Few-shot Learning for Joint Classification of Instrument, Pitch, and Playing Technique of Tones Produced by Bowed String Instruments

by

Pieter Cornelius Kok

Submitted in partial fulfillment of the requirements for the degree

Master of Engineering (Computer Engineering)

in the

Department of Electrical, Electronic and Computer Engineering

Faculty of Engineering, Built Environment and Information Technology

UNIVERSITY OF PRETORIA

December 2023

SUMMARY

Few-shot Learning for Joint Classification of Instrument, Pitch, and Playing Technique of Tones Produced by Bowed String Instruments

by

Pieter Cornelius Kok

Supervisor(s):	Prof. J. Pieter Jacobs
Department:	Electrical, Electronic and Computer Engineering
University:	University of Pretoria
Degree:	Master of Engineering (Computer Engineering)
Keywords:	Instrument playing technique classification, pitch classification, instrument recognition, few-shot learning, prototypical network, Bayesian optimization, transfer learning.

Instrument playing technique classification is a problem in music information retrieval (MIR) that has only selectively been explored in the context of specific instrumentations or datasets. Classifying playing techniques with pitch is a further challenge that takes a step closer to automatic music transcription (AMT) with playing technique annotation. Traditional deep learning methods have been used for the problems of instrument classification, playing technique classification and multiple-instrument transcription, however, annotated data for the combined problems are scarce, thus it is hard to train a sufficiently complex deep neural network that would be able to generalize to many different instruments, playing styles and recording conditions. This study presents a few-shot learning model for joint instrument, playing technique and pitch classification of single tones using prototypical networks. The few-shot nature of the model allows it to be trained on what data are available and to adapt to new instruments, playing techniques or recording conditions at inference time from a few examples. This model could form part of a tutorial system where a music student would record scales of a given playing technique under the supervision of a music teacher, which would later be used to match and evaluate a performance with the technique.

Different deep neural network (DNN) architectures and both log-mel spectrogram and constant-Q transform (CQT) input features are compared. The few-shot models are compared to standard neural network classifier models with transfer learning to show how the few-shot models generalize better to previously unseen playing techniques. Model training is optimized with Bayesian optimization. Prototypical models outperform standard classifier models with transfer learning on all experiments.

The 3-shot CQT convolutional neural network (CNN) model performs the best on the joint classification task and achieves a macro F-score of .64 on the Studio On Line (OrchideaSOL) string instrument playing technique dataset of previously unseen playing technique classes, which shows an ability for the prototypical model to generalize to a new dataset without much loss of performance compared to evaluation on the training classes. The model also achieves a macro F-score of up to .855 on individual instruments, which shows promise for its use in a tutorial set up for any of the string instruments. The models perform just as well when evaluated on extracts from YouTube tutorials and examples of clarinet playing techniques from the Real World Computing (RWC) dataset. The few-shot model also functions as a multitask model, capable of classifying pitch, playing technique or instrument from a recorded sample. The best joint instrument, playing technique and pitch classification prototypical model can accurately classify both playing technique and pitch, and do so just as well or better than models trained more specifically on these problems when compared on the same data. Furthermore, the scenario of instrument, playing technique and pitch classification in the presence of piano accompaniment is investigated, which resulted in some loss of generalization, but still shows promise for the task of main melody extraction, as pitch classification remains high.

LIST OF ABBREVIATIONS

k-NN	*k* nearest neighbours.
ADT	automatic drum transcription.
AMT	automatic music transcription.
ANN	artificial neural network.
ARD	automatic relevance determination.
CCE	categorical cross-entropy.
CNN	convolutional neural network.
CQT	constant-Q transform.
CRNN	convolutional recurrent neural network.
DNN	deep neural network.
F0	fundamental frequency.
FFNN	feed forward neural network.
FFT	fast Fourier transform.
GP	Gaussian process.
GRU	gated recurrent unit.
HMM	hidden Markov model.
MFCC	mel frequency cepstral coefficient.
MIR	music information retrieval.
MPE	multi-pitch estimator.
NMF	non-negative matrix factorization.
OrchideaSOL	Studio On Line.
PLCA	probabilistic latent component analysis.
PYIN	probabilistic YIN.
RNN	recurrent neural network.
RWC	Real World Computing.

STFT	short-time Fourier transform.
SVM	support vector machine.

TABLE OF CONTENTS

CHAPTER 1 INTRODUCTION

1.1 PROBLEM STATEMENT

1.1.1 Context of the problem

Automatic music transcription (AMT) is a prominent research problem in the larger field of music information retrieval (MIR). The goal of AMT is to convert an audio representation of music into a probable transcription of music events in the piece, such as the notes played, instruments present in the piece, or instrument playing techniques employed. AMT has many applications in music processing; it is useful for replicating an improvised performance but is also useful as a preprocessing step for further music processing [1].

Playing techniques are of interest in MIR as secondary transcription labels. The techniques used to play a piece of music or phrases in a piece can change the tone and mood of the music and is an important part of an expressive musical performance. Playing technique transcription would thus be an important part of a complete general purpose AMT system in order to capture the complete information from a recording in order to be able to accurately replicate the performance. Playing technique classification is also very useful in teaching tools in order to evaluate a student's mastery of different playing techniques.

Traditional deep learning methods have been used for the problems of instrument classification, playing technique classification and multiple-instrument transcription, however, annotated data for the combined problems are scarce [1], thus it is hard to train a sufficiently complex deep neural network that would be able to generalize to many different instruments, playing styles and recording conditions [2]. Few-shot learning is a machine learning task whereby a model classifies new classes or performs a new function after training is complete by being provided examples at inference time [3]. Few-shot learning allows models to generalize to examples that are different from those present in the training data and are thus a way to train a model without having large amounts of training data for all target tasks. There only needs to be enough data to train a model that is able to distinguish features necessary for the target tasks.

This study presents a few-shot learning model for joint instrument, playing technique and pitch classification using prototypical networks. The few-shot nature of the model allows it to be trained on what data are available and to adapt to new instruments, playing techniques or recording conditions at inference time from a few examples.

1.1.2 Research gap

Playing technique classification is a relatively unexplored problem in MIR [4]. Many investigations focus on a single instrument or family of instruments, such as guitar [5, 6, 7, 8, 9, 10, 11, 12], violin [13, 14, 15], piano [16, 17], drums [18, 19, 20], singing techniques [21, 22, 23], bagpipe [24], guqin [25], guzheng [26] or Chinese bamboo flute [27, 28]. Full transcription systems that account for both pitch and playing technique are desirable for e.g. tutorial purposes; yet there seems to be a paucity of such studies. Examples include [13, 6]. Hand-crafted pitch tracking and technique detection rules and features limit these approaches to the instruments and techniques they were designed for. Methods tuned to the spectral envelope of a violin, for example, would not be readily applicable to clarinet. Thus there is a research gap for algorithms that can identify both pitch and playing technique. A general-purpose transcription system would also be able to identify the instrument in question.

While prototypical networks [3] have been applied to sound event detection [29, 30], musical instrument recognition [31] and automatic drum transcription (ADT) [32], applying them to playing technique classification in combination with instrument and pitch classification remains unexplored.

1.2 RESEARCH OBJECTIVE AND QUESTIONS

The objective of this research is to establish whether few-shot learning is suitable for the joint classification of instrument, pitch and playing technique for string instrument tones and whether few-shot leaning performs better than standard neural network classifiers on this task. To this end prototypical few-shot models are trained for this task, as well as standard classifier models for the same problem. Models are compared to show the strengths and shortcomings of the approaches to this problem. The following research questions are considered:

- Will a few-shot model jointly classify instrument, pitch and playing technique for string instrument tones better than a standard classifier model?
- Does a few-shot model trained on the joint classification task perform better than a standard classifier model that was trained on the joint classification task when evaluated on the tasks of playing technique classification, pitch classification or instrument recognition?
- Does a few-shot model generalize better to previously unseen data, problems, instruments, performers or recording conditions than a standard classifier model which has been further trained in the new domain?

Some secondary research questions that are also of interest are: how well do the models perform when given more realistic data than studio recordings meant for research, such as string instrument examples with accompaniment or recordings from YouTube that were not necessarily recorded under studio conditions; and how do the models perform on the sub-problems of instrument recognition, playing technique classification or pitch classification, for which state-of-the-art techniques already exist that perform relatively well?

1.3 HYPOTHESIS AND APPROACH

Since the approach should ideally be of use towards a general-purpose transcription system, this is looked at as a few-shot learning problem where the model needs to be able to adapt to new instruments and playing techniques from a handful of examples, rather than needing hand-crafted features for any new instruments and techniques. Prototypical networks have shown promise in few-shot learning for sound event detection [33], thus this approach is adapted to this problem. These few-shot models are compared to standard neural network classifiers that employ transfer learning to adapt to new datasets. The hypothesis is that prototypical few-shot models would perform better than standard classifier models when generalizing to previously unseen data.

Models are trained and evaluated on string instrument examples of isolated notes or tones from the Real World Computing (RWC) [34] dataset. In order to demonstrate the versatility of the system, models are also tested on examples with accompaniment added and examples from YouTube that were recorded under more realistic conditions (what would be expected from a tutorial setting). Cross-instrument evaluation is also done using RWC clarinet playing technique examples and the Studio On Line (OrchideaSOL) [35] dataset string examples are used for cross-dataset evaluation to evaluate the few-shot nature of the approach.

1.4 RESEARCH CONTRIBUTION

This research contributes prototypical few-shot models, as well as standard artificial neural network (ANN) classifier models for joint instrument, pitch and playing technique classification of string instrument tones. A conference paper was submitted and presented at Southern Africa Telecommunication Networks and Applications Conference (SATNAC) 2021 — a conference concerning progress achieved in applied research in the information and communications technology sector in South Africa — that detailed multiple instrument pitch transcription using ANNs [36]. A journal paper comparing the prototypical few-shot models to standard ANN classifier models on the problems of joint instrument, pitch and playing technique classification, instrument playing technique classification, and instrument pitch classification is to be submitted to the IEEE/ACM Transactions on Audio, Speech, and Language Processing. This paper also discusses the performance of prototypical networks compared to transfer learning on adapting to previously unseen data, problems, instruments, performers and recording conditions.

1.5 OVERVIEW OF STUDY

Chapter 2 gives an overview of the relevant literature on this subject. Chapter 3 describes the implemented systems and the theory behind them. Chapter 4 gives an overview of the models that were trained for different experiments and the datasets used for training and evaluation. The results of all experiments are given and findings are discussed in chapter 5. Chapter 6 concludes and summarizes the study.

CHAPTER 2 LITERATURE STUDY

2.1 CHAPTER OVERVIEW

This chapter summarizes the relevant literature on the problems of playing technique classification, instrument recognition and pitch transcription in automatic music transcription. An overview is given of playing techniques for string instruments. State-of-the-art approaches to the individual and combined problems of playing technique classification, instrument recognition and pitch transcription are discussed. Machine learning classifier techniques are explained and finally the approaches of few-shot learning and transfer learning, and their applications to music transcription are discussed.

2.2 STRING INSTRUMENT PLAYING TECHNIQUES

The violin family of musical instruments predominantly produces sound by means of a bow drawn (usually by the right hand) over tightly wound strings to produce vibrations on the string, which reverberates in the hollow body. The pitch of sounds produced is controlled by pressing down on strings with the fingers of the other hand while they are bowed in order to change the length of the string that is able to vibrate. The quality of the sound, i.e. the perceived timbre, can be changed through different playing techniques that could involve some combination of altering how the fingers press down on the strings on the fingerboard or different bowing or plucking methods. Some common playing techniques considered in this investigation are:

- *Vibrato*[1]: The finger pressing down on the string is moved back and forth slightly while the bow is drawn over the string, producing a modulation of the note pitch [37].
- *Non-vibrato*: Notes are not played with *vibrato*, thus the finger pressing down on the string is kept still while the bow is drawn, resulting in a sound with less frequency modulation.

[1]In this investigation this is considered the normal way of playing. Thus where not otherwise indicated for string instruments, assume examples are played with vibrato.

- *Sul ponticello*: Strings are bowed close to the bridge, which produces a glassy sound that emphasizes higher harmonics [38].
- *Sul tasto*: Strings are bowed close to or over the fingerboard, producing sound with fewer high harmonics [39].
- *Pizzicato*: The strings are plucked instead of bowed [40]. If the sound is dampened by lifting the finger off the fingerboard after playing it is called *secco*.
- *Spiccato*: The bow is bounced on the string which results in a staccato effect [41].
- *Sautillé*: A bow stroke similar to *spiccato*. The stroke is played in the middle of the bow and one stroke is used per note. The bow is bounced slightly on the string [42].
- *Martelé*: A percussive stroke produced by heavy initial pressure of the stroke, followed by an explosive release [43].
- *Collé*: A stroke in between *martillé* and *spiccato*. The stoke starts above the string and briefly touches the string before lifting again. The resultant note has a sharp onset and offset [44].
- *Tremolo*: A note is repeated in quick succession by rapidly moving the bow back and forth [45].
- *Flageolet*: Harmonics are played by lightly touching the string at the harmonic position of the pressed note or open string [46].
- *Staccato*: Audible separation between notes is the main distinctive quality of *staccato*. This can be played with a dead stop on the string or with a release. It is often achieved by playing notes shortly and sharply. Since we are interested in single notes, we focus on this as the audible quality we are detecting when considering staccato notes in datasets [47].
- *Sordino*: Sound is muted by adding a three-pronged clamp to the bridge, which absorbs some of the vibrations [48]. This can be combined with other playing techniques.
- *Glissando*: A continuous slide from one note to another. All pitches between the two notes are played with a single bow stroke while the finger on the string slides from the beginning note to the end note [49].
- *Portamento*: A similar technique to *glissando*. Before playing a note, the player pauses at a pitch just above or below the note and slides the pitch to the final note [50].

This list is not exhaustive. Note that playing techniques that incur multiple pitches, such as *glissando* and *portamento* in the RWC playing technique dataset [34] (see Section 4.2.1), are outside the scope of this investigation, as the primary interest is classifying playing technique alongside singular pitch labels.

2.3 INSTRUMENT, PLAYING TECHNIQUE AND PITCH CLASSIFICATION

Playing technique identification research in MIR has for the large part focused on specific instruments or techniques, often because features can be specifically designed for the characteristics of the instrument. A lot of investigation has been done for common instruments in Western classical music such as guitar [5, 6, 7, 8, 9, 10, 11, 12], violin [13, 14, 15] and piano [16, 17], which have very distinct harmonic and spectral profiles. Studies have also been done for percussion and drums [18, 19, 20], as well as singing techniques [21, 22] and traditional music and instruments which are not as common in MIR, such as bagpipe [24], guqin [25], guzheng [26], Chinese bamboo flute [27, 28] and makam music [51].

A limited number of studies have considered playing technique detection in conjunction with transcription, although as two separate systems based on the same features [13, 6]. In [13] pitch extraction is done from fast Fourier transform (FFT) thresholding and playing techniques are detected from the note envelope and spectrograms according to their characteristics. In [6], note onset detection with fundamental frequency (F0) tracking is done and these features, along with additional spectral envelope features are used to train a support vector machine (SVM) model for playing technique classification. This has also been applied to matching a recording to a score to find the alignment of notes and ornamentations [7].

Previous work on each of the sub-problems is summarized in the next sections. This review is not exhaustive and only serves to give an idea of the state-of-the-art in the field and to highlight approaches related to the approach taken in this study. To give an idea of the relevant difficulty of the different sub-problems: the compilers of the OrchideaSOL dataset did some benchmarking on the dataset and investigated instrument recognition, playing technique classification and note classification using some common classifier approaches. 20 mel frequency cepstral coefficients (MFCCs) were used as input features. Of all approaches, their random forest estimator performed the best, achieving 95 %, 90 % and 90 % respectively on the three problems with 16 different instruments and 143 different playing techniques in the evaluated dataset [35].

2.3.1 Playing technique classification

Playing technique classification for specific playing techniques has been done by taking advantage of knowledge about how the playing techniques change frequency spectra over time, like detecting *glissando* using a hidden Markov model (HMM) [27], *vibrato* using the frequency diagonalization method [52] or *protamento* with a HMM and Gaussian mixture model [52].

Playing technique recognition mostly relies on frequency representation features to detect playing techniques. Spectral envelope modeling can be used to differentiate between playing techniques [6, 13]. The features need to capture changes in frequency over time, thus time-frequency spectra are often employed, such as a constant-Q transform (CQT) [53, 25] or a scattering transform — a cascade of constant-Q wavelet transforms alternated with modulus operators [54, 4]. Other techniques used to model frequency changes are power spectral density [5] and cepstrums [10], especially MFCCs [55, 11].

Templates can be used to extract occurrences of events from a recording using machine learning algorithms such as dynamic time warping [24] and non-negative matrix factorization (NMF) [18]. Otherwise, classification methods are used to make predictions based on the features. Common clustering classifiers are SVMs [16, 56] or *k* nearest neighbours (*k*-NN) [15]. *k*-NN has been extended using a large margin nearest neighbors algorithm with a learned matrix that is used to add weights to the distance function. This was used for a query-by-example algorithm for instrument recognition and playing technique classification from isolated note examples [4].

Deep neural network (DNN) models have also been employed [25, 20]. More recent implementations have taken advantage of larger models and datasets. A convolutional neural network (CNN) was used to classify violin bowing techniques on a dataset consisting of a combination of data extracted from existing datasets, YouTube and new recordings [57]. A convolutional recurrent neural network (CRNN) approach to playing technique classification evaluated their models on test sets that were generated from similar data to what the models were trained on and on test sets that were generated from new data and found a significant reduction in accuracy on the new data [53]. Training on one dataset and testing on another is prone to uncover difficulty in generalization that isn’t easily detected by training and testing on data from the same set [2].

2.3.2 Instrument recognition

Approaches to instrument recognition are very similar to playing technique classification. The same frequency features are useful for instrument recognition, such as cepstrums [58, 59, 60], MFCCs [61], line spectral frequencies [62] and CQT [60].

NMF can also be used to match instrument examples [63]. Machine learning approaches include statistical approaches like a Gaussian classifier [59], Gaussian mixture models [62] and naïve Bayesian classifier [64], as well as *k*-NN classification [59, 62, 64, 60], SVMs [64, 11] and binary trees [64].

ANN approaches are also common [64], with some deeper neural networks being used more recently, such as CNNs [61]. An investigation was done on different techniques of spectrogram compression and pre-processing for instrument family classification on mel spectrogram patches using a CNN on the OrchideaSOL dataset [65]. A few-shot learning approach classified instruments at different granularities by training a multi-task model that can operate at different levels of a musical instrument taxonomy hierarchy [66].

2.3.3 Pitch transcription

Pitch transcription can be considered as an F0 estimation problem or as a frame-level transcription (or multi-pitch estimator (MPE)) problem. The difficulty of the problem is tied to the polyphony of the music to transcribe. Multiple pitches or even pitches from multiple instruments can sound at the same time, complicating the transcription. The two camps of approaches to pitch transcription are traditional signal processing methods and deep learning methods. The deep learning methods generally have better accuracy on specific instruments, but signal processing methods generalize better and are faster [1].

Some approaches with traditional signal processing methods for single instrument or instrument agnostic approaches model spectral peaks and match them to the expected profile of instrument harmonics [13, 6]. Often some maximum-likelihood estimation modeling or F0 tracking is employed [67, 68]. A widely used F0 estimation algorithm is probabilistic YIN (PYIN), which is a probabilistic adaptation of the YIN algorithm [69]. It is still a state-of-the-art F0 estimation algorithm to compare to [70] and is also often used for annotating transcription datasets [71].

Template-based models have also shown success in MPE, such as NMF [72, 73] or probabilistic latent component analysis (PLCA) [74], which can model specific instruments based on templates. More recent approaches have taken advantage of advances in neural networks to develop piano transcription systems that outperform state-of-the-art traditional methods on standardized piano transcription problems [75, 76].

2.3.4 Pitch and instrument transcription

Assigning transcriptions to instrument streams adds an additional layer of complexity. Instrument recognition can give some indication of which instruments are active in a frame, but still leaves the problem of matching streams to active instruments when the polyphony is greater than one.

PLCA is a convenient way to model instrument notes played in a recording using a probabilistic model of dependent latent variables. The latent components can be extended to more than just pitches — instrument sources can be incorporated as a dependent latent variable. A shift-invariant model can be used with PLCA to extract shifted structures, such as note templates, from a non-negative spectrogram. Such a model can jointly determine pitches and instrument streams of pitches [77].

Assigning pitches to instrument streams can also be approached with constrained clustering, where pitches are clustered into streams so as to minimize an objective. The clustering is constrained to cluster pitches that are close in time and frequency, but preserve a maximum number of streams present at any time. The objective is to cluster pitches so as to minimize the difference in timbre between sound objects in the same stream. MFCCs are used as representations of sound object timbres [78].

Posteriograms of single instrument transcriptions, such as those produced by NMF and PLCA transcription, can be converted to binary piano roll by Viterbi decoding, which tracks probable notes using a two-state HMM for each pitch [79]. This approach has been extended to instrument stream assignment. The joint observation of F0 streams — which have already been determined from a different transcription algorithm — and features such as MFCCs can be modeled as a HMM where the observations are dependent on hidden states of instrument stream labels for each F0. A hidden Markov random field can be used to assign stream labels to estimated F0s based on minimizing an energy function that simultaneously models the temporal continuity of notes and MFCC source characteristics of F0s [80].

More recent work has applied deep learning to the problems of MPE and note tracking for both problems where the instrument classes are known and where instrument stream identification is done within a closed set of instruments. This was done using an encoder-decoder model with a self-attention mechanism [81].

2.4 MACHINE LEARNING CLASSIFIERS

Machine learning is the broad field in artificial intelligence concerned with developing mathematical models and algorithms that learn from examples. It can be seen as approximating some (often non-linear and intractable) function from example data points. In general, approaches model the unknown function as a weighted combination of basis functions; the learning then involves finding optimal weights and/or basis functions under the constraints of the model [82].

ANNs model complicated functions as linear combinations of basis functions. Nodes in an ANN are activated by a weighted combination of inputs from previous nodes or network inputs. The activations of output nodes can then be taken as multi-variable outputs. A single layer of inputs connected to outputs by a weighted sum and activation function is similar to linear regression, but more layers can be added to get a non-linear function of the inputs controlled by the weights. With enough connected nodes an ANN can learn arbitrarily complex functions. Learning is done by adjusting the weights of node inputs, which is usually done by comparing example input and target data to the estimated output the ANN produces from the input to obtain an error or cost. A common algorithm for adjusting the weights over epochs of a set of training data is backpropagation, which adjusts weights based on the derivative of the error with respect to each weight. ANNs are often used for regression, but are also very useful for classification problems: non-linear hyper-planes that segment classes based on input features can be approximated for binary classification, which can be extended to multi-class classification by having multiple outputs. Making the classification paradigm more explicit to the training procedure, such as by using a softmax output activation can even output class probabilities [83, 82].

A basic form of ANN is the feed forward neural network (FFNN) for which each layer is connected to each node of the previous layer and a bias node (to anchor the network in the solution space). Adding more nodes to a layer increases the accuracy with which the layer can approximate a function. Increased layers allow the network to approximate higher-order functions [83, 82]. More complex ANN architectures each have their own advantages and applications, some of which are discussed here.

2.4.1 Deep neural networks

More complex neural networks with more weights and layers (usually three or more layers) are referred to as DNNs. More layers allow the networks to learn more complex relations between the network inputs and outputs. With deeper layers, networks can learn to recognize simple features at early layers, which are aggregated into more complicated features at later layers. The more layers the network has, the more complex the features can be, and with more weights per layer, networks can learn a wider array of different features, which for classifiers can mean that DNNs are able to differentiate between more classes and more specific classes. The downside is that larger networks take much more data to properly train. It has been shown that increasing the amount of training data has a direct correlation with increased performance on some tasks [84]. DNNs are also more resilient to errors in training data labels when trained on larger datasets [85].

2.4.2 Convolutional neural networks

A problem with fully connected neural networks is the very large amount of weights that come with increases in network size. Large amounts of weights introduce difficulties in training and more weights need more data and epochs to train properly. Another problem comes in with generalization: consider a problem of identifying a pattern in a larger sequence — the same construct of weights and connections could identify the pattern, regardless of its position in the sequence. However, to identify its position, the construct needs to be repeated through the network to identify the pattern at different parts of the sequence input to the network. These problems can be addressed by introducing convolutional layers to a neural network [86]. A convolutional layer consists of weight kernels that are applied to a layer input by sliding them over the input and summing the weighted inputs for each kernel at each position. This allows feature extraction to be generalized by the kernels and the spatial positions of features to be maintained. The features extracted by convolutional layers are often further processed by fully connected layers [87]. A dense layer is not shift- or scale-invariant, but CNN kernels maintain spacial relations, so multiple convolution layers with subsampling (such as max-pooling) can be applied to capture larger templates with some distortion [86].

2.4.3 Recurrent neural networks

Recurrent neural networks (RNNs) are another expansion to the neural network concept that aims to improve processing of sequence data. Observations are sometimes time-dependent and the state of the inputs to a neural network is sometimes dependent on previous time steps or all previous states. For a neural network to take advantage of these dependencies to make predictions at a specific time step would require inputs of not just the current time step, but previous time steps. This is very impractical for large inputs, long time frames and sequences of observations of varying time lengths. RNNs address this by keeping some form of internal memory. The most basic form of RNN expands on FFNNs by having layer activations be dependent on both the layer inputs and the layer activations at the previous time step. More recent approaches have improved on this by adding gates to determine whether to overwrite the internal memory with new information [88].

2.5 FEW-SHOT LEARNING

Few-shot learning is a paradigm in machine learning that aims to allow machine learners to learn tasks after training is completed from only a handful of examples presented at inference time, much like humans can. For classification tasks, rather than outputting class probabilities, few-shot DNNs match queries with prototypes of classes. Some notable approaches to few-shot learning on specific tasks or a narrow set of tasks are summarized below. These approaches have all been applied to audio and music applications.

2.5.1 Siamese networks

Siamese networks are matching networks that compare how closely matched two inputs are using a neural network feature extractor that is duplicated and used on both inputs with the same weights. A distance function computes the distance between the twin output features, which is then converted into a probability for the inputs being a match [89]. This can be used for one-shot learning by having a definitive example for each novel class to which query examples are matched. Classification inference selects the class with the highest probability of a match between the new input and the class example [90].

2.5.2 Matching networks

A matching network maps a support set of example input and label pairs to a classifier. The classifier gives the predicted label of an input as a linear combination of the labels in the support set according to an attention mechanism. The attention mechanism used is a softmax over the cosine distance between embeddings of the classifier input and support set inputs. The embedding functions are DNNs trained over episodes during which the loss for predicting a batch of examples given a support set is minimised [91].

2.5.3 Relation network

The mechanism of matching embeddings of query examples to a support set is extended in relation networks by introducing a relation model, which gives a relation score between two embeddings which aims to be 1 for embeddings of related inputs and 0 for unrelated inputs. Episodic training is again utilized. The embedding feature map and relation model are trained on the objective of minimizing the mean squared error loss of the relation scores between a support set and query set for each episode [92].

2.5.4 Prototypical networks

A prototypical network is a type of few-shot learning ANN for classifying large amounts of classes based on relatively few training examples per class compared to a standard DNN [93]. A prototypical network is also easily adaptable to previously unseen classes without needing to retrain the network. A network is trained to transform input features into a higher dimensional embedding space so that the distance between embeddings of the same class is minimized and the distance between different classes is maximized. At inference time, embeddings for a set number of examples of each class are computed, called the class prototypes. These prototype embeddings are averaged to get one definitive class prototype in the embedding space. Classification is done by computing the embedding for an input and finding the class prototype with the smallest distance to the embedding [3].

2.5.5 Few-shot learning for audio and music applications

While deep learning has long been a consideration for sound event detection, such as using CRNNs to detect different sounds in a polyphonic context [94], few-shot learning has also recently been applied to the problem. This has been applied to keyword detection [33] by comparing the performance of Siamese Networks [90], Matching Networks [91], Prototypical Networks [3], and Relation Networks [92]. Another approach compared prototypical networks for sound event detection to two meta learning approaches: learning feature embeddings that would allow a SVM to generalize well to a new dataset based on a few support examples per class [95] and training a model to be easily fine-tuned by using a cost function that evaluates model performance on a new task after a few more steps of gradient descent on the new task [96]. They compared the few shot learners to standard ANN classifiers that were fine-tuned on the support set as a baseline [29]. A region proposal network in combination with a prototypical network was also used to find regions of interest for sound event detection that were then classified using the prototypical network [30]. Prototypical networks have also been applied to musical instrument recognition [31]. This approach trained a multi-task model capable of classifying at different levels of a taxonomy hierarchy by aggregating prototypes from lower levels in order to classify higher levels. Prototypical networks have also been shown to be applicable to transcription problems for drums [32]. A few-shot model was trained on a synthetic dataset and evaluated on multiple real-world datasets with accompaniment. Their model was able to generalize to finer-grained transcription tasks and out of vocabulary transcription.

2.6 TRANSFER LEARNING

Transfer learning aims to improve machine learning performance in one domain by transferring knowledge from another domain. This is very useful when there are existing models that perform very well on a related task to the task of interest or when there is not sufficient data for properly training a model on the target task, but a lot of data exists for a related task. Approaches to transfer learning can broadly be categorized into the following approaches:

- *Instance-based approaches*: Transfer learning applies models from a source domain on a target domain by re-weighing the samples from the source domain to match the distribution of the target domain samples [97].
- *Feature-based approaches*: Features are transformed to the same domain, either by finding a common latent feature space between the source and target domains or by transforming features from one domain to the other [97].
- *Parameter-based approaches*: Knowledge is transferred between models for different problem domains by using parameters from a model trained on one problem for a model on a different problem. The source model parameters are either frozen while new parameters are added and trained for the new model, or the source parameters are used as a starting point to reduce the training requirements for learning suitable features in the new domain [97].
- *Relational-based approaches*: For problems in relational domains, the logical rules and relations are learned by a model in the source domain. These relations are then transferred to the target domain [97].

Transfer learning is often applied in MIR in order to use feature extractors from one domain in a MIR problem where the training data is limited. For example, because spectrogram representations of sound can be treated as images, an existing image CNN model can be used as a feature extractor for genre classification by feeding the activations of the penultimate network layer into a SVM classifier [98].

Transfer learning can also be used to aggregate features at multiple levels and scales. Models trained at different scales of input features can be combined by using them as feature extractors and then training another model on the combined features. Since the deep feature learning has already been done, the aggregate model can be much simpler, such as a shallow FFNN [99].

Alternatively, representation transfer can be done for a CNN model trained on a tagging task by average pooling the activations of each convolution layer and concatenating them into a feature vector for SVMs to apply on various target tasks in a similar problem domain [100, 101]. This is very useful even when transferring knowledge between data domains in the same problem. As noted, generalization to new data can be a problem for machine learning models trained on narrow data [2]. Transfer learning enables models to be trained on small, synthesized, or otherwise non-ideal datasets that still allow the model to learn good feature representations and then fine-tune the model on examples of the real data of interest.

2.7 CHAPTER CONCLUSION

A lot of research has been done on instrument and playing technique classification, but many investigations focus on specific instruments and only recently have machine learning techniques been explored over signal processing techniques. The same was found for pitch transcription. Some investigation has been done on combining instrument and playing technique classification, as well as on multiple-instrument pitch transcription with instrument stream assignment. No literature was found on jointly classifying instrument, playing technique and pitch.

Research on instrument and playing technique classification, as well as pitch transcription, has utilized a variety of machine learning techniques, including DNNs, CNNs, RNNs and CRNNs. Few-shot learning has also been used in audio and music applications, including ADT. Prototypical networks show the most promise in the audio and music domain of all the few-shot methods encountered, providing the best results in two studies comparing few-shot learning algorithms on audio problems [33, 95]. Alternatively, transfer learning has been shown to be effective for feature learning in low-resource problem domains, including music.

CHAPTER 3 METHODS AND THEORY

3.1 CHAPTER OVERVIEW

This chapter gives an overview of the classification systems used in this investigation and the theory behind the methods. The few-shot and standard classifier systems for instrument, playing technique and pitch classification are described. The theory of spectrogram features is given and the log-mel spectrogram and constant-Q transform input features are described. Standard classifier training and prototypical network training are described. The process of training neural network models with Bayesian optimization for hyperparameter tuning is explained. Methods for pitch classification with the PYIN algorithm are also given.

3.2 CLASSIFICATION SYSTEM OVERVIEW

The goal of this investigation is to implement a classification system that takes in a recorded example of a string instrument note and to classify the example according to the string instrument and playing technique used to play the note, as well as the pitch of the note.

Two different approaches to this are designed and implemented: prototypical few-shot models and standard classifier ANN models. Both approaches take in a time-frequency representation of an isolated string instrument note example recording and output a class label describing the predicted instrument, pitch and playing technique.

3.2.1 Few-shot classifier

The few-shot classification system uses prototypical models that are trained on the meta-learning problem of matching query examples to prototypes of the classes of interest. An overview of the classification system is shown in Figure 3.1.

For each dataset or problem to evaluate, prototypes are calculated for all the classes. Time-frequency

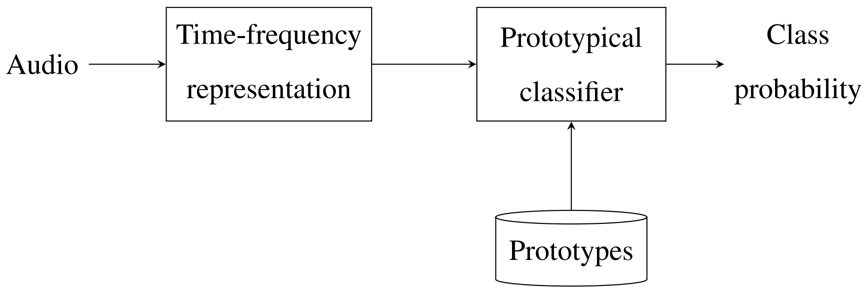


Figure 3.1. An overview of the proposed prototypical few-shot classification system.

representations of query examples are converted to a high-dimensional embedding by the prototypical model and class probabilities are approximated by finding the class prototypes that are the closest in embedding space to the query example. A mel spectrogram or a CQT is used as the time-frequency representation.

3.2.2 Standard classifier

The standard classifier models are trained to approximate class probability saliences for a given time-frequency input. For each new dataset or problem, where the classes do not directly map onto the classes seen during training, transfer learning is applied to adapt the models to the new class domain. At inference time, the class with the highest output activation for a time-frequency input to a model is taken as the predicted label.

3.3 TIME-FREQUENCY REPRESENTATION

Both a log-mel spectrogram and a CQT are compared as input features to the models. These time-frequency representations are adaptations of the short-time Fourier transform (STFT). The first 100 frames of each input feature are used as model inputs. At a sampling frequency of $F_s = 22050$ Hz and a hop size of 256 this is 1.16 s. An examination of the data showed that this is enough to include the onset event and some sustain for all techniques, thus the size of model inputs are restricted to 100 frames to reduce model sizes. The chosen sampling frequency is a common value used in MIR to reduce the number of samples to process, while still avoiding aliasing of harmonics. It is the suggested sampling frequency in the Librosa signal processing library [102].

3.3.1 Spectrogram

The STFT is a Fourier transform of a signal offset over time to produce a 2-dimensional mapping of frequency component intensities in the signal over time. It is given by

$$X(f,t) = \int_{-\infty}^{\infty} x(u)w(u-t)e^{-2\pi fj(u-t)}du, \tag{3.1}$$

for a signal $x(t)$ and a window function $w(t)$ [103]. In discrete-time, the STFT is given by

$$X[k,m] = \sum_{n=0}^{N-1} x(n)x(n+mH)w(n)e^{-2\pi j \frac{kn}{N}}, \tag{3.2}$$

where m is the frame index and k is the frequency bin index. H is the hop size, which determines the time resolution. This can be implemented by shifting the window function by H samples at a time over the signal and taking the FFT of the windowed signal. The energy spectrogram of a signal is given by the magnitude of the STFT of the signal,

$$\mathbf{S} = |\mathbf{X}|, \tag{3.3}$$

and the power spectrogram by the squared magnitude

$$\mathbf{S} = |\mathbf{X}|^2. \tag{3.4}$$

An example magnitude spectrogram is shown in Figure 3.2 for a violin note from the RWC dataset.

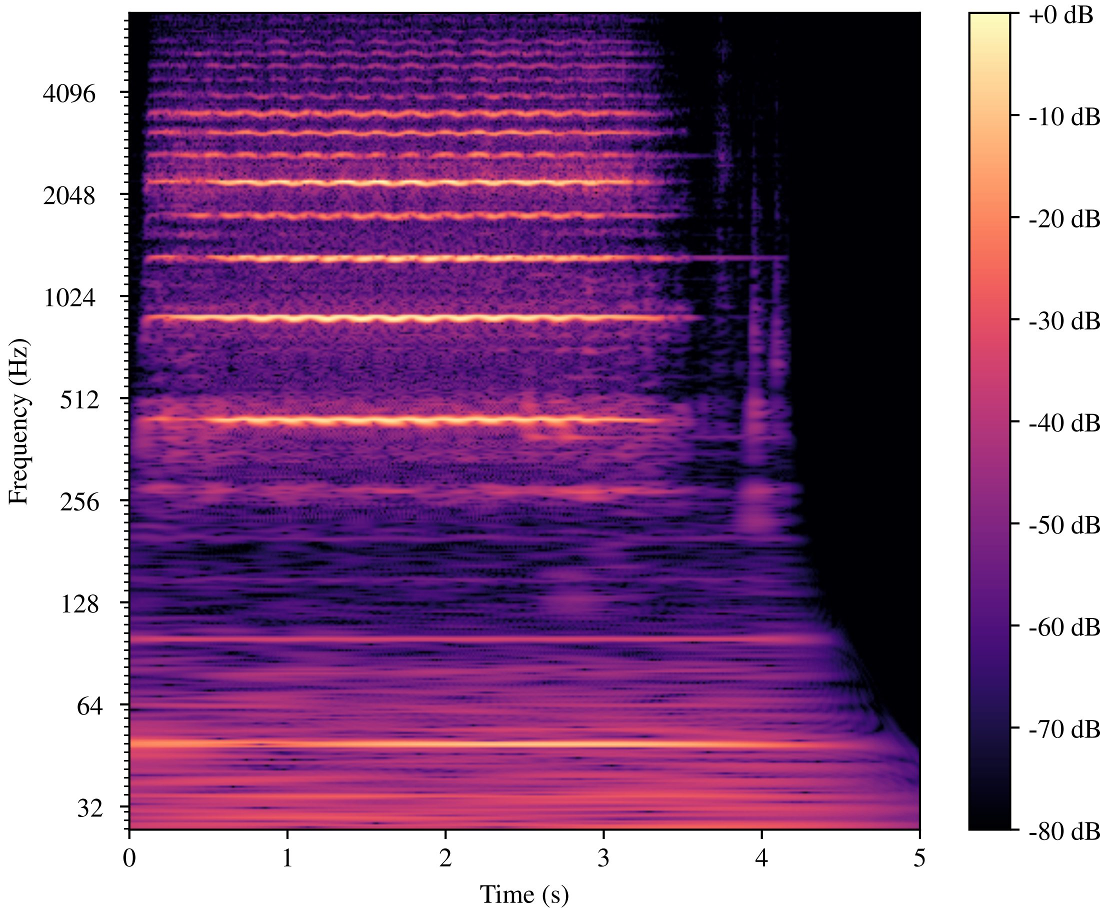


Figure 3.2. The spectrogram of an A4 violin example from the RWC dataset, played using the *vibrato* technique.

3.3.2 Mel spectrogram

The log-mel spectrogram is a common input feature for playing technique classification [65, 32, 33] and is often used for timbre analysis [104]. The mel frequency scale aims to map the non-linear perception of audio frequencies by humans to a linear scale [105]. The mapping of a frequency to the mel scale is given by

$$mel(f) = 2595\log_{10}(1 + f/700). \tag{3.5}$$

A log-mel spectrogram is obtained by mapping a power spectrogram into mel scale by multiplying it with a mel filter bank. The band-pass filters used to characterize sources are L triangular filters that are linearly spaced on a mel scale. The triangular bandpass filters are defined by their centers, which are equally spaced on the mel scale between 0 Hz and $F_s/2$, with filter edges at each extreme and a half overlap between filters in mel scale. The filters are normalized by dividing the filter weights by the width of the mel band. The filter transfer functions for 8 mel filters are shown in Figure 3.3 for demonstration.

Mel spectrogram inputs for classifiers are calculated using 128 mel filters evenly spaced from 0 Hz to $Fs/2$. The input spectrogram is computed from a STFT using a 1024 sample Hamming window and 1024 FFT coefficients with a 256 sample hop size. An example log-mel spectrogram of a violin note from the RWC dataset is shown in Figure 3.4. The Librosa Python library was used to calculate mel spectrograms [106].

3.3.3 Constant-Q transform

A CQT is a time-frequency representation that has proven useful in transcription implementations [107, 75]. The CQT is a frequency transform that results in frequency bins with geometrically spaced center frequencies that all have equal Q-factors. This means a better frequency resolution for low-frequency bins and better time resolution for high-frequency bins [108]. The CQT is well suited for applications where F0s need to be accurately identified [86].

The constant-Q transform is given by

$$X^{CQ}[k,m] = \sum_{n=m-\lfloor N_k/2 \rfloor}^{m+\lfloor N_k/2 \rfloor} x[n]a_k^*[n-m+N_k/2], \tag{3.6}$$

where k is the frequency bin index and $a_k^*[n]$ is the complex conjugate of a time-frequency atom. These are basis functions of complex-valued waveforms:

$$a_k(n) = \frac{1}{N_k} w\left(\frac{n}{N_k}\right) e^{-i2\pi f_k/F_s}. \tag{3.7}$$

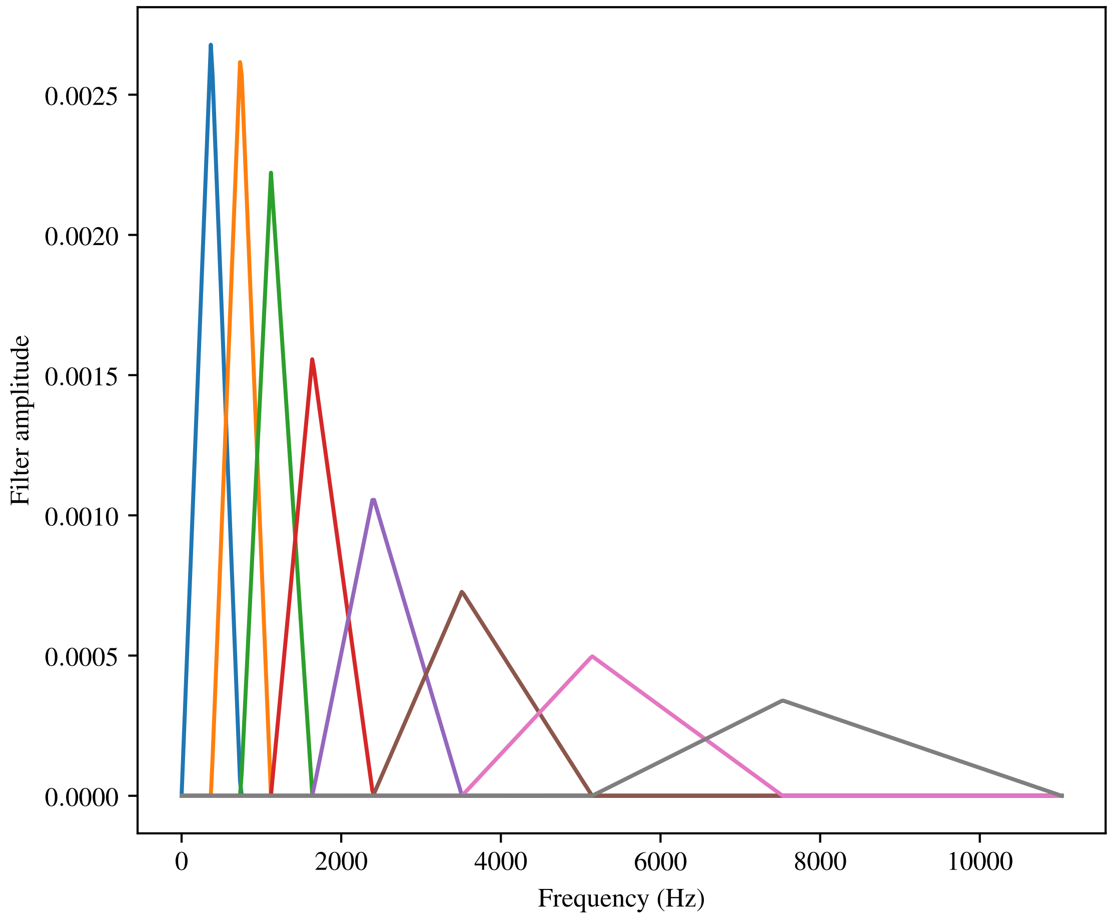


Figure 3.3. 8 half-overlap normalized band-pass filters that are linearly spaced on the mel frequency scale.

f_k is the center frequency of bin k, F_s is the sampling frequency and $w(t)$ is a continuous window function that is only non-zero in the interval $t \in [0,1]$. For B bins per octave and a lowest bin center frequency f_1, the center frequency of any bin is given by

$$f_k = f_1 2^{\frac{k-1}{B}}. \tag{3.8}$$

The window width for each bin that minimizes frequency smearing (thus maximizing the Q-factor) that still allows signal reconstruction is

$$N_k = \frac{qF_s}{f_k(2^{\frac{1}{B}} - 1)}, \tag{3.9}$$

where $q \leq 1$ is a scaling factor that improves time resolution for lower values. $q = 1$ is often used [108].

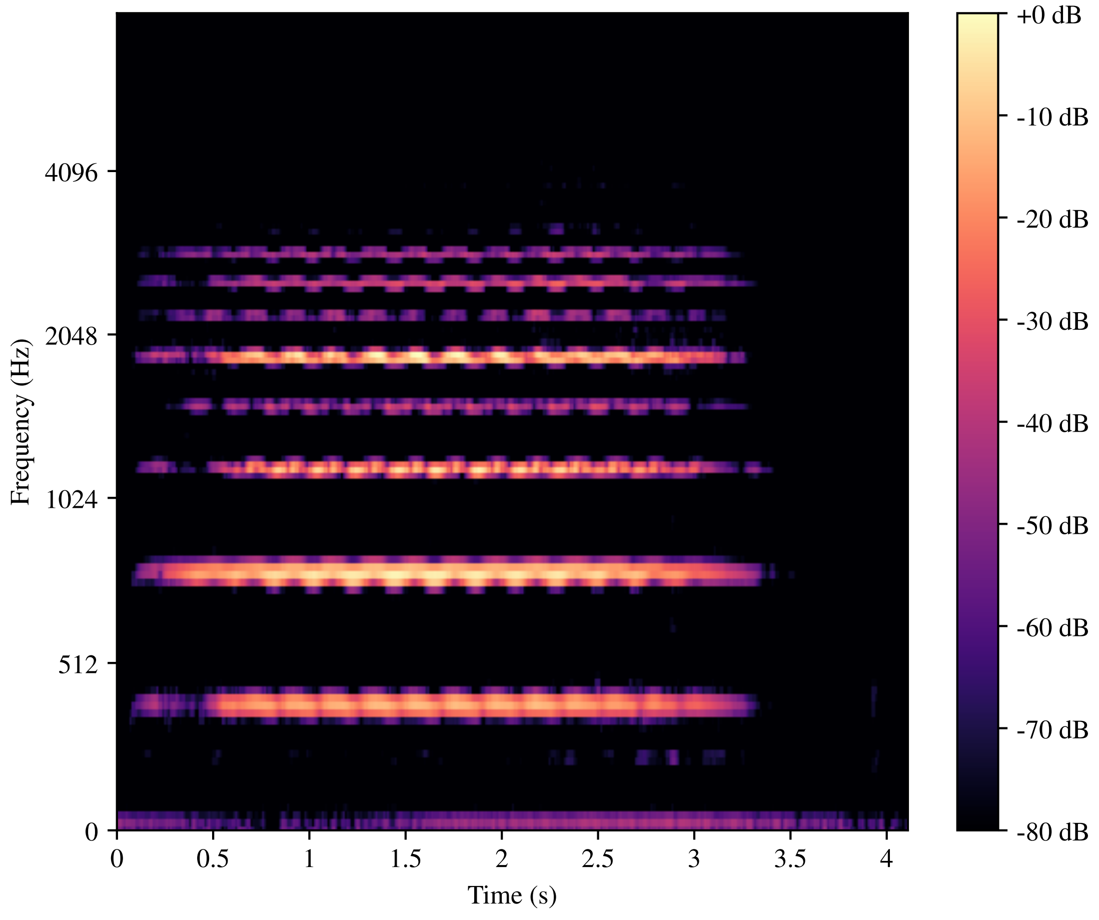


Figure 3.4. Log-mel spectrogram playing technique and pitch classification input features for an A_4 violin example from the RWC dataset, played using the *vibrato* technique.

The direct computation of the constant-q transform is very inefficient. Using Parseval's equation, the CQT for a signal of N windowed points can be calculated using an FFT [109]:

$$X^{CQ}[k,N/2] = \sum_{n=0}^{N-1} X[n]A_k^*[n]. \tag{3.10}$$

$A_k[n]$ is the discrete Fourier transform of the basis function centered in an N point window. If the basis function FFTs are packed into the columns of a matrix and the FFT of the input is expressed as a column matrix, then it can be computed using matrices as

$$\mathbf{X}^{CQ} = \mathbf{A}^*\mathbf{X}. \tag{3.11}$$

The computation can be further improved by the realization that computing the CQT for one octave of frequencies[1] is the same as computing the CQT for an octave higher that is downsampled by a factor of 2. The CQT is thus computed one octave at a time, starting at the highest octave. After calculating each octave, the input signal is downsampled and passed through a low-pass filter. The same kernel of basis functions can be used for all the octaves and only has to be computed once.

CQT coefficients are calculated with a Hamming window, 60 bins per octave and 8 octaves starting at A_0 from audio sampled at a rate of $Fs = 22050$ Hz. This results in an input to the ANN models with 480 bins in the frequency dimension. A hop size of 256 samples is used again to be consistent across features. Figure 3.5 shows a CQT over the frequency spectrum for the same RWC violin note example used previously.The Librosa Python library was used to calculate CQTs [106].

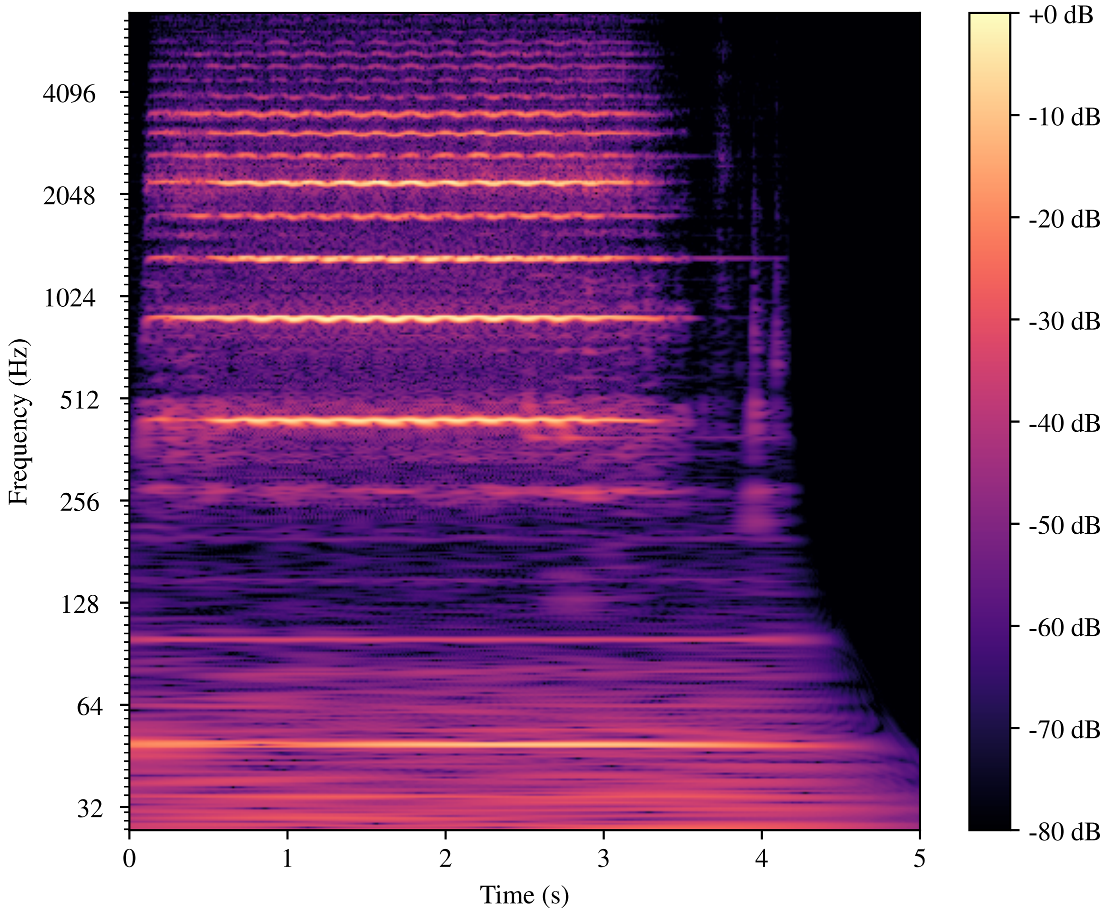


Figure 3.5. CQT playing technique and pitch classification input features for an A_4 violin example from the RWC dataset, played using the *vibrato* technique.

[1]An octave is the range it takes for the fundamental frequency to double.

3.4 STANDARD CLASSIFIER NEURAL NETWORKS

Standard classifier ANNs are used as a baseline to compare the few-shot models to. The models map a time-frequency input to an output vector of one node for each class in the problem space, approximating the class probabilities of the input. Models are trained with the objective of maximizing the output activation of the true class while minimizing the activation of all other classes. A categorical cross-entropy (CCE) loss function is used and models are trained using the Adam optimizer [110] and a learning-rate scheduler. The architectures for the different models that were trained are given in Section 4.3.

3.4.1 Loss function

The CCE loss function is often used for models with categorical outputs that are one-hot encoded [111]. For an output vector $\hat{\mathbf{y}}$ of size N, which approximates a one-hot encoded classification $\mathbf{y}$, the average error over the weights, ϕ, is

$$E(\phi) = \frac{-1}{N}\sum_{n=1}^{N}\left[y_n \log \hat{y}_n + (1-y_n)\log(1-\hat{y}_n)\right]. \tag{3.12}$$

3.4.2 Adam optimizer

Adam is an algorithm for first-order gradient-based optimization of stochastic objective functions. The algorithm updates exponentially weighted moving averages of the gradient and squared gradient. It takes as hyper-parameters: α, the step size or learning rate; β_1 and β_2, hyper-parameters that control the exponential decay rates of the moving averages; ε, a constant for numerical stability. The suggested default parameters of $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\varepsilon = 10^{-8}$ are used [110]. An exponentially decaying learning rate is used.

3.4.3 Learning rate scheduler

An exponentially decaying learning rate scheduler is used in order to promote exploration of the weight parameter space early on and fine-tune weights at later epochs. The scheduler is given for epoch x by

$$\alpha = \alpha_0 e^{-\lambda x}, \tag{3.13}$$

where α_0 is the initial learning rate and λ is the decay rate. $\lambda = 0.023$ is used, which approximately reduces the learning rate by a factor of 10 every 100 epochs. For prototypical network training, the learning rate is adapted every 500 episodes (thus 500 episodes is considered one epoch).

3.5 PROTOTYPICAL NETWORKS

Prototypical networks are used as few-shot models in order to easily generalize to new data, instruments or sub-problems. CNN, CNN, RNN and CRNN models were trained. The model architectures are discussed in Section 4.3.

3.5.1 Episodic training

A prototypical network is trained in episodes. For each episode, a subset of classes from a training set is selected for which support and query examples are sampled. The episode loss is calculated based on how well the network minimizes the distance in embedding space between query examples and true class prototypes, and maximizes the distance between queries and the other class embeddings.

For N_C classes, N_S support examples and N_Q query examples per episode, the class prototypes for a training episode are given for each class $k \in \{1, \ldots, N_C\}$ by

$$\mathbf{c}_k = \frac{1}{N_S} \sum_{(\mathbf{x}_i, y_i) \in S_k} f_\phi(\mathbf{x}_i), \tag{3.14}$$

where S_k is the support set of N_S examples belonging to class k, randomly sampled from the training set and $f_\phi(\mathbf{x})$ is the output of the neural network for the current weights ϕ [3].

The loss for a training episode is given by

$$J = \frac{1}{N_C N_Q} \sum_{k \in \{1,\ldots,N_C\}} \sum_{(\mathbf{x}_i, y_i) \in Q_k} \left[d(f_\phi(\mathbf{x}), \mathbf{c}_k) + \log \sum_{k' \in \{1,\ldots,N_C\}} \exp\{-d(f_\phi(\mathbf{x}), \mathbf{c}_{k'})\} \right], \tag{3.15}$$

where $d(.,.)$ is a distance function, and Q_k is the query set of N_S examples belonging to class k, randomly sampled from the training set, excluding the support set [3].

The training loop for episodic training of a prototypical network is shown in detail in Algorithm 1.

The Euclidean distance, given by

$$d(\mathbf{p}, \mathbf{q}) = \sqrt{\sum_{i=1}^{N} (q_i - p_i)^2}, \tag{3.16}$$

for two vectors $\mathbf{p}$ and $\mathbf{q}$ of size N, is used for the distance function. $N_S = 3$ and $N_Q = 5$ are used for the number of support and query samples respectively during training.

Algorithm 1 Procedure for training a prototypical model with N_C classes, N_S support examples and N_Q query examples per episode. Adapted from [3].

```
Input: Training set $D = \{(x_1, y_1), \ldots, (x_N, y_N)\}$ of paired inputs and class labels. $D_k$ denotes the
    subset of $D$ for which the class label is $y_i = k$. A learning rate scheduling function $r(e \mid \theta)$ with a
    set of hyperparameters $\theta$. A neural network $f_\phi(\mathbf{x})$ with weights $\phi$. A distance function $d(\cdot,\cdot)$. A
    weight update function $A(\phi|J,\alpha)$ dependent on a loss $J$ and learning rate $\alpha$.
Output: Model weights $\phi$.
    $\phi \leftarrow$ Random initial values.
    for each episode, $e$ do
        $\alpha \leftarrow r(e \mid \theta)$.
        $\mathbf{V} \leftarrow N_C$ random class indices from the training set.
        for $k$ in $\{1, \ldots, N_C\}$ do
            $S_k \leftarrow N_S$ random examples from $D_{V_k}$.
            $Q_k \leftarrow N_S$ random examples from $\{D_{V_k} \setminus S_k\}$.
            $\mathbf{c}_k \leftarrow \frac{1}{N_S} \sum_{(\mathbf{x}_i, y_i) \in S_k} f_\phi(\mathbf{x}_i)$.
        end for
        $J \leftarrow \frac{1}{N_C N_Q} \sum_{k \in \{1,\ldots,N_C\}} \sum_{(\mathbf{x}_i, y_i) \in Q_k} \left[ d(f_\phi(\mathbf{x}), \mathbf{c}_k) + \log \sum_{k' \in \{1,\ldots,N_C\}} \exp\{-d(f_\phi(\mathbf{x}), \mathbf{c}_{k'})\} \right]$.
        $\phi \leftarrow A(\phi|J,\alpha)$.
    end for
```

3.6 BAYESIAN OPTIMIZATION

Bayesian optimization is a method for optimizing parameters with minimal evaluations of a cost or utility function. This is especially useful for optimizing neural network hyperparameters for training since training is costly to evaluate because of the time and computation it takes. Gaussian process (GP) regression is used to estimate a utility function. The GP is sampled to find likely next points in the parameter space to evaluate. The GP is then iteratively updated. The estimation of the utility function can be used to find the optimal parameters [112].

Let $f(\mathbf{x})$ be the utility function to maximize in a bounded parameter space. $\{\mathbf{x}_i, y_i\}_{i=1}^n$ are observations of this function at time step n. A GP regression is defined by a mean function $\mu(\mathbf{x} \mid \{\mathbf{x}_i, y_i\}_{i=1}^n, \theta)$ and a covariance $\sigma^2(\mathbf{x} \mid \{\mathbf{x}_i, y_i\}_{i=1}^n, \theta)$, which are dependent on a kernel function $k(\mathbf{x}, \mathbf{x}')$, defined in terms of a set of basis functions, and θ, which is a set of hyperparameters for the kernel that can be found as part of the GP regression using maximum likelihood estimation [83]. To simplify the notation, from here on $\{\mathbf{x}_n, y_n\}$ will denote $\{\mathbf{x}_i, y_i\}_{i=1}^n$ and the set of observations and hyperparameters for the GP mean and covariance functions will be inferred. If $a(\mathbf{x} \mid \{\mathbf{x}_n, y_n\}, \theta)$ is an acquisition function that is dependent on the GP regression, then the next point to evaluate is

$$\mathbf{x}_{n+1} = \underset{\mathbf{x}}{\operatorname{argmax}}\, a(\mathbf{x} \mid \{\mathbf{x}_n, y_n\}, \theta). \tag{3.17}$$

After $\mathbf{x}_{n+1}$ is evaluated, the GP is updated with the new observation [112].

3.6.1 Optimizing neural network hyperparameters with Bayesian optimization

Google Cloud ML Engine was used to perform hyperparameter optimization and model training with Bayesian optimization [113]. Their default parameters are used, including a Matérn kernel with automatic relevance determination (ARD) for the GP and an expected improvement acquisition function [114]. $N_O = 10$ observations are made for each model with $N_I = 3$ initial observations. The evaluation function used was the *micro* F-measure over the validation set (see Section 4.5). The process for optimizing the hyperparameters for training an ANN model is given in detail in Algorithm 2.

The expected improvement acquisition function used in the implementation estimates the expected increase in the maximum observed objective value for any potential next observation point. It is given analytically for a GP with mean $\mu(\mathbf{x})$ and standard deviation $\sigma(\mathbf{x})$ as

$$a_{EI}(\mathbf{x} \mid \{\mathbf{x}_n, y_n\}, \theta) = \begin{cases} (\mu(\mathbf{x}) - y^+)\Phi(Z) + \sigma(\mathbf{x})\phi(Z) & \text{if } \sigma(\mathbf{x}) > 0, \\ max(0, \mu(\mathbf{x}) - y^+) & \text{if } \sigma(\mathbf{x}) = 0, \end{cases} \tag{3.18}$$

where y^+ is the maximum observed value, Z is the standard normal distribution

$$Z = \frac{\mu(\mathbf{x}) - y^+}{\sigma(\mathbf{x})}, \tag{3.19}$$

and $\Phi(Z)$ and $\phi(Z)$ are the cumulative distribution function and probability density function of the distribution respectively [115, 113].

The Matérn kernel used for the GP is defined as

$$k(x_i, x_j) = \frac{1}{\Gamma(\nu)2^{\nu-1}} \left(\frac{\sqrt{2\nu}}{\ell} d(x_i, x_j) \right)^{\nu} K_\nu \left(\frac{\sqrt{2\nu}}{\ell} d(x_i, x_j) \right), \tag{3.20}$$

where $d(\cdot,\cdot)$ is the Euclidean distance, $K_\nu(\cdot)$ is a modified Bessel function and $\Gamma(\cdot)$ is the gamma function. ν and ℓ are parameters for smoothness and length scaling respectively. When applying ARD, the length scaling parameter is replaced with a diagonal matrix of length scaling hyperparameters, which allows the kernel to be independent of inputs for which the corresponding matrix entry is very small [116].

Algorithm 2 Bayesian optimization hyperparameter tuning for machine learning model training. N_I is the number of initial observations and N_O is the total number of observations.

Input: Boundaries on the hyperparameter search space $\mathbb{H}^d$, where d is the number of hyperparameters. Acquisition function $a(\mathbf{h} \mid \{\mathbf{h}_i, u_i\}_{i\in\{1,\ldots,n\}}, u_n\}, \theta)$, where $\mathbf{h} \in \mathbb{H}^d$, u is an observation and θ is the acquisition function hyperparameters. Model training function $T(D \mid \mathbf{h})$ and evaluation function $E(D \mid \phi)$, where D is a dataset and ϕ is the model parameters. Training set D_{train} and validation set $D_{validation}$.

Output: The optimized model weights ϕ_{opt}.

```
for $n$ in $\{1,\ldots,N_I\}$ do
    $\mathbf{h}_n \leftarrow$ randomly sampled from $\mathbb{H}^d$.
    $\phi_{\mathbf{h}_n} \leftarrow T(\mathbf{h}_n, D_{train})$ .
    $u_n \leftarrow E(D_{validation} \mid \phi_{\mathbf{h}_n})$.
end for
for $n$ in $\{N_I+1,\ldots,N_O\}$ do
    $\mathbf{h}_n \leftarrow \underset{\mathbf{h}}{\operatorname{argmax}}\, a(\mathbf{h} \mid \{\mathbf{h}_i, u_i\}_{i\in\{1,\ldots,n\}}, \theta)$.
    $\phi_{\mathbf{h}_n} \leftarrow T(\mathbf{h}_n, D_{train})$ .
    $u_n \leftarrow E(D_{validation} \mid \phi_{\mathbf{h}_n})$.
end for
$\mathbf{h}_{opt} \leftarrow \underset{\mathbf{h}}{\operatorname{argmax}} \{\mathbf{h}_i, u_i\}_{i\in\{1,\ldots,N_O\}}$.
$\phi_{opt} \leftarrow \phi_{h_{opt}}$.
```

3.7 PYIN F0 ESTIMATION

PYIN is a probabilistic adaptation of the YIN F0 estimation algorithm. The original YIN algorithm works on the intuition that for a fundamental period $\tau = 1/f_0$, the cumulative difference between periodic signal points would be small. This difference is calculated using auto-correlation and cumulative normalization is applied. While the original YIN algorithm applied a threshold to decide if a minimum in the normalized difference is a true F0 or an unvoiced frame, PYIN applies a probabilistic threshold, resulting in a voiced probability for each frame. A HMM is then used to track pitches [69].

Pitch classifications are obtained from the PYIN F0s and voiced frame probabilities by first finding a definitive F0 for the entire example. This is done in three ways, which are compared:

- Take the mean F0 over all frames of each input example.

- Take the weighted mean F0 over all frames of each input example, weighted by the probability of the frames being voiced.
- Take the F0 at the first frame of each input example with the maximum probability of being voiced.

The F0 of an example is converted to a pitch classification by finding the pitch with the closest frequency to the example's F0. Only the frames matching the input to the classifiers are considered for F0 estimation.

3.8 CHAPTER CONCLUSION

Two classification systems for classifying instrument, playing technique and pitch from examples of string instrument tones were given: a few-shot approach with prototypical networks and a transfer learning approach with standard neural network classifiers. The calculation of log-mel spectrogram and CQT input features were discussed. The training procedure for standard neural networks was given, as well as the episodic adaption of the training procedure for prototypical networks. Optimization of training and network hyperparameters using Bayesian optimization was also set out. Methods for calculating the pitch classification of a note example using PYIN to compare the other classification methods on were finally given.

CHAPTER 4 MODELS AND EXPERIMENTS

4.1 CHAPTER OVERVIEW

The datasets used during this investigation are described. An overview is given of the data sources, as well as how they are compiled into training, validation and test sets. The different neural network architectures that are trained and evaluated are given, followed by an explanation of how the prototypical networks and standard classifier networks were trained. The procedure for evaluating the models on the data sets is given.

4.2 DATASETS

Few-shot prototypical networks were designed and trained to classify examples of single notes for violin, viola, cello or double bass. Multiple training and evaluation sets were compiled from different data sources for this. The classification labels are the joint label of the example instrument, playing technique and pitch, for example, {violin, *pizzicato*, B_5} The few-shot classifiers should be able to classify new classes not seen during training based on only a handful of examples provided at inference time+, thus the models are evaluated on datasets that contain classes that were not seen during training. Standard neural network classifiers were also trained on the same tasks and employed transfer learning to adapt the standard classifiers to new classes.

4.2.1 Data sources

The RWC [34] instrument dataset contains recordings of scales played on a number of different instruments and with different playing techniques. For each instrument and playing technique, three different performers each recorded scales with *piano*, *mezzo forte* or *forte* dynamics. Bowed string instrument playing techniques were considered, thus the dataset examples for violin, viola, cello and double bass were used. The dataset contains 1393 different combinations of string instrument, playing technique and pitch (see example above). Subsets of these examples were compiled for network training, validation and evaluation from the dataset.

The OrchideaSOL [35] dataset consists of examples of instrument notes played with different playing techniques on different instruments, including string instrument techniques not seen in the RWC dataset. The OrchideaSOL dataset contains 2630 different combinations of string instrument, playing technique and pitch. Playing techniques combined with a mute are considered separate playing techniques. The OrchideaSOL string instrument subset is used for cross-dataset evaluation.

Additionally, real-world examples were compiled by extracting data from YouTube tutorials of violin, viola, cello and contra-bass playing techniques. Tutorials for *collé*, *martelé*, *pizzicato*, *sautillé*, *staccato* and *vibrato* were annotated by hand and individual note examples were extracted. This dataset consists of 53 combinations of instrument, playing technique and pitch played by a mix of different performers.

Piano note examples from the RWC [34] dataset are also used for adding accompaniment to string examples, which is described in more detail in Section 4.2.3.4.

4.2.2 Training and validation data

Training and validation subsets of string instrument playing technique note examples were compiled from the RWC dataset, which were used to train prototypical few-shot classifiers and standard neural network classifiers (together, the training and validation data formed the total body of data available during the training phase).

The RWC string training subsets contain recordings by two of the three performers. Recordings from the remaining performer are left out for evaluation. Prototypical networks require enough examples per class to provide support and query examples in a training episode. $N_S = 3$ support examples and $N_Q = 5$ query examples per class were chosen due to the constraint of the number of class examples available in the dataset. A subset of 228 instrument, playing technique and note combinations, which satisfy the 8 example minimum for support and query examples, are used for network training. The playing techniques present in this set are shown in Figure 4.1(b). Piano accompaniment is also added to these examples to form an additional training set for models evaluated on the accompaniment set described in Section 4.2.3.4. An 8 : 2 training data to validation data split is employed, with classes allocated to the validation set in such a way that 20 % of examples are used for validation. The validation data were used during Bayesian optimization of the hyperparameters of the few-shot learning framework. The RWC training examples with either the pitch or the playing technique class labels ignored were used to train prototypical and standard classifier models on the problems of instrument playing technique classification and instrument pitch classification respectively.

Ideally more than 8 examples would be available per class during training [93], but restrictions on the data available prevent setting this minimum much higher for risk of significantly reducing the total amount of training data. The fact that the number of training examples per playing technique and per pitch are much higher on average, reduces the concern for this constraint, as playing technique classification, as well as pitch classification are the true problems of interest when considering the tutorial scenario these models are aimed at.

4.2.3 Test data

Independent test sets were compiled from different datasets, that were used to evaluate the prototypical few-shot classifiers and standard neural network classifiers. Generalization to new performers was evaluated as well as generalization to new playing techniques and instruments.

The shot number of $N_S = 3$ was chosen to match the training shot. Matching the shot number used in training when testing has been shown to yield good results for prototypical networks [3]. Table 4.1 shows how many classes in each dataset satisfy this minimum, as well as how many of the classes in each dataset were not seen during training. The support examples for each evaluation of a prototypical model on a test set are randomly drawn from the test set and left out during evaluation. Similarly, the same number of randomly drawn examples per class are used to further train standard classifier networks during transfer learning and left out during evaluation.

Table 4.1. A breakdown of the number of valid training and validation $[N_S = 3, N_Q = 5]$, and test $[N_S = 3, N_Q = 1]$ classes in each dataset.

Dataset	**Valid training and validation classes**	**Valid test classes**	**Unseen classes**	**Total Examples**
RWC string instrument training set	228	—	—	2699

Continued on next page

Table 4.1 Continued from previous page

Dataset	**Valid training and validation classes**	**Valid test classes**	**Unseen classes**	**Total Examples**
Accompaniment RWC string instrument training set	228	—	—	2699
Techniques RWC string instrument training set	38	—	—	6813
Notes RWC string instrument training set	166	—	—	6793
OrchideaSOL string instrument test set	—	439	325	3156
OrchideaSOL violin test set	—	158	122	1134
OrchideaSOL viola test set	—	118	82	840
OrchideaSOL cello test set	—	67	31	468
OrchideaSOL double bass test set	—	96	90	714
YouTube string instrument test set	—	46	33	494
Accompaniment RWC string instrument test set	—	234	12	1422
Techniques RWC string instrument test set	—	24	2	2658
Notes RWC string instrument test set	—	153	0	2601
RWC clarinet test set	—	120	120	1080

The datasets used for evaluation, except the RWC test set, contain playing techniques not seen during training, as seen in Figure 4.1(b), as generalization to new performers and playing styles is one of the main research questions. The number of examples for each instrument in the datasets is not balanced, as is shown in Figure 4.1(a), thus experiments are also done for one instrument at a time. The number of examples for each pitch across all instruments and playing techniques is shown in Figure 4.2; as can be seen, generalization to new pitches is also evaluated.

4.2.3.1 RWC test set

A subset of 234 combinations of instruments, playing techniques and pitches from the RWC dataset, played by a musician different from the musicians of the training subset, is used for testing. Instrument, playing technique and pitch combinations were chosen for the test set that have at least six examples per class, which is needed to compute prototypes and evaluate on unseen examples ($N_S + 3$) for the chosen shot number. The cutoff for minimum number of class examples was chosen so as to have a significant number of test examples for each class, without excluding too many classes. This set does not contain new playing techniques not seen during training, but all examples are recorded by a different performer. This set evaluates generalization to new performers within the same dataset and to new data, as well as accommodating previously unseen classes. All classifiers are trained on a subset of pitches that omits some pitches that are present in the RWC evaluation sets, since more classes meet the requirement of 6 examples to be included in the evaluation set, even when compiled from just one performer than meet the requirement of 8 examples needed to be included in the training set. Figure 4.1(b) shows the playing technique classes relevant to the RWC evaluation set.

A subset of 24 combinations of instrument and playing technique were used to evaluate models for instrument playing technique classification. The resulting subset contains 2658 examples across 9 playing techniques. To evaluate the models across different shot numbers, a subset was used with classes restricted to those with a minimum of 18 examples, to be able to compare shot-numbers up to 15 with at least 3 test examples. This subset contains 2658 examples across 9 playing techniques.

Another subset of 153 combinations of instrument and pitch were used to evaluate models for instrument pitch classification. This subset contains 2601 examples across 68 pitches. Again, a subset with classes restricted to those with a minimum of 18 examples was used to evaluate the models across different shot numbers, resulting in a set containing 1152 examples across 35 pitches.

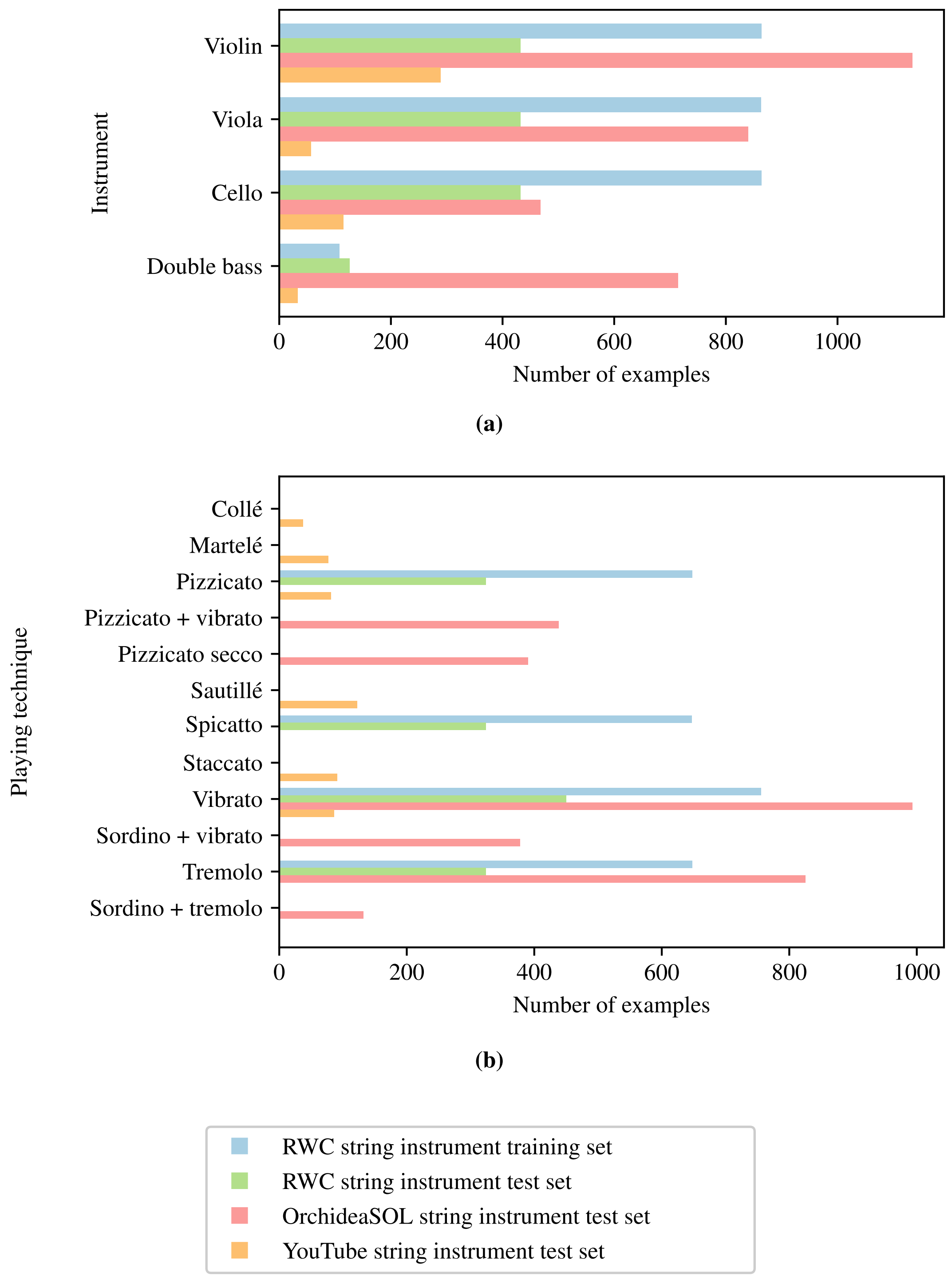


Figure 4.1. A breakdown of the total number of examples for each (a) instrument and (b) playing technique in the RWC subsets used for few-shot classifier training and standard classifier training, and the RWC, OrchideaSOL and YouTube string instrument test sets.

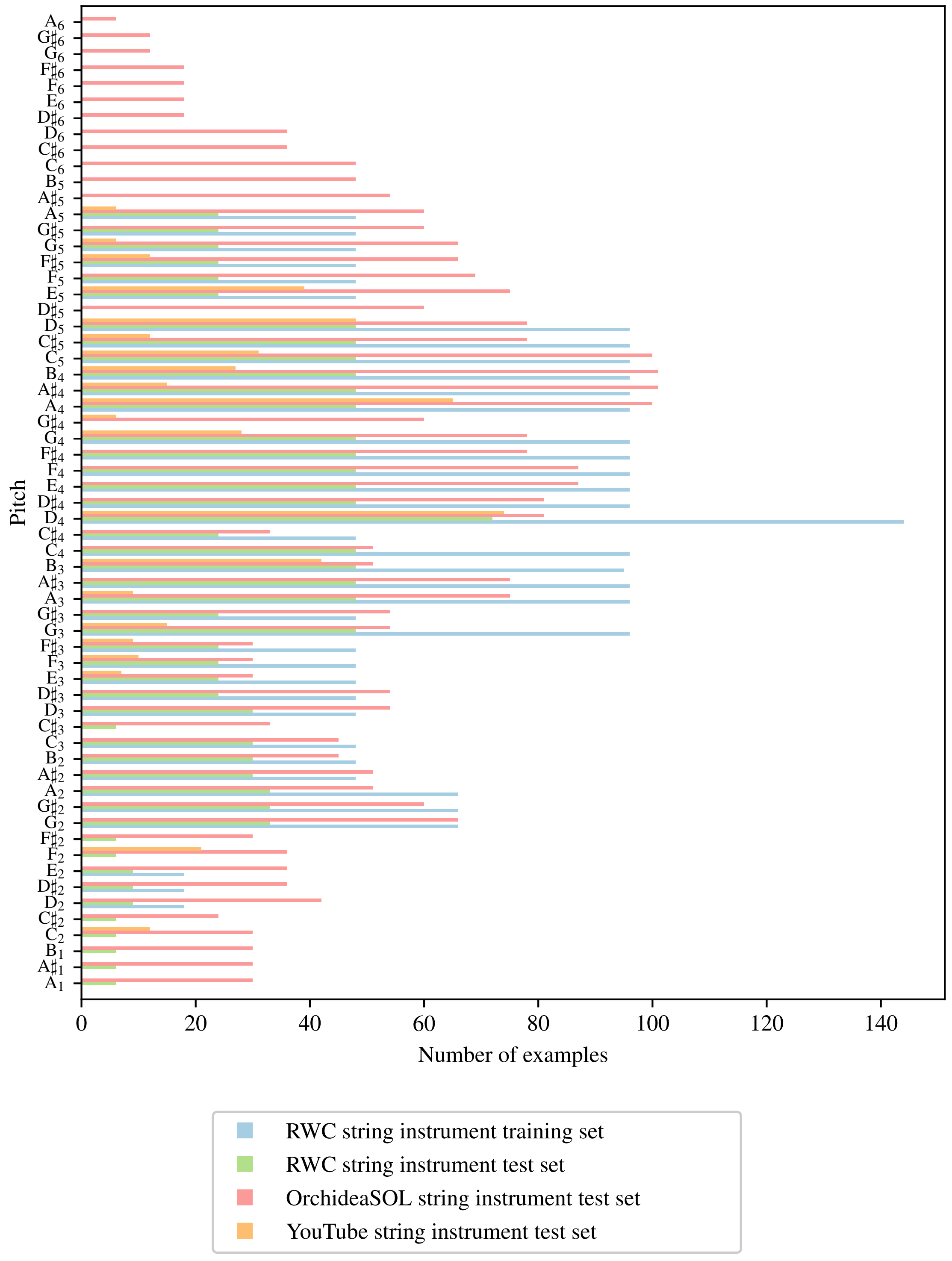


Figure 4.2. A breakdown of the total number of examples for each pitch in the RWC subsets used for few-shot classifier training and standard classifier training, and the RWC, OrchideaSOL and YouTube string instrument test sets.

4.2.3.2 OrchideaSOL test set

The OrchideaSOL string instrument subset is used for cross-dataset evaluation and as examples of previously unseen classes. The classes were restricted to those with at least six examples per class ($N_S + 3$). All string instruments are seen during both training and evaluation, however, the dataset, contains different playing techniques, as seen in Figure 4.1(b), as generalization to new playing styles is one of the main problems of interest. This set also contains pitches not seen during training, as seen in Figure 4.2. Subsets of this set containing only examples from single instruments are also used for single-instrument evaluation.

4.2.3.3 Real-world YouTube test set

The data extracted from YouTube tutorials contain new techniques not seen during training, as is shown in Figure 4.1(b), but are restricted to fewer pitches per playing technique and instrument (see Figure 4.2). The tutorials are biased towards some pitches that are commonly used for demonstration, thus less variety in pitch is seen compared to the other datasets. The classes were restricted to those with at least six examples per class ($N_S + 3$).

4.2.3.4 RWC Test set with accompaniment

Polyphonic dual-instrument sets were compiled from the RWC subset for the training and evaluation of instrument, playing technique and pitch classification of string instrument examples that could realistically be encountered in music with accompaniment. The training and evaluation sets with accompaniment contain the same classes as the RWC string instrument training and test sets, as shown in Table 4.1. Piano accompaniment was chosen as there is a vast chamber music literature in classical music for violin and piano works. The RWC string instrument training sets and test set were adapted by adding two piano notes, also from the RWC instrument dataset, to each example to form chords with the string instrument note as the tonic note. The chords formed were randomly chosen to be major, minor, augmented or diminished triads, as is shown in Figure 4.3. The chords can also be inverted.

Relative scaling of the piano notes was done through listening tests and the resultant mixtures were checked by ear to ensure that the accompaniment does not overwhelm the string instrument, nor gets drowned out.

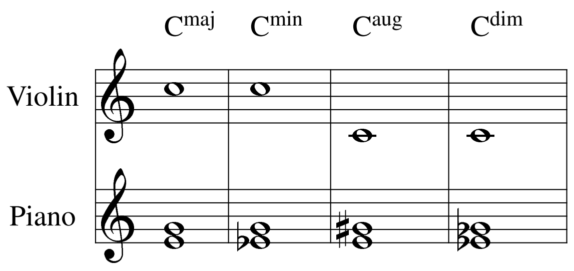


Figure 4.3. Piano accompaniment example chords for accompanying RWC string instrument playing technique note examples. Major, minor, augmented and diminished triads are shown respectively with C as the tonic note.

4.2.3.5 RWC cross-instrument clarinet test set

A subset of clarinet playing technique examples from RWC was also considered for cross-instrument evaluation. Examples from all three performers were used. There are 120 suitable clarinet playing technique and pitch combinations for this.

4.3 NEURAL NETWORK ARCHITECTURES

Prototypical networks are compared with standard classifier models, as well as different architectures and input features. The same architectures are shared between the prototypical models and the standard classifier models, only differing in the output layers. For the prototypical models, the output layer is a high-dimensional embedding, the size of which is determined with Bayesian optimization; for the standard classifier models, the output layer is a vector of saliences indicating class probabilities for each of the classes. For each architecture and input feature combination, a prototypical model and a standard classifier model were trained.

ANN architectures were selected based on previous few-shot learning implementations that have performed well on sound event detection or similar problems. DNN, CNN, RNN and CRNN networks were trained and compared on joint classification of string instrument, playing technique and pitch. For both prototypical and standard classifier models, two models were trained per architecture: one with a log-mel spectrogram input and one with a CQT input. The first 100 frames of an input feature are used, with zero padding for very short examples, which corresponds to 1.16 s at a sampling frequency of $F_s = 22050$ Hz, generally covering the onset and some sustain and fade. Zero mean unit variance normalization is applied to inputs, with normalization applied to each sample [65]. The normalization is given by

$$\mathbf{X} \leftarrow \frac{\mathbf{X} - \mu(\mathbf{X})}{\sigma(\mathbf{X}) + \varepsilon}, \tag{4.1}$$

where $\mu(\cdot)$ and $\sigma(\cdot)$ are the mean and standard deviation respectively. ε is a near-zero constant for computational stability.

The models were implemented and trained using the Keras library in Python [117]. The loss function for the prototypical networks was implemented using Tensorflow [118]. Custom algorithm implementations were verified through unit testing.

4.3.1 DNN

The simplest ANN classifier used is a FFNN. A three hidden layer DNN architecture is used with a flattening layer, three 1024 node fully connected layers with *tanh* activations and a *sigmoid* activation output layer. The architecture is shown in Figure 4.4.

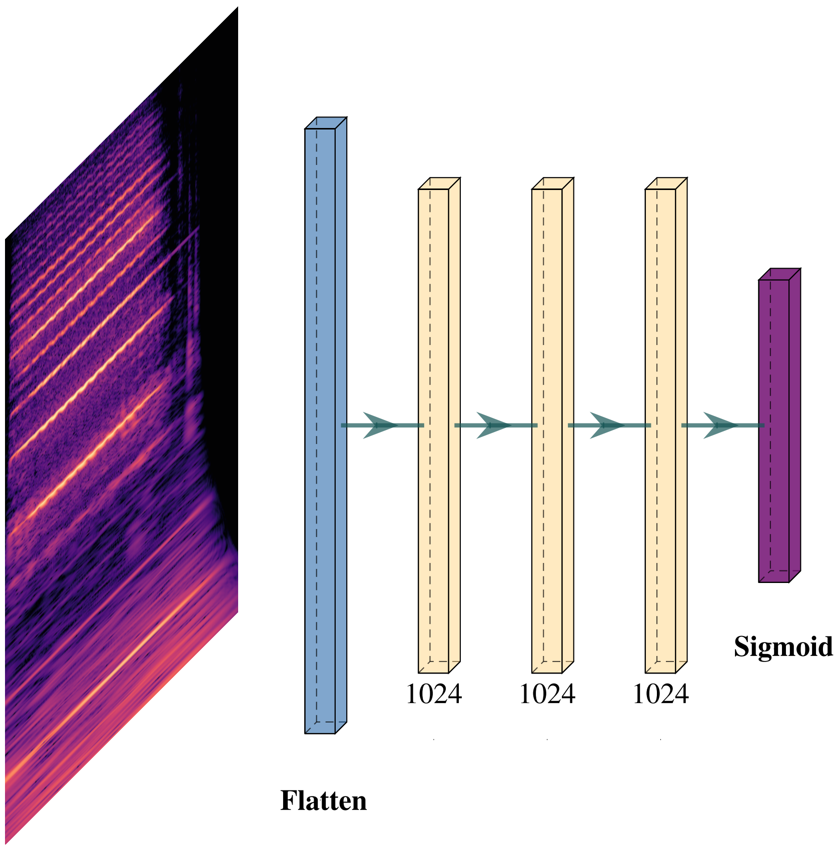


Figure 4.4. The DNN architecture of three fully connected hidden layers with *tanh* activations. The size of the flattening layer is dependent on the input feature dimensions and the output layer dimensions are found using Bayesian optimization for prototypical networks.

4.3.1.1 Activation functions

A *tanh* activation function is used for DNN hidden layer activations. This was chosen to mitigate weight explosion problems that were encountered with initial experiments with prototypical networks. The *tanh* function is given by

$$tanh(x) = \frac{e^{2x}-1}{e^{2x}+1} \tag{4.2}$$

A *sigmoid* activation function is used for the output activations, so as to approximate a class activation that is close to one for a true class and close to zero for all others. For prototypical networks this represents an encoding of the input; for standard classifiers the predicted class is given by the highest output activation. A logistic function is used:

$$f(x) = \frac{1}{1+e^{-x}}. \tag{4.3}$$

4.3.2 CNN

The CNN employs four blocks consisting of a 2-dimensional convolution with a 3×3 filter with a stride of 2 and valid padding, followed by a batch normalization layer, a leaky ReLu activation, a 2×2 max-pooling layer and finally a dropout layer with a rate of 0.25. The architecture is based on a CNN used for sound event detection [33]. The size of the convolution blocks were experimentally tuned by evaluating the performace of the model as a prototypical network on the validation set without training. The blocks have 128, 128, 64 and 32 filters respectively. The output of the last block is flattened and fed into a *sigmoid* activation output layer, as shown in Figure 4.5.

4.3.2.1 Leaky ReLU activation function

The leaky ReLU activation function is used for CNN filter activations. It is an adapted linear rectifier,

$$f(x) = \begin{cases} x, & \text{if} \, x \geq 0 \\ \alpha x, & \text{otherwise,} \end{cases} \tag{4.4}$$

where α is a parameter that controls how much a negative activation “leaks" through. Advantages include sparser activations and better gradient propagation compared to *sigmoid* and *tanh* activation functions [119].

4.3.3 RNN

Figure 4.6 shows the RNN architecture: two bi-directional gated recurrent unit (GRU) layers with 128 and 64 nodes in each direction operating across the time dimension are used, followed by a flattening

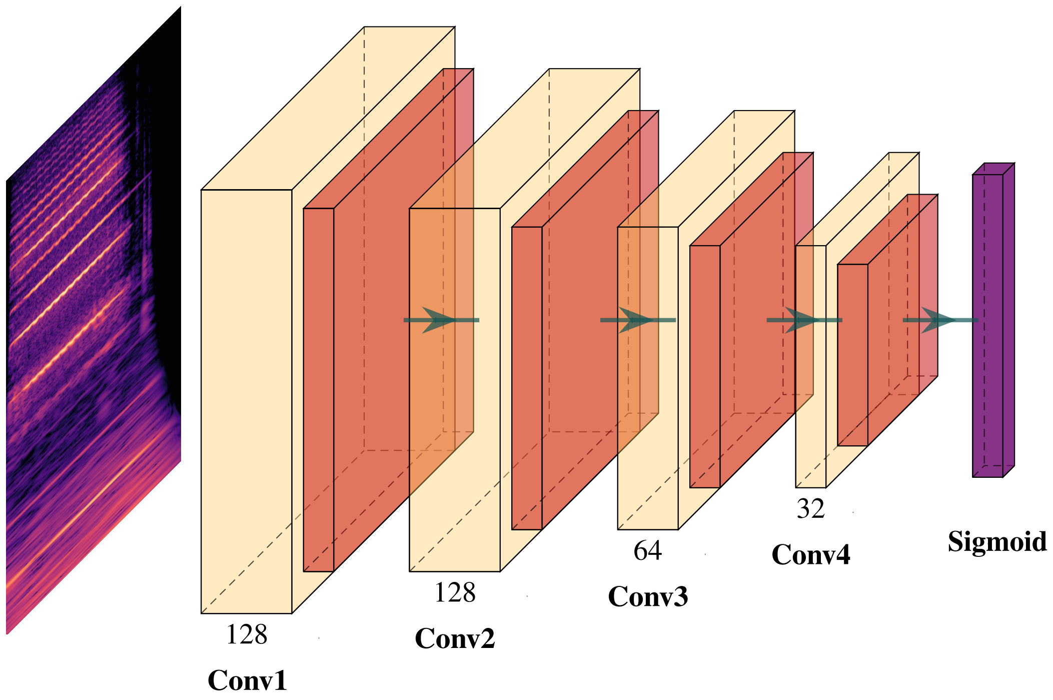


Figure 4.5. The CNN architecture of four convolution blocks with 3×3 filter with a stride of 2 with ReLu activation, followed by a 2×2 max-pooling layer. The output layer dimensions are found using Bayesian optimization for prototypical networks.

layer and a *sigmoid* activation output. The architecture is based on the RNN part of the architecture that inspired the CRNN [94].

4.3.3.1 GRU block

A GRU is an RNN encoder-decoder block that maintains a hidden state over time steps. Two gates are included in the unit: an update gate that decides whether to update the hidden state with a candidate hidden state and a forget gate that decides whether to forget the previous hidden state [88]. The hidden state and gate vectors at time t are then given by

$$\mathbf{z}_t = \sigma(\mathbf{W}_z\mathbf{x}_t + \mathbf{U}_z\mathbf{x}_{t-1} + \mathbf{b}_z), \tag{4.5}$$

$$\mathbf{r}_t = \sigma(\mathbf{W}_r\mathbf{x}_t + \mathbf{U}_r\mathbf{x}_{t-1} + \mathbf{b}_r), \tag{4.6}$$

$$\hat{\mathbf{h}}_t = tanh(\mathbf{W}_h\mathbf{x}_t + \mathbf{U}_h(\mathbf{r}_t \cdot \mathbf{h}_{t-1}) + \mathbf{b}_h), \tag{4.7}$$

$$\mathbf{h}_t = \mathbf{z}_t \cdot \mathbf{h}_{t-1} + (1 - \mathbf{z}_t) \cdot \hat{\mathbf{h}}_t, \tag{4.8}$$

where $\mathbf{x}_t$ is the input vector, $\mathbf{h}_t$ is the output vector, $\hat{\mathbf{h}}_t$ is the candidate updated hidden state vector, $\mathbf{z}_t$ is the update gate vector and $\mathbf{rz}_t$ is the forget gate vector. $\mathbf{W}$, $\mathbf{U}$ and $\mathbf{b}$ are weight vectors. $\cdot$ denotes element-

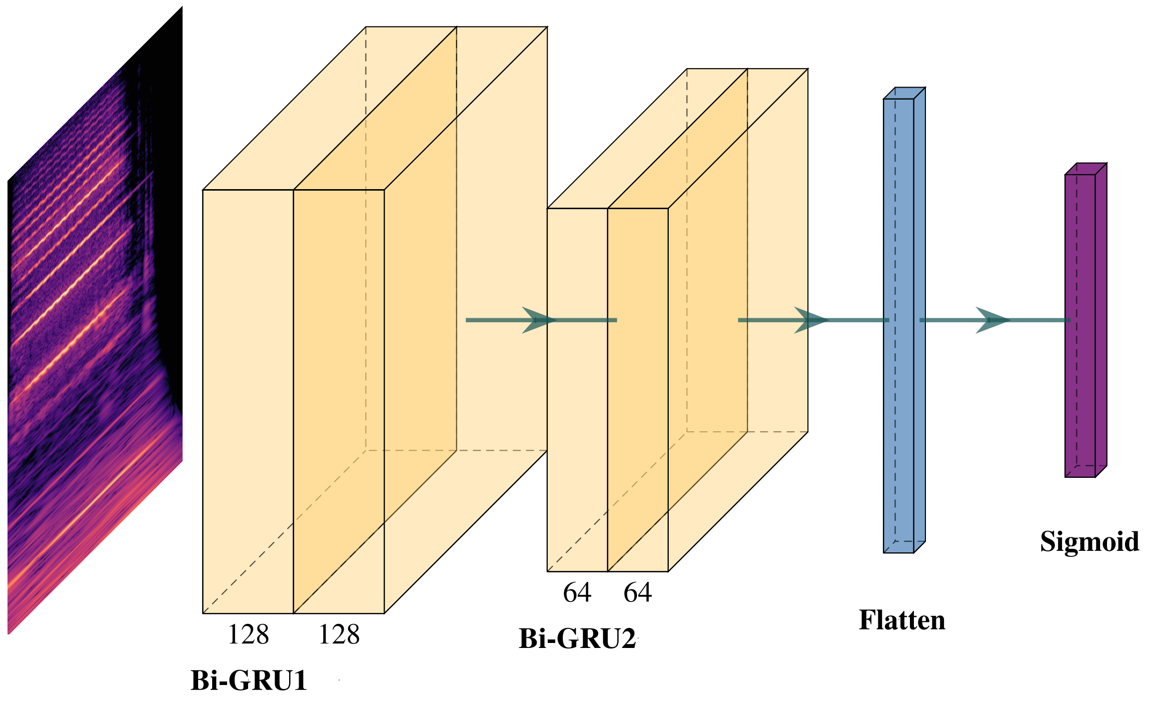


Figure 4.6. The RNN architecture of two bi-directional GRU layers operating over the time dimension. The size of the flattening layer is dependent on the input feature dimensions and the output layer dimensions are found using Bayesian optimization for prototypical networks.

wise multiplication. The original proposed algorithm uses *sigmoid* and *tanh* activation functions for the gates and the hidden state respectively, which are also used in this implementation [88].

4.3.4 CRNN

The CRNN combines the CNN and RNN architectures, as shown in Figure 4.7. Two CNN blocks with 128 filters each are used. The output of the second CNN block's filters are concatenated on the frequency axis before two bi-directional GRU layers with 128 and 64 nodes are applied in the time direction. This is followed by a flattening layer, a 512 node fully connected layer with *tanh* activations and a *sigmoid* activation output layer. The architecture is based on a CRNN used for sound event detection [94] with the sizes and number of convolution and GRU blocks based on experiments with untrained networks.

4.4 MODEL TRAINING FOR JOINT INSTRUMENT, PLAYING TECHNIQUE AND PITCH CLASSIFICATION

The prototypical networks were trained for 500 episodes using the Adam optimizer [110] (see Section 3.4.2) with default parameters and a learning rate that exponentially decreases over the course of

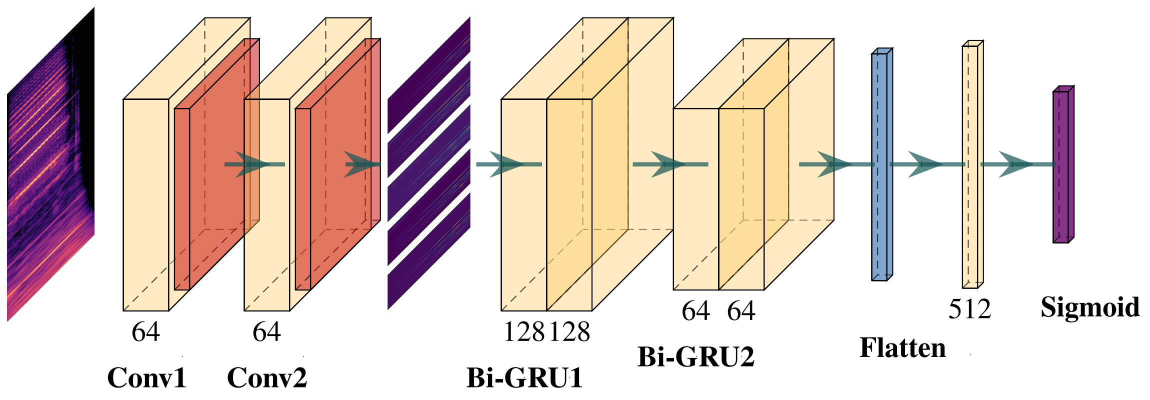


Figure 4.7. The CRNN architecture. Two convolution blocks with 3×3 filters with a stride of 2 and ReLu activations, followed by a 2×2 max-pooling layer are employed. Convolution block 2's output is concatenated along the frequency axis and fed into two bi-directional GRU layers operating over the time dimension. After flattening, a fully connected hidden layer with *tanh* activations is also added. The size of the flattening layer is dependent on the input feature dimensions and the output layer dimensions are found using Bayesian optimization for prototypical networks.

training. Bayesian optimization was used to find the optimal number of classes per episode, embedding size and learning rate for each prototypical network over 10 trials per network architecture and input feature. The optimized hyperparameters that were found are summarized in Addendum A.1. All prototypical networks were trained on a subset of the RWC dataset consisting of string instrument examples recorded by two of the three performers. Recordings from the remaining performer are left out for evaluation. An 8 : 2 training to validation split is employed for Bayesian optimization to maximize classification *micro* F-measure. Prototypical networks are trained with a shot of $N_S = 3$ and $N_Q = 5$ query examples per episode class.

Standard classifier networks were also trained on the same data. A categorical cross-entropy loss was used with the Adam optimizer [110] (see Section 3.4.2). The best learning rate and batch size for the classifiers were determined using Bayesian optimization with each trial trained for 10 epochs. The optimized hyperparameters that were found are summarized in Addendum A.2. Each classifier was retrained for 100 epochs using the best parameters determined by Bayesian optimization. Transfer learning was employed to adapt standard classifiers for evaluation on new datasets.

4.4.1 Transfer learning for standard classifiers

In order to be able to compare the standard models to the prototypical models on evaluation datasets with different classes from the training set, transfer learning is applied. The output layer of each standard classifier model is replaced with a new output layer corresponding to the classes in the new set and the model is trained for a further 10 epochs on the subset of support examples for the new set, using the original hyperparameters. The number of transfer learning epochs was chosen as a balance between the expected model improvement on the new data and the training time.

4.4.2 Instrument playing technique classification models

A 3-shot prototypical CQT CNN model and standard classifier model CQT CNN were also trained on the RWC training set for the problem of instrument playing technique classification without regard for pitch. The RWC training set was again used for training, but with the class pitch label ignored. The models were trained with the best hyperparameters found from Bayesian optimization.

4.4.3 Instrument pitch classification models

A 3-shot prototypical CQT CNN model and standard classifier model CQT CNN were also trained on the RWC training set for the problem of instrument pitch classification without regard for playing technique. The RWC training set with the class playing technique label ignored was used for training. The models were trained with the best hyperparameters found from Bayesian optimization.

4.4.4 Classification with accompaniment models

A 3-shot prototypical CQT CNN model and standard classifier model CQT CNN were also trained on the RWC training set with piano accompaniment for the problem of joint instrument, playing technique and pitch classification with accompaniment.

4.5 EVALUATION PROCEDURE

Prototypical models are evaluated by randomly sampling prototype examples for each class in the test set, from which prototypes are computed. Any class examples used for prototype calculation are left out during testing. Results on each dataset are averaged over five trials with prototype examples newly drawn for each trial. The number of query examples for each class is kept constant for different shot numbers, as well as for transfer learning.

Prototypical models are often evaluated in episodes, similarly to how they are trained [3], however, it was elected not to use this evaluation method, both to test the models under more realistic conditions, as would be expected when used in a real-world application where the number of potential classes is not constrained, but also to be able to compare to standard classifier models that are not evaluated in episodes. The models were evaluated with an F-measure, as has been done for prototypical models for drum transcription [32] and both the *micro* F-measure and *macro* F-measure are reported. The F-measure is given by the harmonic mean of precision and recall or by

$$F_1 = \frac{2tp}{2tp + fp + fn}, \tag{4.9}$$

where tp, fp and fn are the true positive count, false positive count and false negative count respectively. The *micro* F-measure is calculated by taking the F-measure for the global count of true positives, false positives and false negatives. The *macro* F-measure is obtained by taking the mean of the F-measures for each class.

True positives are taken as the count of examples for which a model is able to correctly predict the instrument, playing technique and pitch. This problem is denoted as INT. Other metrics of interest that are also recorded are the F-measures for playing techniques correctly predicted, regardless of instrument or pitch, denoted as T; pitches correctly identified, denoted as N, as well as instruments correctly identified, denoted as I.

Results for transfer learning are also averaged over five trials by retraining the models on three randomly sampled examples per class in each test set before evaluation.

4.6 CHAPTER CONCLUSION

The datasets for training and evaluating prototypical and standard classifier models were discussed. The DNN, CNN, RNN and CRNN architectures were given in detail and the training and evaluation procedures for prototypical classifiers and standard classifiers with transfer learning were given.

CHAPTER 5 RESULTS AND DISCUSSION

5.1 CHAPTER OVERVIEW

The results for all experiments are given and discussed. Results are given for evaluation of prototypical models and standard classifier models on the RWC string instrument set, OrchideaSOL set, YouTube tutorial examples and RWC clarinet set. Confusion matrices for the best models are shown and results for transfer learning for the standard classifier models are given for all datasets, along with a comparison of pitch classification results using the PYIN algorithm. Results are also given for the problems of single instrument playing technique and pitch classification, classification of examples with accompaniment, playing technique classification and pitch classification. The strengths and limitations of the models, the choice of time-frequency features, few-shot model shot and how few-shot learning compares to transfer learning are discussed.

Note on model descriptors

Models in results are denoted by classification algorithm, input feature and ANN architecture. E.g. *Prototypical 3-shot CQT CNN* is a prototypical model with a shot of $N_S = 3$, that takes a CQT as input feature and has a CNN architecture, while *Standard mel RNN* is a standard classifier model that takes a log-mel spectrogram as input, has an RNN architecture and has an output layer that correlates to class probabilities. Where models are trained on data other than the training sets for either prototypical or standard classifier training, this is also indicated, e.g. *Techniques prototypical 3-shot CQT CNN* was trained on the prototypical training set with class labels that only indicate instrument and playing technique.

5.2 EVALUATION ON RWC STRING INSTRUMENT EXAMPLES

Prototypical models were evaluated on string instrument recordings from the RWC dataset performer left out during training. From each class, 3 examples are used for prototypes and are not considered during evaluation. The evaluation set contains 4 playing techniques and 47 pitches across the 4 string instruments for a total of 234 classes and 720 examples, excluding examples used to compute prototypes. None of the playing techniques are unseen during training for any of the models, only the performer (see Figure 4.1(b)). Table 5.1 compares results for the prototypical models and Table 5.2 for standard classifier models. The 3-shot prototypical models are also evaluated with a shot of 1.

Table 5.1. Prototypical models classification F-measure for instrument, playing technique and note classification on string instrument examples from the RWC dataset that were recorded by a performer unseen during training. This set consists of 234 classes of which 12 were unseen before evaluation. Micro and macro F-measures are shown respectively for instrument, note and technique classification (INT), technique classification (T), note classification (N) and instrument classification (I). The best 3-shot prototypical model results are shown in bold and the best 1-shot prototypical model results are underlined.

Model	***micro* F-measure**				***macro* F-measure**			
	INT	T	N	I	INT	T	N	I
Prototypical 3-shot mel DNN	.576	.769	.832	.831	.539	.754	.79	.84
Prototypical 3-shot CQT DNN	.686	.877	.887	.854	.659	.87	.838	.85
Prototypical 3-shot mel CNN	.655	.917	.797	.863	.632	.911	.698	.87
Prototypical 3-shot CQT CNN	**.833**	**.954**	**.943**	**.92**	**.814**	**.953**	**.927**	**.925**
Prototypical 3-shot mel RNN	.695	.877	.879	.873	.669	.874	.823	.877
Prototypical 3-shot CQT RNN	.774	.908	.927	.901	.758	.906	.88	.896
Prototypical 3-shot mel CRNN	.665	.89	.848	.839	.641	.886	.771	.849
Prototypical 3-shot CQT CRNN	.778	.94	.936	.857	.764	.938	.897	.848
Prototypical 1-shot mel DNN	.488	.723	.78	.795	.443	.705	.742	.806
Prototypical 1-shot CQT DNN	.6	.832	.847	.833	.564	.823	.785	.832
Prototypical 1-shot mel CNN	.543	.887	.733	.804	.513	.879	.631	.811
Prototypical 1-shot CQT CNN	.743	.923	.905	.887	.711	.919	.879	.894
Prototypical 1-shot mel RNN	.588	.828	.827	.834	.549	.823	.771	.842
Prototypical 1-shot CQT RNN	.665	.852	.882	.86	.641	.848	.816	.852
Prototypical 1-shot mel CRNN	.574	.855	.81	.795	.541	.85	.729	.809
Prototypical 1-shot CQT CRNN	.701	.909	.9	.818	.673	.906	.85	.82

Table 5.2. Standard classifier models classification F-measure for instrument, playing technique and note classification on string instrument examples from the RWC dataset that were recorded by a performer unseen during training. This set consists of 234 classes of which 12 were unseen before evaluation. Micro and macro F-measures are shown respectively for instrument, note and technique classification (INT), technique classification (T), note classification (N) and instrument classification (I). The best standard classifier results are shown in italics.

Model	***micro* F-measure**				***macro* F-measure**			
	INT	T	N	I	INT	T	N	I
Standard mel DNN	.484	.724	.707	.831	.431	.701	.655	.832
Standard CQT DNN	*.662*	*.844*	*.891*	*.84*	*.618*	*.834*	*.859*	*.827*
Standard mel CNN	.0759	.535	.153	.443	.0543	.51	.116	.42
Standard CQT CNN	.319	.751	.493	.792	.27	.743	.462	.789
Standard mel RNN	.432	.755	.68	.781	.382	.743	.65	.792
Standard CQT RNN	.344	.617	.675	.716	.283	.61	.609	.681
Standard mel CRNN	.391	.76	.546	.672	.329	.754	.486	.68
Standard CQT CRNN	.567	.781	.77	.76	.513	.775	.726	.753

The prototypical models consistently outperform the standard classifier models, with the best model across all metrics being the prototypical 3-shot CNN with CQT input features. Among the standard models, however, the CQT DNN model performs the best. Models with CQT features also generally outperform models with mel features. The prototypical models all perform better with 3-shot prototypes for evaluation — on which they were trained — compared to 1-shot prototypes for evaluation. The 1-shot prototypes still generally outperform the standard classifier models.

Figures 5.1 and 5.2 show the F-measures of the standard classifier models on the RWC string instrument evaluation set after each of the additional training epochs during transfer learning. For all models and architectures the F-measure keeps climbing for each additional epoch.

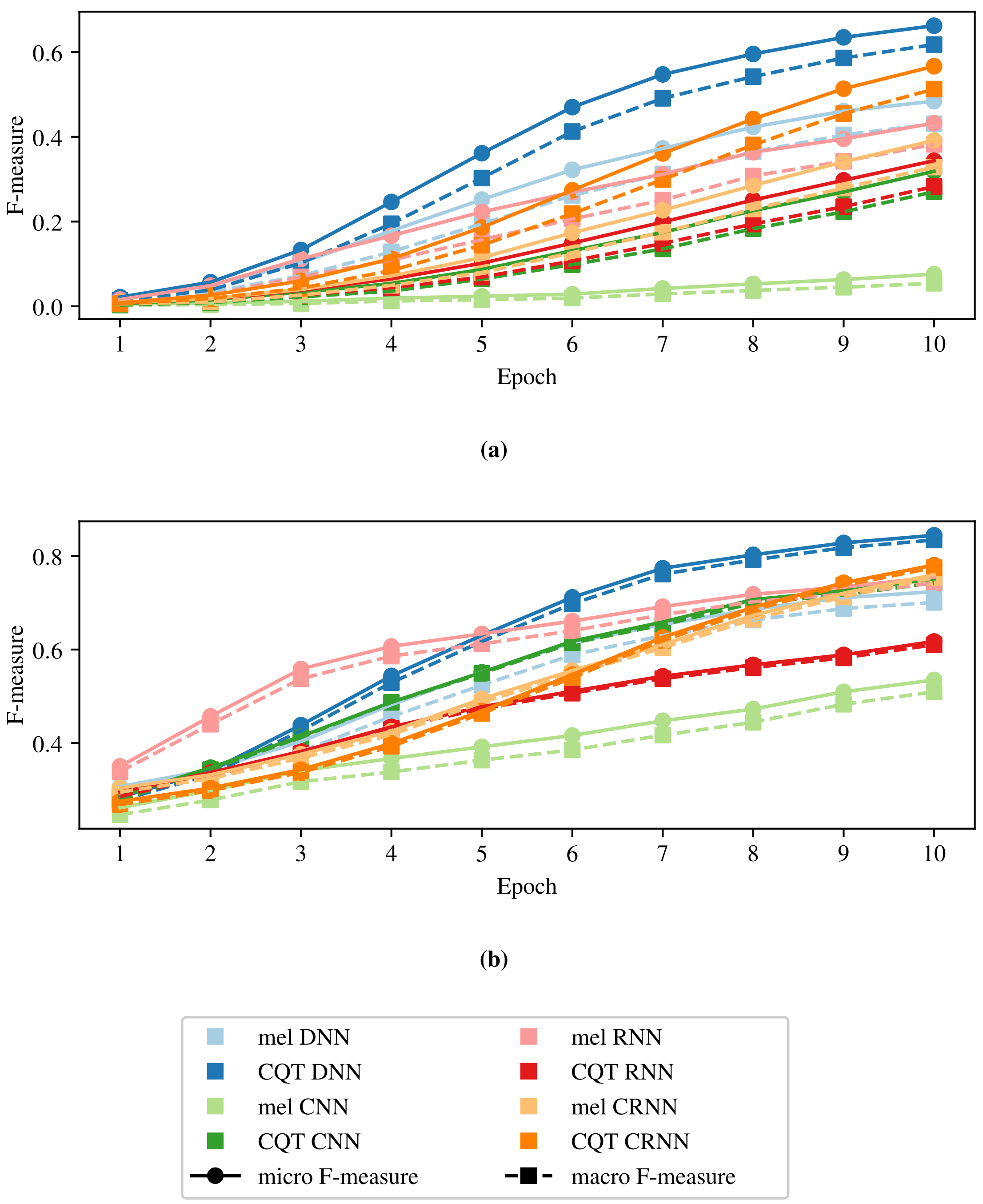
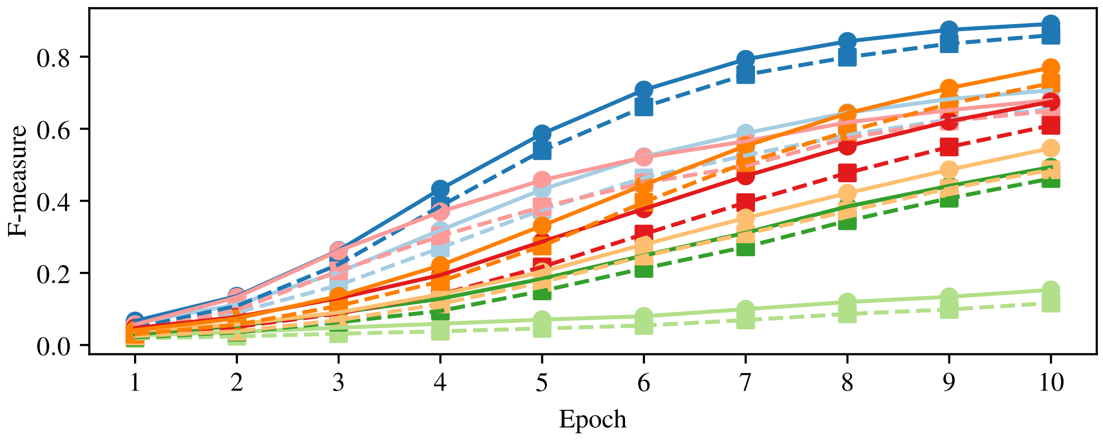
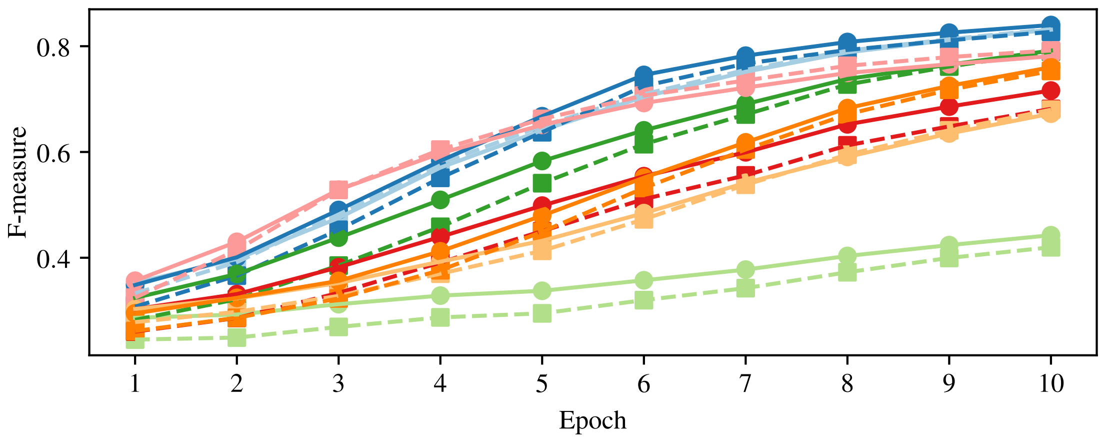


Figure 5.1. Classification F-measure for (a) joint instrument, pitch and technique classification (INT) and (b) technique classification (T) on string instrument examples from the RWC dataset for standard classifier models that were trained further on the prototype subset of the RWC string instrument test set. Classification F-measure is shown after each of the 10 additional training epochs.



(a)



(b)

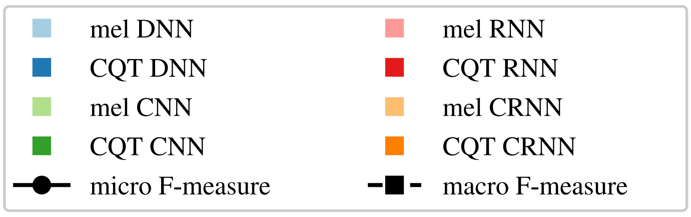


Figure 5.2. Classification F-measure for (a) pitch classification (N) and (b) instrument classification (I) on string instrument examples from the RWC dataset for standard classifier models that were trained further on the prototype subset of the RWC string instrument test set. Classification F-measure is shown after each of the 10 additional training epochs.

The confusion matrices for classification on each of the sub-problems is shown in Figures 5.3 and 5.4 for the best prototypical model — the 3-shot CQT CNN. Confusion matrices for the CQT DNN standard classifier model are shown in Figures 5.5 and 5.6. The confusion matrices show some confusion between violin and viola for the prototypical model, but very little confusion on any of the other problems. The standard classifier model shows more confusion than the prototypical model, with more noticeable confusion between *tremolo* and *vibrato*, as well as some errors with pitch classification.

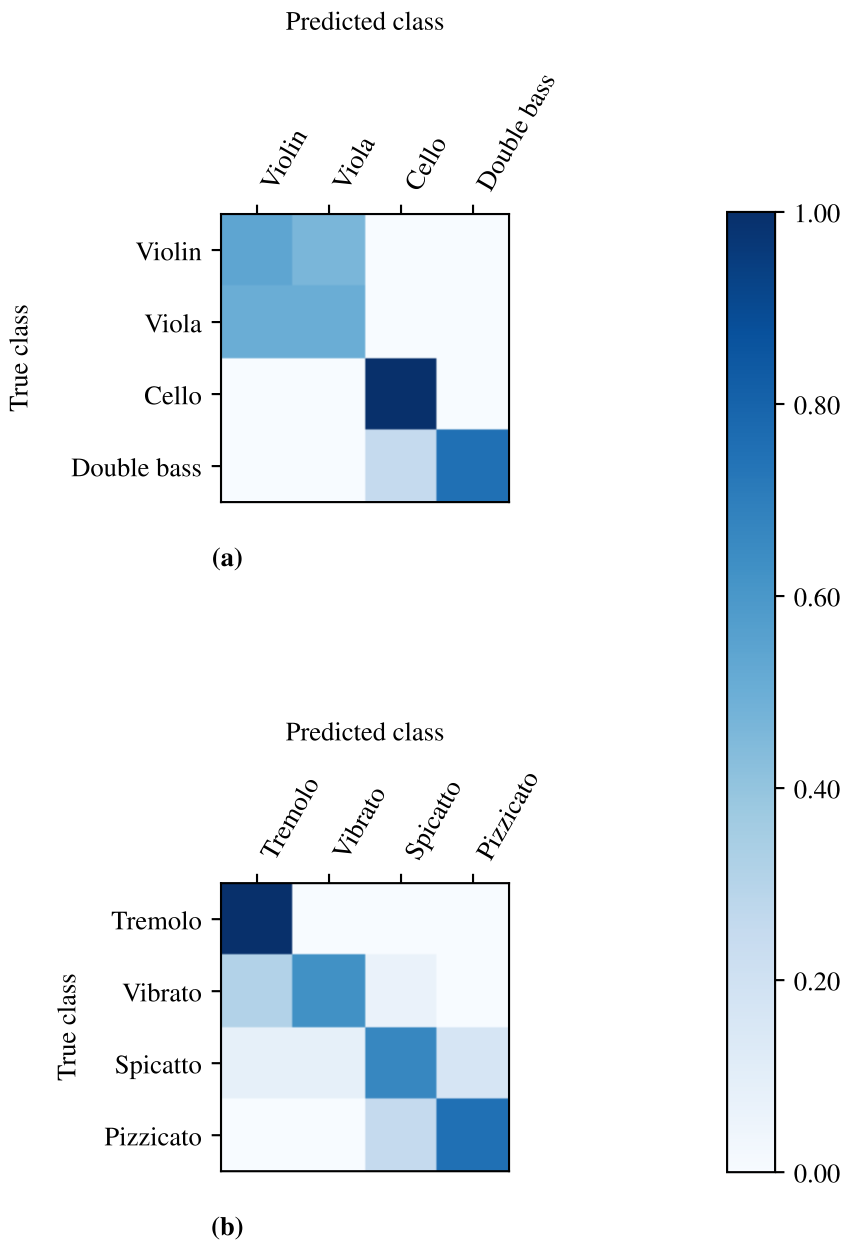


Figure 5.3. Confusion matrices for the prototypical 3-shot CQT CNN model evaluated on the RWC strings evaluation set on (a) instrument classification (I) and (b) playing technique classification (T). Confusion matrices are normalized to the total number of examples for each true class in the set.

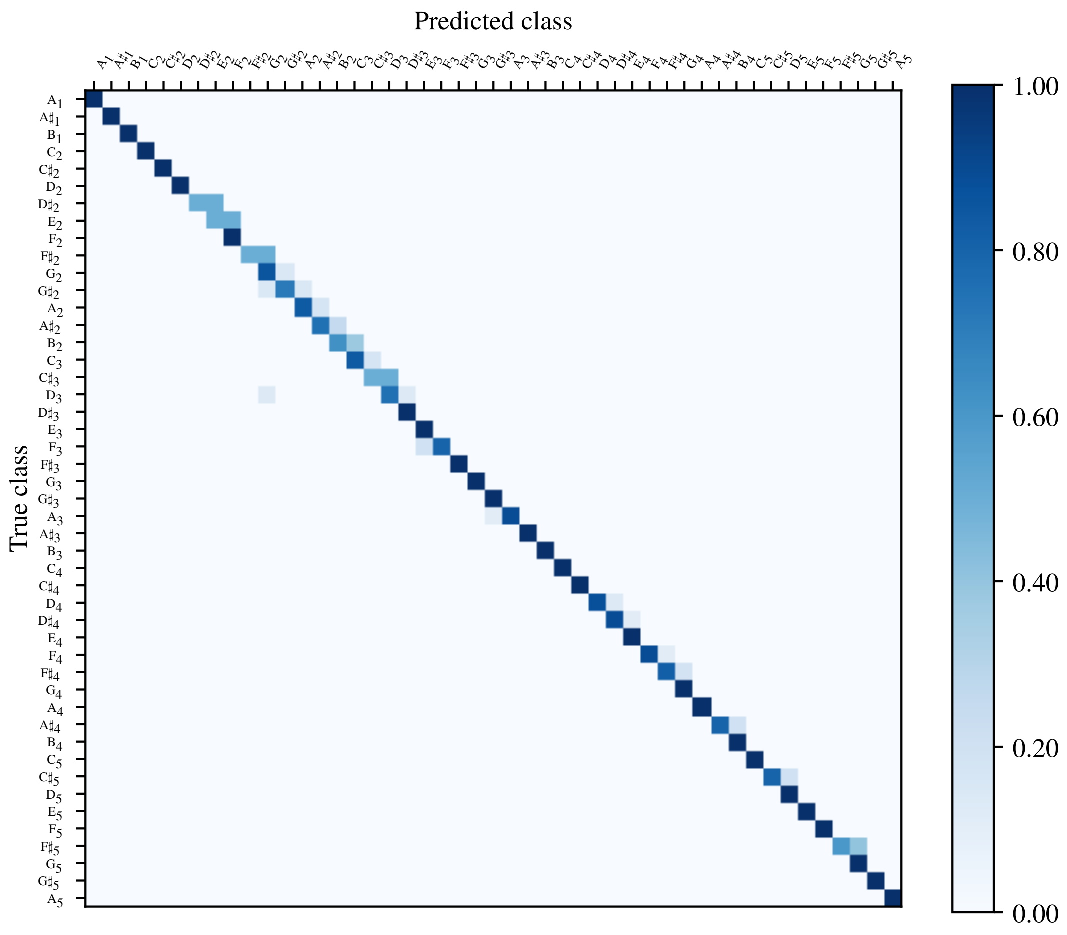
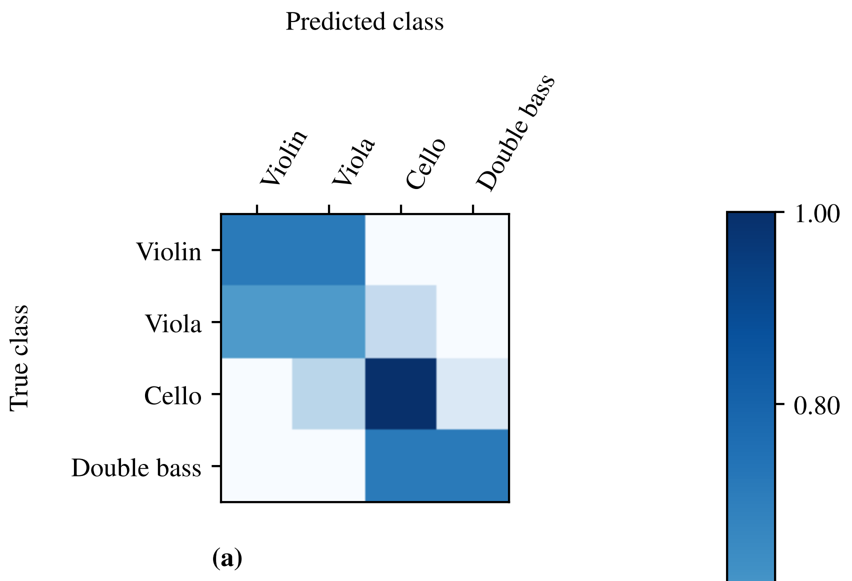


Figure 5.4. Confusion matrix for the prototypical 3-shot CQT CNN model evaluated on the RWC strings evaluation set on pitch classification (N). The confusion matrix is normalized to the total number of examples for each true class in the set.



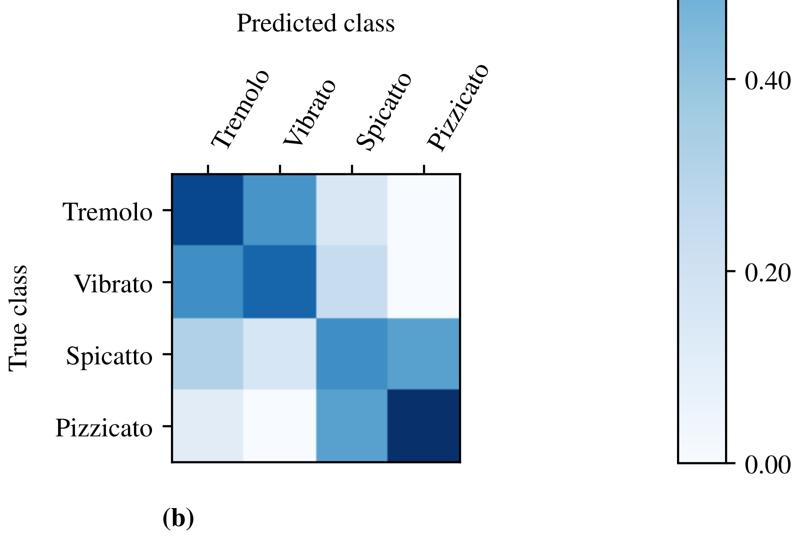


Figure 5.5. Confusion matrices for the standard classifier CQT DNN model evaluated on the RWC strings evaluation set on (a) instrument classification (I) and (b) playing technique classification (T). Confusion matrices are normalized to the total number of examples for each true class in the set.

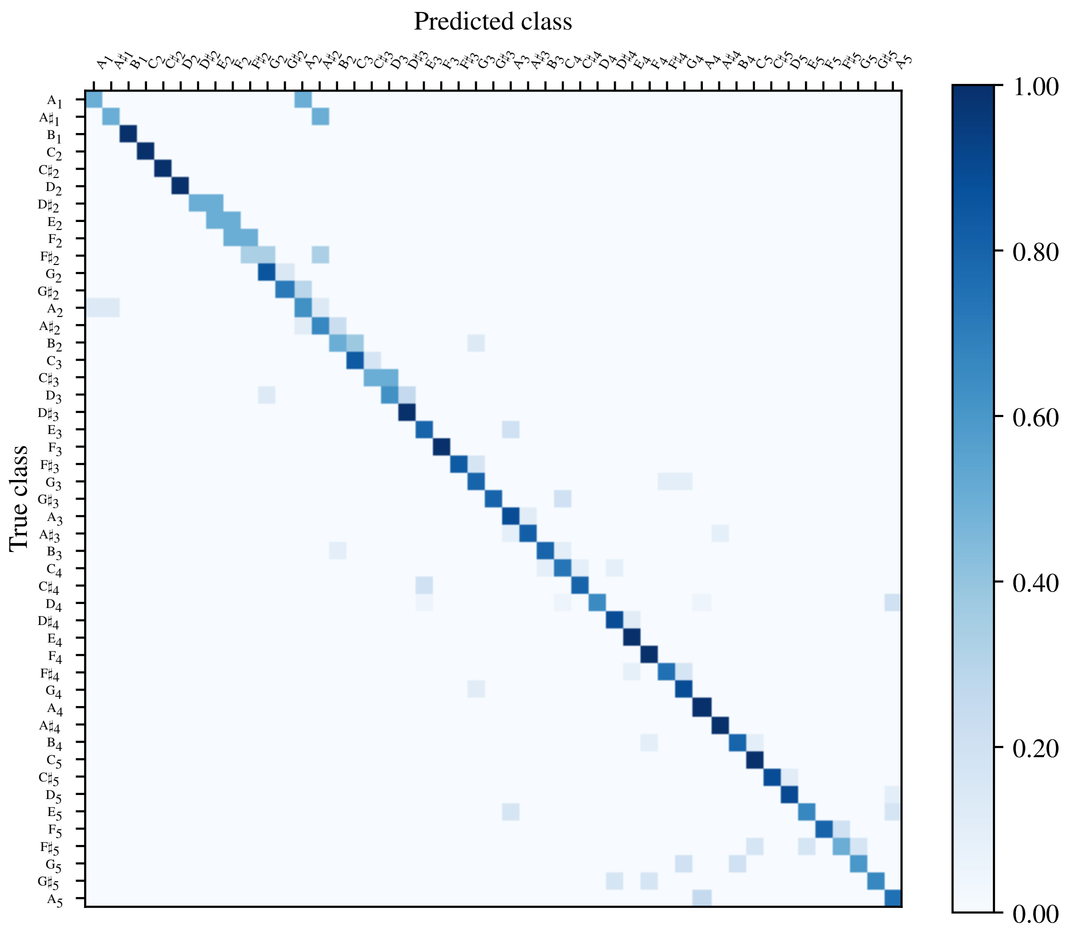


Figure 5.6. Confusion matrix for the standard classifier CQT DNN model evaluated on the RWC strings evaluation set on pitch classification (N). The confusion matrix is normalized to the total number of examples for each true class in the set.

5.3 CROSS-DATASET EVALUATION ON ORCHIDEASOL

The models were cross-evaluated on the string instrument playing technique note recordings of the OrchideaSOL dataset. The OrchideaSOL evaluation set contains 61 pitches and 6 playing techniques of which some were unseen during training (see Figure 4.1(b)). In total, the set contains 439 classes and 2278 examples, excluding prototypes. Transfer learning was first applied to standard models before evaluation. These results are shown in Table 5.3 for prototypical models and in Table 5.4 for standard classifier models. The results of transfer learning for the standard classifiers over the additional epochs are shown in Figures B.1 and B.2.

Table 5.3. Prototypical model classification F-measure for instrument, playing technique and pitch classification on previously unseen string instrument examples from the OrchideaSOL string instrument subset, including playing techniques not seen during training. This set consists of 439 classes of which 325 were unseen before evaluation. Micro and macro F-measures are shown respectively for instrument, note and technique classification (INT), technique classification (T), pitch classification (N) and instrument classification (I). The best 3-shot prototypical model results are shown in bold and the best 1-shot prototypical model results are underlined.

Model	***micro* F-measure**				***macro* F-measure**			
	INT	T	N	I	INT	T	N	I
Prototypical 3-shot mel DNN	.438	.572	.904	.782	.416	.521	.867	.77
Prototypical 3-shot CQT DNN	.499	.632	.883	.857	.475	.556	.811	.847
Prototypical 3-shot mel CNN	.528	.743	.839	.816	.51	.668	.788	.806
Prototypical 3-shot CQT CNN	**.664**	.731	**.964**	**.929**	**.64**	.667	**.947**	**.929**
Prototypical 3-shot mel RNN	.55	.693	.921	.835	.535	.634	.885	.827
Prototypical 3-shot CQT RNN	.653	**.736**	.951	.908	.635	**.676**	.925	.906
Prototypical 3-shot mel CRNN	.515	.7	.891	.791	.495	.636	.839	.784
Prototypical 3-shot CQT CRNN	.554	.68	.92	.849	.533	.602	.869	.845
Prototypical 1-shot mel DNN	.37	.531	.867	.743	.346	.479	.811	.728
Prototypical 1-shot CQT DNN	.428	.598	.85	.813	.39	.517	.767	.802
Prototypical 1-shot mel CNN	.442	.705	.794	.772	.415	.629	.725	.756
Prototypical 1-shot CQT CNN	.564	.682	.932	.877	.536	.61	.9	.878
Prototypical 1-shot mel RNN	.448	.634	.886	.779	.424	.574	.834	.767
Prototypical 1-shot CQT RNN	.536	.665	.925	.854	.507	.596	.899	.849
Prototypical 1-shot mel CRNN	.445	.664	.847	.763	.413	.594	.77	.751
Prototypical 1-shot CQT CRNN	.464	.623	.895	.8	.439	.545	.839	.792

Table 5.4. Standard classifier model classification F-measure for instrument, playing technique and pitch classification on previously unseen string instrument examples from the OrchideaSOL string instrument subset, including playing techniques not seen during training. This set consists of 442 classes of which 328 were unseen before evaluation. Micro and macro F-measures are shown respectively for instrument, note and technique classification (INT), technique classification (T), pitch classification (N) and instrument classification (I). The best standard classifier results are shown in italics.

Model	***micro* F-measure**				***macro* F-measure**			
	INT	T	N	I	INT	T	N	I
Standard mel DNN	.379	.535	.654	.768	.32	.486	.634	.746
Standard CQT DNN	*.566*	.669	*.802*	.884	*.52*	.602	*.793*	.878
Standard mel CNN	.177	.618	.29	.547	.136	.514	.266	.533
Standard CQT CNN	.478	.668	.665	*.922*	.422	.611	.649	*.918*
Standard mel RNN	.399	.583	.654	.774	.354	.542	.619	.757
Standard CQT RNN	.382	.585	.655	.745	.317	.523	.639	.726
Standard mel CRNN	.363	.623	.571	.721	.299	.556	.546	.698
Standard CQT CRNN	.528	*.671*	.739	.826	.481	*.615*	.724	.818

The 3-shot prototypical CQT CNN also performs best on cross-evaluation. The F-measure for all models on the INT and T problems are notably lower than on the N and I problems, which was not seen during evaluation on the RWC test set.

Figures 5.7 and 5.8 show the confusion matrices for each sub-problem on the OrchideaSOL cross-dataset evaluation set for the prototypical 3-shot CQT CNN. There is still some confusion between violin and viola, but also some confusion between the variations on *pizzicato* and *tremolo* with and without a mute. *Vibrato* with or without a mute is also confused for *tremolo*. There is little significant pitch confusion, except for a few octave errors. Additional results on OrchideaSOL for all four string instruments are shown in Appendix B.1.

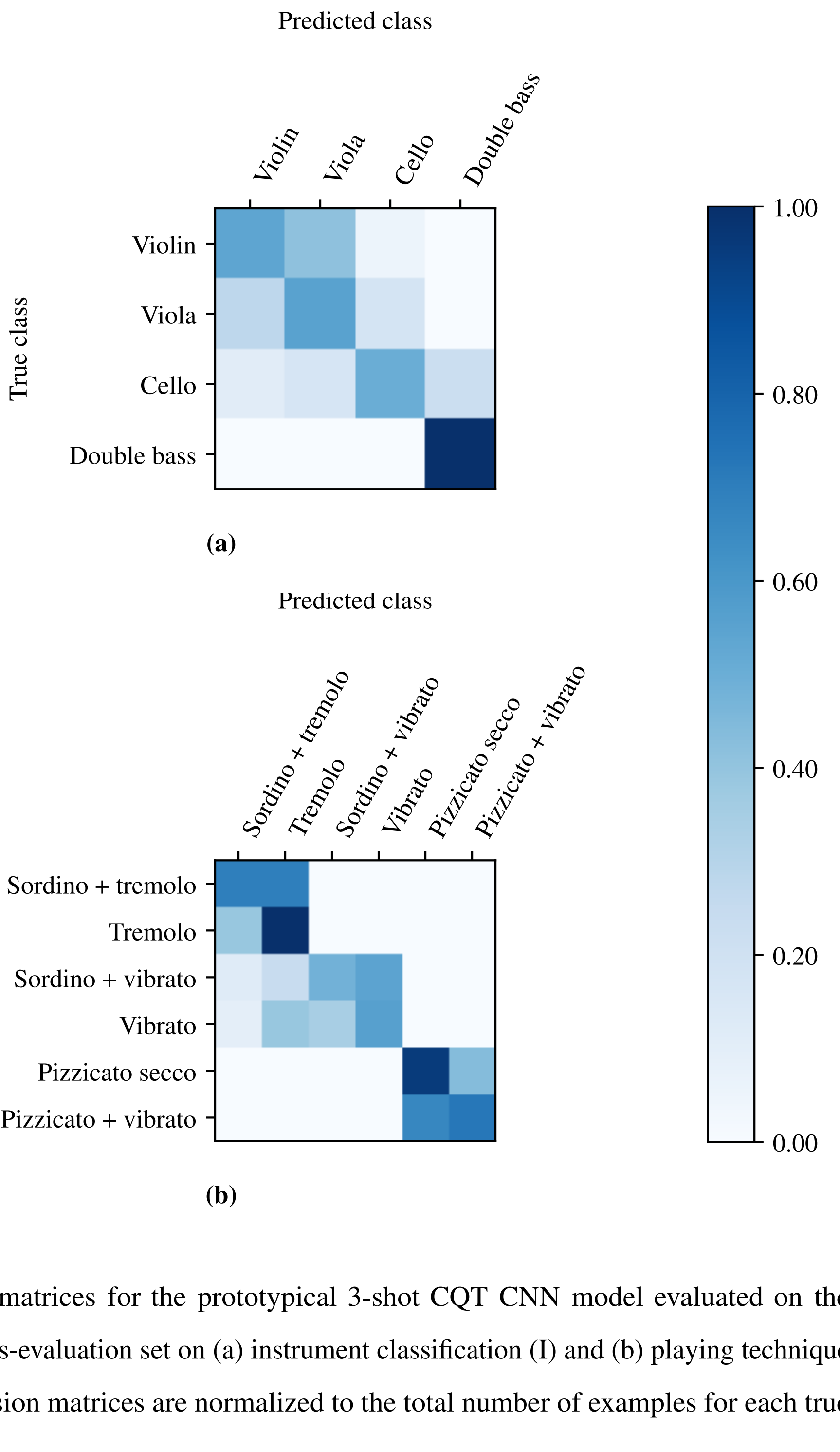


Figure 5.7. Confusion matrices for the prototypical 3-shot CQT CNN model evaluated on the OrchideaSOL strings cross-evaluation set on (a) instrument classification (I) and (b) playing technique classification (T). Confusion matrices are normalized to the total number of examples for each true class in the set.

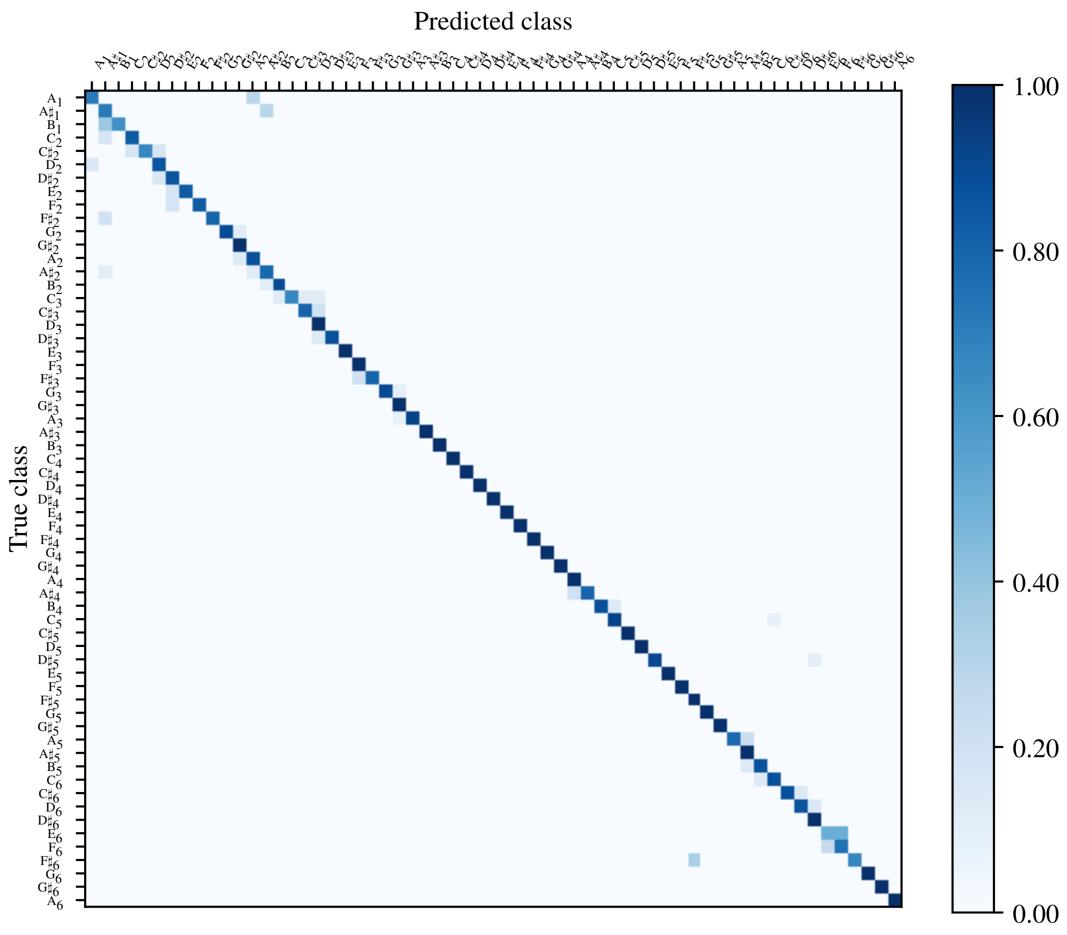


Figure 5.8. Confusion matrix for the prototypical 3-shot CQT CNN model evaluated on the OrchideaSOL strings cross-evaluation set on pitch classification (N). The confusion matrix is normalized to the total number of examples for each true class in the set.

The best 3-shot prototypical model — the 3-shot prototypical CQT CNN — was also evaluated on examples from only one instrument at a time from the OrchideaSOL dataset. The prototypical model was evaluated for both the case of classifying single instrument classes with all string instrument prototypes and with only the prototypes for the target instrument. The results are shown in Table 5.5.

Table 5.5. Classification F-measure for instrument, playing technique and pitch classification on previously unseen examples for single instruments from the OrchideaSOL string instrument subset, including playing techniques not seen during training. Micro and macro F-measures are shown respectively for instrument, pitch and technique classification (INT), technique classification (T), pitch classification (N) and instrument classification (I). The best results are shown in bold.

Model	***micro* F-measure**				***macro* F-measure**			
	INT	T	N	I	INT	T	N	I
OrchideaSOL violin test set								
Prototypical 3-shot CQT CNN with violin prototypes[1]	**.598**	**.648**	.941	—	**.583**	**.641**	.905	—
Prototypical 3-shot CQT CNN with string instrument prototypes[2]	.552	.631	**.951**	.917	.472	.621	**.924**	.351
OrchideaSOL viola test set								
Prototypical 3-shot CQT CNN with viola prototypes	**.745**	**.747**	.996	—	**.693**	**.68**	**.996**	—
Prototypical 3-shot CQT CNN with string instrument prototypes	.656	.712	**.997**	.898	.529	.551	.991	.315
OrchideaSOL cello test set								
Prototypical 3-shot CQT CNN with cello prototypes	**.855**	**.87**	**.982**	—	**.843**	**.748**	**.979**	—
Prototypical 3-shot CQT CNN with string instrument prototypes	.742	.813	.977	.877	.579	.707	.964	.234
OrchideaSOL double bass test set								
Prototypical 3-shot CQT CNN with double bass prototypes	**.769**	.825	.935	—	**.721**	.759	.95	—
Prototypical 3-shot CQT CNN with string instrument prototypes	.757	**.825**	**.938**	.987	.687	**.761**	**.952**	.497

[1] CNN with constant-Q transform input evaluated with only prototypes computed from OrchideaSOLviolin classes; other abbreviations are similarly explained.

[2] CNN with constant-Q transform input evaluated with prototypes computed from all OrchideaSOL string instrument classes.

The 3-shot prototypical CQT CNN performed much better on only a single instrument than on all string instruments, except for on violin. Only providing prototypes for the target instrument improved results on most instruments and problems.

5.4 EVALUATION ON REAL-WORLD YOUTUBE EXAMPLES

The prototypical models and standard classifier models were also evaluated on single-note violin, viola, cello and double bass examples extracted from YouTube tutorials. This gives an idea of how the models would perform in a practical real-world scenario such as recording a new performer outside of a studio environment and then evaluating their playing techniques. This also gives an indication on how the models perform when prototypes and query examples are not restricted to being recorded by the same performer. This set contains 6 playing techniques of which 4 are unseen before evaluation (see Figure 4.1(b)), and 21 pitches across violin, viola, cello and double bass. The set has a total of 46 classes with 356 examples, excluding prototypes. Evaluation results are shown in Tables 5.6 and 5.7.

Table 5.6. Prototypical model classification F-measure for instrument, playing technique and pitch classification on string instrument examples extracted from YouTube tutorials, including playing techniques not seen during training. This set consists of 46 classes of which 33 were unseen before evaluation. Micro and macro F-measures are shown respectively for pitch and technique classification (NT), technique classification (T), pitch classification (N) and instrument classification (I). The best 3-shot prototypical model results are shown in bold and the best 1-shot prototypical model results are underlined.

Model	***micro* F-measure**				***macro* F-measure**			
	INT	T	N	I	INT	T	N	I
Prototypical 3-shot mel DNN	.723	.793	.898	.884	.701	.776	.888	.883
Prototypical 3-shot CQT DNN	.753	.827	.918	.919	.728	.808	.884	.916
Prototypical 3-shot mel CNN	.686	.779	.843	.86	.696	.768	.843	.863
Prototypical 3-shot CQT CNN	**.852**	**.912**	**.963**	**.953**	**.829**	**.909**	**.962**	**.951**
Prototypical 3-shot mel RNN	.762	.812	.915	.899	.756	.793	.915	.915
Prototypical 3-shot CQT RNN	.799	.869	.923	.928	.775	.859	.906	.919
Prototypical 3-shot mel CRNN	.728	.795	.902	.864	.724	.776	.906	.865
Prototypical 3-shot CQT CRNN	.761	.815	.946	.949	.723	.808	.932	.939
Prototypical 1-shot mel DNN	.664	.751	.87	.863	.625	.729	.833	.855
Prototypical 1-shot CQT DNN	.664	.784	.872	.873	.644	.756	.824	.871
Prototypical 1-shot mel CNN	.602	.714	.798	.824	.592	.703	.78	.825
Prototypical 1-shot CQT CNN	<u>.772</u>	<u>.849</u>	<u>.936</u>	<u>.924</u>	<u>.748</u>	<u>.841</u>	<u>.917</u>	<u>.921</u>
Prototypical 1-shot mel RNN	.694	.758	.882	.867	.683	.735	.871	.877
Prototypical 1-shot CQT RNN	.723	.811	.898	.907	.703	.797	.879	.896
Prototypical 1-shot mel CRNN	.641	.711	.873	.823	.636	.692	.855	.824
Prototypical 1-shot CQT CRNN	.696	.772	.902	.922	.655	.762	.866	.911

Table 5.7. Standard classifier models classification F-measure for instrument, playing technique and pitch classification on string instrument examples extracted from YouTube tutorials, including playing techniques not seen during training. Micro and macro F-measures are shown respectively for pitch and technique classification (NT), technique classification (T), pitch classification (N) and instrument classification (I). The best standard classifier results are shown in italics.

Model	***micro* F-measure**				***macro* F-measure**			
	INT	T	N	I	INT	T	N	I
Standard mel DNN	.476	.582	.493	.734	.443	.579	.5	.704
Standard CQT DNN	*.786*	*.827*	*.793*	*.926*	*.747*	*.809*	*.799*	*.923*
Standard mel CNN	.054	.258	.105	.445	.0395	.218	.0666	.3
Standard CQT CNN	.107	.285	.135	.476	.0941	.265	.112	.38
Standard mel RNN	.19	.345	.234	.585	.158	.343	.208	.493
Standard CQT RNN	.111	.321	.154	.506	.0967	.301	.138	.335
Standard mel CRNN	.155	.312	.192	.511	.137	.322	.168	.35
Standard CQT CRNN	.553	.632	.568	.788	.538	.64	.59	.765

As with the previous evaluations, the retrained standard models suffer, except for the DNN models, with the standard CQT DNN having the best *macro* F-measure on the joint instrument, pitch and technique (INT) problem. The few-shot models, especially the 3-shot models perform well, matching the high cross-evaluation results seen on other cross-dataset evaluation tasks. The prototypical 3-shot CQT CNN model is again the standout, performing the best of the prototypical models for both the joint instrument, pitch and technique (INT), the technique (T) classification and instrument classification (I) tasks. On the pitch (N) classification task, the 3-shot prototypical CQT RNN performs the best.

The confusion matrices for the 3-shot CQT CNN are shown for all the sub-problems on this set in Figures 5.9 and 5.10. There is less instrument confusion than on other sets, as well as less clear playing technique confusion. There are some bigger mistakes with pitch detection, such as F$\sharp_3$ exclusively being classified as G$_3$. Additional results are shown in Appendix B.2.

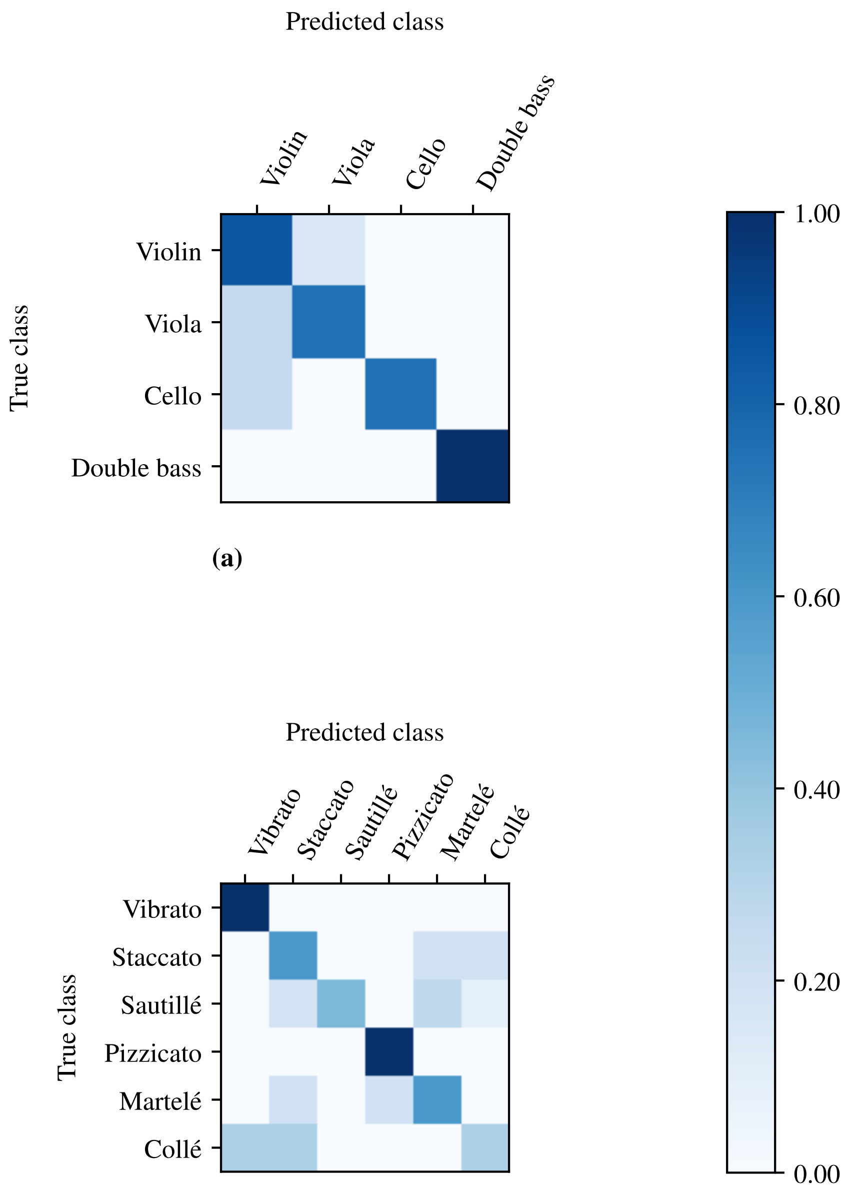


Figure 5.9. Confusion matrices for the prototypical 3-shot CQT CNN model evaluated on the YouTube strings cross-evaluation set on (a) instrument classification (I) and (b) playing technique classification (T). Confusion matrices are normalized to the total number of examples for each true class in the set.

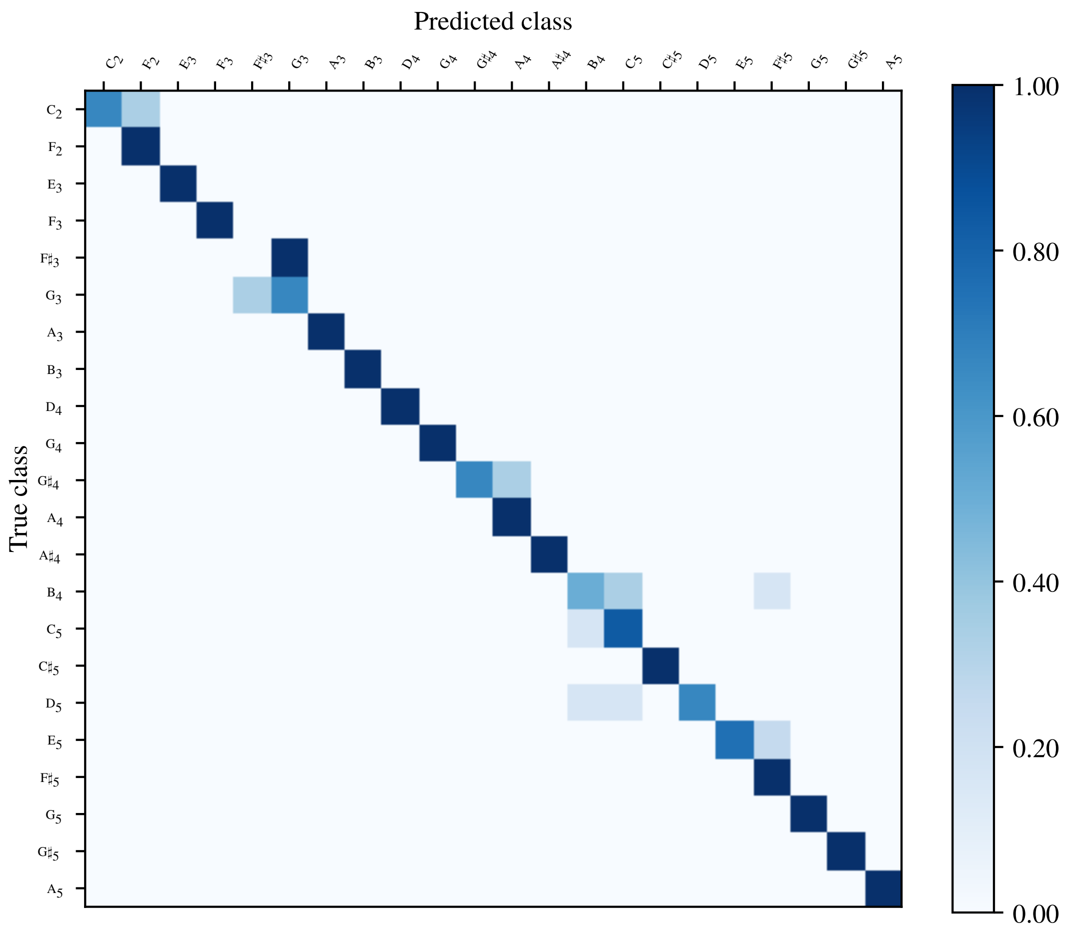


Figure 5.10. Confusion matrix for the prototypical 3-shot CQT CNN model evaluated on the YouTube strings cross-evaluation set on pitch classification (N). The confusion matrix is normalized to the total number of examples for each true class in the set.

5.5 EVALUATION WITH ACCOMPANIMENT

To evaluate how the models perform on data similar to what can be expected from a real-world recording with accompaniment, the existing CQT CNN prototypical and standard classifier models are evaluated on the RWC test set with added piano chords. The evaluation set contains 4 playing techniques and 47 pitches across the 4 string instruments for a total of 234 classes and 720 examples, excluding examples used to compute prototypes — the same as the RWC test set without accompaniment. The models are compared to models with the same architecture and training parameters that were trained on the RWC training set with additional piano chords. Results are shown in Table 5.8.

Table 5.8. Classification F-measure for instrument, playing technique and pitch classification on string instrument examples with piano accompaniment from the RWC test set. This set consists of 234 classes of which the playing technique and pitch combination of 12 were unseen before evaluation. Micro and *macro* F-measures are shown respectively for instrument, note and technique classification (INT), technique classification (T), pitch classification (N) and instrument classification (I). The best 3-shot prototypical model results are shown in bold, the best 1-shot prototypical model results are underlined and the best standard classifier results are shown in italics.

Model	***micro* F-measure**				***macro* F-measure**			
	INT	T	N	I	INT	T	N	I
Prototypical 3-shot CQT CNN	.41	.59	.759	.828	.384	.584	.694	.821
Accompaniment prototypical 3-shot CQT CNN	**.505**	**.619**	**.831**	**.865**	**.478**	**.599**	**.755**	**.858**
Prototypical 1-shot CQT CNN	.305	.525	.673	.781	.276	.512	.608	.776
Accompaniment prototypical 1-shot CQT CNN	<u>.402</u>	<u>.552</u>	<u>.788</u>	<u>.822</u>	<u>.366</u>	<u>.537</u>	<u>.706</u>	<u>.812</u>
Standard CQT CNN	.198	.382	.31	.758	.167	.381	.291	.744
Accompaniment standard CQT CNN	*.219*	*.394*	*.327*	*.753*	*.183*	*.392*	*.316*	*.745*

The 3-shot prototypical models again outperform all other models. Training the models with accompaniment present makes a significant improvement in performance across all tasks compared to training on data without accompaniment.

5.6 EVALUATION ON A PREVIOUSLY UNSEEN INSTRUMENT

The models were also cross-evaluated on a new instrument using clarinet playing technique note recordings from the RWC dataset, which can be seen in Tables 5.9 and 5.10. This gives an indication of how models perform when evaluated on new playing techniques and a new instrument. This set contains 3 clarinet playing techniques played in 40 different pitches. In total there are 120 classes with 720 examples, excluding prototypes. The results of transfer learning for the standard classifiers over the additional epochs are shown in Figure B.5.

Table 5.9. Prototypical models classification F-measure for playing technique and pitch classification on clarinet instrument examples from the RWC dataset that were not seen during training. This set consists of 120 classes of which 120 were unseen before evaluation. Micro and macro F-measures are shown respectively for note and technique classification (NT), technique classification (T) and pitch classification (N). The best 3-shot prototypical model results are shown in bold and the best 1-shot prototypical model results are underlined.

Model	***micro* F-measure**			***macro* F-measure**		
	NT	T	N	NT	T	N
Prototypical 3-shot mel DNN	.415	.484	.864	.386	.49	.862
Prototypical 3-shot CQT DNN	.475	.501	**.936**	.449	.501	**.934**
Prototypical 3-shot mel CNN	.478	.609	.821	.461	.615	.828
Prototypical 3-shot CQT CNN	.554	.638	.896	.545	.646	.904
Prototypical 3-shot mel RNN	**.632**	**.703**	.912	**.604**	**.699**	.912
Prototypical 3-shot CQT RNN	.587	.647	.92	.565	.646	.919
Prototypical 3-shot mel CRNN	.505	.648	.796	.479	.648	.795
Prototypical 3-shot CQT CRNN	.511	.593	.853	.487	.592	.853
Prototypical 1-shot mel DNN	.36	.44	.828	.325	.443	.824
Prototypical 1-shot CQT DNN	.412	.459	<u>.887</u>	.378	.458	<u>.88</u>
Prototypical 1-shot mel CNN	.378	.549	.737	.364	.555	.738
Prototypical 1-shot CQT CNN	.449	.57	.829	.436	.584	.828
Prototypical 1-shot mel RNN	<u>.53</u>	<u>.68</u>	.817	<u>.494</u>	<u>.672</u>	.816
Prototypical 1-shot CQT RNN	.484	.605	.838	.46	.593	.838
Prototypical 1-shot mel CRNN	.432	.619	.728	.403	.617	.724
Prototypical 1-shot CQT CRNN	.442	.561	.782	.415	.559	.777

Table 5.10. Standard classifier models classification F-measure for playing technique and pitch classification on clarinet instrument examples from the RWC dataset that were not seen during training. Micro and macro F-measures are shown respectively for pitch and technique classification (NT), technique classification (T) and pitch classification (N). The best standard classifier results are shown in italics.

Model	***micro* F-measure**			***macro* F-measure**		
	NT	T	N	NT	T	N
Standard mel DNN	.384	.476	.787	.337	.479	.786
Standard CQT DNN	*.474*	.493	*.953*	*.44*	.503	*.953*
Standard mel CNN	.0469	.466	.0972	.0318	.464	.0823
Standard CQT CNN	.0963	.498	.191	.0759	.505	.179
Standard mel RNN	.322	*.612*	.607	.269	*.621*	.608
Standard CQT RNN	.162	.471	.357	.127	.476	.341
Standard mel CRNN	.313	.607	.505	.259	.6	.486
Standard CQT CRNN	.455	.597	.747	.411	.599	.739

Models perform worse when evaluated on a new instrument than on new playing techniques from the same instrument. For both the NT and T problems on the clarinet data, the prototypical 3-shot mel RNN outperforms the prototypical 3-shot CQT CNN. Additional results are shown in Appendix B.3.

5.7 PLAYING TECHNIQUE CLASSIFICATION

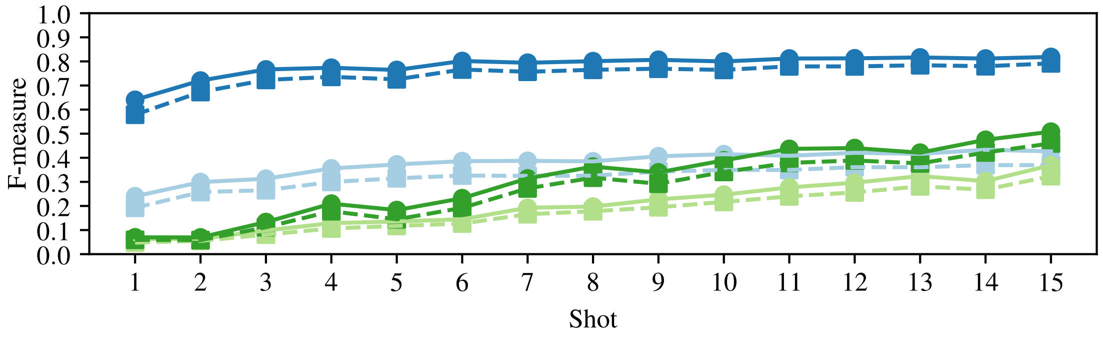
The best prototypical model — the CQT CNN — and its standard classifier counterpart were evaluated on the problem of instrument technique classification and compared to standard classifier and prototypical models specifically trained on the problem. For evaluation the RWC test set was used with only the instrument and playing technique class labels used. This set contains 24 classes with 9 techniques and 2586 examples, excluding prototypes. Results for instrument playing technique classification are shown in Table 5.11.

Table 5.11. Classification F-measure on the RWC test set for the problems of joint instrument and playing technique classification (IT), playing technique classification (T) and instrument classification (I), compared between the prototypical 3-shot CQT CNN model, standard classifier CQT CNN model and prototypical and standard classifier models with the same architectures trained on the joint instrument and playing technique classification (IT) problem. This set consists of 24 classes of which 2 were unseen by the models trained on instrument playing technique classification before evaluation. The best results are shown in bold.

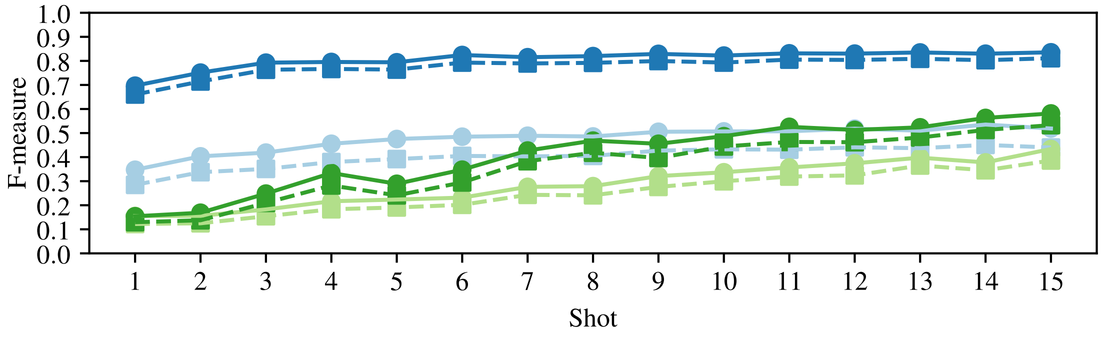
Model	***micro* F-measure**			***macro* F-measure**		
	IT	T	I	IT	T	I
Prototypical 3-shot CQT CNN	.319	.42	.663	.284	.359	.625
Techniques prototypical 3-shot CQT CNN	**.768**	**.798**	**.949**	**.732**	**.777**	**.94**
Standard CQT CNN	.0752	.156	.413	.0656	.13	.337
Techniques standardCQT CNN	.146	.276	.467	.116	.222	.415

The prototypical model trained for the problem significantly outperforms the other models on all 3 sub-problems. The prototypical model trained on joint instrument, playing-technique and pitch classification still outperforms the standard classifier models with only 3 examples per class used to adjust the models to the problem and dataset.

Figures 5.11 and 5.12 show the performance of the models for different shot numbers on the instrument playing technique classification problem. The standard classifier models are compared for different numbers of examples per class used for transfer learning. The set for this evaluation only contains classes that have enough examples for a shot of 15 with minimum 3 evaluation examples and the number of evaluation samples per class was kept constant for each shot number. It contains 24 classes with 9 techniques and 2226 examples, excluding those left out for prototypes. F-measure increases slightly with shot number for the techniques prototypical network, but increases a lot more for the other models.



(a)



(b)

■	Prototypical 3-shot CQT CNN
■	Techniques prototypical 3-shot CQT CNN
■	Standard CQT CNN
■	Techniques standard CQT CNN
—●—	micro F-measure
-■-	macro F-measure

Figure 5.11. Classification F-measure on the RWC test set for the sub-problems of (a) joint instrument and technique classification (IT) and (b) technique classification (T), compared between the prototypical 3-shot CQT CNN model, standard classifier CQT CNN model and prototypical and standard classifier models with the same architectures trained on the joint instrument and playing technique classification (IT) problem. Model performance is compared for different shot numbers at evaluation time.

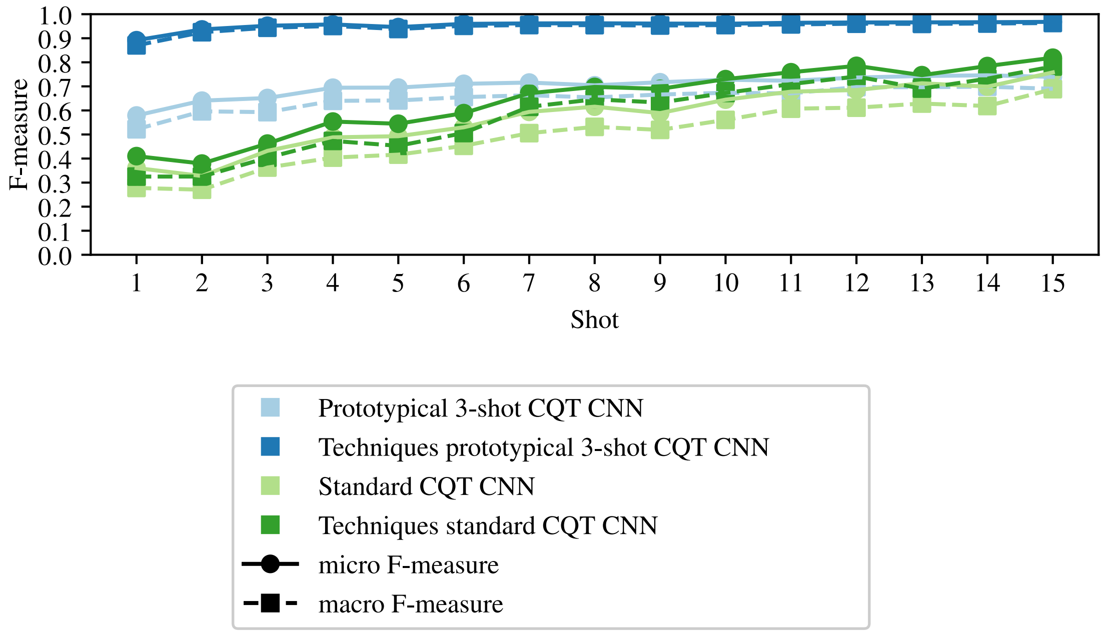


Figure 5.12. Classification F-measure on the RWC test set for the sub-problem of instrument classification (I), compared between the prototypical 3-shot CQT CNN model, standard classifier CQT CNN model and prototypical and standard classifier models with the same architectures trained on the joint instrument and playing technique classification (IT) problem. Model performance is compared for different shot numbers at evaluation time.

5.8 PITCH CLASSIFICATION

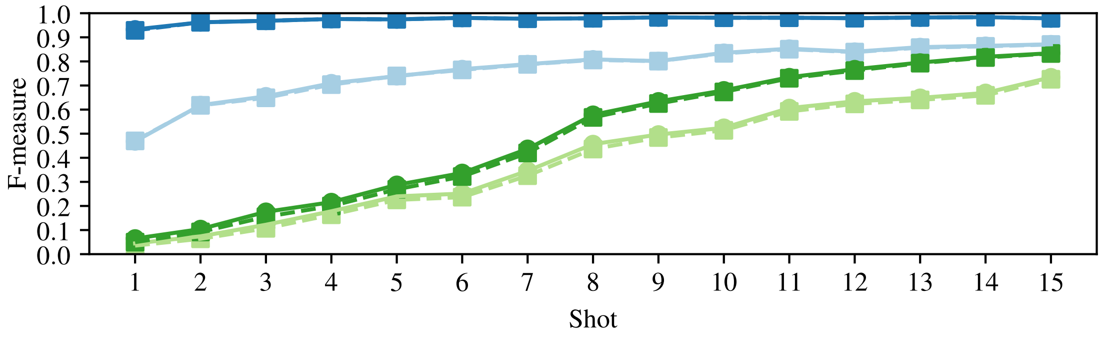
The CQT CNN prototypical model and standard classifier model are also evaluated on the problem of instrument pitch classification and again compared to standard classifier and prototypical models specifically trained on the problem. For evaluation, the RWC test set was again used with the playing technique label ignored. This set contains 153 classes with 68 pitches and 2142 examples, excluding prototypes. Results for instrument pitch classification are shown in Table 5.12.

Table 5.12. Classification F-measure on the RWC test set for the sub-problems of joint instrument and pitch classification (IN), pitch classification (N) and instrument classification (I), compared between the prototypical 3-shot CQT CNN model, standard classifier CQT CNN model and prototypical and standard classifier models with the same architectures trained on the joint instrument and pitch classification (IN) problem. This set consists of 153 classes of which none were unseen by the models trained on instrument pitch classification before evaluation. The best results are shown in bold.

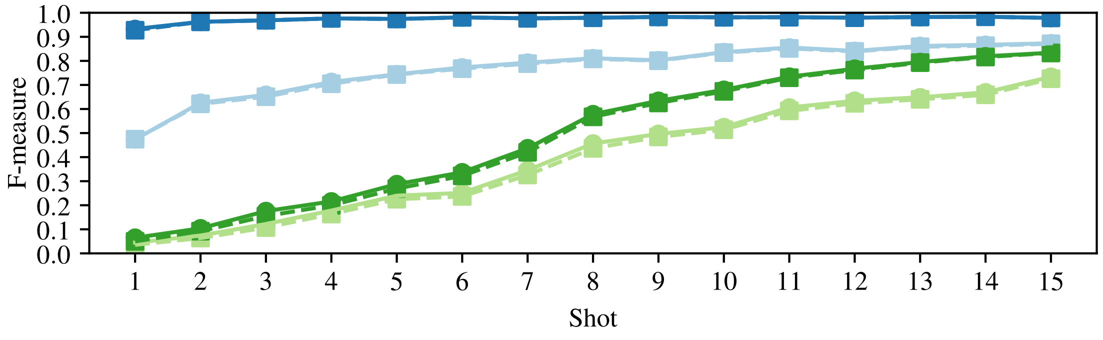
Model	***micro* F-measure**			***macro* F-measure**		
	IN	N	I	INT	N	I
Prototypical 3-shot CQT CNN	.419	.586	.708	.37	.524	.654
Notes prototypical 3-shot CQT CNN	**.853**	**.923**	**.919**	**.774**	**.901**	**.895**
Standard CQT CNN	.256	.276	.697	.221	.28	.641
Notes standard CQT CNN	.355	.376	.707	.299	.374	.646

The 3-shot prototypical CQT CNN models again significantly outperform the retrained standard classifier models. The models specifically trained on the pitch classification problem again outperform the more general models.

Figures 5.13 and 5.13 show the performance of the models for different shot numbers on the instrument pitch classification problem. The standard classifier models are compared for different numbers of examples per class used for transfer learning. The set for this evaluation only contains classes that have enough examples for a shot of 15 with minimum 3 evaluation examples and the number of evaluation samples per class was kept constant for each shot number. It contains 36 classes with 35 pitches and 504 examples, excluding those left out for prototypes. F-measure increases very slightly with shot number for the notes prototypical network, but increases a lot more for the other models. Model performance across the board remain consistently near perfect for all models and all shot numbers for the sub-problem of instrument classification.



(a)



(b)

■	Prototypical 3-shot CQT CNN
■	Notes prototypical 3-shot CQT CNN
■	Standard CQT CNN
■	Notes standard CQT CNN
—●—	micro F-measure
-■-	macro F-measure

Figure 5.13. Classification F-measure on the RWC test set for the sub-problems of (a) joint instrument and pitch classification (IN) and (b) pitch classification (N), compared between the prototypical 3-shot CQT CNN model, standard classifier CQT CNN model and prototypical and standard classifier models with the same architectures trained on the joint instrument and pitch classification (IN) problem. Model performance is compared for different shot numbers at evaluation time.

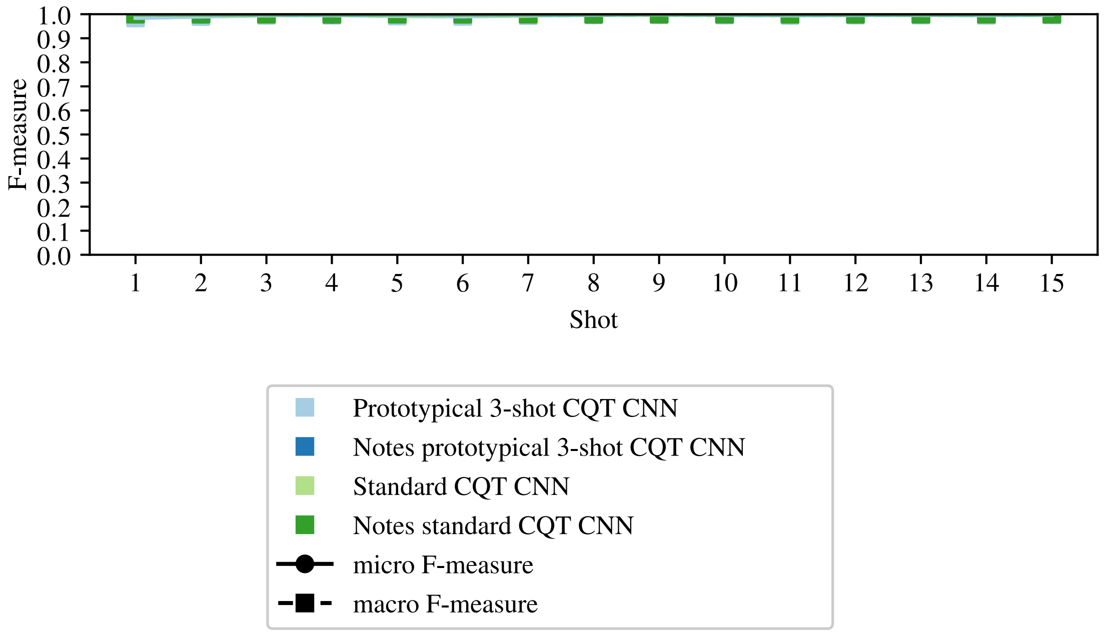


Figure 5.14. Classification F-measure on the RWC test set for the sub-problem of instrument classification (I), compared between the prototypical 3-shot CQT CNN model, standard classifier CQT CNN model and prototypical and standard classifier models with the same architectures trained on the joint instrument and pitch classification (IN) problem. Model performance is compared for different shot numbers at evaluation time.

5.8.1 Pitch classification using PYIN

The pitch classification problem (N) was also evaluated from F0s estimated using the PYIN algorithm. Table 5.13 shows evaluation results on the same datasets that were used to evaluate the classifier models (excluding the examples used to calculate prototypes) for finding a predicted pitch class from F0s. Three different methods of extracting a pitch classification from the PYIN F0 contour are evaluated: taking the mean, taking the weighted mean or taking the maximum voiced probability.

Table 5.13. Pitch classification (N) F-measure using the PYIN algorithm on the RWC string instrument test set, OrchideaSOL string instruments data, YouTube tutorials data and RWC clarinet data. The best prototypical classifier on each evaluation set is compared.

Dataset	***micro* F-measure**			
	PYIN			
	Mean	**Weighted Mean**	**Maximum Probability**	**Prototypical Classifier**
RWC string instrument test set	.819	.86	.929	.943[1]
OrchideaSOL string instrument test set	.906	.916	.941	.964[1]
YouTube string instrument test set	.891	.898	.906	.963[1]
RWC clarinet test set	.731	.906	.906	.92[2]

Dataset	***macro* F-measure**			
	PYIN			
	Mean	**Weighted Mean**	**Maximum Probability**	**Prototypical Classifier**
RWC string instrument test set	.617	.666	.62	.927[1]
OrchideaSOL string instrument test set	.86	.874	.869	.947[1]
YouTube string instrument test set	.571	.635	.684	.962[1]
RWC clarinet test set	.708	.879	.883	.919[2]

[1] Prototypical 3-shot CNN with constant-Q transform input.

[2] Prototypical 3-shot RNN with constant-Q transform input.

The maximum voiced probability provides the highest F-measure over all datasets. On each dataset PYIN is outperformed by the best prototypical models.

5.9 MODEL PERFORMANCE

Prototypical networks prove to perform well when cross-evaluating on new playing techniques and instruments. There is some loss of generalization, but the best models still perform quite well on the datasets containing playing techniques and instruments that were not seen during training. Evaluating the cross-dataset problems shows very little decrease in F-measure on the OrchideaSOL string instrument set or YouTube examples. Playing technique classification is the problem that is the hardest for the models. Where the models struggle most with playing technique classification is with playing techniques that sound similar, such as percussive bow strokes (see *staccato* being confused with *sautillé*, *martelé* and *collé* in Figure 5.9(b)) or playing techniques with variations, such as adding a mute (see Figure 5.7(b)). This is much more prevalent with the standard classifier models, as can be seen in Figure 5.5(b). A similar confusion is seen between violin and viola, which are much closer in range and timbre than the other string instruments. There is some confusion for the prototypical CQT CNN between these instruments on the RWC test set, as seen in Figure 5.3(a), but again, Figure 5.5(a) shows even more confusion for the standard classifier CQT CNN.

The models do maintain a consistently high F-measure for pitch transcription, even on clarinet. Generalization to new instruments and datasets is still a hard problem [2], so it is impressive to see such consistency across datasets. When examining confusion matrices, it can be seen that errors with pitch are either half-step errors or octave errors. The prototypical models are able to outperform signal processing algorithms, such as PYIN, which has previously been hard for standard neural network models to do on cross-evaluation [1].

Examining the results in Table 5.5 shows substantial improvement for prototypical networks for single instruments. A tutorial scenario would most likely only involve a single instrument, thus this bodes well for that application. Of all the single instrument string evaluation sets, OrchideaSOL violin has the most classes and the second highest percentage of new classes, thus the model performing worse on violin than on the other instruments, compared to the full OrchideaSOL string instrument test set shows that differentiating between fewer classes is advantageous. It is interesting to note that pitch classification results (N) are worse with only violin prototypes than with all string instrument prototypes on violin evaluation data, and other instrument evaluations show a much smaller decrease in F-measure on N than on other problems when using all the string instrument prototypes, which suggests that even though providing instrument specific prototypes reduces playing technique confusion, pitch classification features are not as bound to a specific instrument, thus even if the instrument may be classified incorrectly, the pitch is still likely to be correct.

Models evaluated on data with accompaniment compare favorably to models evaluated on new playing technique data and pitch classification remains high, which bodes well for applying these models to main melody extraction. The marked improvement for the models trained on data with accompaniment indicates that the important features for playing technique and pitch classification with accompaniment differ from classification without accompaniment. While the models perform well as multi-task models, there is some improvement to be gained by attuning the models to a specific task during training.

With the goal of building a tutorial system in mind, the prototypical approach performs very well as a multi-task model. The best joint instrument, playing technique and pitch classification prototypical model is able to accurately classify both playing technique (the T problem) and pitch (the N problem). When compared to models trained more specifically on these problems (compare Table 5.1 with Tables 5.11 and 5.12) the model trained on join instrument, playing technique and pitch classification suffers when given prototypes for only instrument and playing technique or only instrument and pitch. Since the model was not trained for these kinds of prototypes, it is expected that classification would suffer and because performance on IT and IN is not as good as on INT, performance on T and N decreases. Thus for good multi-task performance it is better to provide prototypes similar to what the models were trained on.

The sustained high performance of the few-shot models on the real-world cross-evaluating task of classifying YouTube playing technique tutorial note examples also shows great promise for the practical application of these models. The few-shot models thus do not suffer under less ideal recording conditions and could be used, for example, by a music student who would record prototypes of playing techniques under the supervision of a teacher, using a smartphone or inexpensive microphone setup. The student could then try to match the playing techniques and the models would give a good indication of whether the student closely matched the examples.

Comparing these results to other studies on similar problems is hard, since as far as can be told, the number of classes in jointly classifying instrument, playing technique and pitch in this dissertation is unprecedented. No direct comparison can be drawn, but the prototypical models perform near what would be expected of state-of-the-art algorithms on each of the sub-problems. If desired, the results in Table 5.3 can be compared to benchmark results discussed in Section 2.3. Similar to previous few-shot approaches in MIR, these prototypical models are able to generalize to new performers and datasets, as well as to new playing techniques, as has been shown for automatic drum transcription [32].

5.10 TIME-FREQUENCY FEATURES

For most problems the CQT models perform the best or at least very close to the best model. A CQT maintains consistent frequency resolution for each musical pitch, even as the pitch frequency increases exponentially, whereas a log-mel spectrogram will vary the frequency bins corresponding to each pitch, as it is an offset exponential mapping; this gives some advantage to the CQT, especially with the convolutional models where feature extraction kernels can be applied to any pitch. Increasing the number of mel bands for the log-mel spectrograms to match the frequency resolution of the CQT features was tested, but this did not yield any significant improvement, thus it was chosen to match input feature hyperparameters that yielded good results in previous implementations for log-mel spectrogram [65, 33] and CQT [107].

CQT features thus work well for incorporating pitch classification into the playing technique classification problem, since it provides good frequency resolution across the frequency spectrum for extracting pitch information and differentiating harmonics from fundamental frequencies.

5.11 PROTOTYPICAL MODEL SHOT

Prototypical models perform better when evaluated on the same shot as they are trained on [3]; this is also seen in this investigation, with prototypical models trained with a shot of 3 performing better when evaluated with a shot of 3 than with a shot of 1. Figures 5.11 and 5.13 also show a general, if not substantial improvement for larger shot numbers on problems where enough examples were available to accommodate larger shot numbers. It would be expected that larger shot numbers would improve generalization, as the class prototypes would average out any irregularities in some of the prototype examples. For some problems the model performance compared to the shot number plateaus after a while, giving very little performance increase for further shot number increase. Shot number is thus a good parameter to tune for better performance where allowed by the available data when training prototypical models.

5.12 PROTOTYPICAL NETWORKS COMPARED TO TRANSFER LEARNING

Unlike prototypical networks, where the main attraction is easy generalization to new data, standard DNN classifiers needed to be retrained when applied to new classes. Transfer learning is a close comparison to the philosophy of prototypical networks of applying the feature extraction capability of a pre-trained network to a new problem. The best standard classifiers with transfer learning performed well, but were surpassed by the prototypical models. The best models overall for transfer learning were the DNN models. The DNN architecture has the advantage of fewer weights and being able to train with fewer data than deeper networks, thus retraining with a new output layer was much easier than the other deeper network architectures.

Compared to prototypical networks, transfer learning is also much less efficient and practical. Training has higher hardware requirements than inference, so calculating new prototypes is a lot easier than retraining a network. Transfer learning also requires some time commitment; F-measure increased significantly with more epochs of the new dataset (see Figures 5.1 and 5.2). This again speaks to the practicality of the prototypical network approach for real-world applications, such as a tutorial system.

5.13 LIMITATIONS OF THE APPROACH

The prototypical models function better on problems with more specificity. Although prototypical few-shot learners generalize well to new datasets and problems, they might also not perform as well as models specifically trained for the new problem or data domain. Where enough training data is available and less flexibility is needed, a standard classifier might function just as well or better.

The performance of prototypical models has been shown to correlate to the number of examples per class in the training set [93]. Suggested examples per class from previous implementations are in the order of $N_S + N_Q = 20$ [3]. The datasets available restricted the number of examples for some classes to less than that for the problem of joint instrument, playing technique and pitch classification. Although the trained models performed well on the joint classification problem, as well as the sub-problems for which the number of examples per class were increased, performance could still be improved further with a larger training set with more examples per class.

Prototypical models are also dependent on prototype examples. Figures 5.11 and 5.13 show that model performance can be improved by providing more prototypes, especially when the model was not trained on the problem. Prototypes require annotation, however, which is not always available.

5.14 CHAPTER CONCLUSION

After comparing different prototypical and standard classifier models, it was found that the best model on instrument, playing technique and pitch classification is the 3-shot CQT CNN prototypical model. It displays minimal loss of accuracy on new datasets and classes, and is able to perform well on the tasks of playing technique classification and pitch classification, as well as instrument, playing technique and pitch classification with accompaniment when compared to models specifically trained on the tasks.

CHAPTER 6 CONCLUSION

A few-shot classification model is presented for instrument, playing technique and pitch classification that achieves a *macro* F-score of .814 on RWC string instrument examples for a 3-shot CQT CNN prototypical model and generalizes better to new problems and data than standard classifier models. Few-shot models function as multi-task learners and these models perform well on the tasks of playing technique classification, pitch classification and instrument classification from single note examples, achieving a *macro* F-score of .953, .927 and .925 respectively on the RWC test set. The best few-shot model outperforms the best standard classifier model on both the joint instrument, pitch and playing technique classification task and the individual classification tasks of playing technique classification, pitch classification and instrument recognition.

The few-shot models perform well on cross-dataset evaluation and are able to adapt to new playing techniques and instruments from only a few prototype examples per new class. The best few-shot model performs better on previously unseen datasets, playing techniques, instruments and tasks than any of the standard classifier models that were finetuned on the new data or tasks. The models also perform very well on realistic data and are able to classify playing technique and pitch in the presence of accompaniment as well as with non-ideal recording conditions, such as from YouTube videos. This approach could function well as part of a tutorial system for evaluating the playing technique performance of a student.

A new aspect of playing technique classification was investigated, namely joint classification with instrument and pitch. Few-shot learning with prototypical networks is a promising solution to this problem that is able to generalize well to new classes, even with an unprecedented amount of classes for the problem domain.

6.1 FURTHER WORK

Further work of interest would be to apply these models and methodologies to more instruments to achieve a more general model that could transcribe any instrument or playing technique. Models would also be greatly improved by larger training sets with more examples per class. Investigation of the application of these models towards a general-purpose transcription system should be done by applying the models to the problem of stream-level transcription with playing technique. As noted, a major obstacle to this is a lack of playing technique annotation for transcription datasets in MIR. With a general-purpose transcription system in mind, an investigation should also be done on adapting these methods to polyphonic multiple-instrument transcription.

REFERENCES

[1] E. Benetos, S. Dixon, Z. Duan, and S. Ewert, "Automatic music transcription: An overview," *IEEE Signal Processing Magazine*, vol. 36, no. 1, pp. 20–30, 2019.

[2] A. Livshin, "Automatic musical instrument recognition and related topics," Ph.D. dissertation, Université Pierre et Marie Curie-Paris VI, 2007.

[3] J. Snell, K. Swersky, and T. R. Zemel, "Prototypical networks for few-shot learning," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[4] V. Lostanlen, J. Andén, and M. Lagrange, "Extended playing techniques: The next milestone in musical instrument recognition," in *Proceedings of the 5th International Conference on Digital Libraries for Musicology*, vol. 18, 2018, pp. 1–10.

[5] J. Abeßer, H. Lukashevich, and G. Schuller, "Feature-based extraction of plucking and expression styles of the electric bass guitar," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2010, pp. 2290–2293.

[6] J. Abeser and G. Schuller, "Instrument-centered music transcription of solo bass guitar recordings," *IEEE/ACM Transactions on Audio Speech and Language Processing*, vol. 25, no. 9, pp. 1741–1750, Sept. 2017.

[7] S. Giraldo and R. Ramírez, "Performance to score sequence matching for automatic ornament detection in jazz music," in *International Conference of New Music Concepts ICMNC*, 2015.

[8] T. H. Özaslan and J. L. Arcos, "Legato and glissando identification in classical guitar," in *7th Sound and Music Computing Conference (SMC)*, 2010.

[9] L. Reboursière, O. Lähdeoja, T. Drugman, S. Dupont, C. Picard-Limpens, and N. Riche, "Left and right-hand guitar playing techniques detection," in *Proceedings of the International Conference on New Interfaces for Musical Expression (NIME)*, 2012.

[10] L. Su, H.-M. Lin, and Y.-H. Yang, "Sparse modeling of magnitude and Pphase-derived spectra for playing technique classification," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 22, no. 12, pp. 2122–2132, 2014.

[11] Y.-P. Chen, L. Su, and Y.-H. Yang, "Electric guitar playing technique detection in real-world recording based on F0 sequence pattern recognition," in *Proceedings of the International Society for Music Information Retrieval (ISMIR) Conference*, 2015, pp. 708–714.

[12] R. Foulon, P. Roy, and F. Pachet, "Automatic classification of guitar playing modes," in *Sound, Music, and Motion: 10th International Symposium, CMMR*, vol. 8905, Oct. 2014, pp. 58–71.

[13] I. Barbancho, C. de la Bandera, A. M. Barbancho, and L. J. Tardon, "Transcription and expressiveness detection system for violin music," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASP)*, 2009, pp. 189–192.

[14] J. Charles, "Playing technique and violin timbre: Detecting bad playing," Ph.D. dissertation, Dublin Institute of Technology, Dublin, Jan. 2010.

[15] D. Young, "Classification of common violin bowing techniques using gesture data from a playable measurement system," in *Proceedings of the International Conference on New Interfaces for Musical Expression (NIME)*, 2008, pp. 44–48.

[16] B. Liang, G. Fazekas, A. Mcpherson, and M. Sandler, "Piano Pedaller: A measurement system for classification and visualisation of piano pedalling techniques," in *Proceedings of the International Conference on New Interfaces for Musical Expression (NIME)*, 2017.

[17] M. Bernays and C. Traube, "Expressive production of piano timbre: Touch and playing techniques for timbre control in piano performance," in *Proceedings of the 10th Sound and Music Computing Conference (SMC2013)*, 2013, pp. 341–346.

[18] C. Wu and A. Lerch, "On drum playing technique detection in polyphonic mixtures," in *Proceedings of the International Society for Music Information Retrieval (ISMIR) Conference*, 2016, pp. 218–224.

[19] P. Herrera, A. Yeterian, and F. Gouyon, "Automatic classification of drum sounds: A comparison of feature selection methods and classification techniques," in *Music and Artificial Intelligence: Second International Conference (ICMAI)*, vol. 2445, 2002, pp. 69–80.

[20] G. Tzanetakis, A. Tindale, A. Kapur, and I. Fujinaga, "Retrieval of percussion gestures using timbre classification techniques," in *Proceedings of the International Society for Music Information Retrieval (ISMIR) Conference*, 2004.

[21] A. Neocleous, G. Azzopardi, C. N. Schizas, and N. Petkov, "Filter-based approach for ornamentation detection and recognition in singing folk music," in *Computer Analysis of Images and Patterns: 16th International Conference (CAIP)*, vol. 9256, 2015, pp. 558–569.

[22] J. Wilkins, P. Seetharaman, A. Wahl, and B. Pardo, "VocalSet: A singing voice dataset," in *Proceedings of the International Society for Music Information Retrieval (ISMIR) Conference*, 2018, pp. 468–474.

[23] Y. Yamamoto, J. Nam, and H. Terasawa, "Analysis and detection of singing techniques in repertoires of J-POP solo singers," in *Proceedings of the 23rd International Society for Music Information Retrieval (ISMIR) Conference*, 2022.

[24] D. W. H. Menzies and A. P. Mcpherson, "Highland piping ornament recognition using dynamic time warping," in *Proceedings of the International Conference on New Interfaces for Musical Expression (NIME)*, 2015, pp. 50–53.

[25] Y. Huang, J. Liang, I. Wei, and L. Su, "Joint analysis of mode and playing technique in Guqin

performance with machine learning," in *Proceedings of the International Society for Music Information Retrieval (ISMIR) Conference*, 2020, pp. 85–92.

[26] D. Li, Y. Wu, Q. Li, J. Zhao, Y. Yu, F. Xia, and W. Li, "Playing Technique Detection by Fusing Note Onset Information in Guzheng Performance," in *Proceedings of the 23rd International Society for Music Information Retrieval (ISMIR) Conference*, 2022.

[27] C. Wang, E. Benetos, X. Meng, E. Chew, and others, "HMM-based glissando detection for recordings of chinese bamboo flute," in *Sound and Music Computing*, May 2019.

[28] C. Wang, E. Benetos, V. Lostanlen, E. Chew, E. Chew Adaptive, S. Member, S. Member, and I. Vincent Lostanlen, "Adaptive scattering transforms for playing technique recognition," *IEEE/ACM Transactions on Audio, Speech and Language Processing*, vol. 30, pp. 1407–1421, 2022.

[29] B. Shi, M. Sun, K. C. Puvvada, C.-C. Kao, S. Matsoukas, and C. Wang, "Few-shot acoustic event detection via meta learning," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 76–80.

[30] P. Wolters, C. Daw, B. Hutchinson, and L. Phillips, "Proposal-based few-shot found event detection for speech and environmental sounds with perceivers," *arXiv preprint arXiv:2107.13616*, 2021.

[31] H. F. Garcia, A. Aguilar, E. Manilow, and B. Pardo, "Leveraging hierarchical structures for few-shot musical instrument recognition," in *Proceedings of the International Society for Music Information Retrieval (ISMIR) Conference*, 2021, pp. 220–228.

[32] Y. Wang, J. Salamon, M. Cartwright, N. J. Bryan, and J. P. Bello, "Few-shot drum transcription in polyphonic music," in *Proceedings of the International Society for Music Information Retrieval (ISMIR) Conference*, 2020.

[33] Y. Wang, J. Salamon, N. J. Bryan, and J. Pablo Bello, "Few-shot sound event detection," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020,

pp. 81–85.

[34] M. Goto, H. Hashiguchi, T. Nishimura, and R. Oka, “RWC music database: Popular, classical, and jazz music databases,” in *Proceedings of the International Society for Music Information Retrieval (ISMIR) Conference*, 2002, pp. 287–288.

[35] C. E. Cella, D. Ghisi, V. Lostanlen, F. Lévy, J. Fineberg, and Y. Maresz, “OrchideaSOL: A dataset of extended instrumental techniques for computer-aided orchestration,” *arXiv preprint arXiv:2007.00763*, 2020.

[36] P. C. Kok and J. P. Jacobs, “Computationally efficient note activation classifier for multiple-instrument automatic music transcription,” in *Southern Africa Telecommunication Networks and Applications Conference (SATNAC)*, 2021, pp. 192–197.

[37] G. Moens-Haenen, “Vibrato,” *Oxford Music Online*, 2001.

[38] “Sul ponticello,” *Oxford Music Online*, 2001.

[39] “Sul tasto,” *Oxford Music Online*, 2001.

[40] S. Monosoff, “Pizzicato,” *Oxford Music Online*, 2001.

[41] D. D. Boyden and P. Walls, “Spiccato,” *Oxford Music Online*, 2001.

[42] ——, “Sautillé,” *Oxford Music Online*, 2001.

[43] W. Bachmann, R. E. Seletsky, D. D. Boyden, J. Liivoja-Lorius, P. Walls, and P. Cooke, “Bow,” *Oxford Music Online*, vol. 1, 2001.

[44] M. Thiriot, “Developing right-hand finger flexibility in young violinists: Teaching collé, martelé, spiccato, and sautillé through the Suzuki literature,” Ph.D. dissertation, Liberty University, Lynchburg, 2022.

[45] D. Fallows, “Tremolo (i),” *Oxford Music Online*, 2001.

[46] G. Grove, *A Dictionary of Music and Musicians*, 1900.

[47] G. Chew and C. Brown, “Staccato,” *Oxford Music Online*, 2001.

[48] D. D. Boyden, C. Bevan, and J. K. Page, “Mute,” *Oxford Music Online*, 2001.

[49] D. D. Boyden and R. Stowell, “Glissando,” *Oxford Music Online*, Jan. 2018.

[50] E. T. Harris, “Portamento (i),” *Oxford Music Online*, 2001.

[51] T. H. Özaslan, X. Serra, and J. L. Arcos, “Characterization of embellishments in ney performances of makam music in Turkey,” in *Proceedings of the 13th International Society for Music Information Retrieval (ISMIR) Conference*, 2012.

[52] L. Yang, “Computational modelling and analysis of vibrato and portamento in expressive music performance,” Ph.D. dissertation, Queen Mary University of London, London, Apr. 2017.

[53] J. Ducher and P. Esling, “Folded CQT RCNN for real-time recognition of instrument playing techniques,” in *Proceedings of the International Society for Music Information Retrieval (ISMIR) Conference*, 2019.

[54] J. Andén, V. Lostanlen, and S. Mallat, “Classification with joint time-frequency scattering,” *IEEE Transactions on Signal Processing*, vol. 67, pp. 3704–3718, 2019.

[55] L. Su and Y. H. Yang, “Combining spectral and temporal representations for multipitch estimation of polyphonic music,” *IEEE Transactions on Audio, Speech and Language Processing*, vol. 23, no. 10, pp. 1600–1612, 2015.

[56] A. Kruger and J. Jacobs, “Playing technique classification for bowed string instruments from raw audio,” *Journal of New Music Research*, vol. 49, no. 4, pp. 320–333, 2020.

[57] H. S. Alar, R. O. Mamaril, L. P. Villegas, and J. R. D. Cabarrubias, “Audio classification of violin bowing techniques: An aid for beginners,” *Machine Learning with Applications*, vol. 4, p. 100028, June 2021.

[58] J. C. Brown, “Computer identification of musical instruments using pattern recognition with cepstral coefficients as features,” *The Journal of the Acoustical Society of America*, vol. 105, no. 3, p. 1933, Mar. 1999.

[59] A. Eronen and A. Klapuri, “Musical instrument recognition using cepstral coefficients and temporal features,” in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, vol. 2, 2000, pp. 753–756.

[60] I. Kaminskyj, “Automatic recognition of isolated monophonic musical instrument sounds using kNNC,” *Journal of Intelligent Information Systems*, vol. 24, no. 3, pp. 199–221, 2005.

[61] V. Lostanlen and C.-E. CelláCellá, “Deep convolutional networks on the pitch spiral for musical instrument recognition,” *Proceedings of the International Society for Music Information Retrieval (ISMIR)*, 2016.

[62] A. G. Krishna and T. V. Sreenivas, “Music instrument recognition: From isolated notes to solo phrases,” in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, vol. 4, 2004.

[63] E. Benetos, M. Kotti, and C. Kotropoulos, “Musical instrument classification using non-negative matrix factorization algorithms and subset feature selection,” in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, vol. 5, 2006, pp. V–V.

[64] P. Herrera-Boyer, G. Peeters, and S. Dubnov, “Automatic classification of musical instrument sounds,” *International Journal of Phytoremediation*, vol. 21, no. 1, pp. 3–21, 2003.

[65] M. Taenzer, J. Abeßer, S. I. Mimilakis, C. Weiß, M. Müller, and H. Lukashevich, “Investigating CNN-based instrument family recognition for western classical music recordings,” in *Proceed-*

ings of the International Society for Music Information Retrieval (ISMIR) 2019, Delft, The Netherlands, Nov. 2019.

[66] V. Garcia and J. Bruna, "Few-shot learning with graph neural networks," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018, pp. 1–13.

[67] Z. Duan, B. Pardo, and C. Zhang, "Multiple fundamental frequency estimation by modeling spectral peaks and non-peak regions," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 18, no. 8, pp. 2121–2133, 2010.

[68] V. Emiya, R. Badeau, and B. David, "Multipitch estimation of piano sounds using a new probabilistic spectral smoothness principle," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 18, no. 6, pp. 1643–1654, 2010.

[69] M. Mauch and S. Dixon, "PYIN: A fundamental frequency estimator using probabalistic threshold distributions," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2014, pp. 659–663.

[70] J. W. Kim, J. Salamon, P. Li, and J. P. Bello, "Crepe: A convolutional representation for pitch estimation," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018, pp. 161–165.

[71] X. Li, Y. Guan, Y. Wu, and Z. Zhang, "Piano multipitch estimation using sparse coding embedded deep learning," *Eurasip Journal on Audio, Speech, and Music Processing*, vol. 2018, no. 1, 2018.

[72] P. Smaragdis and J. C. Brown, "Non-negative matrix factorization for polyphonic music transcription," in *IEEE Workshop on Applications of Signal Proccssing to Audio and Acoustics*, no. 1, New PaltzIf, 2003, pp. 177–180.

[73] E. Vincent, N. Bertin, and R. Badeau, "Adaptive Harmonic Spectral Decomposition for Multiple Pitch Estimation," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 18, no. 3, pp. 528–537, 2010.

[74] B. Fuentes, R. Badeau, and G. Richard, “Harmonic adaptive latent component analysis of audio and application to music transcription,” *IEEE Transactions on Audio, Speech and Language Processing*, vol. 21, no. 9, pp. 1854–1866, 2013.

[75] S. Sigtia, E. Benetos, and S. Dixon, “An end-to-end neural network for polyphonic piano music transcription,” *IEEE/ACM Transactions on Audio Speech and Language Processing*, vol. 24, no. 5, pp. 927–939, 2016.

[76] R. Kelz, M. Dorfer, F. Korzeniowski, S. Böck, A. Arzt, and G. Widmer, “On the potential of simple framewise approaches to piano transcription,” *Proceedings of the 17th International Society for Music Information Retrieval (ISMIR) Conference*, pp. 475–481, 2016.

[77] E. Benetos and S. Dixon, “Multiple-instrument polyphonic music transcription using a temporally constrained shift-invariant model,” *The Journal of the Acoustical Society of America*, vol. 133, no. 3, pp. 1727–1741, 2013.

[78] Z. Duan, J. Han, and B. Pardo, “Multi-pitch streaming of harmonic sound mixtures,” *IEEE Transactions on Audio, Speech and Language Processing*, vol. 22, no. 1, pp. 138–150, 2014.

[79] G. E. Poliner and D. P. W. Ellis, “A discriminative model for polyphonic piano transcription,” *EURASIP Journal on Advances in Signal Processing*, vol. 48317, pp. 1–9, 2007.

[80] V. Arora and L. Behera, “Multiple F0 estimation and source clustering of polyphonic music audio using PLCA and HMRFs,” *IEEE/ACM Transactions on Audio Speech and Language Processing*, vol. 23, no. 2, pp. 278–287, 2015.

[81] Y. T. Wu, B. Chen, and L. Su, “Multi-instrument automatic music transcription with self-attention-based instance segmentation,” *IEEE/ACM Transactions on Audio Speech and Language Processing*, vol. 28, pp. 2796–2809, 2020.

[82] S. Russel and P. Norvig, *Artificial Intelligence—A Modern Approach*, 3rd ed. Pearson Education Limited, 2012.

[83] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006.

[84] C. Sun, A. Shrivastava, S. Singh, and A. Gupta, “Revisiting unreasonable effectiveness of data in deep learning era,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 843–852.

[85] D. Rolnick, A. Veit, S. Belongie, and N. Shavit, “Deep Learning is Robust to Massive Label Noise.” *arXiv preprint arXiv:1705.10694*, 2017.

[86] K. Choi, G. Fazekas, K. Cho, and M. Sandler, “A tutorial on deep learning for music information retrieval,” *arXiv preprint arXiv:1709.04396*, 2017.

[87] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[88] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, “Learning phrase representations using RNN encoder-decoder for statistical machine translation,” *arXiv preprint arXiv:1406.1078*, 2014.

[89] J. Bromley, I. Guyon, Y. Lecun, E. Sickinger, R. Shah, A. Bell, and L. Holmdel, “Signature verification using a "Siamese" time delay neural network,” *Advances in Neural Information Processing Systems*, vol. 6, 1993.

[90] G. Koch, R. Zemel, and R. Salakhutdinov, “Siamese neural networks for one-shot image recognition,” in *ICML Deep Learning Workshop*, 2015.

[91] O. Vinyals, G. Deepmind, C. Blundell, T. Lillicrap, K. Kavukcuoglu, and D. Wierstra, “Matching networks for one shot learning,” *Advances in Neural Information Processing Systems*, vol. 29, 2016.

[92] F. Sung, Y. Yang, L. Zhang, T. Xiang, P. H. Torr, and T. M. Hospedales, “Learning to compare: Relation network for few-shot learning,” in *Proceedings of the IEEE Conference on Computer*

Vision and Pattern Recognition, 2018, pp. 1199–1208.

[93] J. Pons, J. Serrà, and X. Serra, "Training neural audio classifiers with few data," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 16–20.

[94] E. Çakır, G. Parascandolo, T. Heittola, H. Huttunen, and T. Virtanen, "Convolutional recurrent neural networks for polyphonic sound event detection," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 25, no. 6, pp. 1291–1303, Feb. 2017.

[95] K. Lee, S. Maji, A. Ravichandran, S. Soatto, W. Services, U. C. San Diego, and U. Amherst, "Meta-learning with differentiable convex optimization," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 10 657–10 665. [Online]. Available: https://github.com/kjunelee/MetaOptNet

[96] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *International Conference on Machine Learning*. PMLR, 2017, pp. 1126–1135.

[97] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He, "A comprehensive survey on transfer learning," *Proceedings of the IEEE*, vol. 109, no. 1, pp. 43–76, Jan. 2021.

[98] G. Gwardys and D. Grzywczak, "Deep image features in music information retrieval," *International Journal of Electronics and Telecommunications*, vol. 60, no. 4, pp. 321–326, Dec. 2014.

[99] J. Lee and J. Nam, "Multi-level and multi-scale feature aggregation using pretrained convolutional neural networks for music auto-tagging," *IEEE Signal Processing Letters*, vol. 24, no. 8, pp. 1208–1212, Aug. 2017.

[100] K. Choi, G. Fazekas, M. Sandler, and K. Cho, "Transfer learning for music classification and regression tasks," *arXiv preprint arXiv:1703.09179*, Mar. 2017.

[101] B. Liang, G. Fazekas, and M. Sandler, "Transfer learning for piano sustain-pedal detection," in *International Joint Conference on Neural Networks (IJCNN)*, 2019, pp. 1–6.

[102] 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.

[103] M. Müller, *Fundamentals of Music Processing*. Springer International Publishing, 2015.

[104] J. Pons, O. Slizovskaia, R. Gong, E. Gómez, and X. Serra, "Timbre analysis of music audio signals with convolutional neural networks," in *25th European Signal Processing Conference (EUSIPCO)*, 2017, pp. 2744–2748.

[105] S. Theodoridis and K. Koutroumbas, *Pattern recognition*, 4th ed. Academic Press, 2009.

[106] B. McFee, A. Metsai, M. McVicar, S. Balke, C. Thomé, C. Raffel, F. Zalkow, A. Malek, Dana, K. Lee, O. Nieto, D. Ellis, J. Mason, E. Battenberg, S. Seyfarth, R. Yamamoto, viktorandreevichmorozov, K. Choi, J. Moore, R. Bittner, S. Hidaka, Z. Wei, nullmightybofo, D. Hereñú, F.-R. Stöter, P. Friesch, A. Weiss, M. Vollrath, T. Kim, and Thassilo, "librosa/librosa: 0.8.1rc2," May 2021. [Online]. Available: https://doi.org/10.5281/zenodo.4792298

[107] E. Benetos, S. Cherla, and T. Weyde, "An effcient shift-invariant model for polyphonic music transcription," *Proceedings of the 6th International Workshop on Machine Learning and Music*, 2013.

[108] C. Schörkhuber and A. Klapuri, "Constant-Q transform toolbox for music processing," in *Proceedings of the 7th Sound and Music Computing Conference, SMC*, 2010, pp. 3–64.

[109] J. C. Brown and M. S. Puckette, " An efficient algorithm for the calculation of a constant Q transform ," *The Journal of the Acoustical Society of America*, vol. 92, no. 5, pp. 2698–2701, 1992.

[110] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference for Learning Representations*, San Diego, 2015, pp. 1–15.

[111] E. Gordon-Rodriguez, G. Loaiza-Ganem, G. Pleiss, and J. P. Cunningham, "Uses and abuses of the cross-entropy loss: Case studies in modern deep learning," in *Proceedings on "I Can't Believe It's Not Better!" at NeurIPS Workshops*, 2020, pp. 1–10.

[112] J. Snoek, H. Larochelle, and R. P. Adams, "Practical Bayesian optimization of machine learning algorithms," *Advances in Neural Information Processing Systems*, vol. 25, 2012.

[113] G. K. P. Kaul, D. Golovin, and G. Kochanski, "Hyperparameter tuning in cloud machine learning engine using bayesian optimization," *Google Cloud Platform*, 2017.

[114] D. Golovin, B. Solnik, S. Moitra, G. Kochanski, J. Karro, and D. Sculley, "Google Vizier: A service for black-box optimization," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2017, pp. 1487–1495.

[115] J. Mockus, V. Tiesis, and A. Zilinskas, "The application of Bayesian methods for seeking the extremum," *Towards Global Optimization*, vol. 2, pp. 117–128, 1978.

[116] E. Brochu, V. M. Cora, and N. De Freitas, "A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning," *arXiv preprint arXiv:1012.2599*, 2010.

[117] F. Chollet *et al.*, "Keras," https://keras.io, 2015.

[118] TensorFlow Developers, "Tensorflow," Aug. 2021. [Online]. Available: https://doi.org/10.5281/zenodo.5181671

[119] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," *Proceedings of Machine Learning Research*, vol. 31, 2013.

ADDENDUM A HYPERPARAMETERS FOUND WITH BAYESIAN OPTIMIZATION

Hyperparameters for training models, found using Bayesian optimization, are summarized here.

A.1 PROTOTYPICAL MODELS

Bayesian optimization for prototypical model training optimized the number of classes in each training episode, the size of the output embedding and the initial learning rate of the learning rate scheduler. The ranges and scale of each hyperparameter to search are shown in Table A.1. The optimal parameters found that were used to train the models used in evaluation are shown in Table A.2.

Table A.1. The hyperparameter search space for training prototypical models. The scaling and range for each parameter are given.

Hyperparameter	Scale	Min	Max
Episode classes	Linear	20	100
Embedding size	Linear	640	2048
Initial learning rate	Log	0.00001	0.001

A.2 STANDARD CLASSIFIER MODELS

Bayesian optimization for standard classifier model training optimized the size of batches in training epochs and the initial learning rate of the learning rate scheduler. The ranges and scale of each hyperparameter to search are shown in Table A.3. The optimal parameters found that were used to further train the models used in evaluation are shown in Table A.4.

Table A.2. Training and architecture hyperparameters for prototypical models found with Bayesian optimization.

Model	**Episode classes**	**Embedding size**	**Initial learning rate**
Prototypical 3-shot mel DNN	72	1045	0.0001
Prototypical 3-shot CQT DNN	76	1115	0.0034
Prototypical 3-shot mel CNN	24	1054	0.00008
Prototypical 3-shot CQT CNN	24	1054	0.00008
Prototypical 3-shot mel RNN	65	963	0.00018
Prototypical 3-shot CQT RNN	100	1060	0.0001
Prototypical 3-shot mel CRNN	28	1056	0.0001
Prototypical 3-shot CQT CRNN	28	1056	0.0001

Table A.3. The hyperparameter search space for training standard classifier models. The scaling and range for each parameter are given.

Hyperparameter	**Scale**	**Min**	**Max**
Batch size	Linear	16	128
Initial learning rate	Log	0.00001	0.001

Table A.4. Training hyperparameters for standard classifier models found with Bayesian optimization.

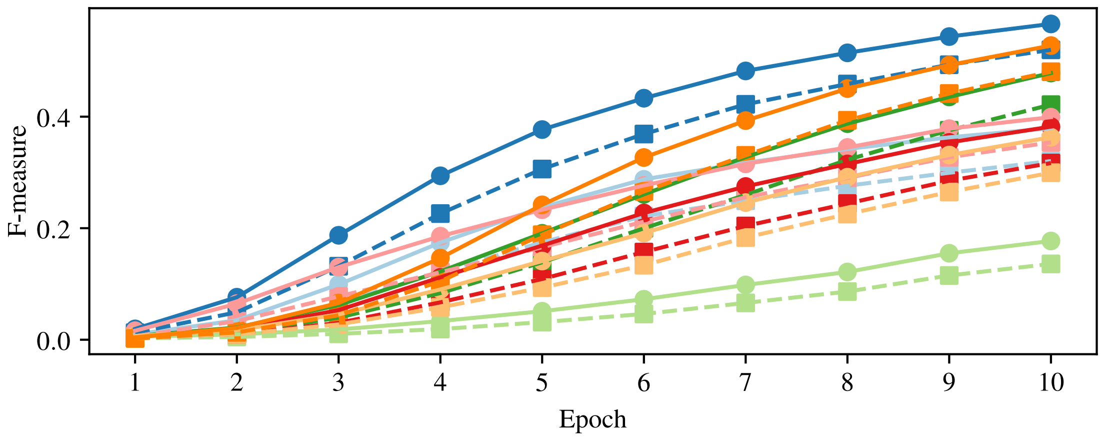
Model	**Batch size**	**Initial learning rate**
Standard mel DNN	50	0.00022
Standard CQT DNN	33	0.00028
Standard mel CNN	28	0.001
Standard CQT CNN	25	0.00037
Standard mel RNN	50	0.00022
Standard CQT RNN	63	0.00014
Standard mel CRNN	48	0.0007
Standard CQT CRNN	32	0.00054

ADDENDUM B ADDITIONAL RESULTS

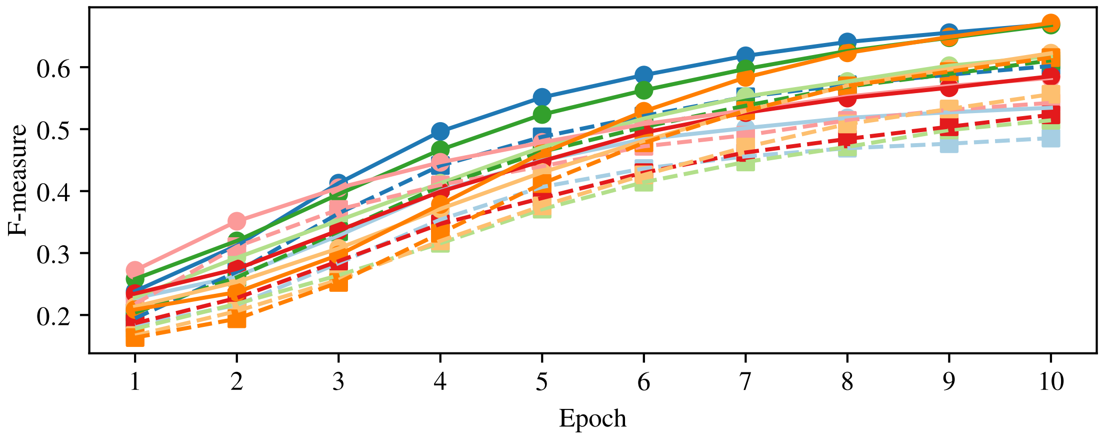
Additional results for evaluation on the OrchideaSOL dataset, YouTube tutorial examples and RWC clarinet examples are shown here. These results do not contribute information to the discussion that could not be gleamed from previous results, but are presented here for the sake of interest.

B.1 CROSS-DATASET EVALUATION ON ORCHIDEASOL

Figures B.1 and B.2 show the F-measure of the standard classifiers on the OrchideaSOL set after each of the additional training epochs after transfer learning. An increase in F-measure for additional training epochs is also observed on cross-evaluation data.



(a)



(b)

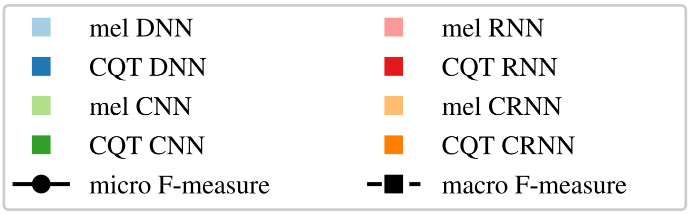
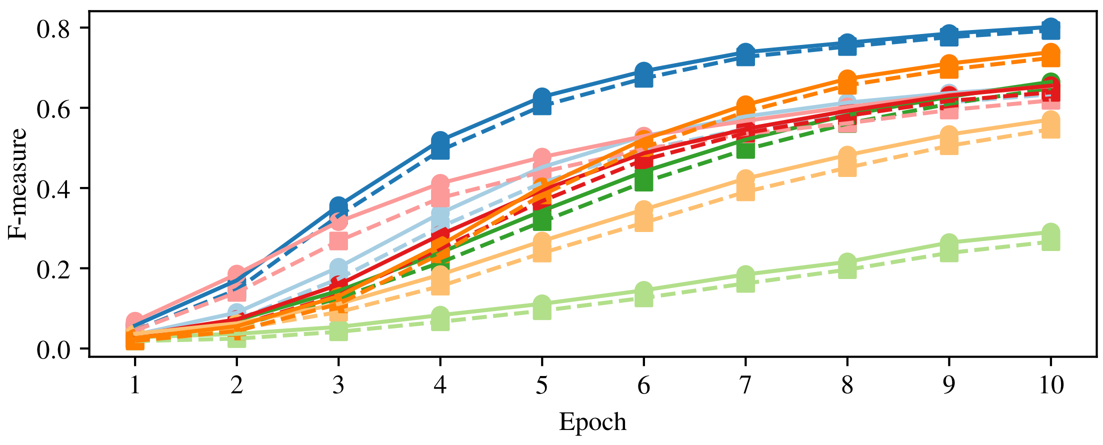
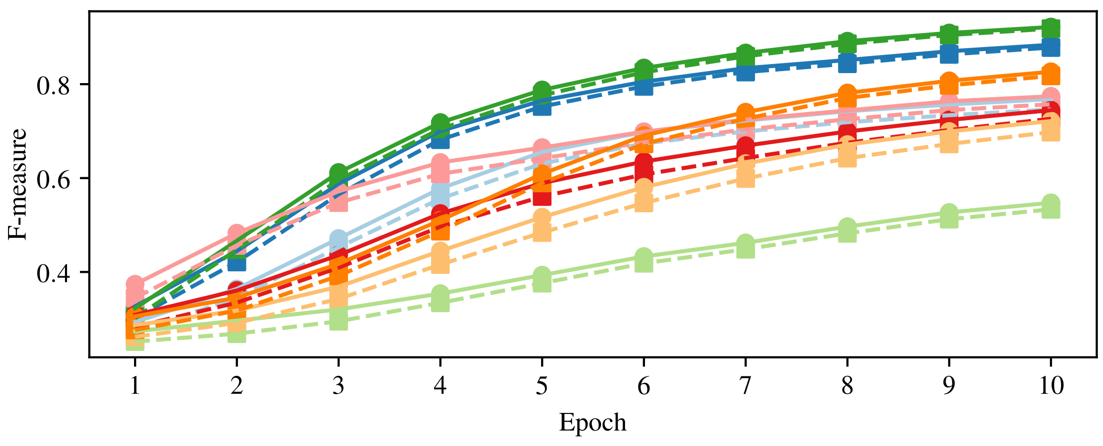


Figure B.1. Classification F-measure for (a) joint instrument, pitch and technique classification (INT) and (b) technique classification (T) on string instrument examples from the OrchideaSOL dataset for standard classifier models that were trained further on the prototype subset of the OrchideaSOL test set. Classification F-measure is shown after each of the 10 additional training epochs.



(a)



(b)

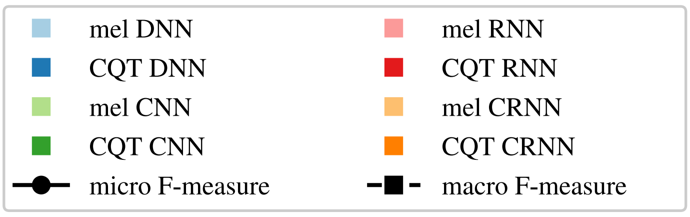
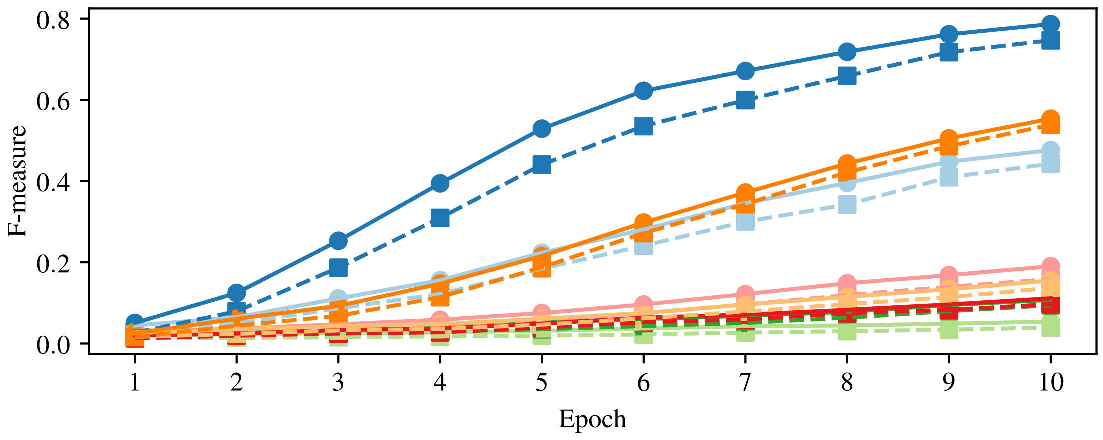


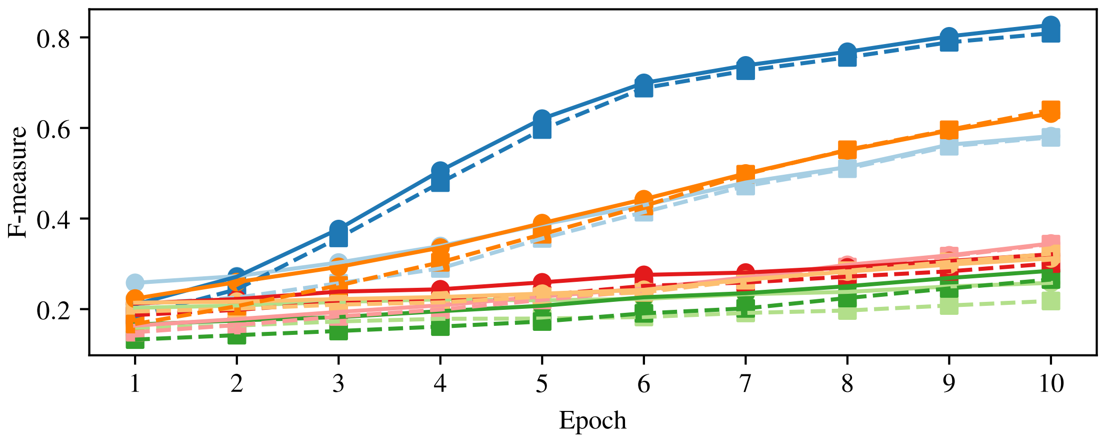
Figure B.2. Classification F-measure for (a) pitch classification (N) and (b) instrument classification (I) on string instrument examples from the OrchideaSOL dataset for standard classifier models that were trained further on the prototype subset of the OrchideaSOL test set. Classification F-measure is shown after each of the 10 additional training epochs.

B.2 EVALUATION ON REAL-WORLD YOUTUBE EXAMPLES

The F-measures on each problem after each additional training epoch after transfer learning on this set is shown in Figures B.3 and B.4. Again, F-measures improve with each additional epoch, however, some models show much steeper improvement than others, especially the DNN models and the CQT CRNN.



(a)



(b)

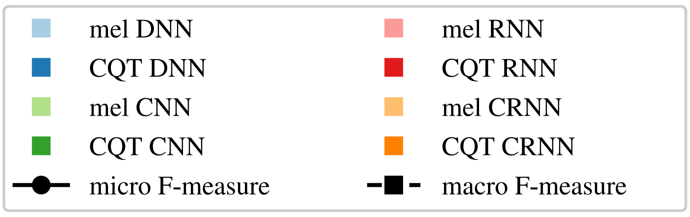
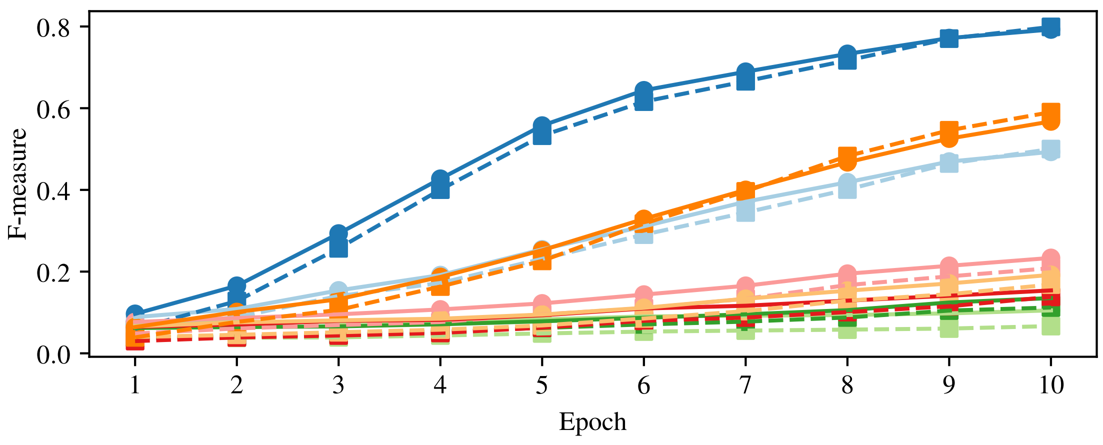
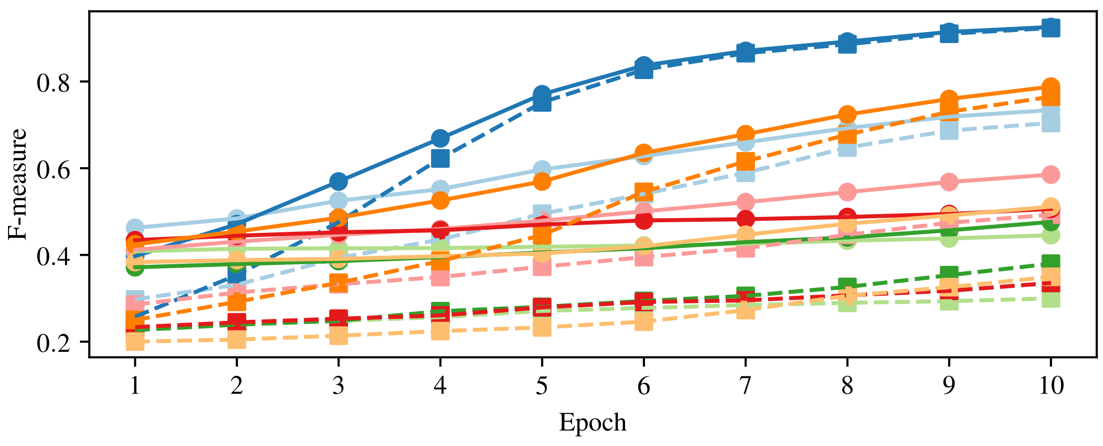


Figure B.3. Classification F-measure for (a) joint instrument, pitch and technique classification (INT) and (b) technique classification (T) on string instrument examples from the YouTube dataset for standard classifier models that were trained further on the prototype subset of the YouTube test set. Classification F-measure is shown after each of the 10 additional training epochs.



(a)



(b)

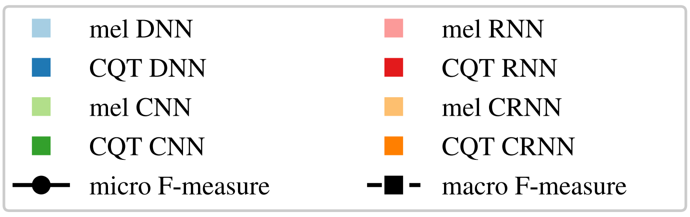
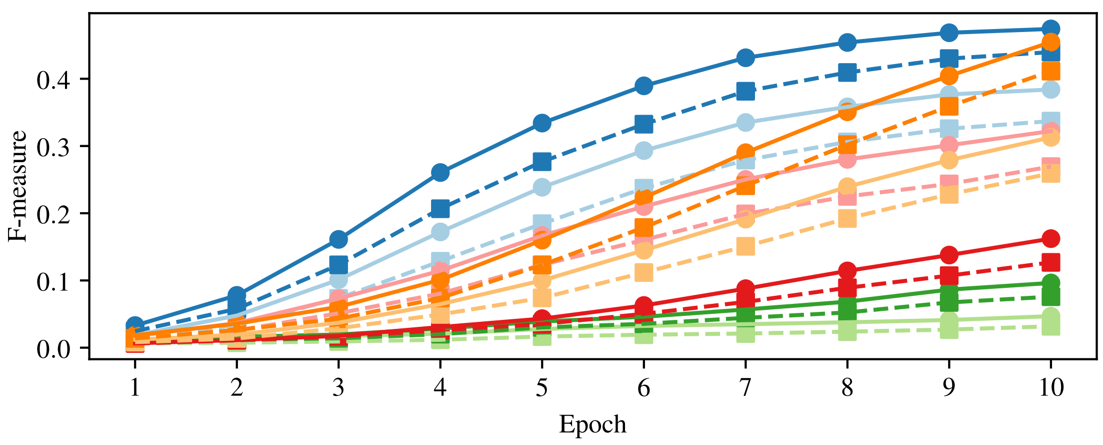


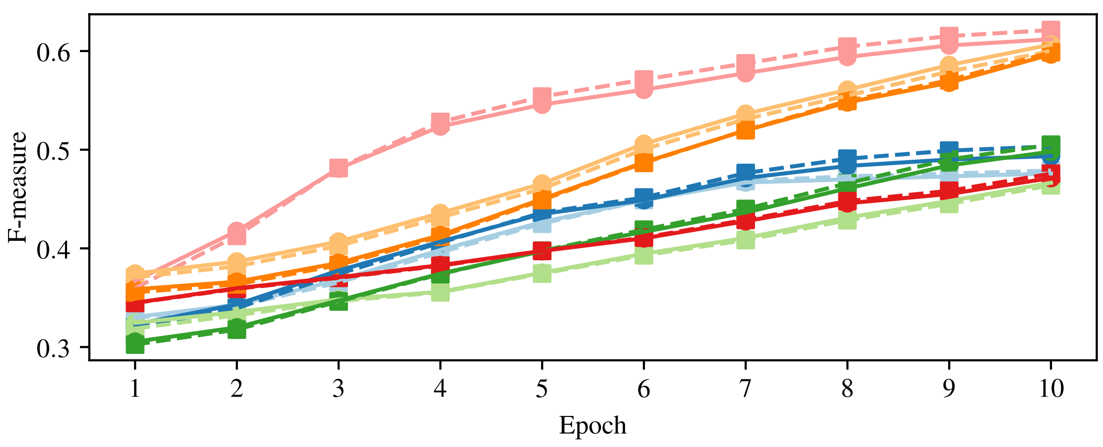
Figure B.4. Classification F-measure for (a) pitch classification (N) and (b) instrument classification (I) on string instrument examples from the YouTube dataset for standard classifier models that were trained further on the prototype subset of the YouTube test set. Classification F-measure is shown after each of the 10 additional training epochs.

B.3 EVALUATION ON A PREVIOUSLY UNSEEN INSTRUMENT

The transfer learning accuracies on the RWC clarinet set, evaluated for each of the standard classifier models after every additional epoch, is shown in FIgures B.5 and B.6. Each model again improves with additional epochs, but the mel CNN briefly performs worse on the sub-problems before improving again.



(a)



(b)

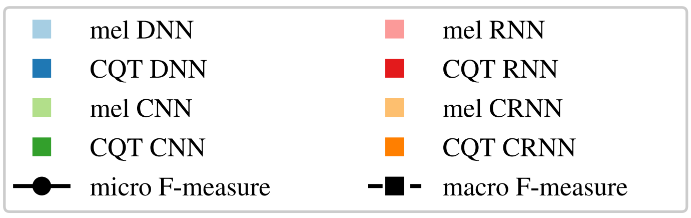


Figure B.5. Classification F-measure for (a) joint pitch and technique classification (NT) and (b) technique classification (T) on examples from the RWC dataset clarinet subset for standard classifier models that were trained further on the prototype subset of the RWC clarinet set. Classification F-measure is shown after each of the 10 additional training epochs.

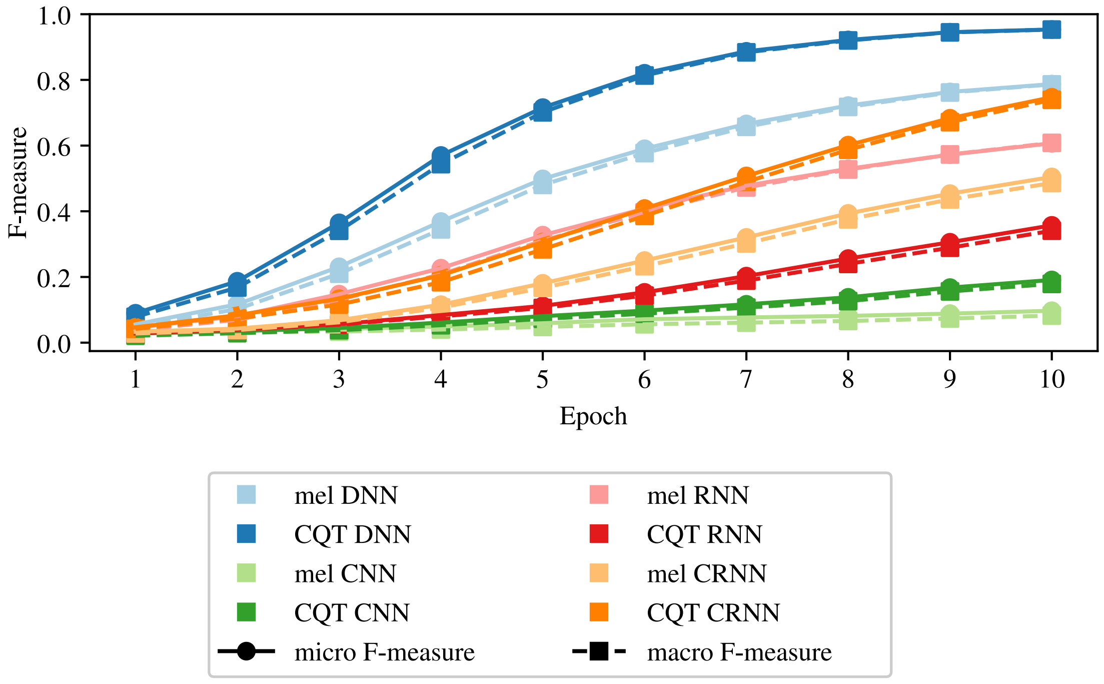


Figure B.6. Classification F-measure for pitch classification (N) on examples from the RWC dataset clarinet subset for standard classifier models that were trained further on the prototype subset of the RWC clarinet set. Classification F-measure is shown after each of the 10 additional training epochs.