OMR - A deep learning approach

1st Christoffer Vang Roed
Advanced machine learning
IT University of Copenhagen
Copenhagen, Denmark
chvr@itu.dk

2nd Filip Peška
Advanced machine learning
IT University of Copenhagen
Copenhagen, Denmark
filp@itu.dk

3rd Hadrien Tramoni
Advanced machine learning
IT University of Copenhagen
Copenhagen, Denmark
hadt@itu.dk

Abstract

Optical music recognition deals with the task of computationally reading of music notation in documents. A large part of music is still stored as manual written notes today. Transcribing the large amount of handwritten music scores to digital interpretable music score is of high interest for scientific musicological society [1]. Furthermore, it provides an opportunity to preserve and translate music scores written in e.g. mensural notation to modern notation. In this article, we propose a convolutional recurrent neural network (CRNN) for optical music recognition (OMR) and evaluate its performance on the PrIMuS dataset, which includes 87,678 real-music incipits. This approach was presented by [2] in 2018 which contributed significantly to the research field of OMR. We use normalized edit distance as the evaluation metric to measure the similarity between the predicted and ground-truth music scores. Using this metric, our CRNN model was able to achieve an average normalized edit distance of 1.0964 % on the test dataset, corresponding to accuracy of almost 99 %, indicating very good performance in recognizing music notation, close to the results achieved by [2]. We also compare the performance of our CRNN model to the currently available state-of-the-art methods presented by previous literature, to look for significant improvements. Our results demonstrate the effectiveness of using CRNN for optical music recognition.

***Index Terms*—OMR, Deep learning, CTC, Convolutional recurrent neural networks**

I. Introduction

Optical music recognition (OMR) refers to a computer's ability to read music, i.e. musical notation symbols, and decode these [1]. A large part of music is today stored as handwritten symbols, which is difficult to work with on computers. This alone would be highly beneficial for the scientific musicological domain to have an interpretable digital encoder. To this day, this transformation has not been possible due to the time-consuming nature of manual human based approach [2]. To digitalize these handwritten music scores, a less costly method was needed, which is where OMR came into place.

The term OMR is a broad term which has been used to describe different sub-tasks and has not yet been commonly defined in the literature [1]. However, these sub-tasks have a common focus on enabling computers to interpret music scores [1]. Some of the divergence in definition of OMR has occurred because OMR has not been considered a research field on its own, therefore research in this field has been limited and fractured though time [1].

Optical Character Recognition (OCR) is a related field that represents the research field of automatic processing of handwritten texts. It has gained much more focus, mostly due to higher funding and wider user base. Despite both OMR and OCR deal with recognition of characters, OCR techniques cannot be directly transferred and used for OMR, mainly because of complex long term dependencies between characters in OMR, e.g. the clef is important for how the following notes has to be interpreted. In this paper, the definition of OMR would be based on the one proposed by [1], which is defined as follows: "Optical Music Recognition is a field of research that investigates how to computationally read music notation in documents."

Recently some breakthroughs have been made in the field of OMR, which can be attributed to easy availability of large high quality datasets and the emergence of Neural networks, such as Convolutional neural networks, and recurrent neural networks [2]. This paper explores the potential of using deep learning to solve OMR tasks, more specifically using a convolutional recurrent neural network (CRNN), and the "Printed Images of Music Staves" (PrIMuS) dataset inspired by [2]. The PrIMuS is a large, high quality dataset that contains 87 678 real-music incipits, which makes it ideal for a machine learning task. It will be further detailed in section III. The use of CRNN is not new. CRNN combines the strengths of convolutional neural networks (CNNs) and recurrent neural networks (RNNs), and is typically used for tasks that involve sequential data. Previous studies have shown that CRNN has added significant improvement for Handwritten Music Recognition for Modern and Mensural notation, compared to previous approaches [2], [3]. Convolutions neural-networks are especially good for visual data processing, which is ideal for capturing the written symbols in the image, which then can be fed to the recurrent neural network [2]. The recurrent neural networks have shown promising results in the text processing domain, thanks to, among other things, its ability to preserve dependencies between features, which addresses the issue of capturing long term dependencies between the musical symbols [2]. The Whole CRNN network can be trained in end-to-end approach using Connectionist Temporal Classifi-

cation (CTC) loss function, which is suitable for generating sequences out of sequences.

II. Methods

Our model's core has the following structure:

- CNN block to slice the relevant features,
- RNN block made out of a BiLSTM to carry the meaning over long sequences,
- CTC as loss function,
- normalized edit distance as an evaluation metric.

A. CTC

The Connectionist Temporal Classification (CTC) loss is a loss function used for training sequence generating tasks, such as speech recognition, where the input is a sequence of frames or features and the output is a sequence of labels. The goal of the CTC loss is to align the input sequence with the output sequence in co-called *extended labeling*, such that each input frame is associated with a suitable output label. As CTC can generate a sequence of size m for input of size n, where $m \leq n$, it is perfect for our type of task.

To understand CTC loss, we need to define a so-called *CTC alphabet* which includes all possible output labels (further denoted as Σ), as well as a special "blank" symbol that can be generated instead of any "valid" output symbol. We can then perform a classification of each entry of input sequence to produce the already mentioned extended labeling. This labeling is then processed using the CTC as follows:

1) Merge multiple consecutive instances of the same symbol into a single instance.
2) Remove the "blank" symbol.

Application of these rules will be further denoted as a function $\mathcal{B}$.

To put it simple, we compute probability for each symbol from CTC alphabet in each time-stamp t. Probability of the extended labeling is then simply a product of probabilities of symbols that are part of it in corresponding time-steps.

As generally exponentially many extended labeling lead to the same output sequence, we have to sum the probabilities of every extended labeling that would result in the given output sequence after application of $\mathcal{B}$ to it.

For input sequence x and output sequence y, the CTC loss is then defined as

$$-\log(P(y|x)).$$

This loss can be then minimized by the classic gradient-descend-based algorithms. Our goal is to train the model in a way that it outputs high probability (ideally close to 1) for correct classifications.

To make the computation effective, dynamic programming based algorithms are commonly used.

The last part of the CTC loss is decoding. The simplest algorithm would be to find the most probable path, i.e. to take the most probable symbol at each time-step. We then apply $\mathcal{B}$ to the resulting extended labeling. As this approach is also the fastest one, we decided to use it in our project because it is perfectly enough for our problem.

More complex algorithms are commonly used, such as beam-search or prefix-search decoding.

B. BiLSTM

Bidirectional Long Short-Term Memory (BiLSTM) is a type of recurrent neural network that is trained to process sequential data by considering both past and future context. One advantage of BiLSTM over other types of RNNs, including traditional Long Short-Term Memory (LSTM) networks, is that it is able to capture long-distance dependencies in the data more effectively.

It is capable of contextualizing the whole sequence better, as even the information from the future may be important for the musical symbol's recognition (even it may not sound intuitive at the first glance). For example, the duration of a note may depend on the presence of ties or dots in the notation, which may not be immediately apparent when processing the input sequence from left to right. Also, we have to consider that every bar mush get fixed amount of beats, for which information from the future may also be important.

C. CRNN, an end to end approach

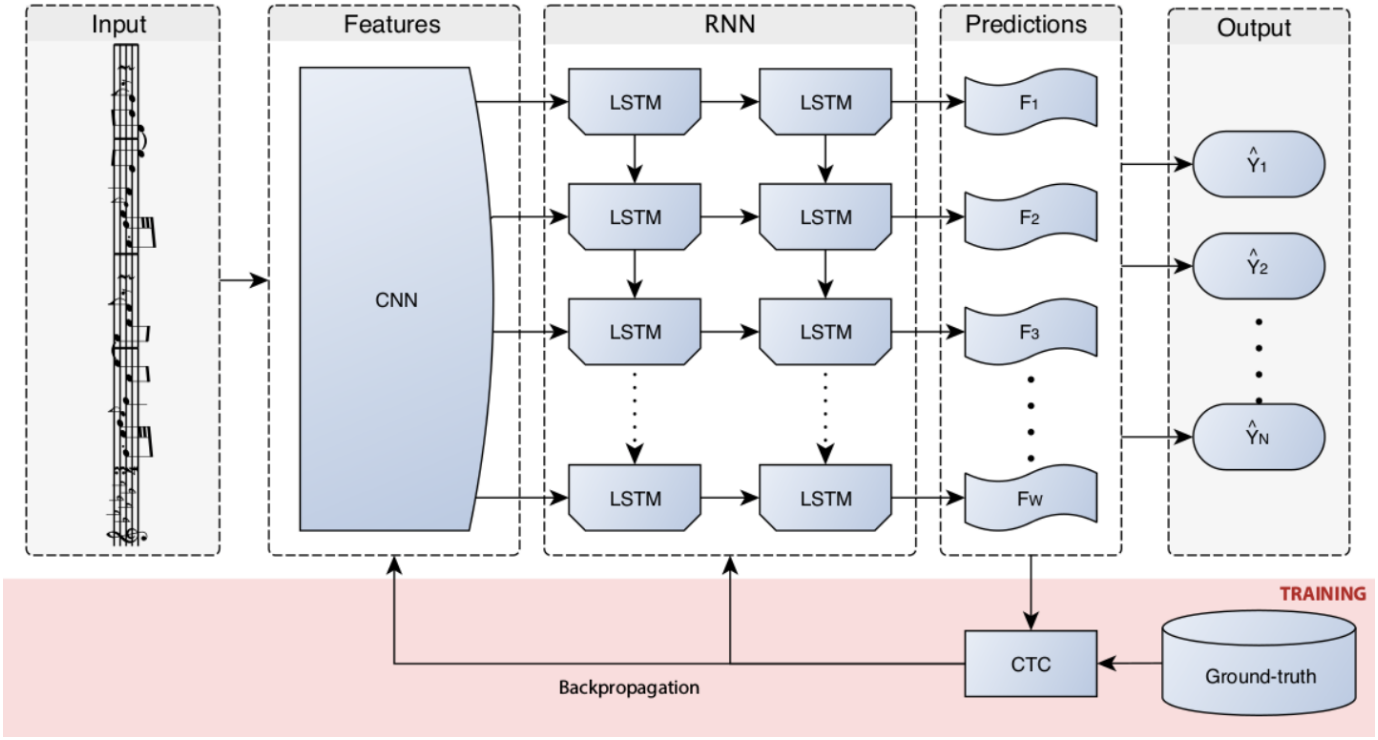
The CRNN, which represents an end to end approach for performing OMR, is covered at high level in this section. An end to end approach is defined as a solution where no manual labour is needed in the process besides from the training phase [2]. The CNN handles the extraction of the features, the RNN is used for their contextualization.

Connecting CNN outputs to RNN is not straightforward, as CNNs produce 4D data in format (batch, rows, columns, filters) and RNNs takes 3D data as input in format (batch, time-steps, features). This gets solved by first transposing the dimensions of the last CNN output to take the form (batch, column, rows, filters) and then reshaping to $(|\text{batches}|, |\text{columns}|, |\text{rows}| \cdot |\text{filters}|)$. It is kind of logical to do it this way as the columns represent the time-steps in music score images, not the rows, hence the transposition.

Each RNN cell is then connected to the dense layer with $|\Sigma|+1$ units without activation. This way, logits are generated, which then serve as an input for the CTC. CTC can then translate them to probabilities and produce already mentioned extended-labeling.

Note that for better regularization, dropout layers are used before and after the BiLSTM layer.

A graphical illustration of the whole architecture can be seen in II-C. Note that the model on the illustration has few differences. The first one is that the image fed into the model is rotated 90 degrees. This way, no transposition is needed, as the rows represent the time-steps. The second difference is that it stacks two LSTM layers with the same direction, not the opposite one (which is what we did).



An overview of the used architecture i.e. a convolutional recurrent neural network similar to the one used in this paper presented by [2]

D. Evaluation metric

Edit distance, also known as Levenshtein distance, is a measure of the similarity between two strings of words or characters. It is defined as the minimum number of insertions, deletions, or substitutions required to transform one string into the other [4]. Edit distance is often used as an evaluation metric for tasks that involve comparing sequences of data, such as machine translation, speech recognition, and music transcription. It is particularly useful when the sequences are of variable length and may contain errors or variations.

The normalized edit distance (NED) is calculated by measuring the predicted notes against the ground truth, where a value of 0 means there is no difference between the predicted nodes and ground truth, and a value of 1 means there is no similarity [4]. Our results are presented as the percent of the edit distance, which is the percent of wrongly classified nodes. The selected evaluation metric edit distance is similar to what was used by [2], where it is denoted as Symbol Error rate. Furthermore, [2] includes a second metric, sequence error rate, which measures the amount of correctly classified sequences. This has not been included in this study.

III. Experiments

We trained our models on the Printed Images of Music Staves (PrIMuS) dataset. For an OMR task, it is the simplest case for the following reasons:

- The samples are printed scores with a single staff
- The scores are perfectly clear (no blur)
- The notes are normalized and perfectly aligned

Therefore, the preprocessing will be minimalist. As the scores are already binarized, we only have to pad them to have a normalized shape.

We have trained different models based on the same architecture. Let's review two of them. The models are available in the appendix as they take too much space. Our first model is 4 CNN layers into a bidirectional LSTM. We trained one for 3 epochs and the other for 10 epochs.

For our second model, we use the first model as a basis and add an Inception block between the CNNs and the RNN. The idea with implementing an Inception block is that it is an important component from one of the current best image classifier and thanks to its (1x1) convolution's filters, the number of parameters does not increase exponentially. Also, the Inception block is designed to allow the model to learn different features at different scales, which could be useful for OMR, since the musical symbols can appear in a variety of sizes - for example grace notes or rests.

We trained this model for 10 epochs so that we can compare with the base model trained for the same amount of time.

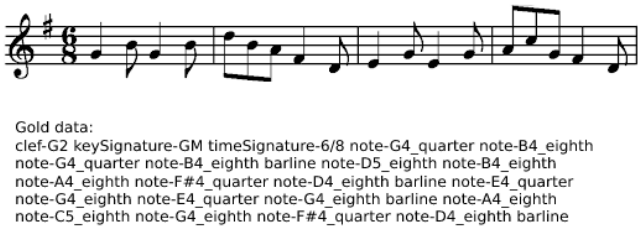
The training time for the inception model is close to the training time of the base model, however, it has almost twice as many parameters. It is also better in every metric - including edit distance on validation dataset, edit distance on test dataset and loss on train and validation dataset.

It can be concluded from the figures 3 and 4 that the base model overfitted during the last epoch, which led to the typical U-shaped curve on the loss and NED on the validation dataset. This may be caused by the fact that the model has learned the training data too much, and it learned the noise from it rather than the underlying signal, which leads to a poor performance on unseen data. To prevent this, more regularization could be done, e.g. larger dropout rates. The simplest solution would be to stop the training earlier. The results on the test dataset for base model after 10 and 3 epochs and for the inception model after 10 epochs can be seen in the following table I.

Model	Number of epochs	NED on test dataset (%)
Base	3	2.0861
Base	10	19.4453
Inception	10	1.0964

TABLE I
MODELS PERFORMANCE COMPARISON

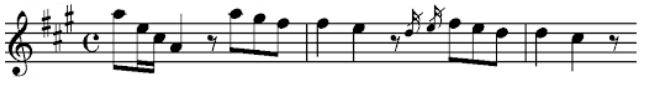
Figures 1 and 2 feature images from the test dataset, their annotation and the prediction made by the base model after 3 epochs. It can be seen that the model is fully capable of predicting a variety of music symbols correctly even after 3 epochs with just minor occasional mistakes. The results support the general low error rate archived using the CRNN.



Gold data:
clef-G2 keySignature-GM timeSignature-6/8 note-G4_quarter note-B4_eighth note-G4_quarter note-B4_eighth barline note-D5_eighth note-B4_eighth note-A4_eighth note-F#4_quarter note-D4_eighth barline note-E4_quarter note-G4_eighth note-E4_quarter note-G4_eighth barline note-A4_eighth note-C5_eighth note-G4_eighth note-F#4_quarter note-D4_eighth barline

Base model prediction:
clef-G2 keySignature-GM timeSignature-6/8 note-G4_quarter note-B4_eighth note-G4_quarter note-B4_eighth barline note-D5_eighth note-B4_eighth note-A4_eighth note-F#4_quarter note-D4_eighth barline note-E4_quarter note-G4_eighth note-E4_quarter note-G4_eighth barline note-A4_eighth note-C5_eighth note-G4_eighth note-F#4_quarter note-D4_eighth barline

Fig. 1. Prediction on a random sample. The model recognized the notes correctly.



Gold data:
clef-G2 keySignature-AM timeSignature-C note-A5_eighth note-E5_sixteenth note-C#5_sixteenth note-A4_quarter rest-eighth note-A5_eighth note-G#5_eighth note-F#5_eighth barline note-F#5_quarter note-E5_quarter rest-eighth gracenote-D5_sixteenth gracenote-E5_sixteenth note-F#5_eighth note-E5_eighth note-D5_eighth barline note-D5_quarter note-C#5_quarter rest-eighth

Base model prediction:
clef-G2 keySignature-AM timeSignature-C note-A5_eighth note-E5_sixteenth note-C#5_sixteenth note-A4_quarter rest-eighth note-A5_eighth note-G#5_eighth note-F#5_eighth barline note-F#5_quarter note-E5_quarter rest-eighth gracenote-E5_sixteenth note-F#5_eighth note-E5_eighth note-D5_eighth barline note-D5_quarter note-C#5_quarter rest-eighth

Fig. 2. Prediction on a random sample. The model made one mistake - missed one of the grace notes.

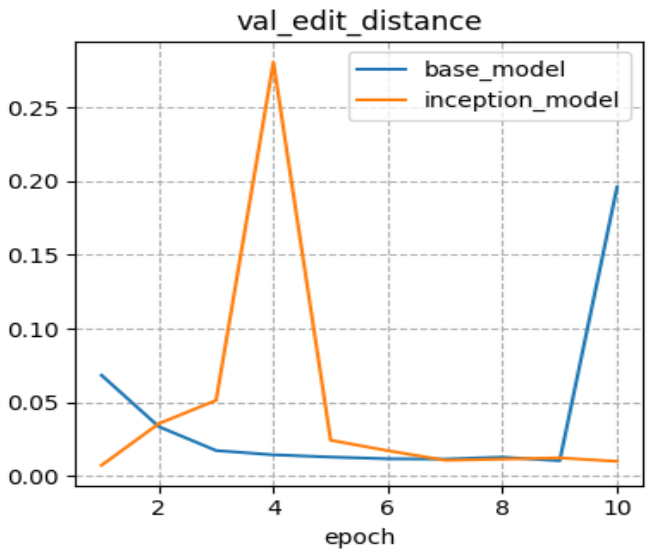


Fig. 3. Edit distance comparison between the models

Table II shows the sizes of the base and the inception model, it seems that the inception model is consisting of nearly the double amount of parameters compared to the base model. This seems to have limited effect on the results, but it also only demands limited extra training time, so the additional computational time and the benefits in performance seems to add up.

Model	Number of parameters	Training time for 1 epoch (s)
Base	7.334.070	1000
Inception	13.168.374	1100

TABLE II
MODELS SPECS

IV. DISCUSSION

Our best model achieved 1.0964 % NED on the test dataset. This result matches the results achieved by Zaragoza & Rizo (2018) [2] which yielded an accuracy of 1% evaluated using symbol error rate equivalent to our edit distance. The differences between the result archived by this study and the one conducted by Zaragoza & Rizo (2018) [2] seems to be partly attributed to the amount of epochs conducted in the training phase. Our results are based on 10 epochs where Zaragoza & Rizo (2018) [2] did 90 epochs, it may potentially have had a positive effect on the model to extend the training with more regularization, which wasn't done due to lack of available computational powers, but we also showed that even models trained for less time achieve very close results.

Not many studies have carried out an end to end approach for OMR, where most previous papers only dealing with sub parts of the problem, staff removal, symbol classification, contrast enhancement etc. [1], [2]. Therefore, there are not many approaches to compare these results against, which also states the fact that CRNN must be considered a breakthrough within the area of OMR. The previous presented approach

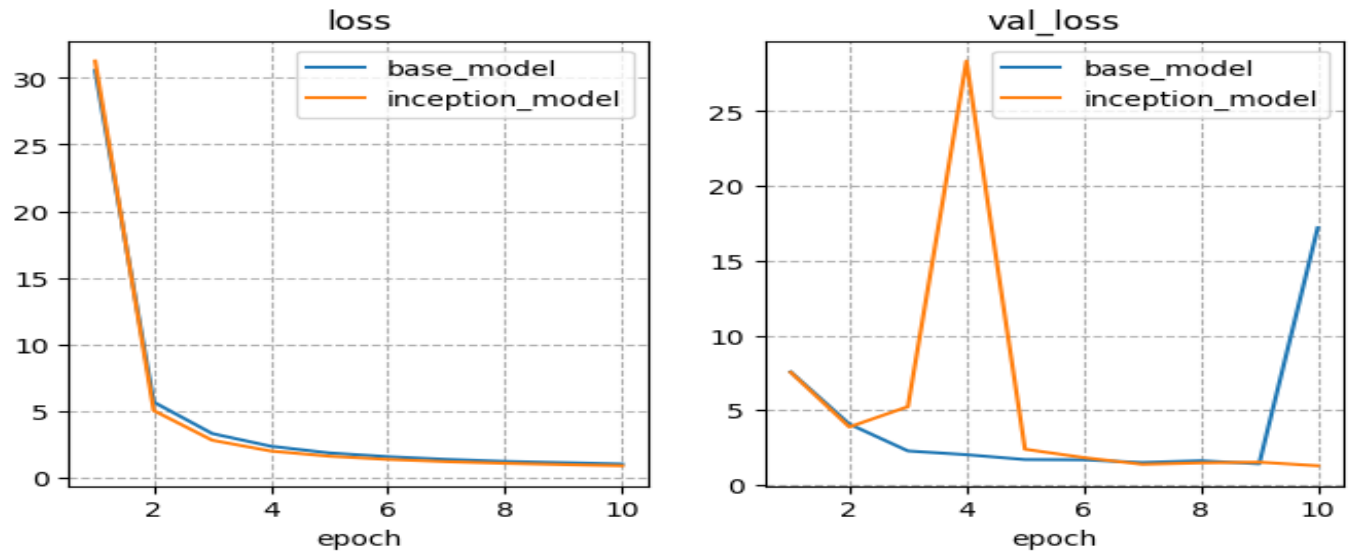


Fig. 4. Loss comparison between the models

which came closest to offer an end to end approach was developed by Audiveris [5], who offered a tool to complete this task of staff detection and classification using the PrIMuS dataset. Using the approach offered by Audiveris [5], it was tested by Zaragoza & Rizo (2018) [2] who managed to achieve an accuracy of 44.2% symbol error rate, significantly lower than the one achieved using the CRNN framework [2].

The results achieved by [2] demonstrated that the most frequent error was denoted to the symbol bar line being the cause of the error 45.5% percent of the time when using an agnostic representation of the music scores. It would be interesting to test if the symbol bar line also is the primary cause of errors in our model.

Even though CRNN has yielded very promising results, it has to be noted that these results have to be attributed to the availability of a very large high quality data consisting of simple single staff nodes. It would be interesting to further investigate, how the model would perform on a dataset with more complicated staffs and noise such as semi visible notes from other pages which is common among most of not properly cleaned handwritten notes [1]. Would it be possible to retrain the model on a smaller more complicated dataset using transfer learning and still utilize the benefits learned from the PrIMuS dataset, to approach real world scenarios.

Further research and lessons learned in the field of OCR could possibly contribute to utilize CRNN in real world scenarios, because the robustness of the model and data quality would have to be addressed in further research.

V. Conclusions

In conclusion, the OMR model developed in this study shows promising results for accurately predicting the outcomes of musical scores. The model was trained on a large dataset PrIMuS of musical scores and achieved an impressive normalized edit distance of 1.0964 %. Furthermore, the best model demonstrated strong generalization abilities, performing well on both the training, validation and test datasets. The combined CRNN demonstrated that an end to end approach is possible without compromising the quality of the OMR task. These findings serve as a potential important stepping stone for a cost-effective approach for transforming larger parts of handwritten music scores into digital music scores. The data used in this study was the cleanest as possible, which highly contributed to the achievement of the model. An extension to this project would be to reach the same results using rawer data, including more complicated staffs and blurred images, such data that would suffer from lack of extensive preprocessing.

REFERENCES

[1] J. Calvo-Zaragoza, J. H. Jr, and A. Pacha, "Understanding optical music recognition," *ACM Computing Surveys (CSUR)*, vol. 53, no. 4, pp. 1–35, 2020.

[2] D. R. Jorge Calvo-Zaragoza, "End-to-end neural optical music recognition of monophonic scores," *Applied Sciences*, vol. 8, p. 606, 2018.

[3] J. Calvo-Zaragoza, A. H. Toselli, and E. Vidal, "Handwritten music recognition for mensural notation with convolutional recurrent neural networks," *Pattern Recognition Letters*, vol. 128, pp. 115–121, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167865519302338

[4] S. Rani and J. Singh, *Enhancing Levenshtein's Edit Distance Algorithm for Evaluating Document Similarity*, R. Sharma, A. Mantri, and S. Dua, Eds. Singapore: Springer Singapore, 2018.

[5] H. A. Bitteur, https://github.com/Audiveris/audiveris, accessed: 2023-01-01.

APPENDIX

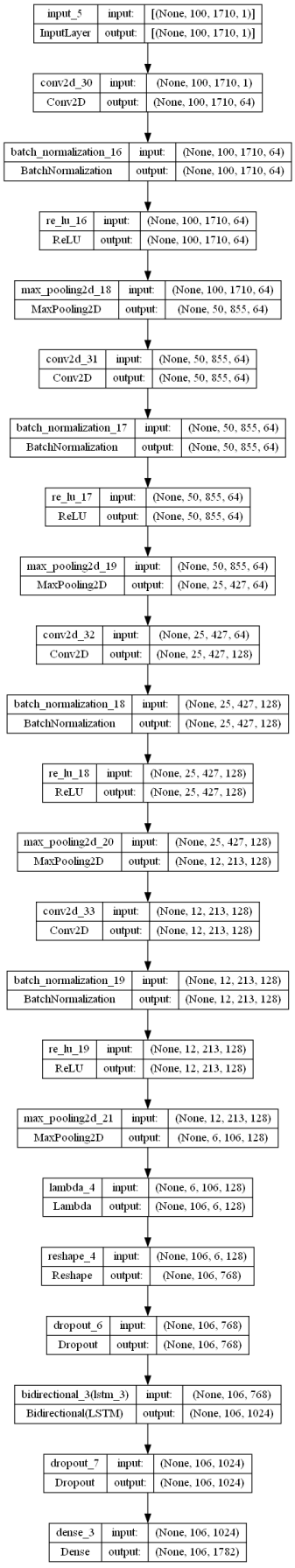


Fig. 5. Architecture of the base model.

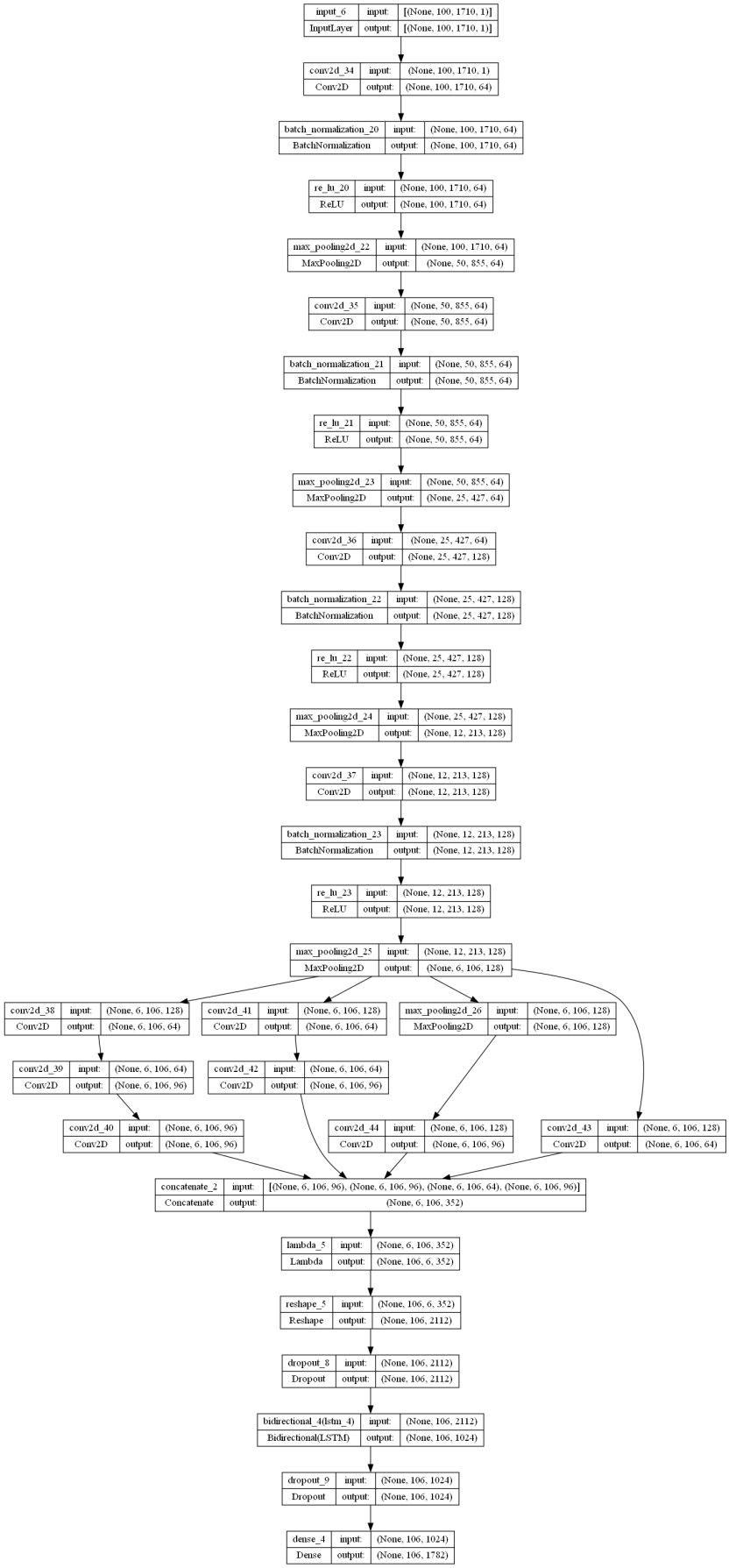


Fig. 6. Architecture of the inception model.