4. Music tagging models

This section focuses on exploring the architectures and results of the various prominent models in state of the art music tagging. Each models' architecture and strengths will be overviewed, as well as its performance on the dataset used in the relevant papers' experiments. A last subsection will be reserved for model comparison on a standardized split as seen in [51].

Papers from literature use three input representation lengths : 3s, 15s, and 30s. Though input preprocessing and length are standardized in our case, preprocessing steps vary slightly from paper to paper, so these preprocessing steps and hyperparameters will also be considered in this section.

4.1. Fully Convolutional Network

This architecture was implemented in [6], and was the first use of convolutional neural networks for end-to-end music automatic tagging. Each convolution layer of size $H \times W \times D$ learns D features of $H \times W$, where H and W refer to the height and the width of the learned kernels respectively. The kernel size determines the maximum size of a component it can precisely capture.

Base architecture

The general stucture of the architecture proposed in [6] is a stack of convolutional kernels with a dense head which is referred to as the classification head. The convolutional front-end acts as a feature extractor while the dense fully-connected back-end acts as a discriminator head. [6] proposes 5 variants of the base network (**FCN** - fully convolutional network): FCN 3-4-5-6-7 referring to the number of convolutional blocks in the front-end of the network.

Each convolutional block is comprised of a convolutional layer, a batch Normalization layer and a Max Pooling layer to allow for feature size reduction throughout the process. based on performance on the MagnaTagATune dataset ([19]), FCN-4 is selected as the final model. The architecture for this model is shown below (Figure [**FCN**]):

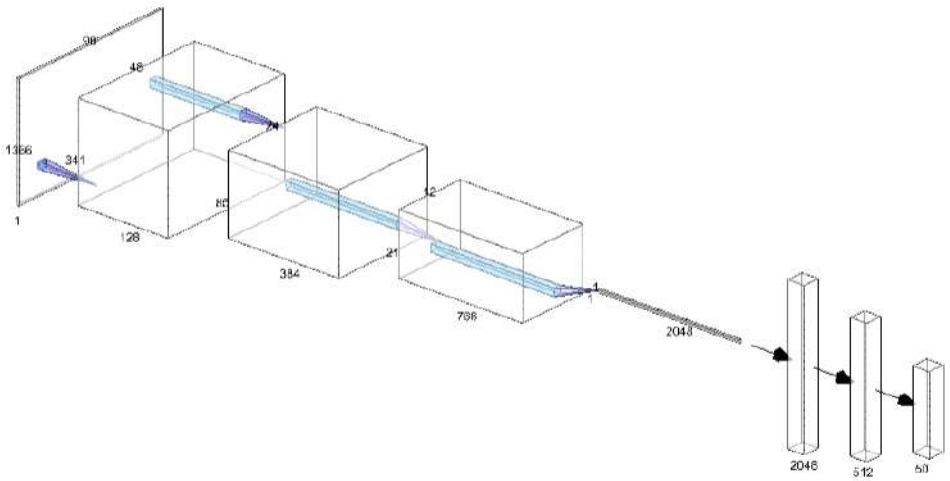


Figure 4.1.: Architecture of the FCN model

Preprocessing

Multiple experiments were run in [6], including STFTs, MFCCs and Mel-Spectrograms:

> "A pilot experiment demonstrated similar performances with 12 and 16 kHz sampling rates and melbins of 96 and 128 respectively. As a result, the audio in both datasets was trimmed as 29.1s clips (the shortest signal in the dataset) and was downsampled to 12 kHz. The hop size was fixed at 256 samples (21 ms) during timefrequency transformation, yielding 1,366 frames in total. STFT was performed using 256-point FFT while the number of mel-bands was set as 96. For each frame, 30 MFCCs and their first and second derivatives were computed and concatenated."[6]

This yields an input shape of 1366×96 for input mel spectrograms

Performance

The models for this paper were evaluated against previous works using the MagnatagATune dataset, against different input representations on the magnatagatune dataset, and against all the k-variants of the different FCN-k architectures on the million-song datasets. The obtained results are shown in the tables below 4.2:

Methods	ROC-AUC
The proposed system, FCN-4	**0.894**
2015, Bag of features and RBM	0.888
2014, 1D convolutions	0.882
2014, Transferred learning	0.88
2012, Multi-scale approach	0.898
2011, Pooling MFCC	0.861

(a) Results of [6] compared to past work

Methods	Representation	ROC-AUC
FCN-3,	mel-spectrogram	.852
FCN-4,	mel-spectrogram	**.894**
FCN-5,	mel-spectrogram	.890
FCN-4,	STFT	.846
FCN-4,	MFCC	.862

(b) results of [6] varying representation and model

Methods	Representation	ROC-AUC
FCN-3,	mel-spectrogram	.786
FCN-4,	—	.808
FCN-5,	—	.848
FCN-6,	—	**.851**
FCN-7,	—	.845

(c) Results of [6] on the million song dataset

Table 4.1.: Results from [6]

4.2. CRNN

[8] Introduces A Convolutional Recurrent Neural Network (CRNN) for audio music tagging with a convolutional front-end acting as a feature extractor and a Recurrent Neural Network acting as a temporal feature aggregator. The intuition here is that since music is an inherently sequential media (a sequence of chords, notes, lyrics, structures, etc.) It is beneficial to include a temporal learning structure to the successful convolutional one.

Base Architecture

[8] Proposes a structure with the same convolutional front-end as [6], with less temporal pooling to keep a set of temporal features at the end of the front-end. A two-layer recurrent neural network (RNN) is then stacked on top with 32 units per layer to extract temporal features (As shown in figure 4.2) [1]

[1] The two RNN layers are represented as Flat blocks along the temporal axis in the above figure

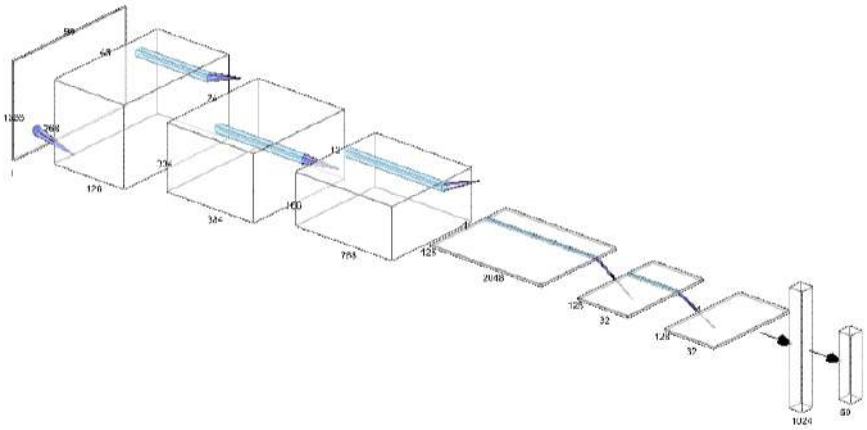


Figure 4.2.: Architecture of the CRNN model

Results

Input representations were only log-Mel-Spectrograms as their effectiveness has been demonstrated in [6]. The preprocessing hyperparameters are kept the same from the previous study. CRNN is compared to 3 other models (k2c1, k1c2, k2c2), which respectively use time-convolution filters, frequency-convolution filters, and rectangular filters (shown in figure 4.3):

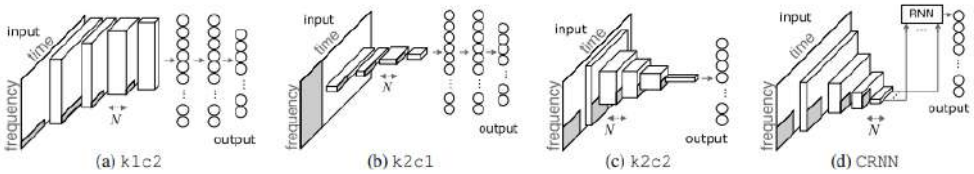


Figure 4.3.: Models implemented in [8]

Results are evaluated on the million-song dataset for all 4 models and compared to SOTA (state of the art). All models are also evaluated using different parameters scalings (less filter depth) and compared (Shown in figure 4.4):

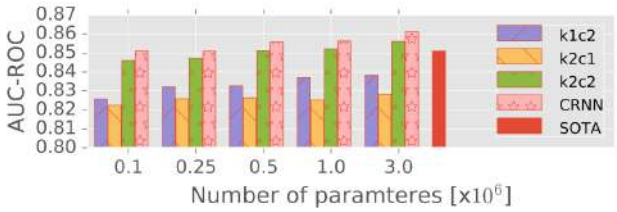


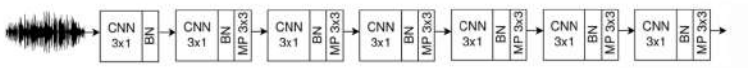
Figure 4.4.: Results from [8]

CRNN effectively beats SOTA on music tagging tasks and represents the last used model with an input length of 30s in literature.

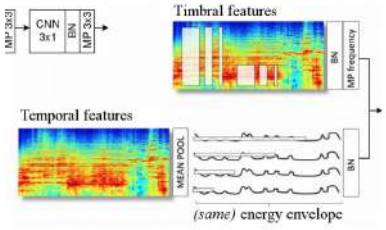
4.3. MusiCNN

Architecture

MusiCNN, implemented in [32], aims to take advantage of domain knowledge to improve music tagging performance. MusiCNN is adaptable to two input representations : Melgrams and raw audio. To achieve this, it is comprised of a common back-end for both architectures and a convolutional architecture front-end : in 1D for raw audio and 2D as previously for 2D melgrams. These two front-ends can be seen in the following figures (**??**)



(a) Front end for pure audio in [32]



(b) Front end for melgrams in [32]

The front-end for raw audio is a series of 1D convolutional filters used to extract relevant features from pure audio. The front-end for Spectrograms, on the other hand, takes from domain knowledge of musical analysis that music is a combination of temporal and timbral features [32], and thus is designed with this in mind as two branches concatenated, one a series of vertical (timbral) filters and the other a series of horizontal (temporal) features of varying lengths (to capture features at different scales). It is shown that having multiple filter scales is more beneficial than having one. In both representation architectures, the input from the front-end is fed into a common back-end, shown in figure 4.6

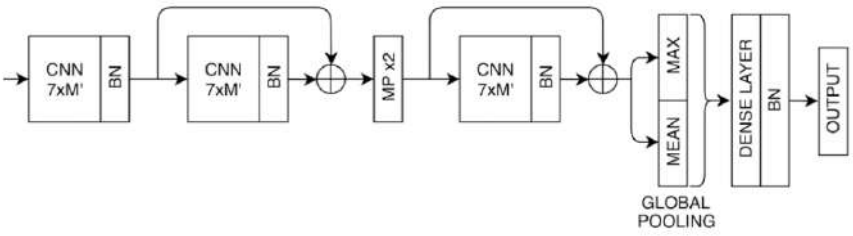


Figure 4.6.: MusicNN backend [32]

Results

Input representations are taken as chunk-level samples of songs (3s input length). The audio is resampled to 16kHz and the parameters for melgram computing is 512 fft bins with a hop value of 256.

Multiple experiments are run for this paper, including:

- experiments on a proprietary 1.2M song dataset on which the raw audio model performs better than the spectrogram one (table 4.2a)
- experiments on the million song dataset comparing to previous work including previously mentioned models here (4.2b)
- results on the MTAT dataset [19] compared to previous work, where Musicnn comes close to previous results with high reduction in the size of the model (4.2c)

MusicNN demonstrates that domain knowledge in both music and signal processing is a valuable insight when conceptualizing model architectures for audio tasks as it can enhance performance while greatly reducing model size. It also demonstrates that using 128 mel bins instead of 96 is preferable for performance.

1.2M-songs	trained on	ROC-AUC	PR-AUC
Baseline	1.2M	91.61%	54.27%
Waveform	1M	**92.50%**	**61.20%**
Spectrogram	1M	92.17%	59.92%
Waveform	500k	91.16%	56.42%
Spectrogram	500k	91.61%	58.18%
Waveform	100k	90.27%	52.76%
Spectrogram	100k	90.14%	52.67%

(a) Results of [32] on their proprietary dataset

MSD	ROC	PR	#params
Models			
Waveform	87.41	28.53	5.3M
SampleCNN [22]	88.12	-	2.4M
SampleCNN multi level and scale [21]	88.42	-	-
Spectrogram (ours)	88.75	31.2	5.9M
CRNN [8]	86.2	-	3M
Multi-level and multiscale [25]	88.78	3M	-

(b) Results of [32] compared to past work on the million song datasset

MTAT	ROC	PR	#
Waveform	AUC	AUC	param
SampleCNN [22]	90.55	-	2.4M
SampleCNN (128mels) [22]	88.56	34.38	2.4M
MusiCNN raw audio	84.87	-	-
MusiCNN raw audio (128mels)	85.58	29.59	194k

MTAT	ROC	PR	#
Spectrogram	AUC	AUC	param
Fully convolutional [6]	89.4	-	22M
Fully convolutional (128mels) [6]	89.99	37.6	450k
MusiCNN	89.30	-	191k
MusiCNN (128mels)	89.07	34.92	220k

(c) results of [32] on the MTAT dataset

Table 4.2.: Results from [22]

4.4. Sample-level CNN

SampleCNN [22] and SampleCNN+Squeeze [17] propose a novel approach to music tagging by 1) considering raw audio waveforms as their input (requires no preprocessing), and 2) using strided convolutional filters with very high granularity to decompose audio signals at into high frequency features. This allows for learning low-dimensional filters to apply onto the audio signal to extract relevant features.

4.4.1. Base model

Architecture

The base model for sampleCNN is a 1-D convolutional neural network using chunk-level audio lengths (2.6s) as an input but using frame-level convolutional filters (of size 3) on the audio signal. In the terminology of this paper, a frame is comprised of 3 samples. The architecture for samplecnn is shown in figure 4.7 [22]:

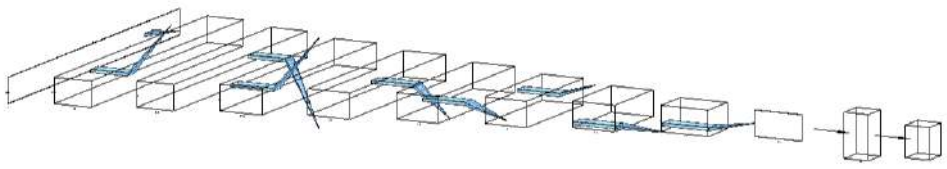


Figure 4.7.: Samplecnn architecture

Sample CNN is compared to an equivalent 2D network at the sample level as well as previous work with frame-level convolutional filters ([8], [6]), and another proposed 2D CNN with low-level convolutional filters (241 samples of filter width). It is evaluated on the million song dataset and the MTAT dataset, as previous works were:

input type	model	MTAT	MSD
Frame-level	Persistent CNN	0.9013	-
(mel-spectrogram)	2D CNN [6]	0.894	0.851
	CRNN [8]	-	0.862
	Proposed DCNN	0.9059	-
Frame-level waveform	1D CNN	0.8487	-
Sample-level waveform	Proposed DCNN	0.9055	0.8812

Table 4.3.: Results from [22]

4.4.2. Adding Squeeze and Excitation

[17] builds upon [22] by taking inspiration from successful 2D convnets (specifically SENets and ResNets) and implement residual connection blocks and squeeze-excitation blocks. Furthermore, it uses multi-level feature aggregation before the dense classification head to ensure the discriminatory dense head sees features from multiple scale levels. This paper focuses only on frame-level raw audio waveforms. The added modifications to the architecture shown in [22] are shown in figure [**samplecnn2**]

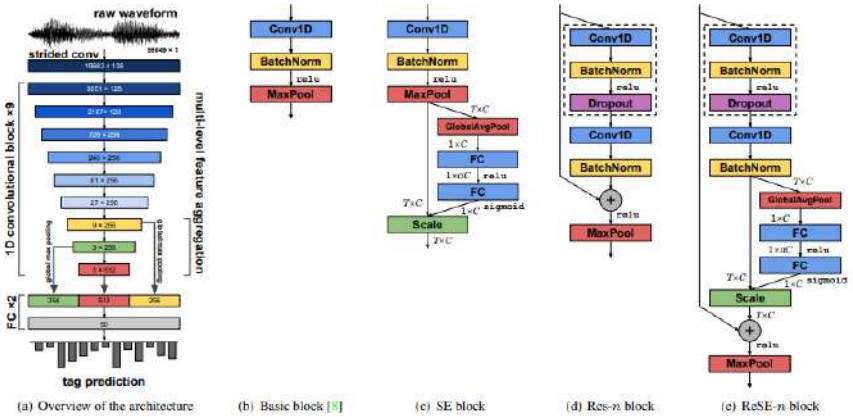


Figure 4.8.: Addition of residual blocks and Squeeze-Excitation blocks to samplecnn [17]

This additional leads to substantial gains in performance compared to previous work and the previous samplecnn on both the MTAT dataset and the Million song dataset (4.4):

Model	MTAT	MSD
Timbre CNN [32]	0.8930	-
2D CNN [6]	0.8940	0.8510
CRNN [8]	-	0.8620
Multi-level & multi-scale [20]	0.9017	0.8878†
SampleCNN multi-features [21]	0.9064	0.8842
SampleCNN [22]	0.9055	0.8812
SE [This work]	0.9111	0.8840
ReSE [This work]	0.9113	0.8847

Table 4.4.: Results for SampleCNN-ReSE [17]

4.5. HarmoniCNN

Intuition and architecture

Similar to [32], [49] aims to incorporate domain knowledge into the very architecture of their proposed model. The harmonic constant-Q transform (HCQT) is a 3-dimensional representation whose dimensions are harmonic(H), frequency (F), and time (T). By stacking standard constant-Q transform (CQT) representations, one harmonic at a time, the output representation (i.e., HCQT) can preserve the harmonic structure while having spectro-temporal locality

In previous work, two learnable sinc functions (i.e., $sin(x)/x$) were used to form each band-pass filter of the first convolutional layer, such that the set of harmonics can be learned. By aligning the convolution band-pass filters in each harmonic, the first layer outputs an H × F × T tensor. When the first harmonic center frequencies are initialized with a MIDI scale, this can be interpreted as an extended, more flexible version of HCQT.

However, the convolution band-pass filter approach to get harmonic spectro-temporal representations requires many convolutions (H × F), including redundant ones (e.g., a 440Hz filter is equivalent to the second harmonic filter of 220Hz). To overcome these efficiency limitations, in this work they replace the convolution band-pass filters of our previous work with an STFT module followed by learnable triangular filters, the so-called Harmonic filters (See figure 4.9).

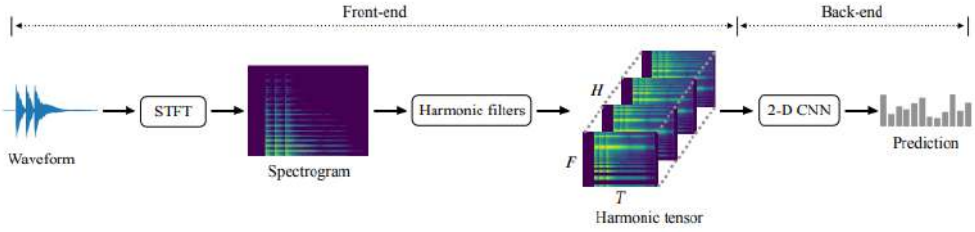


Figure 4.9.: Structure of the harmonicnn network [49]

Where a harmonic filter can be formulated as:

$$\Lambda_n(f; f_c, \alpha, \beta, Q) = \left[1 - \frac{Q|f - nf_c|}{(n\alpha f_c)}\right]_+ \tag{4.1}$$

Where $[]_+$ denotes a rectified linear unit function, and α β, Q are learnable model parameters. The proposed learnable filter banks is the set of harmonic filters [49]:

$$\{\Lambda_n(f; f_c)|n = 1, ..., H, fc \in \{f_c^{\{}(1)\}, ..., f\{(F)\}_c\}\} \tag{4.2}$$

Center frequencies of all the harmonic filters are also learnable paramaeters. This approach differs from previous approaches in that it proposes a F-channel input to the 2D CNN backend instead of a single channel input as before. Furthermore, the approach is novel even in classic computer vision as the channel representations are learned at the same time as the CNN backend.

Results

HarmonicNN is tested on multiple tasks : **music tagging, keyword spotting and sound event tagging**. Only the results for the first task will be presented, though it should be noted that HarmoniCNN generalizes well to in-domain tasks due to the incorporation of domain knowledge to the architecture. STFTs are generated with $n_fft = 512$. HarmoniCNN is evaluated on the MTAT dataset, with the same split as [22], [50], which is the first acknowledgment of a standardized split for inter-model performance comparison on the MTAT dataset. The results are shown in the following table (Table 4.5):

Methods	Music Tagging	
	MTAT	
	ROC-AUC	PR-AUC
Musicnn [32]	0.9089	0.4503
Sample-level [22]	0.9054	0.4422
+ SE [17]	0.9083	0.4500
+ Res +SE [17]	0.9075	0.4473
Proposed	0.9141	0.4646

Table 4.5.: results from [49]

4.6. Short-chunk CNN

4.6.1. Two architectures

It is intuited in [49] and shown in [51] that a simple VGG-like CNN trained on short chunks of audio can provide state of the art results on audio music tagging tasks. Though no paper is explicitly dedicated to it, it is implemented in [51] for comparison's sake with other state of the art models.

The architecture for this model is quite similar to that seen in [6] and in figure 4.1 : a series of 2D convolutional layers deepen the input representation while squeezing it horizontally andvertically. However, it differs in that[6] is trained on 29.1 long second audio (song-level evaluation), which means that the time-wise max pooling dimension reduction steps are much more reductive. This is due to the need of the model to have a large receptive field to capture features on all 29.1s. The **Short-Chunk CNN** - as is dubbed this architecture - however, trains on 3.69s long audio excerpts, which greatly restricts the necessary receptive field of the model and allows for smaller max-pooling size (2 versus 4 in [6].)

An additional step to increasing the already state of the art performance of the Short-Chunk model is adding residual skip-connections. These connections, similar to the **Res block** in [17], allow for the flow of original information from one convolutional block instead of only residuals. This alleviates the gradient vanishing problem often found in deep CNNs by shortening the path between information in the network, and shows better results as shown in [51][2]

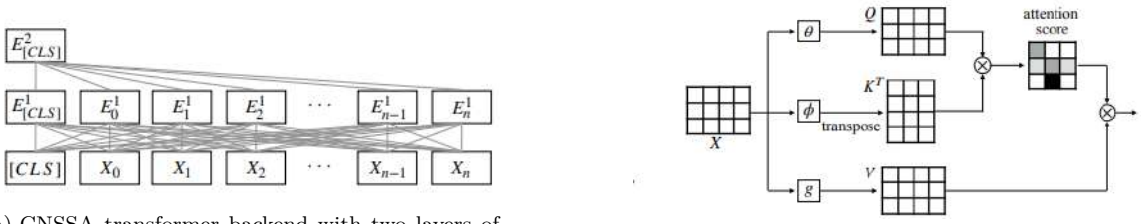
4.7. Convolutional front-end with self-attention

[50] introduces the transformer model for the first time in music auto tagging. Transformers are a class of ML models that rely only on the self-attention mechanism, which is a mechanism allowing them to learn the relationships between features of input representations. This has proven invaluable in fields such as Natural Language Processing, where transformer models have revolutionized the landscape of the domain. Transformer models are highly skilled at dealing with sequential data, and at interpretability. [50]'s goal is to build upon these innate capabilities by introducing a transformer encoder back-end as a temporal feature aggregator for a convolutional front-end for music tagging. The intuition resembles that of [8] in that a temporal feature aggregator on top of a proven feature extractor is a good idea for an inherently sequential media. It differs mostly in two ways :

- The feature aggregator is, as discussed, no longer a RNN but a transformer. This, as shown in literature [44] allows for faster computation times within the model due to parallelization and shorter paths between inputs in the model, leading to less gradient vanishing (important for sequences of long lengths)
- The length of the audio used for training in [50] is between 5 and 15s, with [51] citing it as 15s, which represents a lower length than the 29.1s of [8]

The self-attention back-end as well as a schematic of the self-attention mechanism (which can also be found in [44]) are shown in the following figure (4.10):

[2]As this model was only implemented in a paper that recaps most known music tagging models to date and their performance on one same standardized dataset, these results will be presented in section 4.12 along with those of the paper which are of most interest, the comparison results.



(a) CNSSA transformer backend with two layers of self-attention encoders [50]

(b) Self-attention mechanism in CNSSA [50]

Figure 4.10.: Architecture proposed in [50]

Results

[50] implements two front-ends to go with the novel back-end, musicnn's frontend (2D) [32] and samplecnn's front-end (1D) [22], and compares results with their respective back-ends and the transformer back-end on MTAT and Million song:

		MTAT		MSD	
Front-end	Back-end	AUROC	AUPR	AUROC	AUPR
Raw	CNNL	90.62	44.20	88.42	-
Raw	Att (Ours)	90.66	44.21	88.07	29.90
Spec	CNNP	90.89	45.03	88.75	31.24
Spec	Att (Ours)	**90.80**	**44.39**	**88.14**	**30.47**

Table 4.6.: results for CNSSA [50]

4.7.1. Semi-supervised music tagging transformer

Another paper which elaborates on CSSA is [48], implementing a semi-supervised learning version of [50]. The back-end architecture doesn't change, the front end is swapped for a 7-layer short-chunk CNN. Melgrams are extracted using 1024 n_fft and 128 mel bins for 15s audio clips resampled to 22.5kHz.

[48] Uses noisy student training to use the full 1.2M songs of the million-song dataset. Put simply, noisy student training is explianed in [48]:

> First, we train a teacher model T with a conventional supervised learning approach. Then, we train a student model S with two types of losses. The first loss, L_1, is coming from the typical supervised approach with labeled data as done for the teacher model. The other loss, l2, is from unlabeled inputs and the corresponding pseudo-labels provided by the teacher model . In order to make the student model perform beyond mimicking the teacher model, data augmentation is applied.

Through knowledge distillation (i.e when the student model is smaller than the teacher model), [48] reaches state of the art performance on a new proposed split on the million-song dataset with 9 times less parameters than the largest previous model. The results on the conventional split are shown in the following table (4.7):

Models	ROC-AUC	PR-AUC
FCN	0.8742	0.2963
Musicnn	0.8788	0.3036
Sample-level	0.8789	0.2959
Sample-level+SE	0.8838	0.3109
CRNN	0.8460	0.2330
CNNSA	0.8810	0.3103
Harmonic CNN	0.8898	0.3298
Short-chunk CNN	0.8883	0.3251
Short-chunk ResNet	0.8898	0.3280
Transformer (proposed)	0.8916	0.3358
Transformer (proposed) + DA	**0.8972**	**0.3479**

Table 4.7.: Results for the semi-supervised CNSSA [50]