Comparative Analysis of Time–Frequency Representations and Neural Architectures for Polyphonic Piano Transcription

Oliver Björklund
Politecnico di Torino
Torino, Italy
s350431@studenti.polito.it

Abstract

Automatic music transcription (AMT) aims to produce symbolic notation such as MIDI from audio recordings. This study explores polyphonic piano transcription which requires detection of multiple simultaneous pitches. The study does this by comparing different time-frequency representations such as Mel, CQT and STFT alongside different architectural designs such as CNN and CRNN. The impact of network depth, residual connections and temporal modeling is investigated. A robustness evaluation is also conducted to evaluate how well models adapt to degraded conditions.

The results show that the CQT and Mel representations generally performs very well when using both CNN and CRNN for training. The recurrent structure is very strong for onset detection as well as reducing degradation when exposed to certain types of corruptions. However, deeper recurrent stacks do not provide additional gains, and all models degrade under very severe corruption.

1. Introduction

The task of Automatic Music Transcription (AMT) seeks to transform audio recordings of instruments to structured symbolic notation such as MIDI. In the case of polyphonic piano transcription, this task includes detecting multiple simultanous pitches along with their onset and offset times. The task is inherently challenging due to overlapping harmonic content, wide dynamic ranges, expressive timing variations, and sometimes complex temporal dependencies between notes.

Deep neural network approches has significantly advanced the field of AMT by being applied to time-frequency representations. Later approaches including RNNs, LSTMs and transformers has also been incorperated for improved results. However, the performance of an AMT model depends hugely on the spectral representation used as well as the core network architecture. Differences in frequency resolution, temporal modeling capacity, and network depth can meaningfully influence performance. As well as robustness and generalization.

This study compares three different time-frequency representations, STFT, Mel-spectrograms and Constant-Q Transform (CQT), along several different network architectures based on convulutional neural networks (CNN), as well as convolutional-recurrent neural networks (CRNN). The study analyzes the effect of convolutional depth, residual connections, and recurrent layer configuration on frame-level and note-level transcription performance, using the MAESTRO dataset. Additionally, the model robustness is evaluated on synthetic corruptions with the purpose of assessing model performance under degraded conditions.

By systematically compare different time-frequency representations, architectural choices as well as temporal modeling in a controlled setting, this study aims to investigate how these design choices impact polyphonic piano transcription performance.

2. Data

2.1. Primary Dataset: MAESTRO

2.1.1 Dataset Description

The primary dataset used for training was the MAESTRO v3.0.0 dataset [4], which contains high quality piano recordings paired with precise MIDI notations captured with Yamaha Disklavier. The MAESTRO dataset is very well suited for the task of polyphonic music transcription due to it having a very precise temporal alignment, diverse pieces, as well as being very comprehensive. The precisely aligned MIDI makes supervised learning possible both for frame and note level. The MAESTRO dataset is often considered a standard training dataset in this feild.

2.1.2 Dataset Statistics

The MAESTRO dataset contains over 200 hours of piano recordings spanning over 1 200 different pieces. The dataset has an official train/val/test split which was used without modifications for this study. The frequencies included in the dataset spans from 21 (A0) to 108 (C8) representing all the 88 piano keys. As it is a polyphonic dataset, at a given frame, most pitches will be inactive, resulting in a very sparse representation.

2.1.3 Data Split Strategy

As already mentioned, the official MAESTRO train/val/test split [4] was used without any modifications. All hyperparameter tuning and threshold optimization were performed exclusively on the validation set, and final performance metrics were reported on the test set.

2.2. Class Distribution and Imbalance

This kind of automatic music transcription is defined as an 88-dimensional multi-label classification problem. This means that each frame can contain multiple active pitches. However in piano music, at a given frame, the vast majority of pitches are inactive. This leads to the frame label matrix being very sparse with a big imbalance between active and inactive pitches. This imbalance creates a bias toward negative predictions and motivates the use of weighted loss functions during training, as is described in the Methodology section.

2.3. Data Augmentation

No explicit data augmentation techniques such as pitch shifting or time stretching was used in this study. All models was trained on the clean MAESTRO dataset to preserve the precise temporal and pitch alignment.

However, random cropping of fixed-length sequences during the training can be interpreted as an implicit form of data augmentation. At each epoch different temporal segments of the same performance was sampled. Hence exposing the model to varying musical contexts and boundary conditions. This does increase the diversity of the training while still preserving the spectral and temporal characteristics.

Any additional explicit data augmentation techniques were not used for two main reasons. Augmentations such as pitch shifting and time stretching requires careful handling of the MIDI files to preserve precise frame-level alignment established during preprocessing. Also, due to space constraints, it was not a viable approach to preprocess variations of the representations with different augmentations applied.

2.4. Secondary Dataset: MAPS

2.4.1 Dataset Description

For cross dataset evaluation the MAPS dataset [2] was used. This differs from MAESTRO as it consists of piano recordings of both real pianos and synthesized instruments. Compared to MAESTRO MAPS differs in recording setup, instrument characteristics, and acoustic environment This introduces a domain shift between training and evaluation data.

2.4.2 Evaluation Protocol

The MAPS dataset was used solely for cross-dataset evaluation. The evaluation was conducted without any fine-tuning on the MAPS dataset and the same preprocessing pipeline and feature extraction was used for MAPS to ensure consistency

3. Methods

The goal of this system is to conduct polyphonic piano transcriptions from audio recordings. Audio files are first converted into time-frequency representations and then feeded either to a CNN or CRNN to predict frame-level pitch activations and onsets. Predictions are then decoded into symbolic note events $(\text{pitch}, t_{\text{onset}}, t_{\text{offset}})$.

3.1. Audio Preprocessing

All audio recordings are converted into a standardized representation prior to the time–frequency analysis. Audio files are first loaded using`torchaudio` and then converted to mono by averaging the channels. The audio files are also resampled to 16 kHz to reduce computational cost while still preserving musical content.

The audio waveform is segmented using a hop-size of 512 which approximentaly corresponds to 32 ms per frame. This hop size defines the temporal resolution of the system and is in this study shared across all input representations. This ensures a consistent alignment between audio and symbolic notation.

The magnitude of the time-frequency representation is calculated and then a logarithmic compression $\log(1 + x)$ is applied. The purpose of this is to reduce the dynamic range of the input features as well as improve the numerical stability.

By using the `pretty_midi` library ground truth labels for the dataset is generated. Here only the relevant piano frequencies are considered, which results in an 88-dimensional pitch target. Frame-level pitch activation labels are constructed by discretizing note onset and offset times according to the same frame rate as the audio features. Onset labels are also generated by marking the frame which corresponds to a notes start time.

The dataset is split into training, validation, and test sets according to the official metadata found in the MAESTRO dataset. For training and storage efficiency, all the labels and corresponding time-frequency representations are precomputed and cached to disk.

Normalization statistics are also calculated per frequency-bin for the training set. These statistics are later used to normalize the input to zero mean and unit variance for training. This is done independently for each time-frequency representation.

3.2. Time–Frequency Representations

Time-frequency representations explain how spectral content evolves over time. By keeping the same resampling rate as well as hop length, different time representations can be evaluated while still keeping the same model architecture.

STFT. The Short-Time Fourier Transform is computed with an FFT size of 2048 and hop length of 512. This results in 1025 linearly spaced frequency bins which covers 0–8 kHz. The STFT serves as a baseline spectral representation without any perceptual or musical scaling.

Mel-spectrogram. The Mel-spectrogram setup uses the same FFT, and then projects onto 229 Mel-bins with frequencies between 30 Hz and 8000 Hz [3]. The Mel scale approximates human auditory perception by providing higher resolution at lower frequencies and coarser resolution at higher frequencies.

CQT. The Constant-Q Transform is computed with 36 bins per octave and a minimum frequency of 27.5 Hz (A0). It has a total of 264 frequency bins covering the full piano range. Its logarithmic spacing aligns naturally with musical pitches.

3.3. CNN-Based Architecture

The baseline model which was used is a convolutional neural network (CNN) which is designed for frame-wise multi-pitch classification. The network takes as input a time-frequency representation on the form $X \in \mathbb{R}^{F \times T}$. The network then predicts a vector of 88 logits which correspond to the piano pitch range, for each time frame t as follows:

$$\hat{\mathbf{y}}_t \in \mathbb{R}^{88},$$

Sigmoid activation is then applied independently to each pitch.

Input Representation The input spectrogram to the CNN is treated as a single channel image. This is done by adding a channel dimension resulting in a shape of $(1, F, T)$.

Convolutional Backbone The baseline CNN used consists of four convolutional blocks with channel sizes as follows:

$$[64,\ 128,\ 256,\ 512].$$

Each of the convolutional blocks contains 2 convolutional layers with a kernel size of 3×3. These layers are followed by batch normalization and ReLU activation.

After each block, max pooling is applied using the sizes that follows:

$$[(2,1),\ (2,1),\ (2,1),\ (1,1)].$$

As shown, pooling is primarily used for the frequency axis. This reduced the spectral resolution after most blocks while still preserving the temporal resolution. This design hoice was made to reflect an important point in transcription. Keeping a more precise temporal resolution for the frame level classification, but still a more moderate compression for frequency, which makes the model learn more abstract harmonic patterns.

Frame-Level Prediction Head To go from this to frame-wise prediction, the tensor is reshaped to the size $(T, C \cdot F')$. This produces one feature vector per time frame. Here $C = 512$ and F' is the reduced frequency dimension. A dropout layer with rate 0.3 is then applied for regularization. Ultimately a fully connected layer maps the produced frame representations to 88 output logits.

3.4. CRNN-Based Architecture

To be able to use longer temporal contexts beyond the normal receptive field of convolutions the baseline was extended to a convolutional recurrent neural network (CRNN). This CRNN works similar to the baseline CNN, it uses the same CNN frontend as the baseline to get frame-wise features from the time-frequency input. It then applies recurrent layers to model temporal dependencies.

Convolutional Frontend As already mentioned, the CRNN architecture uses the same CNN frontend as the baseline, with channels and pooling sizes as described in section 3.1. This results in a representation $(T, C \cdot F')$ with one feature vector per time frame.

Projection Bottleneck Before the recurrent part of the network a linear projection is applied. The purpose of this is to reduce the dimensionality of the CNN features. For the experiments conducted, the CNN features are projected 512-dimensional embedding, by using a fully connected layer and ReLU activation. This bottleneck reduces the input size of the recurrent layers and also acts as an additional learnable feature transformation.

Recurrent Temporal Modeling In the baseline CRNN the temporal context is modelled using a 2 layer bidirectional LSTM [5]. In the standard setup the LSTM layer size is 256 and a dropout of 0.3 is used between layers. If $\mathbf{z}_t$ defines the generated CNN features at time t, the LSTM will produce $\mathbf{h}_t$ as a contextualized representation. This representation should integrate both past and future information.

$$\mathbf{h}_t = \mathrm{BiLSTM}(\mathbf{z}_{1:T})_t.$$

Dual Output Heads Instead of using 1 linear output head, the CRNN based approach uses 2 linear output heads [3] which are applied to each recurrent output. First $\mathbf{h}_t$: which is a *frame* head focusing on pitch activation, and also a *onset* head which focuses on note onset detection. Both these heads produce a similar output of 88 logits per frame as follows:

$$\hat{\mathbf{y}}_t^{\text{frame}} \in \mathbb{R}^{88}, \qquad \hat{\mathbf{y}}_t^{\text{onset}} \in \mathbb{R}^{88}.$$

3.5. Training Objective and Loss Functions

Frame-Level Loss For the part of trying to identify pitch activation, the Binary Cross-Entropy with Logits was used (BCEWithLogitsLoss). If we define the logits predicted by the network as $\hat{\mathbf{y}}_t$ and define the labels from the ground truth as $\mathbf{y}_t \in \{0,1\}^{88}$, we can then define the loss for a single frame as follows:

$$\mathcal{L}_{\text{frame}} = \mathrm{BCEWithLogits}(\hat{\mathbf{y}}_t, \mathbf{y}_t).$$

There is a pretty large class imbalance in music transcription seen to active/inactive notes. This is because most frames only have very few active pitches. To make this issue less catastrophic a positive class weight was applied to the loss. This helps as it increases the punishment for false negatives, making the model more reliably detect note activations.

Onset Loss (CRNN Only) When implementing the CRNN model another binary cross-entropy loss was added to the onset prediction head [3].

$$\mathcal{L}_{\text{onset}} = \mathrm{BCEWithLogits}(\hat{\mathbf{y}}_t^{\text{onset}}, \mathbf{y}_t^{\text{onset}}).$$

The total loss for the CRNN is a weighted sum of these two. Using an onset weight of 1.5 for baseline experiments.

$$\mathcal{L} = \mathcal{L}_{\text{frame}} + \lambda\, \mathcal{L}_{\text{onset}},$$

here λ defines the onset weight factor.

Training vs. Evaluation Threshold During training for the reason of monitoring performance, sigmoid activation was applied and predictions thresholded to 0.5 for calculating precision, recall and F1-score. As discussed later, threshold optimization will find the optimal prediction threshold.

3.6. Training Procedure

Models are setup and trained using PyTorch Lightning in combination with Hydra configuration, which makes it easier to reproduce runs and setup different experiments.

During both training and validation to ensure that training is conducted on fixed length segments, all pieces are randomly cropped to a fixed length sequence T. This ensures efficient batching.

The normalization statistics which were previously computed on the training split, is now appplied to both training and validation splits.

The AdamW [6] optimizer is used with a weight decay of 10^{-4} for regularization and a tuned learning rate.

The ReduceLROnPlateau scheduler was used which reduces the learning rate by a factor of 0.5 if the validation F1-score does not improve over 5 epochs.

Early stopping is used if the validation F1-score does not improve over 15 epochs. The best model is selected according to validation F1-score and saved via checkpointing.

Hyperparameter Optimization (Optuna) Optuna [1] was used to tune the hyperparameters using the validation F1-score as tuning objective. Each trial was trained for a maximum of 25 epochs, with the use of pruning as well as early stopping.

The optuna search space was defined on the following parameters. Learning rate (10^{-4}–10^{-3}, log-uniform), dropout (0.1–0.5) and sequence length ($T \in \{625, 1000\}$). Additionally, the positive class weight was tuned (CNN: 1.0–4.0; CRNN: 4.0–7.0) and for the CRNN the onset loss weight was tuned in the range 0.5–2.5.

Architectural parameters such as channels, residual etc was not tuned with optuna, but rather tested separately to draw conclusions from architectural choices.

3.7. Post-processing and Transcription Logic

Sigmoid activation is applied to the logits and then binarized using a threshold τ which is calculated using the validation F1-score.

For each pitch a note is considered to be initiated at a time frame t if both the onset and the frame level activation exceeds this computed threshold, similar to [3]. The note stops when the activation goes below the threshold. To avoid short likely invalid activations, a minimum duration of 5 frames is considered. The resulting $(\text{pitch}, t_{\text{onset}}, t_{\text{offset}})$ tuples form the final transcription and can be exported as MIDI files.

3.8. Evaluation Metrics

The evaluation is conducted on frame level metrics for CNN, and also including note level metrics for CRNN using the best checkpoint saved during training.

Frame-Level Metrics Frame-level performance is measured using precision, recall and F1-score globally over all frames and pitches.

$$\text{Precision} = \frac{TP}{TP + FP}, \quad \text{Recall} = \frac{TP}{TP + FN}, \quad \text{F1} = \frac{2PR}{P + R}.$$

Note-Level Metrics For the CRNN model, additional to the frame level metrics, note level metrics are also used for evaluation.

The predicted notes, as well as the ground truth notes are converted to $(\text{interval}, \text{pitch})$ pairs. These pairs are then used with the `mir_eval` [7] transcription library. For onset correctness a tolerance of 50 ms is used, The offset tolerance is defined to be 20% of the reference notes duration [7]. For this setup, onset F1 as well as onset+offset F1 is reported.

Robustness Evaluation To test the robustness of the models, spectrogram-level corruptionos were applied to the data. Evaluation includes additive Gaussian noise at SNR levels of 20 dB, 10 dB and 5 dB as well as gain variations of -6 dB and -12 dB. The performance degradation is then reported relative to a non-corrupted version.

Zero-shot evaluation is also conducted on the MAPS dataset, using the same preprocessing pipeline.

4. Experiments

4.1. Experimental Setup

All the experiments presented in this section are conducted on the official MAESTRO train/test/validation splits. For cross dataset evaluation, models that are trained on the MAESTRO dataset are evaluated immedietly on the MAPS dataset [2] without any finetuning. All experiments follow the configuration defined in section 3. For experiments on CNN, hyperparameters are choosen using optuna. This hyperparameter search is conducted on the baseline configuration for each time-frequency representation and then reused for all entries in the grid-search. In other words, Optuna was ran once for each time-frequency representation. For CRNN models, the positive class weight was fixed to 5.0 and the onset loss weight to 1.5 to stabilize training and prevent early collapse of F1 scores. All hyper-parameters are reported for reproducibility. The selected hyper-parameters for each representation are summarized in Table 1.

A grid search was conducted on CNN approaches as well as CRNN approaches. The best performing model from each category was then evaluated on robustness as well as the MAPS dataset.

Table 1. Optimized hyper-parameters selected via Optuna.

Model	LR	Drop	PosW	Seq
CNN-CQT	3.99e-4	0.39	1.67	1000
CNN-Mel	2.86e-4	0.49	2.50	1000
CNN-STFT	3.99e-4	0.48	2.52	1000
CRNN-CQT	3.99e-4	0.39	5.00	1000
CRNN-Mel	2.86e-4	0.49	5.00	1000
CRNN-STFT	3.99e-4	0.48	5.00	1000

4.2. CNN Architecture Exploration

4.2.1 Search Space

The CNN architecture was evaluated using 3, 4 and 5 convolutional blocks, with the following channel configuration:

- 3 blocks: [64, 128, 256]
- 4 blocks: [64, 128, 256, 512]
- 5 blocks: [64, 128, 256, 512, 512]

For each number of blocks models were trained both with and without residual connections. This search space was applied to both CQT and Mel. However, due to computational constraints STFT was only evaluated on a 4-block non-residual configuration.

4.2.2 Results Across Representations

Table 2 reports frame-level F1-scores on the MAESTRO test set.

CQT. For CQT the number of block inreased the performance from 0.694 (3 blocks) to 0.722 (5 blocks). For shallow models, residual connections degraded performance, while not having a big performance impact on deeper networks. The best-performing configuration was the 5-block non-residual model (F1 = 0.722).

Mel. For Mel-spectrogram representations a similar depth trend appears, going from 0.665 (3 blocks) to 0.706 (5 blocks). Residual connections also had the same impact on Mel-spectrograms as on CQT. Overall performance remained below CQT across all configurations.

STFT. The 4-block non-residual STFT model achieved F1 = 0.700. Although a full depth exploration was not performed, STFT performed competitively but below CQT.

Summary. Overall performance increased with the number of blocks, with the performance slightly saturating from 4 to 5 blocks. Residual connections did not provide any measurable impact in this setting, which might be caused by the network being to shallow. Among representations

Table 2. Frame-level F1-score on MAESTRO test set for CNN architectures. B = number of blocks, R = residual connections (0 = off, 1 = on).

Model	B3	B4	B5
CQT (R0)	0.694	0.713	**0.722**
CQT (R1)	0.686	0.703	0.722
Mel (R0)	0.665	0.697	**0.706**
Mel (R1)	0.680	0.671	0.703
STFT (B4R0)		0.700	

CQT achieved the highest overall frame-level performance, followed by STFT and Mel. Which is most likely caused by CQT more closely resembeling the musical pitch structure.

4.3. CRNN Architecture Exploration

4.3.1 Search Space

The CRNN models were evaluated using the best performing CNN frontend for each representation. A grid search was conducted using LSTM hidden size $h \in \{128, 256\}$ and number of layers $l \in \{1, 2\}$.

4.3.2 Results Across Representations

Table 3 summarizes CRNN performance on the MAESTRO test set.

Mel. The best performing configuration for Mel was H256L1 with frame F1 = 0.723 and onset F1 = 0.833. Increasing hidden size from 128 to 256 improved both frame and onset F1 for single-layer models, however, adding a second LSTM layer decreased performance over all metrics. Offset F1 remained low across all configurations (0.178–0.220), showing limited improvement from deeper recurrence.

CQT. For CQT the best performing configuration was H128L1 with frame F1 = 0.717 and onset F1 = 0.837. As for Mel the single-layer models outperformed the two-layer models for both hidden sizes 128 and 256. The hidden layer size seemed to have a very small and also inconsistant impact on the models performance. Frame-level performance remained comparable to the CNN baseline (0.722), with differences within ±0.5%.

STFT. The evaluated STFT configuration (H256L1) achieved frame F1 = 0.697 and onset F1 = 0.835. The performance for frame level F1 was very similar to the CNN baseline, but STFT still showed pretty good onset detection.

Summary. Across representations the single-layer CRNNs consistently outperformed the two-layer models for both frame and note-level metrics. Increasing the hidden layer size generally had a small but positive impact on the result.

Table 3. CRNN performance on MAESTRO test set. H = hidden size, L = number of LSTM layers.

Model	Frame F1	Onset F1	Offset F1
Mel H128L1	0.689	0.819	0.215
Mel H128L2	0.615	0.734	0.178
Mel H256L1	**0.723**	**0.833**	**0.220**
Mel H256L2	0.681	0.792	0.202
CQT H128L1	**0.717**	**0.837**	**0.230**
CQT H128L2	0.667	0.778	0.199
CQT H256L1	0.704	0.821	0.223
CQT H256L2	0.717	0.827	0.221
STFT H256L1	0.697	0.835	0.216

When comparing the CRNN architecture to the CNN baseline for Mel-spectrograms, the frame-level F1 increased from 0.706 to 0.723 (+1.7%). With just a minor performance difference for CQT. The most pronounced effect of temporal modeling appears at note level: onset F1 exceeded 0.83 across several configurations, substantially higher than frame-level F1. Offset F1, however, remained low across all models, indicating that note termination remains challenging even with temporal context.

4.4. Robustness Evaluation

4.4.1 Corruption Protocol

A robustness analysis was performed on the best performing CNN and CRNN configurations for each representation. First the model was evaluated on a clean test set, and then under the synthetic corruptions:

- Additive Gaussian noise at 20 dB, 10 dB, and 5 dB SNR
- Gain variation at -6 dB and -12 dB

Due to computational constraints, the robustness tests were conducted on cropped test sequences rather than full length recordings. Because of the law of large numbers, the resulting evaluation should still be representative of the data as a whole. This cropping strategy was used over all corruptions, ensuring a fair comparison.

Frame-level F1 is reported for all models. For the CRNN models, note-level evaluations are also included.

4.4.2 Results and Interpretation

For all the different architectures, the performance decreased when the corruption strength increased. When moderate noise was applied, the performance decreased by approximentaly 7–15% compared to the clean evaluation.

Table 4. Robustness evaluation on MAESTRO test set. For CRNN models, onset and offset F1 are reported in addition to frame-level F1.

	STFT		Mel		CQT	
Corruption	CNN	CRNN	CNN	CRNN	CNN	CRNN
Clean	0.713	0.703 / 0.839 / 0.222	0.716	0.731 / 0.838 / 0.230	0.728	0.729 / 0.846 / 0.241
Noise 20dB	0.639	0.629 / 0.762 / 0.195	0.611	0.665 / 0.778 / 0.196	0.676	0.672 / 0.792 / 0.218
Noise 10dB	0.522	0.579 / 0.658 / 0.166	0.498	0.514 / 0.660 / 0.150	0.547	0.555 / 0.629 / 0.155
Noise 5dB	0.436	0.526 / 0.584 / 0.146	0.412	0.381 / 0.557 / 0.137	0.451	0.435 / 0.486 / 0.115
Gain -6dB	0.691	0.693 / 0.835 / 0.224	0.674	0.712 / 0.817 / 0.228	0.635	0.644 / 0.763 / 0.193
Gain -12dB	0.659	0.665 / 0.819 / 0.210	0.628	0.668 / 0.783 / 0.202	0.394	0.425 / 0.566 / 0.121

At 10 dB SNR, degradation reached 25–30% for most models, and at 5 dB, performance dropped by 38–48%, indicating substantial sensitivity to severe noise.

The effect of the temporal modeling of CRNN was mixed, and varied across representations and corruptions. For STFT however the CRNN consistantly outperformed the CNN when noise was added, showing improved robustness. For Mel and CQT the CRNN had a positive impact on a moderate noise level, but also seems to have a negative impact on more severe noise levels.

For gain variation -6 dB the impact was smaller compared to the noise corruptions. In contrast to the noise setting, CRNN models were consistently more robust than CNN under gain shifts across all representations.

For the CRNN models onset F1 remained high on clean data (0.838–0.846) but decreased with corruption strength. At 20 dB SNR onset F1 remained above 0.76 for all representations, but dropped more on lower SNR levels. The offset F1-score was consistently low during clean runs as well, and got worse when exposed to all corruptions.

Overall, temporal modeling improved robustness, most clearly for STFT under noise, as well as all representations under gain. On higher noise levels on the other representations, the impact of temporal modeling was minimal.

Temporal modeling can help stabilize predictions by incorporating context across frames, which appears particularly beneficial for STFT under noise. At very low SNR, however, the spectral structure itself becomes unreliable, limiting the benefit of recurrence and leading to similar degradation for both architectures.

4.5. Cross-Dataset Generalization (MAPS)

Models trained on MAESTRO were evaluated directly on the MAPS dataset without fine-tuning to assess cross-dataset generalization. Table 5 summarizes the results. All models has a performance drop when exposed to the different conditions of the MAPS dataset, however this performence drop varies a lot depending on architecture and representation. CRNN models has generally a smaller degradation compared to the CNN models. The CNN based STFT model has the largest drop, showing that STFT is less robust to different conditions. Both CQT and Mel shows good robustness to the changes, while STFT generally both for CNN and CRNN performs worse. Offset F1 remains substantially lower across all models, consistent with earlier observations on temporal boundary estimation difficulty. These results indicate that temporal modeling improves robustness to dataset shift.

4.6. Overall Comparative Analysis

This section explores the reason performance differs between different architectural choices over different forms of evaluation.

Effect of Time–Frequency Representation. Seen to both CNN and CRNN CQT achieved the very high frame level F1 score, achieving the highest for CNN and only slightly underperforming Mel for CRNN. Mel followed closely, while STFT generally performed slightly lower. CQTs success most likely stems from its logarithmic frequency spacing which aligns well with musical pitch structure and harmonic organization for the piano signals. This could make it easier to learn pitch-dependant patterns which is more difficult to learn with linear frequency resolution for STFT, or perceptual compression for Mel.

Robustness experiments however shows that CQT was more sensitive to strong gain attenuation (-12 dB). Especially the CNN model. This implies that CQT captures pitch structures well under clean conditions, but might rely to much on magnitude distribution.

Effect of Network Depth and Residual Connections. Inreasing the CNN depth had had a positive impact on all representations, with the impact saturating beyond 4 convolutional blocks. This is because a deeper network increases representational capacity. However residual connections did not provide any measurable difference for this setting. Given that a moderate network depth was used, optimization might already be stable enough.

Impact of Temporal Modeling. When reccurent layers were introduced to the model, the performance of Mel increased for the frame level F1-score, while the results for CQT stayed very similar. But the clear benefit of temporal modellling was shown for the note-level metrics where

Table 5. Cross-dataset evaluation on MAPS. MAESTRO test F1 is shown for comparison. Gap denotes relative F1 drop from MAESTRO to MAPS.

Model	MAESTRO F1	MAPS F1	Gap (%)	MAPS Onset F1	MAPS Offset F1
CNN-Mel	0.720	0.646	10.3	–	–
CRNN-Mel	0.734	0.683	6.9	0.791	0.298
CNN-CQT	0.733	0.669	8.7	–	–
CRNN-CQT	0.730	0.686	6.1	0.793	0.328
CNN-STFT	0.705	0.571	19.0	–	–
CRNN-STFT	0.701	0.642	8.4	0.764	0.273

several configuration achieved a 0.83 note-level F1-score. However, adding a second LSTM layer consistently reduced performance, suggesting that deeper recurrent stacks introduce optimization difficulty or overfitting without proportional gains. So for this task, jst one temporal layer seems sufficient to capture important information. The offset-F1 score was consistently low, likely due to the gradual sustained decay of piano notes.

Robustness and General Behavior. All models performance was degraded when exposed to corruptions, where severe noise had the biggest negative impact. The CRNN architecture made STFT more robust to noise corruptions, and also had a positive impact on all representations when exposed to gain modifications. These observations suggest that temporal modeling can stabilize predictions when spectral structure remains partially reliable, but cannot compensate when noise severely distorts the time–frequency representation.

Failure Cases. Despite the fact that models generally performs well, several failure points were observed. Offset estimation remains particularly challenging, as reflected by consistently lower offset F1 scores compared to onset metrics. By manual inspection, many models seem to struggle with repeating notes in clusters. Under more severe noise levels, models tend to more regularly miss note onsets as well as generally keeping a lower frame level F1 score.

5. Consclusion

This study has investigated the impact of different time-frequency representations as well as different architectural design choices for polyphonic piano transcription. Both CNN and CRNN architectures were investigated with Mel, STFT and CQT representations on the MAESTRO dataset. A robustness evaluation was also conducted with synthetic corruptions and cross-dataset evaluation.

The results shows that CQT in most cases performs best on clean data, suggesting that the logarithmic frequency spacing used aligns well with musical pitch structure. Increasing CNN depth improved performance up to four to five convolutional blocks, after which gains saturated. At the same time, residual connections did not seem to have a big impact on the performance in this setting.

By introducing a bidirectional LSTM Mel representations had modest improvements at fame-level, while other representations did not show any major performance improvements. The biggest gain from using temporal modeling however, was the note-level onset detection, which yielded high results for most models. However, deeper recurrent stacks did not further improve performance, and offset prediction remained challenging across all models.

The robustness experemints that was conducted shows that performance degrades rapidly when models are exposed to noise or gain variations. The recurrent layers seems to have a positive impact on robustness to gain variations, but a much smaller impact on robustness to noise variations, especially when severe noise is used. Cross-dataset evaluation shows that CQT and Mel both performs well under different conditions, while STFT generally adapts less efficient. Temporal modelling also has a big positive performance impact when it comes to cross-dataset generalization.

Overall, the results highlight the importance of representation choice and moderate temporal modeling for automatic music transcription. Future work could explore improved offset modeling, different architectural choices, and controlled data augmentation techniques to further enhance robustness.

References

[1] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.

[2] Valentin Emiya, Roland Badeau, and Bertrand David. Maps – a piano database for multipitch estimation and automatic transcription of music. *Research Report*, 2010. Dataset: https://adasp.telecom-paris.fr/resources/2010-07-08-maps-database/.

[3] Curtis Hawthorne, Erich Elsen, Jialin Song, Adam Roberts, Ian Simon, Colin Raffel, Jesse Engel, Sageev Oore, and Douglas Eck. Onsets and frames: Dual-objective piano transcription. In *Proceedings of the 19th International Society for Music Information Retrieval Conference (ISMIR)*, 2018.

[4] Curtis Hawthorne, Andriy Stasyuk, Adam Roberts, Ian Simon, Cheng-Zhi Anna Huang, Sander Dieleman, Erich Elsen, Jesse Engel, and Douglas Eck. Enabling factorized piano music modeling and generation with the maestro dataset. In *International Conference on Learning Representations (ICLR)*, 2019. Dataset: https://magenta.tensorflow.org/datasets/maestro.

[5] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8), 1997.

[6] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *7th International Conference on Learning Representations (ICLR)*, 2019.

[7] Colin Raffel, Brian McFee, Eric J. Humphrey, Justin Salamon, Oriol Nieto, Dawen Liang, and Daniel P. W. Ellis. mir_eval: A transparent implementation of common mir metrics. In *Proceedings of the 15th International Society for Music Information Retrieval Conference (ISMIR)*, 2014.