initial features consists of a **log-Mel spectrogram**.

The hop length is aligned to the frame rate of the video stream from section 3.3.3 with 25 frames per second. This corresponds to a distance of 40 ms between two frames. Since 40 ms lies inside the range of the selection in previous papers, this is chosen for this thesis while the window length amounts to 46.4 ms. Although this is a common choice by itself, the combination with a hop length of 40 ms leads to only minimal overlapping between the windows of individual samples which was not seen in the other papers. However, a short window length promises faster responsiveness in a live scenario so this approach is pursued.

Due to the smaller sampling frequency of the input data (22.05 kHz instead of 44.1 kHz in the analyzed papers), also the frequency range is reduced to 30 Hz - 11 kHz. For the number of frequency subbands a midsize value of 80 is tried.

The addition of the **positive first order difference** of the spectrograms facilitates the adjustment of the model to short-term events like a drumbeat and doubles the number of input features to 160. Figure 3.7 shows an example excerpt of the song with the audio input, the log-Mel spectrogram and its positive difference.

While the values of the log-Mel spectrogram range from -80 dB to 0 dB, the positive difference contains values between 0 dB to 76 dB. A basic normalization step increases the values of the log-Mel spectrogram by 80 dB in order to adjust the value ranges of the two feature categories.

3.4.2. Model Architecture

Inspired by the common approach of the analyzed beat detection systems, a CRNN forms the base for the lighting automation (pictured in figure 3.8). The architecture promises an appropriate combination for this task: As stated in section 2.1.2, **convolutional layers** are the state of the art for image recognition. Interpreting spectrograms is a similar task: It is of minor importance whether the location of a maximum is a frequency subband higher or lower. The convolutional layer can extract the general structure of the spectrogram instead.

The **recurrent layer** in comparison is predestined for learning the temporal context. It consists of stacked LSTM layers. Following the *BeatNet* architecture [12], a 1D max pooling and a linear layer with a ReLU activation function connect the convolutional and the recurrent layer.

For the final output, another linear layer and a sigmoid activation function follow the recurrent layer. The sigmoid function intrinsically ensures that the output remains between zero and one, which is the desired range for hue as well as for brightness.

Dropout layers inserted after the max pool layer and after each LSTM layer except the last one are supposed to prevent the model from overfitting.

3.4.3. Parameters

The input size of the convolutional layer is determined by the number of input features (160). Whereas *BeatNet* uses a filter size of 10 for processing 272 features, this is proportionally downscaled and rounded to a filter size of 6. So, the kernel in the convolutional layer faces a similar frequency range. Cheng et al. [5] investigated explicitly the optimal filter size where 7 was a good compromise for 216 features over different data sets. Downscaled to 160 features, it would result in a theoretical filter size of 5.18. Although it has limited expressiveness due to differences in the feature extraction process and the use for four convolutional layers, it indicates at least that 6 is a reasonable choice.

The stride is set to one and since there is no benefit of keeping the original dimensions in this application, the convolutional layer is employed without padding. The following max pool layer with kernel size 2 is adopted from *BeatNet* again. The

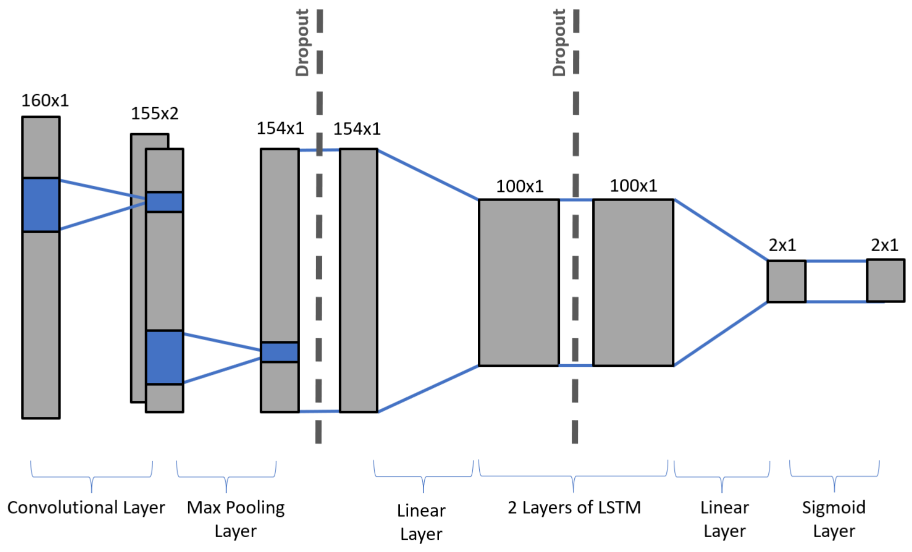


Figure 3.8: Architecture overview and data dimensionality for an example with two stacked LSTM layers and a hidden size of 100

linear layer is fully defined by the output dimension of the max pool layer and the input dimension of the recurrent layer which is set to 160.

Preliminary tests showed that the hidden size of the LSTM, the number of stacked LSTM layers as well as the dropout rate drastically influence the behavior. Therefore, all three are chosen as hyperparameters to be optimized.

3.5. Performance Metrics

Performance metrics are a prerequisite for the training process described in section 3.6. Without a loss function, back propagation would be impossible. In regression problems, the straightforward approach is to measure the difference between expected output y and the prediction $f(x)$. According to Wang et al. [42], one of the most common loss functions for these problems is the square loss

$$L(y, f(x)) = (y - f(x))^2 \tag{3.2}$$

which is also called the L2 loss.

Additional metrics support the evaluation of the final result: The loss function might not be able to represent the quality of the output regarding all aspects. The additional metrics can then be used to compare the results after the learning process. Those additional metrics have an increased importance in this application because the loss function itself contains hyperparameters. So, comparing the loss at the end of the training process of models trained with different hyperparameters would not unveil which model does the best.

3.5.1. Loss Function

Directly feeding hue and brightness into the L2 loss might lead to undesired behavior: For example, there is not a single correct hue at a certain point in time. Although Hsiao et al. [15] suggest to choose the color depending on the emotion and the genre of the song, the results of the paper showed that the correlation is rather limited.

Sometimes, a **change of the color** from red to yellow in the correct moment might be almost as good as a change from green to blue in the ground truth. Therefore, the solution proposed in this thesis is to add the first order difference of hue and brightness together with their absolute values as input for the loss function. So the model tries to output the correct color and brightness values, but it is also rewarded if it predicts a change of hue and brightness correctly. To understand the effect of the absolute and the relative part of the loss function, they are weighted and the ratio between them are changed as hyperparameters.

While the first order difference of the brightness is calculated the standard way by subtracting two consecutive values, the **circular nature** of hue has to be considered: The hue values zero and one represent the same color. Therefore, the distances are calculated clockwise and anticlockwise and the minimum of both is regarded as the difference between the two hue values.

Another issue observed during preliminary tests is the **side effect between hue and value**: According to the definition in 2.1.4, hue is zero when the red, green and blue components are equal. Figure 3.3 shows such an example. At 0.75 s into the segment, a blue light is pulsing and as soon the brightness reaches 0, the hue follows. This would correspond to a red light although the light never shines red in this example. When displaying the state of the light, this has no effect since the light is not shining and it does not matter which theoretical hue value is assigned to it in this moment.

However, when using the hue value in the loss function, this behavior induces unjustified hue changes. The model shall not learn a transition between blue and red in the example above. In order to avoid concept changes like a different color representation, an added mask prevents the model from learning the meaningless hue values. All colors and color differences where the corresponding brightness is below 0.2 are masked out before the loss function.

3.5.2. Evaluation of the Result

The weights in the loss function prevent a comparison of the quality of the different models by having a look solely at the validation loss. In addition, the loss function only compares how close prediction and ground truth are in each point in time individually. For measuring how well the different moments fit together as a sequence, another metric is needed.

The challenge when proposing an additional metric is that light shows can be seen as a form of art where measuring the quality with an objective formula is always a constrained goal[2]. Being aware of the limitations, there are still some indicators that can be used for comparing the results.

Distribution Metrics

Early tests showed that some models tended to produce high frequency flashing or to have a constant output. In order to sort out those models, two kind of performance indicators evaluate the distribution of the output: The **standard deviation** of the output and the **average of the output's first order difference**.

Tempo Metric

The previous two metrics are universally applicable to all data sets. Since there should be a light pulse at each beat, Data Set 1 simplifies measuring how well music and light correspond. So the comparison of the song's tempo and the tempo of the light pulses extend the metrics.

The general idea how for example Hsiao et al. [3] estimate the tempo of a song based on the periodicity of music features works also for the tempo of light pulses. The autocorrelation function

$$R_{vv}(\tau) = \frac{1}{N}\sum_{k=0}^{N-1} v(k)v(k+\tau) \tag{3.3}$$

of the brightness signal v reveals the periodicity of the signal [17]. As the brightness should have a maximum point with each beat, the location of the maximum of the auto-correlation function $\arg\max_{R_{vv}}$ can be interpreted as a measure of the detected tempo. It displays how many samples are typically between two light pulses.

Interpolating the auto-correlation values with a piecewise cubic polynomial before evaluating the periodicity improves the accuracy because it allows an odd number of samples as a result. Figure 3.9 shows an example where the maximum of the interpolation lies between two samples.

[2]For example, the club Waldschänke Dornheim in Würzburg carried this issue very far: When visiting the club on 9th of July 2023, there was a party with just a single yellow bulb on the whole floor. It was slowly blinking and at least I didn't notice any connection to the music. Was this a high quality lightshow? Since the club was full and people enjoyed the evening, it seems it was in this situation.

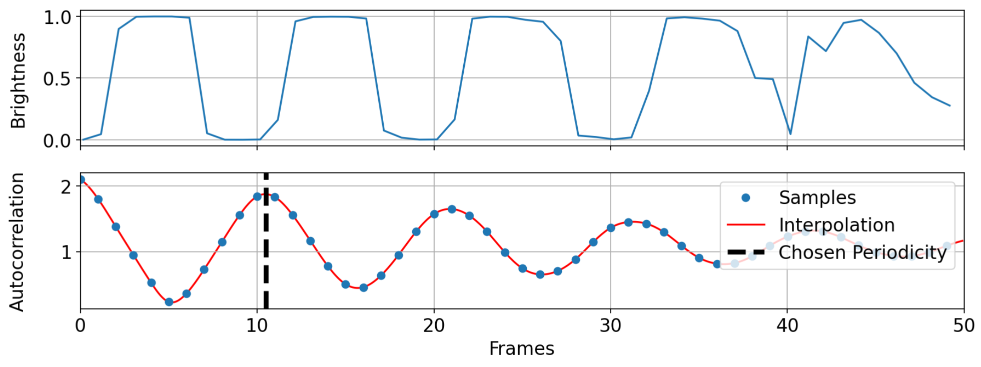


Figure 3.9: Extract of brightness signal and the corresponding auto-correlation curve

In addition to the periodicity, the observed tempo of the prediction

$$t_p = f_{video} \frac{60}{\arg\max_{R_{vv}}} \tag{3.4}$$

in beats per minute (bpm) depends also on the sampling rate f_{video} of the video.

The tempo is computed for each song in the validation data set and compared to the ground truth. The number of songs with correctly detected tempo is the metric. A tolerance of $\pm 3\,bpm$ allows small deviations.

Correlation Metric

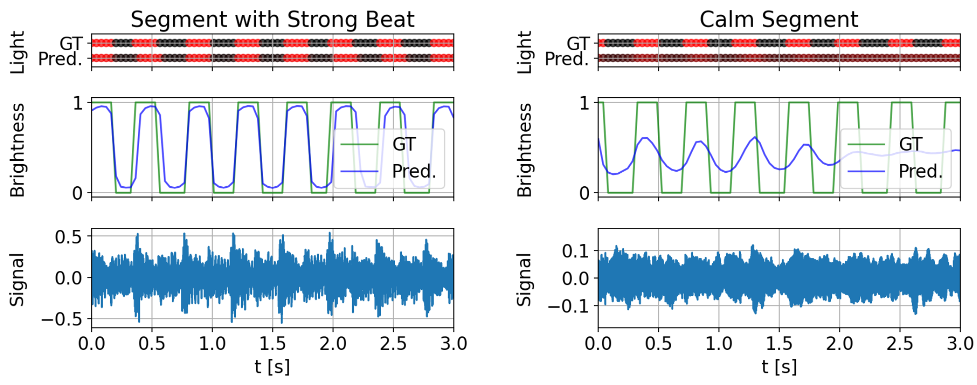
When comparing the brightness output of different models trained with Data Set 1, it becomes visible that the ability to reproduce the output shape of the ground truth varies. To objectify this observation, the zero-normalized cross-correlation (ZNCC)

$$R_{v_p v_t}(\tau) = \frac{1}{N} \sum_{k=0}^{N-1} \frac{1}{\sigma_p \sigma_t} (v_p(k) - \mu_p)(v_t(k+\tau) - \mu_t) \tag{3.5}$$

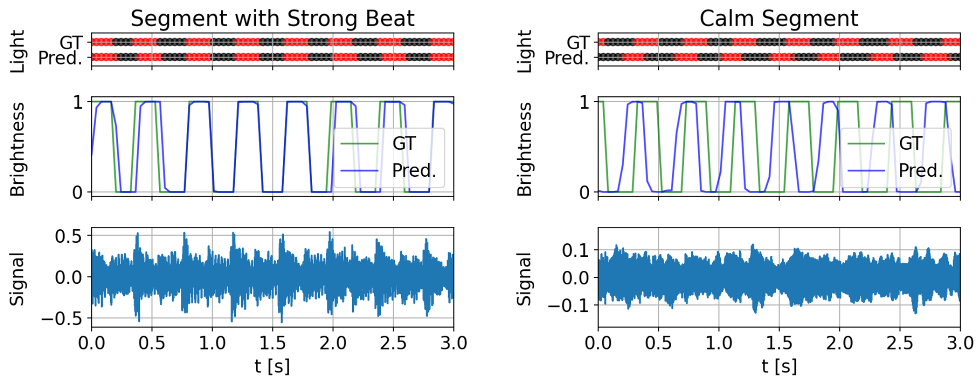
between the brightness signals of the prediction (v_p) and of the ground truth (v_t) is calculated where μ is the average of the corresponding signal and σ the standard deviation. The output of the ZNCC lies between -1 and 1 in which 1 expresses the maximum correlation. For every segment of $10\,\text{sec}$, the maximum correlation is retrieved and the average over all segments in the validation data embodies the metric.

Examples

The predictions of two example models illustrate how the metrics correlate with the actual output of the lights and support that way the interpretation of the results in



(a) Model with a standard deviation of 0.27 and a tempo metric of $50\ \%$



(b) Model with a standard deviation of 0.46 and a tempo metric of $40\ \%$

Figure 3.10: Example predictions (Pred.) of two models, together with the ground truth (GT), for visualizing the relation between output and metrics

chapter 4. The behavior is not homogeneous through the whole data set, but the two song segments from figure 3.2 cover a snippet with a very obvious beat and a calm passage in order to give a general impression.

While the first model from figure 3.10a achieves a tempo metric of $50\ \%$, which is 10 percent points above the score of the second one from figure 3.10b, it is the other way round for the standard deviation and the average difference: The output of the second model has a standard deviation of 0.46 which is $8\ \%$ below the ground truth in Data Set 1 (0.5). The average difference amounts to 0.19 (ground truth: 0.192). In comparison, the standard deviation (0.27) and the average difference (0.11) of the first model are reduced by $\approx 40\ \%$. For those two examples, the correlation metric equals each other (0.5).

3.6. Training Process

The implementation of the training process and the model itself use the Python library *PyTorch* [33]. As an optimizer, the *PyTorch* realization of the resilient back-propagation algorithm *Rprop* is chosen. Details of the optimiser can be found for example in the work of Mushgil et al. [30]. The training lasts 150 epochs with a learning rate of 1×10^{-3}.

3.6.1. Sequencing

As pointed out in section 2.1.1, the needed sequence length for the truncated back-propagation through time differs between applications. For analyzing its influence, the sequence length is also varied during the hyperparameter grid search. However, the model receives the whole input as a single sequence during the validation and test process. This represents the way the module is also employed during inference.

3.6.2. Split between Training, Validation and Test Data

Unlike typical machine learning tasks, the sequences of the data set cannot be divided randomly into training, validation and test data. The sequences which belong to the same song can share similarities. So assigning one of these sequences to the training data and one to the validation data would cause information leakage. As a countermeasure, the selection happens on song level so that all sequences of one song are assigned to either training, validation or test data.

The songs for the **Data Sets 1 and 2** divide up into three splits: Split a) and b) include 20 songs each and they are assigned to the training or validation data in a 2-fold cross validation process. Ten songs remain for the test data set. Table A.1 lists the allocation of the songs to the different splits while figure 3.11 shows that each split contains songs with varying tempo.

Data Set 3 is only used for training while the validation is done with the songs from split b) of the previous data sets. This enables a better comparison across the data sets.

3.6.3. Hyperparameter Grid Search

Due to the number of hyperparameters and the time constraints in this thesis, not every combination could be tested. Instead, an initial hyperparameter grid search varies four of the parameters where an influence on overfitting is assumed before a separate run varies the weight ratios in the loss function.

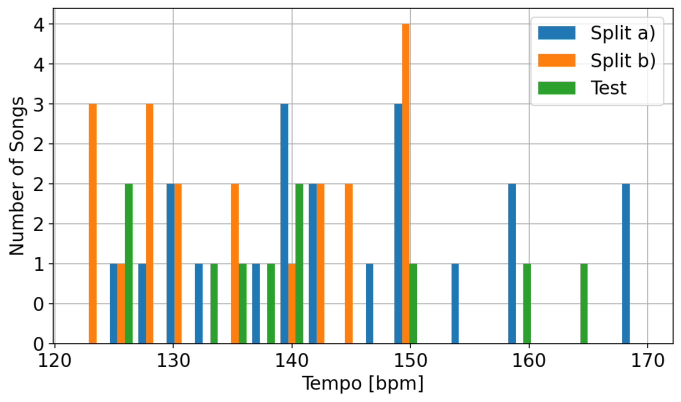


Figure 3.11: Comparison of the tempo distribution across the different splits

Basic Setup

On the one hand, the parameters hidden size and number of stacked layers determine the model's complexity so that a high influence on overfitting is likely. On the other hand, the dropout rate reduces overfitting and the sequence length affects the training process in general.

Each parameter is set to three different values while the weight ratio in the loss function is constantly balanced for this grid search:

Parameter	**Variations**		
Hidden Size	50	100	150
Layers	1	2	3
Dropout	0	0.1	0.3
Sequence Length	25	250	750

Table 3.1.: Variations during hyperparameter grid search for basic setup

Figure 3.12 displays the resulting variation in model complexity by reference to the number of trainable parameters.

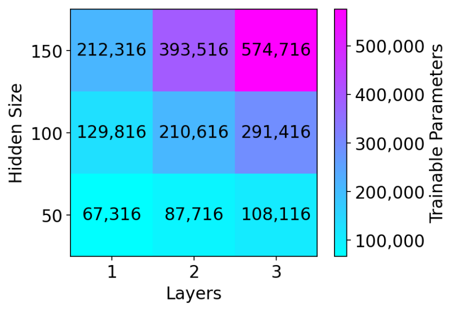


Figure 3.12: Number of trainable parameters depending on hidden size and number of stacked LSTM layers

Weighted Loss Function

After consolidating the first four parameters to the values that performed best during the previous grid search, different weights of the loss function shall reveal whether another ratio between the absolute and the relative component of the loss function improves the results. The following weight ratios are tried:

Parameter	**Variations**						
Weight Ratio	0/1	1/100	1/10	1/1	10/1	100/1	1/0

Table 3.2.: Weight ratios between absolute and relative component during second hyper-parameter variation

Thus, the search covers the extreme options of taking only the absolute or the relative component into account, as well as five combinations with different priorities.

Due to the constant hue in Data Set 1, the model predicts the correct value quickly so that it does not affect the loss afterwards. With Data Set 2 in contrast, the model adopts to two different outputs throughout the training process. Hence, the weight ratio between the brightness and the hue component of the loss function is altered in the same range as above.

3.6.4. Baseline

One potential flaw regarding the tempo metric is a model that learns the typical tempo of the training data and flashes the light with a constant frequency, independently from the currently played music. A simple comparison helps detecting such a behavior: The baseline model turns on the light with a constant frequency, either the mean or the average tempo of the songs from the training data. The number of songs where this behavior is accepted as correct depends on the tempo distribution in the validation data and the tolerance of the tempo metric described in section 3.5.2.

4. Results

4.1. Training Process

4.1.1. Development over the Epochs

An initial analysis of the training process with one combination of hyperparameters helps to interpret the further results. Choosing the median option of each parameter (hidden size=100, layers=2, dropout=0.1, sequence length=250, equal weights in loss function) for the test increases the chance that the results give an indication for the whole hyperparameter grid search.

Figure 4.1 shows two examples of repeated training processes with those parameters and training on Data Set 1. While the cross validation split a) represents the base for training, the validation utilizes split b). The first five epochs reduce the loss for the training and for the validation split in both repetitions equally. In a second phase of about 10 to 25 epochs, the loss stays constant, before the curves separate in the third phase. Here, the training loss decreases in both cases whereas it is almost stable or even increasing on the two validation splits. Run 1 leads to a lower training loss and causes the higher validation loss at the same time.

Although this is a sign for overfitting, the correlation metric and the standard deviation also improve for the validation split in this phase before they reach a plateau.

4.1.2. Reproducibility

The two exemplary training processes point to limited reproducibility of the metrics. Extending the analysis to 60 repetitions provides further insights into the expressiveness of the metrics. For these 60 runs, the 2-fold cross validation is used to visualize dependencies on the splits of the data set, too. Figure 4.2 shows the distribution of the four metrics for the validation data. Compared to the variation within models trained on split a) or b), the differences between the two groups are rather small.

However, the number of outliers creates a challenge for further comparisons. A possible approach for a practical application would be to train several models with the same parameters and use the one that embodies the best compromise between the metrics. Since understanding the general influence of design decisions on the behavior of the model is of higher interest in this thesis, the data is smoothed with

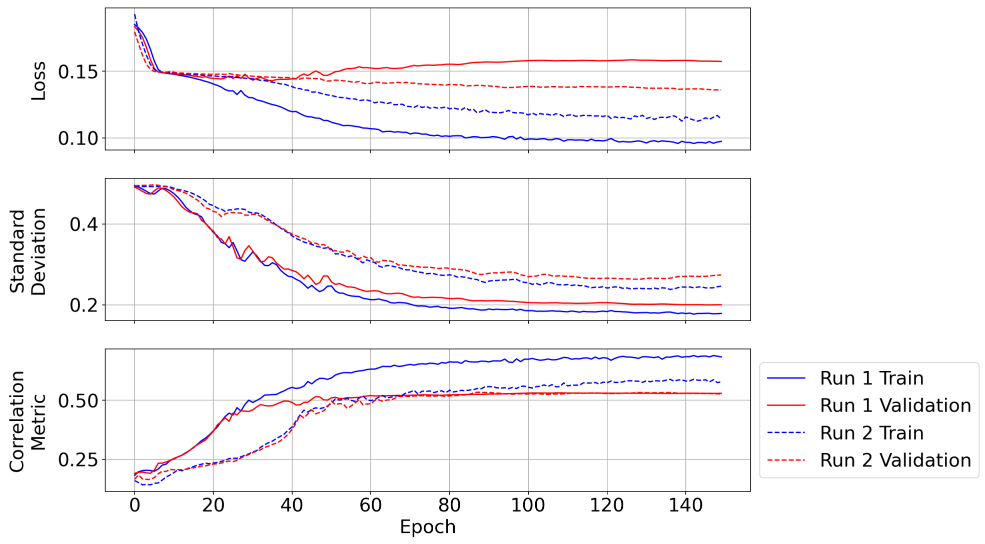


Figure 4.1: Development of loss, standard deviation and correlation metric for two example training processes with the same hyperparameters

the median for the following evaluations: After six training processes (three on each split), the median value of these six runs represents the final metric.

The smoothed metrics still contain a certain variation. This is simulated by grouping the 60 runs from before into 10 chunks of six values each. The figure 4.2 also shows the distribution of the median values of each chunk.

The smoothing process halves the variation approximately: While the relative standard deviation varies between $4.8\,\%$ (correlation metric, trained on split a) and $21\,\%$ (average difference, trained on split b) in the raw data, this range is reduced to $2.3\,\%$ and $10.8\,\%$. A further reduction of the noise in the data by more runs would be beneficial but increases the execution time. Therefore, the median over six runs is kept as a compromise and the remaining uncertainty has to be considered when interpreting the results.

4.1.3. Execution Time

The execution time for a single training process out of the basic hyperparameter grid search fluctuates between 50 s and 269 s on an *Intel Core i7-13700* processor. Figure 4.3 compares the influence of the hyperparameters on the time needed for the training process.

The dropout parameter has only minimal impact on the needed time. Concerning the sequence length, the optimum within this grid search lies at 250 steps. The hidden size and the number of layers influence the duration in the same way: As

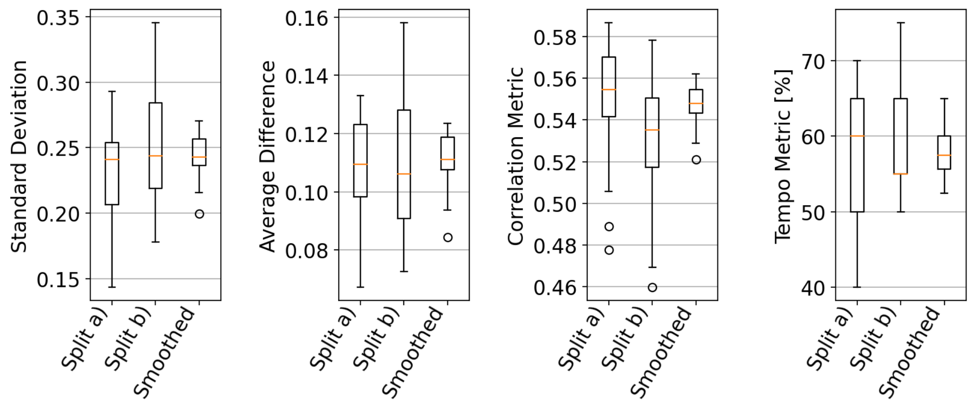


Figure 4.2: Boxplot diagrams of the four metrics with 60 runs trained either on split a) or b) in comparison to the distribution of the smoothed data

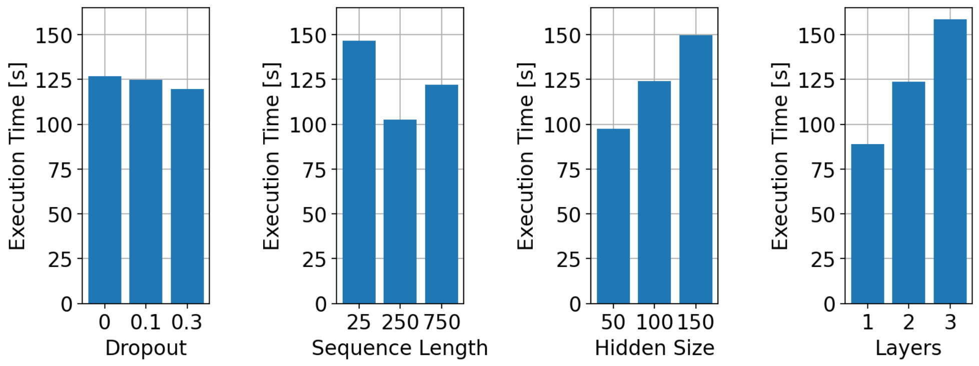


Figure 4.3: Average execution time for the different hyperparameters

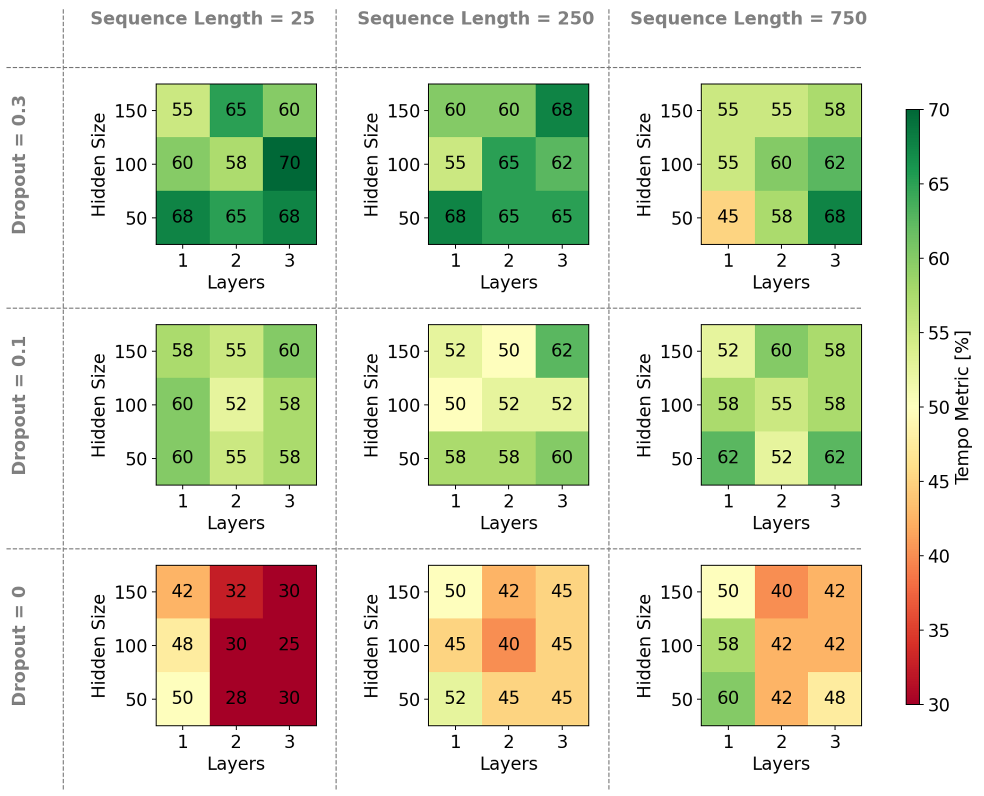
expected, an increased number of trainable parameters also extends the execution time.

4.2. Hyperparameter Grid Search

4.2.1. Basic Setup

Tempo Metric

Figure 4.4 depicts how the four different hyperparameters impact the tempo metric. In general, dropout improves this value. A dropout of 0.3 with three layers, a hidden size of 100 and 25 as sequence length generates the highest score. The model



Dropout	Hidden Size	SL 25: L1	SL 25: L2	SL 25: L3	SL 250: L1	SL 250: L2	SL 250: L3	SL 750: L1	SL 750: L2	SL 750: L3
0.3	150	55	65	60	60	60	68	55	55	58
0.3	100	60	58	70	55	65	62	55	60	62
0.3	50	68	65	68	68	65	65	45	58	68
0.1	150	58	55	60	52	50	62	52	60	58
0.1	100	60	52	58	50	52	52	58	55	58
0.1	50	60	55	58	58	58	60	62	52	62
0	150	42	32	30	50	42	45	50	40	42
0	100	48	30	25	45	40	45	58	42	42
0	50	50	28	30	52	45	45	60	42	48

Figure 4.4: Influence of the four hyperparameters on the tempo metric

detects the tempo of 14 out of the 20 songs correctly which leads to a tempo metric of $70\,\%$.

The influence of higher model complexity on this metric is limited. For example, with a dropout of 0.1 and a sequence length of 25 steps, the metric varies only between $52\,\%$ and $60\,\%$ for the different model configurations. Assuming the underlying distribution is the same as during the reproducibility test in section 4.1.2, the results are within 2.4 standard deviations. There is also no unique picture which of the model configurations inside each of the subplots in figure 4.4 is the best.

However, a clear influence of the model architecture is visible when training the model without dropout. In this case, a single LSTM layer (average tempo metric: $51\,\%$) performs better than two or three with an average of $39\,\%$.

The simple **baseline model** that outputs light pulses with the frequency of the median tempo of the training data achieves between $25\,\%$ when trained on split a) and validated on split b) and $5\,\%$ the other way round. If the median is replaced with the average value, the metric decreases to 10 - $5\,\%$.

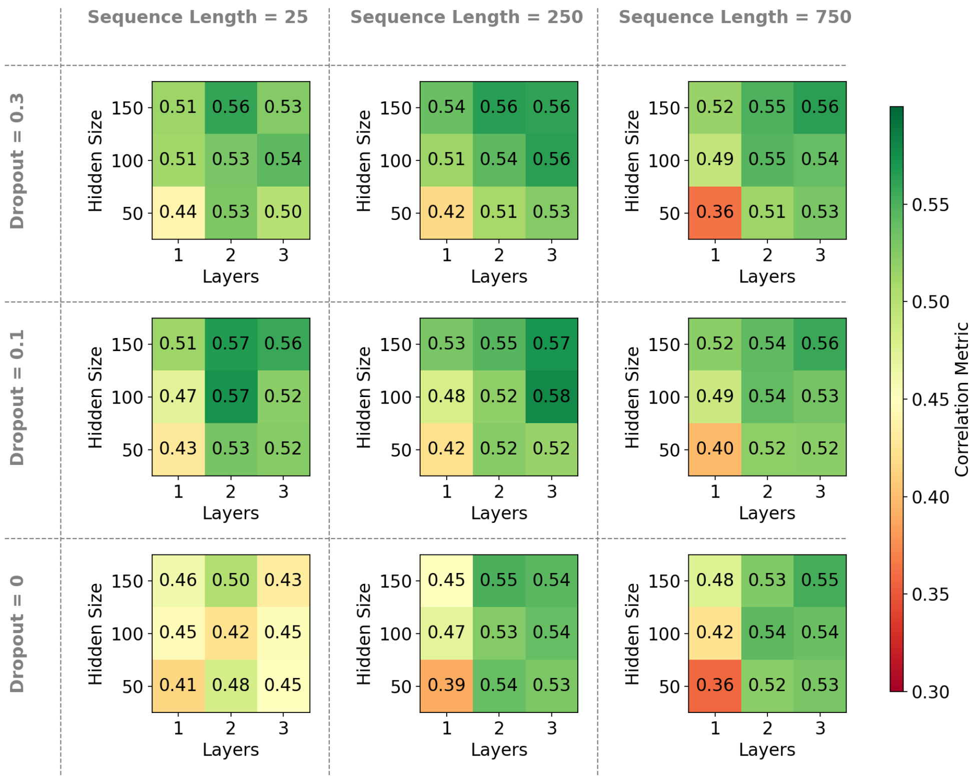


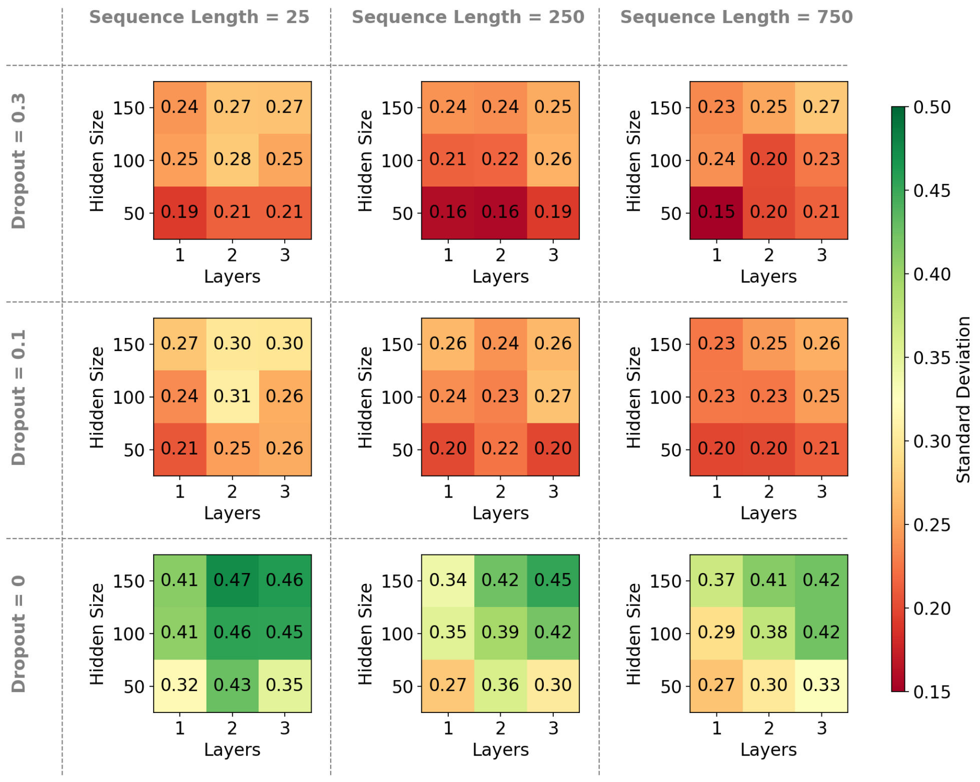
Figure 4.5: Influence of the four hyperparameters on the correlation metric

Correlation Metric

In contrast, the correlation metric in figure 4.5 benefits from a higher model complexity. While the score of models with a hidden size of 50 and a single LSTM layer varies between 0.36 and 0.44, the maximum for a dropout/sequence length combination is always achieved with two or three layers and a hidden size of at least 100. With such a higher model complexity, the maximum score ranges from 0.5 to 0.58.

Standard Deviation

The standard deviation predominantly improves with more complex models, too. For example, the average score is 0.24 when the hidden size is set to 50, whereas the average of all models with a hidden size of 100 or 150 is 0.3. In addition, the dropout rate has a high influence on the metric and causes a target conflict with the tempo metric: Without dropout, the average is 0.38 and the best value 0.47



Dropout	Seq. Length	Hidden Size	Layers 1	Layers 2	Layers 3
0.3	25	150	0.24	0.27	0.27
0.3	25	100	0.25	0.28	0.25
0.3	25	50	0.19	0.21	0.21
0.3	250	150	0.24	0.24	0.25
0.3	250	100	0.21	0.22	0.26
0.3	250	50	0.16	0.16	0.19
0.3	750	150	0.23	0.25	0.27
0.3	750	100	0.24	0.20	0.23
0.3	750	50	0.15	0.20	0.21
0.1	25	150	0.27	0.30	0.30
0.1	25	100	0.24	0.31	0.26
0.1	25	50	0.21	0.25	0.26
0.1	250	150	0.26	0.24	0.26
0.1	250	100	0.24	0.23	0.27
0.1	250	50	0.20	0.22	0.20
0.1	750	150	0.23	0.25	0.26
0.1	750	100	0.23	0.23	0.25
0.1	750	50	0.20	0.20	0.21
0	25	150	0.41	0.47	0.46
0	25	100	0.41	0.46	0.45
0	25	50	0.32	0.43	0.35
0	250	150	0.34	0.42	0.45
0	250	100	0.35	0.39	0.42
0	250	50	0.27	0.36	0.30
0	750	150	0.37	0.41	0.42
0	750	100	0.29	0.38	0.42
0	750	50	0.27	0.30	0.33

Figure 4.6: Influence of the four hyperparameters on the standard deviation

which equals almost the standard deviation of the ground truth (0.5). The average decreases for the tries with dropout to 0.23 and also the highest value of the group with dropout (0.31) is only slightly better than the worst result without dropout (0.27).

To put it into perspective: The orange cells in figure 4.6 that contain values less than 0.25 correspond to models that produce less than half of the standard deviation that is expected.

Correlation between Metrics

A separate figure for the average difference is omitted because this metric correlates strongly with the standard deviation, as visualized in figure 4.7. So this metric provides no information gain at least for this model architecture trained on Data Set 1. The figure also shows the correlation between the standard deviation and the tempo metric. It confirms the trade-off between the two metrics observed before.

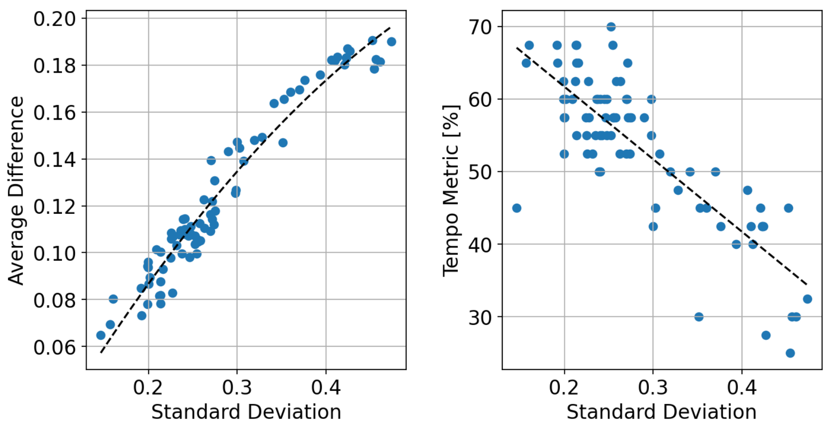


Figure 4.7: Correlation between the standard deviation and the average difference (left subplot), respectively the tempo metric (right subplot)

4.2.2. Weighted Loss Function

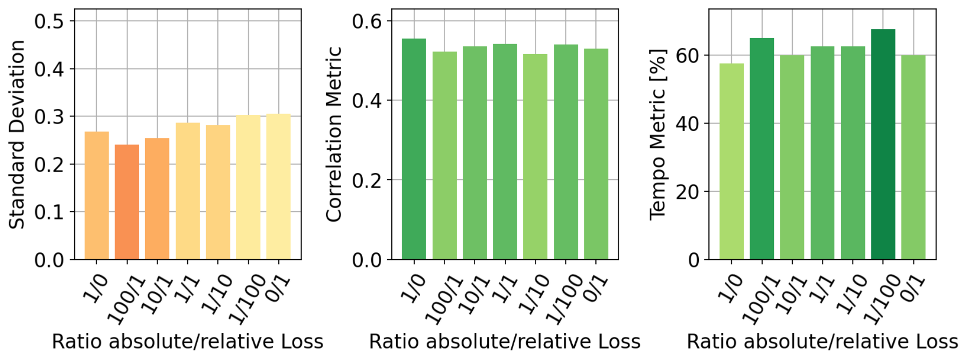
The goal of varied weights in the loss function is to find out, whether different weights are capable of solving the target conflict observed in section 4.2.1. Therefore, it is carried out twice: Once with parameters that led previously to a good score in the tempo metric and once for an example with a high value for the standard deviation. The two combinations that embody the starting points for this second hyperparameter search are listed in table 4.1.

Parameters	**Combination a): High Tempo Metric**	**Combination b): High Standard Deviation**
Hidden Size	150	150
Layers	3	3
Sequence Length	25	250
Dropout	0.3	0

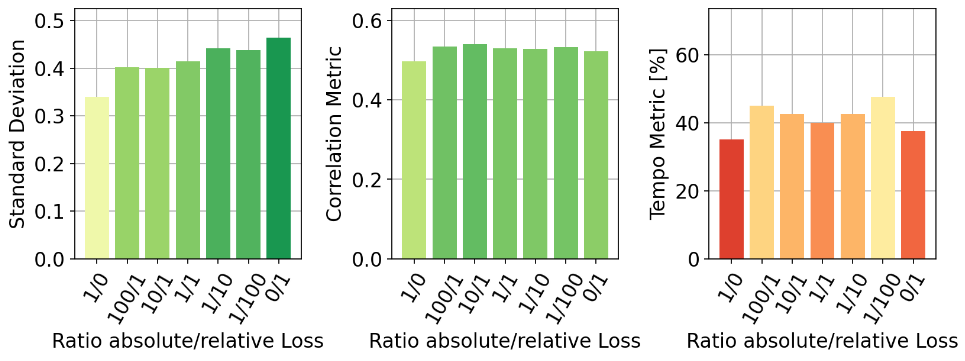
Table 4.1.: Starting points for second hyperparameter search

The results in figures 4.8 reveal little influence of the weight ratio in the loss function. Only the standard deviation improves with higher weights for the relative component of the loss function. A loss function that evaluates only the change of the brightness between the samples achieves the best score in both configurations. The higher importance of this improvement holds for the configuration in figure 4.8a which had a low standard deviation in the grid search. However, the effect is limited in this case, adding only a margin of 0.02 when comparing the balanced weight function and the one with only relative loss.

Although the tempo metric shows lower values for the loss functions with only



(a) Starting point: High tempo metric



(b) Starting point: High standard deviation

Figure 4.8: Influence of the weight ratio between the absolute and relative components in the loss function on the three metrics

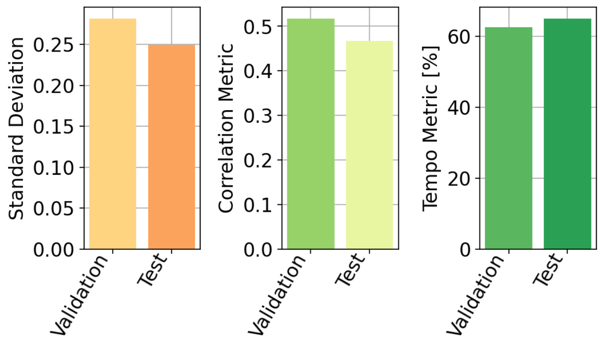


Figure 4.9: Comparison between validation and test results

a relative component compared to the others, there is no clear influence on the tempo nor on the correlation metric visible.

4.2.3. Test Results

The analysis in section 4.2.2 depicts that a different weight ratio in the loss function is not able to resolve the target conflict between the metrics observed in basic hyperparameter grid search in section 4.2.1. Nevertheless, a higher prioritization of the relative component in the loss function turned out to be slightly beneficial. Therefore, the decision is to continue with ten times higher weights on the relative loss component. The combination a) from table 4.1 is selected for the other hyperparameters because it is more promising for future developments to add a post processing step that improves the standard deviation instead of the tempo metric.

For a better comparability with the validation results, the test data set is employed on all six models trained in section 4.2.2 with these parameters. The metric consists again of the median value of the six single results.

Figure 4.9 displays that the performance on the test data set does not meet the results on the validation data for the standard deviation ($-13\,\%$) and the correlation metric ($-11\,\%$). The tempo metric stays almost constant ($+4\,\%$).

4.3. Evaluation with Data Set 2

Besides a different signal form for the brightness (saw tooth instead of rectangular), Data Set 2 also brings in color changes.

When trying the model configuration from 4.2.3 with the same parameters on Data Set 2, the training process fails. The training loss is only reduced by $1.5\,\%$ and the output is almost constant. In this data set, the standard deviation of the color

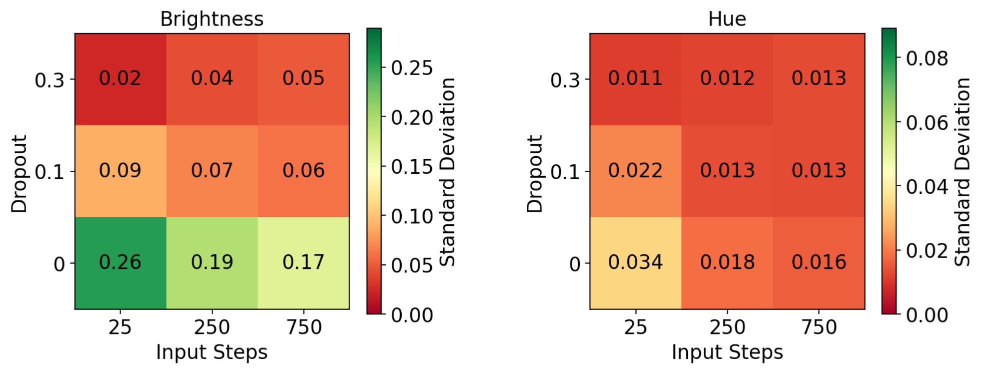


Figure 4.10: The effect of dropout and the sequence length on the standard deviation of brightness and hue

is 7.7 times less during validation than in the ground truth and for the brightness, the factor is 3.7.

4.3.1. Basic Setup

A repetition of parts of the hyperparameter grid search clarifies whether different parameters solve the issue. Since the models with higher complexity were beneficial in most cases for Data Set 1 and especially for the standard deviation, the hidden size stays 150 with three LSTM layers throughout this search.

The color scheme of figure 4.10 is adapted to the ground truth of Data Set 2 (0.29 for brightness and 0.09 for hue). The results for the hue distribution are low for all models during this grid search. For the models trained with dropout, this also applies to the brightness. Without dropout, the metric improves with a declining sequence length. The best score is achieved with a sequence length of 25. However, the previous results from Data Set 1 in section 4.2.1 revealed that the combination of high model complexity, no dropout and a sequence length of 25 tends to detect an incorrect beat. Therefore, the sequence length is set to 250 for an additional variation of the weights in the loss function.

4.3.2. Weighted Loss Function

Since the model also needs to learn changes of the hue in Data Set 2, the weight ratio between hue and brightness in the loss function are altered for this data set. As a result, figure 4.11 exposes a cross effect between brightness and hue: The standard deviation of the hue is above 0.06 if the hue component in the loss function is prioritized by the factor 10 or more. At the same time, these conditions lead to a low standard deviation for the brightness.

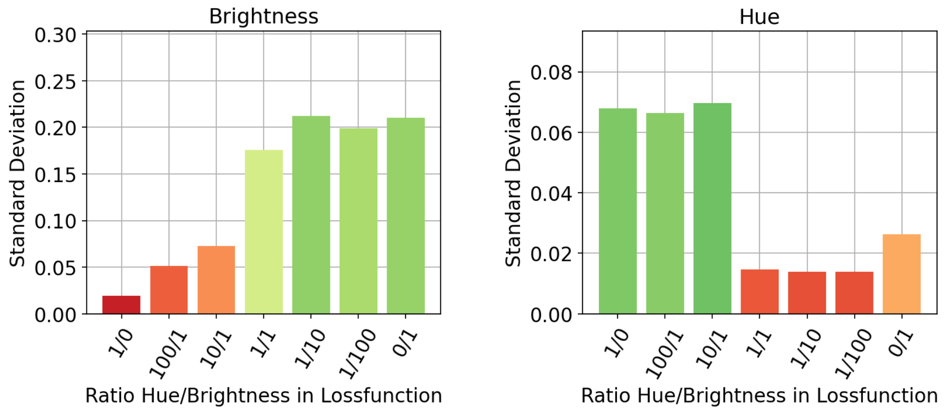


Figure 4.11: Influence of the weight ratio between the hue and brightness components in the loss function

The approach to overcome this dilemma is to train two models individually: The first with a loss function that only takes the brightness into account and the second the other way round. During inference, both variants are combined again.

The example output in figure 4.12 shows how the combined solution behaves. The brightness curve forms the sawtooth profile and also the hue changes between blue, turquoise and purple. In case of the segment with the strong beat, the temporal sequence of the different colors is roughly matching the ground truth but the brightness does not employ the full value range. Moreover, the frequency of the changes does not fit to the ground truth.

The distribution of the combined model's output is similar across the data sets: The standard deviation for brightness and hue varies only by $2\,\%$ between the validation and test data sets.

4.4. Evaluation with Data Set 3

The target of the following section is to give a first impression on how the approach with two independent models for brightness and hue behaves when the training takes place on Data Set 3.

When using Data Set 3 as it is, the model predicts almost constant values. In this case, one reason is that there is little variance in the lights for some songs of the data set. Excluding those songs from the data set enhanced the behavior.

A closer comparison of the two variations of the training data in figure 4.13 exhibits that the distribution of hue and brightness is still similar for both variants. Therefore, the standard deviation of the brightness is almost equal (0.26 vs. 0.27)

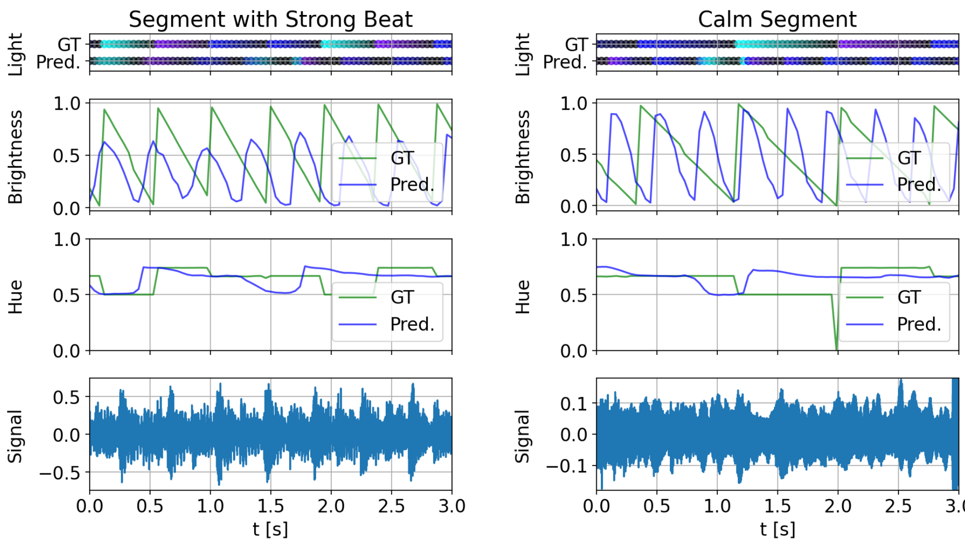


Figure 4.12: Example output for Data Set 2 with two models that predict either brightness or hue independently from each other

and the one for the hue is only slightly increased from 0.26 to 0.32. However, the average difference increases by a factor of 1.8 (brightness), respectively 3 (hue) when using the shortened list.

Figure 4.14 displays the example output for a model combination that is trained on the shortened Data Set 3. The relationship to the music is difficult to distinguish and requires further research. In addition, the brightness equals zero for about $1.5\,\mathrm{s}$ at the beginning of the calm segment.

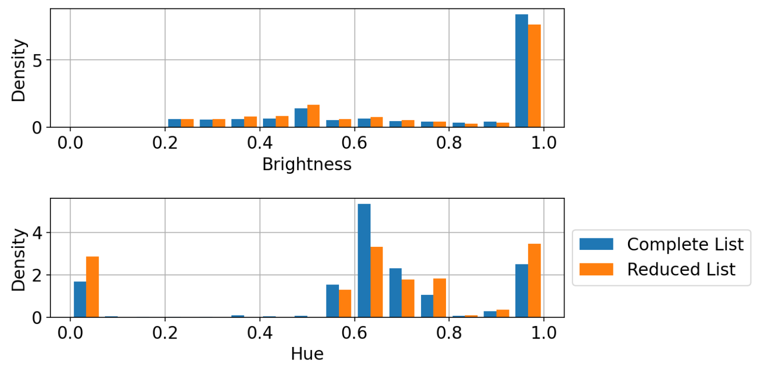


Figure 4.13: Distribution of brightness and hue in the ground truth of the complete Data Set 3 and the shortened variant

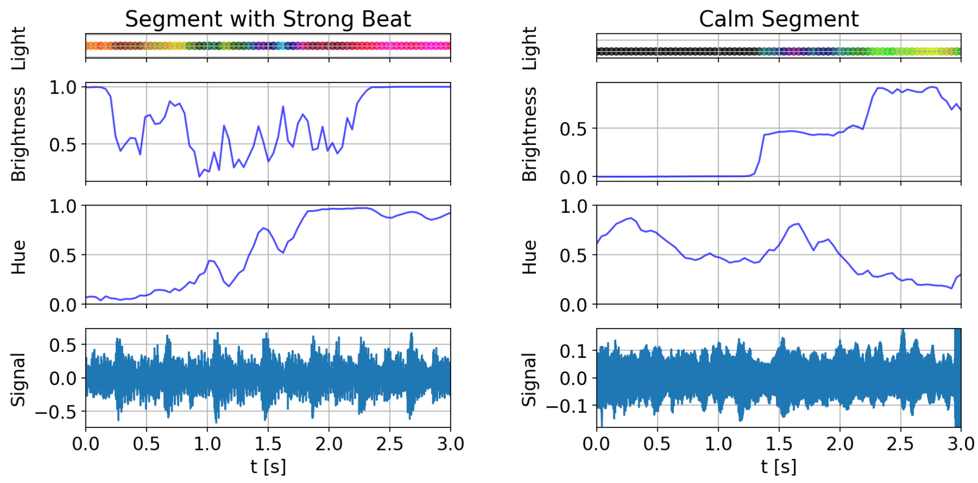


Figure 4.14: Example output for two models trained on Data Set 3 that predict either brightness or hue independently from each other

5. Discussion

The results showed that deep learning proposals intended for beat detection are a promising approach for sound-to-light automation, too. By learning a direct dependency between low level music features and the lighting output, an intermediate estimation of emotions is not necessary. This direct prediction offers the potential of a closer relationship between dynamic changes in the audio input and the lights.

Nevertheless, when only focusing on the current output, a light show with a similar quality could probably be retrieved using classic algorithms with a fraction of the effort. The analysis revealed several limitations that need to be addressed before it can compete with the light shows people are used to on small events these days.

5.1. Research Question 1 – Data Sets

Three different data sets were created during this thesis and all gave new insights concerning sound-to-light automation in a certain phase of the project. Especially the stepwise evolution of the model due to the different data sets turned out as an advantage. However, the experiences varied across data sets.

5.1.1. Data Set 1

The simple structure of Data Set 1 was predestined for debugging purposes at the beginning of the project. The deterministic behavior allowed clear expectations of the predictions and facilitated that way the detection of programming errors. Reducing the complexity with the choice of brightness as the only relevant output turned out as an advantage for that, too. Multiple issues can lead to situations where solving one problem does not improve the overall performance. Setting up the infrastructure would have been more demanding with the additional side-effects between hue and brightness observed with Data Set 2.

So for fundamental changes of the model, the first tests should be carried out with this data set again.

5.1.2. Data Set 2

Data Set 2 was an appropriate second step after the general setup with Data Set 1 and demonstrated that predicting hue and brightness at the same time, is an additional challenge. Until now, the work with this data set was limited to increasing the standard deviation as a first necessary step. For further improvement of the model, optimizing the other metrics on this data set is expected to be the most important.

5.1.3. Data Set 3

Although still not suitable for practical application, the models trained on Data Set 3 made the impression to be closer to this target than the models trained on the other data sets. This is mainly due to the variety of colors in Data Set 2 compared to the other two.

It turned out that only a fraction of light show videos is valuable for the parsing approach proposed in this thesis. The method of parsing the input does not detect when two adjacent fixtures within one segment pulse alternating with the same color. It returns a constant output instead. The same holds for the beam of a moving head that oscillates with the beat of the music but stays in the same segment. That way, information necessary for a close correlation between sound and light gets lost.

Removing songs with a very constant light output improved the behavior, but it is rather a workaround. A better approach would be to create traces of a professional light show with the DMX512 logger directly.

5.2. Research Question 2 – Model

The chosen CRNN architecture was capable of producing output with a good result either for the tempo metric or for the standard deviation, mainly depending on the dropout rate. Solving this conflict should be the first step for further research. A potential approach for an increased standard deviation would be a different activation function or a postprocessing step that amplifies the changes of the output. Regarding the other hyperparameters, the model showed a certain robustness as long as the complexity was sufficiently high (two or three layers with a hidden size of 100 or 150). There is not a specific choice that works while a slight modification brings the learning process to fail.

Data Set 2 highlighted the difficulties of predicting two separate outputs with the different temporal behavior: The brightness pattern repeated at each beat, whereas a complete cycle for hue lasted four beat events. The solution with training two models individually leads to the disadvantage of models not interacting with each other: Effects like a change in color for every light pulse will not be synchronized when hue and brightness are predicted by independent models.

All in all, it was already more complicated than expected to make a CRNN output acceptable distributions. It is worth a try to increase the training data and the model's complexity by an order of magnitude for example. Especially the decision for the structure of the convolutional layer was based on previous research in the beat detecting field. Since the hyperparameter grid search of this thesis focused on the recurrent layer, a potential optimization is to analyze the effect of more filters or a deeper structure of the convolutional layer in order to improve the interpretation of the music features.

Depending on the results, other deep learning architectures for time series forecasting, such as temporal fusion transformers, should be taken into account as an alternative to the stacked LSTMs used in this thesis.

5.3. Research Question 3 – Loss Function and Metrics

The **loss function** made the model learn the general structure of Data Set 1. However, a loss function where the validation loss increases, although other metrics display that the quality of the validation predictions is still improving, does not fulfill a typical expectation within machine learning. One idea for further development would be to add parts of the performance metrics in the loss function additionally.

Those four **performance metrics** fulfilled their task: The behavior observed during example runs matched the messages of the metrics (e.g. figures 3.10a and 3.10b). Therefore, the subjective impression became measurable in an objective form and allowed the comparison of many hyperparameters combination without watching and grading hours of light shows. During this thesis, the tempo metric and the standard deviation provided more insights than the average difference or the correlation metric.

As soon as the model improves so that the proposed metrics reach a state of saturation, **new metrics** would be needed for quantifying changes in the quality. This could be for example the color distribution during a song segment. If a light show alternates between two different colors, it would be good if a change to the next two colors is aligned to the boundary of the song segment. However, such a metric is only helpful if the predictions step up a level.

Bibliography

[1] S. Albawi, T. A. Mohammed, and S. Al-Zawi. “Understanding of a convolutional neural network”. In: *2017 International Conference on Engineering and Technology (ICET)*. 2017, pp. 1–6.

[2] F. Alias, J. C. Socoró, and X. Sevillano. “A review of physical and perceptual feature extraction techniques for speech, music and environmental sounds”. In: *Applied Sciences* 6(5) (2016), p. 143.

[3] S. Böck and M. Schedl. “Enhanced Beat Tracking with Context-Aware Neural Networks”. In: *Proceedings of the 14th International Conference on Digital Audio Effects (DAFx-11)*. 2011.

[4] R. Chen, M. Wang, and Y. Lai. “Analysis of the role and robustness of artificial intelligence in commodity image recognition under deep learning neural network”. In: *PLoS ONE* 15(7) (2020).

[5] T. Cheng, S. Fukayama, and M. Goto. “Joint Beat and Downbeat Tracking Based on CRNN Models and a Comparison of Using Different Context Ranges in Convolutional Layers”. In: *Proceedings of the International Computer Music Conference (ICMC)* (2021).

[6] *Desktop lighting control software for creating and performing lightshows*. inMusic Brands. 2023. URL: `https://www.soundswitch.com/software` (visited on 10/30/2023).

[7] D. P. Ellis. “Beat Tracking by Dynamic Programming”. In: *Journal of New Music Research* 36(1) (2007), pp. 51–60.

[8] *eurolite DMX LED EASY Operator Deluxe - User Manual*. Steinigke Showtechnic. 2018. URL: `https://www.steinigke.de/download/70064574-Manual-116621-1.000-eurolite-dmx-led-easy-operator-deluxe-de_en.pdf` (visited on 10/30/2023).

[9] *grandMA3 User Manual*. MA Lighting International. 2023. URL: `https://help2.malighting.com/Page/grandMA3/grandMA3/en/1.9` (visited on 10/30/2023).

[10] D. J. Hermes. *The Perceptual Structure of Sound*. Springer Nature, 2023.

[11] H. Hewamalage, C. Bergmeir, and K. Bandara. “Recurrent Neural Networks for Time Series Forecasting: Current status and future directions”. In: *International Journal of Forecasting* 37(1) (2021), pp. 388–427.

[12] M. Heydari, F. Cwitkowitz, and Z. Duan. “BeatNet: CRNN and Particle Filtering for Online Joint Beat Downbeat and Meter Tracking”. In: *International Society for Music Information Retrieval* (2021).

[13] S. Hizlisoy, S. Yildirim, and Z. Tufekci. “Music emotion recognition using convolutional long short term memory deep neural networks”. In: *Engineering Science and Technology, an International Journal* 24(3) (2021), pp. 760–767.

[14] S. Hochreiter and J. Schmidhuber. “Long Short-Term Memory”. In: *Neural Computation* 9(8) (1997), pp. 1735–1780.

[15] S.-W. Hsiao, S.-K. Chen, and C.-H. Lee. “Methodology for stage lighting control based on music emotions”. In: *Information Sciences* 412-413 (2017), pp. 14–35.

[16] S. Hwangbo, S.-Y. Chun, S.-Y. Gang, and C.-S. Lee. “Lighting Control using Frequency Analysis of Music”. In: *Journal of Korea Multimedia Society* 16 (2013).

[17] R. Isermann and M. Münchhof. *Identification of Dynamic Systems*. Springer Berlin, Heidelberg, 2010.

[18] A. Khodabakhsh, I. Ari, M. Bakır, and S. M. Alagoz. “Forecasting multivariate time-series data using LSTM and mini-batches”. In: *Data Science: From Research to Application*. Springer. 2020, pp. 121–129.

[19] K. Kroening. *ffmpeg-python*. URL: `https://github.com/kkroening/ffmpeg-python` (visited on 10/30/2023).

[20] C.-C. J. Kuo. “Understanding convolutional neural networks with a mathematical model”. In: *Journal of Visual Communication and Image Representation* 41 (2016), pp. 406–413.

[21] Z. Li, F. Liu, W. Yang, S. Peng, and J. Zhou. “A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects”. In: *IEEE Transactions on Neural Networks and Learning Systems* 33(12) (2022), pp. 6999–7019.

[22] R. Liao, Y. Xiong, E. Fetaya, L. Zhang, K. Yoon, X. Pitkow, R. Urtasun, and R. Zemel. “Reviving and Improving Recurrent Back-Propagation”. In: *Proceedings of the 35th International Conference on Machine Learning*. Ed. by J. Dy and A. Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, 2018, pp. 3082–3091.

[23] M. Loesdau, S. Chabrier, and A. Gabillon. “Hue and Saturation in the RGB Color Space”. In: *Image and Signal Processing*. Springer International Publishing, 2014, pp. 203–212.

[24] N. Loi, KimDonglim, and LimYounghwan. “Emotion-based music visualization using LED lighting control system”. In: *Journal of Korea Game Society* 17(3) (June 2017), pp. 45–52.

[25] J. T. Love, M. Singh Benning, A. Elliot, J. Leben, and G. Odowichuk. *System and method for predictive generation of visual sequences*. LIMBIC MEDIA CORPORATION, U.S. Patent 10,319,395 B2, 2019.

[26] E. P. MatthewDavies and S. Böck. “Temporal convolutional networks for musical audio beat tracking”. In: *2019 27th European Signal Processing Conference (EUSIPCO)*. 2019, pp. 1–5.

[27] B. McFee, C. Raffel, D. Liang, D. P. Ellis, M. McVicar, E. Battenberg, and O. Nieto. “librosa: Audio and music signal analysis in python”. In: *Proceedings of the 14th python in science conference*. Vol. 8. 2015.

[28] A. Meghanani, A. C. S., and A. G. Ramakrishnan. “An Exploration of Log-Mel Spectrogram and MFCC Features for Alzheimer’s Dementia Recognition from Spontaneous Speech”. In: *2021 IEEE Spoken Language Technology Workshop (SLT)*. 2021, pp. 670–677.

[29] T. C. Mills. *Time series techniques for economists*. Cambridge University Press, 1990.

[30] H. M. Mushgil, H. A. Alani, and L. E. George. “Comparison between resilient and standard back propagation algorithms efficiency in pattern recognition”. In: *International Journal of Scientific & Engineering Research* 6(3) (2015), pp. 773–778.

[31] S. Newton. “Art-Net and Wireless Routers”. In: *Asia-Pacific Conference on Communications*. 2005, pp. 857–861.

[32] F. Nouvel, S. Aing, T. Beyou, T. Chea, P. Cauchois-Guiheneuf, F. Millet, and T. Thomas. “Wireless music controlled leds: MUSIC COLORZ”. In: *4th European Education and Research Conference (EDERC 2010)*. 2010, pp. 106–110.

[33] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. “PyTorch: An Imperative Style, High-Performance Deep Learning Library”. In: *Advances in Neural Information Processing Systems 32*. 2019, pp. 8024–8035.

[34] N. Patil, R. M. Yadahalli, and J. Pujari. “Comparison between HSV and YCbCr Color Model Color-Texture based Classification of the Food Grains”. In: *International Journal of Computers and Applications* 34 (2011).

[35] K. J. Peacock. “Instruments to Perform Color-Music: Two Centuries of Technological Experimentation”. In: *Leonardo* 21 (2017), pp. 397–406.

[36] Y. Ri. *PyDMX*. URL: `https://github.com/YoshiRi/PyDMX` (visited on 10/30/2023).

[37] A. R. Smith. “Color Gamut Transform Pairs”. In: *SIGGRAPH Comput. Graph.* 12(3) (1978), pp. 12–19.

[38] R. C. Staudemeyer and E. R. Morris. “Understanding LSTM – a tutorial into long short-term memory recurrent neural networks”. In: *arXiv preprint arXiv:1909.09586* (2019).

[39] S. S. Stevens, J. Volkmann, and E. B. Newman. “A scale for the measurement of the psychological magnitude pitch”. In: *The journal of the acoustical society of america* 8(3) (1937), pp. 185–190.

[40] R. E. Thayer. *The Biopsychology of Mood and Arousal*. Oxford University Press USA, 1989.

[41] R. Vogl, M. Dorfer, G. Widmer, and P. Knees. “Drum Transcription via Joint Beat and Drum Modeling Using Convolutional Recurrent Neural Networks”. In: *Proceedings of the 18th International Society for Music Information Retrieval Conference*. Ed. by S. J. Cunningham, Z. Duan, X. Hu, and D. Turnbull. 2017, pp. 150–157.

[42] Q. Wang, Y. Ma, K. Zhao, and Y. Tian. “A comprehensive survey of loss functions in machine learning”. In: *Annals of Data Science* (2020), pp. 1–26.

[43] Z. J. Wang, R. Turko, O. Shaikh, H. Park, N. Das, F. Hohman, M. Kahng, and D. H. Polo Chau. “CNN Explainer: Learning Convolutional Neural Networks with Interactive Visualization”. In: *IEEE Transactions on Visualization and Computer Graphics* 27(2) (2021), pp. 1396–1406.

[44] R. J. Williams and J. Peng. “An Efficient Gradient-Based Algorithm for On-Line Training of Recurrent Network Trajectories”. In: *Neural Computation* 2(4) (1990), pp. 490–501.

[45] Y. Xu, Q. Kong, W. Wang, and M. D. Plumbley. “Large-Scale Weakly Supervised Audio Classification Using Gated Convolutional Neural Network”. In: *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2018, pp. 121–125.

[46] h. Yang, Y.-F. Su, Y.-C. Lin, and H. Chen. “Music emotion recognition: The role of individuality”. In: *Proceedings of the ACM International Multimedia Conference and Exhibition* (2007), pp. 13–22.

[47] A. Zhang, Z. C. Lipton, M. Li, and A. J. Smola. *Dive into Deep Learning*. Cambridge University Press, 2023.

List of Figures

A. List of Songs in Data Sets

A.1. Data Sets 1 and 2

Data Sets 1 and 2 consist of the *Spotify* playlist *Mainstage*.

The database *Tunebat*[1] provided the information about the tempo. It states a tempo below $80\,\mathrm{bmp}$ for three of the songs. These values are doubled because the songs do not make the impression that they are significantly slower than the others and the software *Virtual DJ* states the doubled tempo, too.

Split	Song	Interpret	Tempo [bpm]
	Vois sur ton chemin - Techno Mix	BENNETT	138
	Boyz In Paris (with VINAI)	Marnik, Naeleck, VINAI	149[2]
	Move It	POLTERGST	160
	Say It Right	Macon, Enny-Mae	143
	Prada	cassö, RAYE, E-Block Europe	142
	SKY	AVA CROWN, Öwes, Joe Kox	155
	Let Me Think About It Again	Showtek, Ida Corr	129
	Rock My Body (with SASH!) [W&W x R3HAB VIP Remix]	R3HAB, INNA, W&W, Sash!	130
a)	Fuego	Timmy Trumpet, Blasterjaxx, Zafrir	132
	About You Now (How I Feel)	Niklas Dee, Luca-Dante Spadafora	170
	Raveship	HBz, Neptunica, MEELA	160
	I'll Be Okay	Neptunica	170
	Where Do We Go	LUNAX	150
	Push Up - Main Edit	Creeds	150[2]
	Cynical	twocolors, Safri Duo, Chris de Sarandy	130
	Sunglasses At Night	Gabry Ponte, Don Diablo	140
	Jungle	Alok, The Chainsmokers, Mae Stephens	140
	It Burns	Alle Farben	126
	La La La	Nicolas Julian	150

[1] Tunebat LLC. *"Key & BPM Database and Music Finder"*. 2023. https://tunebat.com (visited on 10/30/2023).

	Past Life	Felix Jaehn, Jonas Blue	140
b)	Ray Of Solar	Swedish House Mafia	135
	Not Fair	Niklas Dee, Old Jim, Enny-Mae	145
	Love All Night	Marten Hørger	126
	Easy On My Heart	Gabry Ponte	128
	S&M (With Chacel) - HYPER-TECHNO Edit	Macon, Chacel	142
	Esta Vida	Marshmello, Farruko	124
	The Magic Key	Trinix, One-T	122
	Fable	Klaas	142
	Drinkin	Mike Candys	130
	Past Lives	OBS	150
	When You're Lonely	VIZE, Emma Steinbakken	145
	On My Love	Zara Larsson, David Guetta	123
	Fly so High	Zombic, Rocco, Steve 80	150[2]
	Sweet Dreams	La Bouche, Paolo Pellegrino	130
	Weak - Nicolas Julian Remix	AJR, Nicolas Julian	140
	Typa Girl	southstar	150
	One with the Wolves	Robin Schulz	128
	On & On	Armin van Buuren, Punctual, Alika	128
	Sleepless (feat. GoldFord)	Restricted, Topic, GoldFord	150
	Sinner	Nu Aspect	136
Test	Wake Me Up - Radio Edit	MOLOW, Nito-Onna	132
	Cold as Ice	LUNAX, KYANU	160
	Good Life	FAST BOY	126
	Eternity	Timmy Trumpet, KSHMR, Bassjackers	140
	High On Life	le Shuuk, MERYLL	165
	Living On Video (feat. DTale)	Mike Williams, DTale	126
	On The Move	LIZOT, PRISKA	136
	Desire (with Sam Smith)	Calvin Harris, Sam Smith	140
	Lose This Feeling	Armin van Buuren	75
	One in a Million	Bebe Rexha, David Guetta	138

Table A.1.: Music selection for Data Sets 1 and 2

[2] Doubled tempo compared to original *Tunebat* information

A.2. Data Sets 3

Variant	Song	Interpret	Tempo [bpm]
Shortened + Complete	Eat The Rich	Aerosmith	123
	Move	SAINT Motel	101
	m.A.A.d city	Kendrick Lamar, MC Eiht	91
	GDFR	Flo Rida, Sage The Gemini, Lookas	146
	Good Times Roll	Big Gigantic, GRiZ	100
Complete	Something Just Like This	The Chainsmokers, Coldplay	103
	Heavy Is the Head	Zac Brown Band, Chris Cornell	171
	Life In Color	OneRepublic	127
	It Ain’t Me	Kygo, Selena Gomez	100
	All Time Low	Jon Bellion	90
	Can’t Stop	Red Hot Chili Peppers	91

Table A.2.: Songs in Data Set 3