Carnatic Rāga Classification Using Convolutional Neural Networks

Andrés Fernández Rodríguez

Goethe Universität Frankfurt am Main

October 13, 2017

Table of contents

Section 1

About Carnatic Music

General Aspects of Carnatic Music

- Oral tradition from South India (organized in schools called gharānā)
- Specialized ensemble of few soloists with predominance of melody
- Four relevant elements: rāga, tāla, composition and improvisation
    - **Rāga** is the melodic framework
    - Tāla is the rhythmic framework
    - Compositions and improvisation techniques depend on school/person
- Delivered in a concert (1-2 hours), playing several pieces (20-30 min)

A traditional carnatic ensemble (from [3, p.17])

Rāga Building Blocks

Each rāga has a specific set of the following melodic elements:

- **Svaras**: The basic pitches, defined with respect to the fundamental
- **Gamakas**: Ornaments applied to the svaras
- **Nyās svaras** The pitches that can be held longer
- **Ārōhana and Avrōhana**: ascending and descending melodic patterns, respectively
- **Phrases and Chalans**: Phrases are traditional *licks*, chalans are their underlying structure (a phrase without the gamakas)

Implicit features

Due to improvisatoric+oral nature of the repertoire, this building blocks present a high variance: **the svaras and chalans are conceptual and never appear without Gamakas**

Rāga Recognition Example

Example of the svara set identification for three different rāgas:

Rāga	S	R1	R2	G2/R3	G3	M1	M2	P	D1	D2/N1	N2/D3	N3
Śankarābharaṇaṁ	•		•		•	•		•		•		•
Harikāmbhōji	•		•		•	•		•		•	•	
Dhanyāsi	•	•		•		•		•	•		•	

The svara set of three diferent rāgas (from [3, p.244])

- Audio 1 is a clear example of the first rāga
- Audio 2 is a clear example of the last rāga, contrasting with A1
- Audio 3 is a clear example of the middle rāga, note similarity with A1
- Audio 4 is an ambiguous example, since the *Ni* svara never appears in the first minute

Section 2

Related Work on Rāga Classification

The CompMusic Project

- Multi-team project hosted at the UPF Barcelona ([12], [13])
- Effort towards data-driven and culture-aware approaches in the Music Information Retrieval field
- Compilated and curated an open **carnatic corpus** with around 2500 recordings, labeled with rāga, form, artists and tāla among others[14]
- On the rāga domain: at the moment of downloading, 2050 recordings (over 425 hours) distributed over 220 rāgas, the most represented one having over 60 recordings and 30 hours
- The Rāga Recognition Dataset ($\boldsymbol{RRD_{CMD}}$) is a test subset of the carnatic corpus covering 40 rāgas, with 12 recordings each, and around 124 hours of total duration[3, p.84]. "*The largest and most comprehensive ever for this task*"[3, p.183]

Relevant Approaches

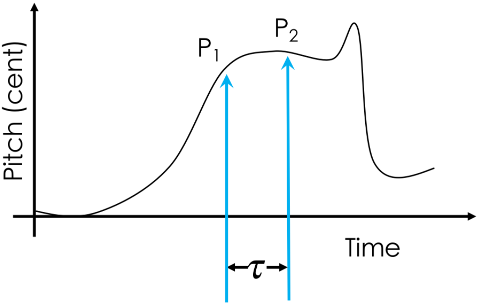
In his 2016 dissertation, Sankalp Gulati develops two model and tests them on the RRD_{CMD} subset. After extracting the melody line:

- Vector Space Modelling (VSM)[3, p.179], inspired in text processing: phrases↔words, rāgas↔topics, recordings↔documents
  1. clustering the phrases into a dictionary of building blocks[3, p.204]
  2. considering the recording a text-like vector of such blocks
  3. applying a classifier on top (semi-supervised)
  4. Accuracy below SoTA (67.3%). Sensitive to allied rāgas

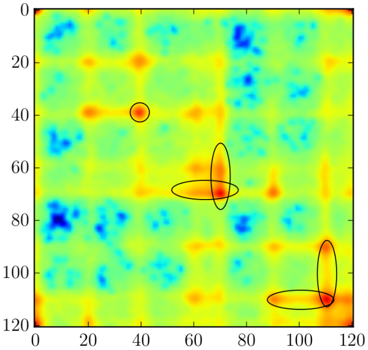
- Time-Delayed Melody Surface (TDMS)[3, p.192]
  1. normalize around tonic and wrap around octaves to get η distinct frequency bins
  2. scroll the normalized melody with two points of a short, fixed delay between them and add their pitch relations to an $\eta \times \eta$ histogram
  3. post-process and apply a classifier on top (semi-supervised)
  4. Accuracy well above SoTA (86.7%)

Characteristics of TDMS Representation

Compact, fast to compute, continuous, meaningful, performative.



The melodic contour of a piece is scrolled with a fixed, small delay τ and the (P_1, P_2) relations added to the TDMS (from [4, p.163])



A TDMS of a a music piece in rāga Yaman, with the octave divided in 120 bins. The frequent nyās and gamakas can be intuitively seen (from [4, p.170])

Section 3

Approach and Task Definition

Approach and Problem Definition

The improvement in performance by the TDMS model is given under the following conditions:

- Most of the carnatic corpus' data remains unused
- Semi-supervised methods were applied to fully labeled data
- Melody extraction supposes a very drastic compression of the data
- The time-frequency relations of the melody contour are enough to capture the rāgas' features

This, added to **my assumptions** that **the corpus is sizable**[11, p.iii], and that the **arguably related field of Music Genre Recognition**[11, p.17] (MGR) presents a lot of successful, recent research applying supervised end-to-end methods, made interesting and feasible the following

Task:

Perform carnatic rāga classification using supervised, end-to-end convolutional neural networks (CNNs) from the MGR field's SoTA.

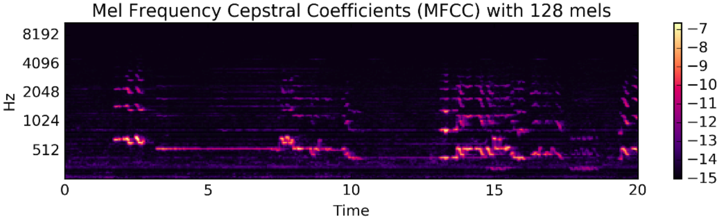
Section 4

Related Work on End-to-end Music Classification

The MGR Field

MGR is also a part of the MIR field. Main considerations[11, p.17]:

- A straightforward top-bottom definition of genre is also impossible. Many features apart from time and frequency (timbre, lyrics...)
- Also lack of unified&global criteria and data. Most of the research is western-based, around labeled song-sets like the Million Song Dataset (MSD)
- Apart from waveforms, popular representations are STFT and MFCC (so-called time/frequency representations, because they capture the sound intensity through time for specific frequency bands)

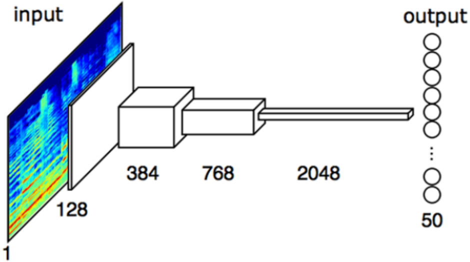


the beginning of a piece in Kalyāṇi rāga: *Sikkil Sisters – Nijadasa Varada* (audio 5)

Recent MGR Approaches

Notable work has been developed in [6], [9], [8], [7] and [10]:

- MSD (training: 201,680 – validation: 12,605 – test: 25,940 songs)
- Multi-tagging on top-50 tags including genres (rock, pop, funk) and moods (sad, happy, chill)
- FCN4 Achieved 0.894 AUC-ROC on the test subset



One of the fully-convolutional neural architectures (FCN4) proposed in [6]. The numbers indicate the number of feature maps (i.e. channels)

Supervised Learning with NNs – Highlights I

- Neural networks are models that alternate linear with non-linear transformations to provide the so-called **hypothesis** $h_\theta(x)$, based on some set of **parameters** θ. CNNs are NNs that have at least one convolution
- For a labeled **sample** (x_i, y_i), given a **cost function** $J(h(x_i), y_i)$ between its **label** and the corresponding hypothesis, it is possible to measure the error of the network. The **learning** process consists in alter the parameters to minimize this error
- J is usually non-convex, so it is optimized using the **gradient descent** algorithm: $\theta^{(t+1)} := \theta^{(t)} - \alpha \frac{\partial}{\partial \theta} J(X, Y)$, that follows the derivative of J downwards proportionally to α, the **learning rate**

Supervised Learning with NNs – Highlights II

- Once the network memorizes the training data, stops learning from it. This lack of generalization ability is known as **overfitting**, and the measures to prevent it as regularization. Two important regularization techniques are **weight decay** and **dropout**
- The size of X, Y (**batch size**) is also a regularization factor. The parameters that aren't trained by the network are called **hyperparameters**, and are usually optimized by splitting the non-training data into **test** and **validation**, and doing a **grid search**
- CNNs usually alternate convolution with **pooling**. This combo helps to capture local features. Usually this is done several times, increasing the number of filters and decreasing their size (**repr. learning**)
- CNNs are based on the idea of **parameter sharing**, which allow deeper architectures that perform empirically better[1, p.202]

Section 5

Experiments and Results

Guidelines and Implementation

Choosing a convolutional architecture seemed feasible to me because of the following reasons:

- CNNs should be able to ignore the background drone and percussion, as they are independent from the labeled rāga, and filter out the melodic contour in a way similar to the input of the VSM and TDMS algorithms
- Being able to detect shapes on a time/frequency representation of a melodic contour is equivalent to being sensitive to its time/frequency changes
- CNNs are locally limited, but Locality is explicitly recognized as an advantage in the TDMS model

The code was implemented on TensorFlow. A first test setup training the CNN suggested in [2] on the MNIST classification problem achieved a 98% accuracy in less than 2 training epochs (100,000 training samples).

Rock vs. Merengue

After the success with the *vanilla* MNIST setup, I tried with some data of my own, selecting 20 songs representative of each style (audios 6 and 7), achieving a performance consistently above 90% after two epochs. This could be due to the great timbrical differences.

Layer	**Shape**
Input	$512{\times}86 \times 1$
Conv512x5x1x4	$1{\times}82 \times 4$
Conv1x10x4x6	$1{\times}73 \times 6$
Conv1x15x6x8	$1{\times}59 \times 8$
Conv1x28x8x12	$1{\times}40 \times 12$
FullyHidden	$40{\cdot}12 \times 24$
Logits	24×2

The best-performing architecture among the tested ones. It has a ReLU after each convolution

Hyperparameter	**Value**
Chunk size	1 sec
Batch size	20
Learning rate (SGD)	10^{-4}
Weight decay rate	0.2
Dropout	none

The corresponding hyperparametrization

Augmented Carnatic Corpus – Approach

After those two toy-setups, I switched to the whole corpus:

- To enforce intra- and inter-class balance, I applied data augmentation to the 40 classes present in the RRD_{CMD} dataset by time-stretching by $unif \sim [0.7, 1.3]$
- I trained on the STFT expecting the network to adapt the exponential frequency scale to its needs, as in [5]

Layer	Shape
Input	$257{\times}625 \times 1$
Conv3x3x1x128 MaxPool2x3	$255{\times}623 \times 128$ $127{\times}207 \times 128$
Conv3x3x128x384 MaxPool4x5	$125{\times}205 \times 384$ $31{\times}41 \times 384$
Conv3x3x384x768 MaxPool5x6	$29{\times}39 \times 768$ $7{\times}6 \times 768$
Conv3x3x768x2048 MaxPool6x4	$3{\times}4 \times 2048$ $1{\times}1 \times 2048$
Logits(Conv1x1x40)	$1{\times}1 \times 40$

Layer	Shape
Input	$257{\times}625 \times 1$
Conv3x3x1x128 MaxPool2x3	$255{\times}623 \times 128$ $127{\times}207 \times 128$
Conv3x3x128x256 MaxPool3x3	$125{\times}205 \times 256$ $41{\times}68 \times 256$
Conv3x3x384x512 MaxPool3x4	$39{\times}66 \times 512$ $13{\times}16 \times 512$
Conv3x3x768x1024 MaxPool3x4	$11{\times}14 \times 1024$ $4{\times}3 \times 1024$
Conv3x2x768x2048 MaxPool2x2	$2{\times}2 \times 2048$ $1{\times}1 \times 2048$
Logits(Conv1x1x40)	$1{\times}1 \times 40$

Adaption of the **FCN4** and **FCN5** architectures[6], respectively. There are ReLU layers between convolution and pooling. There is a ReLU and a dropout layer before the logits, in that order.

Augmented Carnatic Corpus – Some Hyperpars.& Results

- With Adam optimizer and initial learning rate $= 10^{-4}$, batch size $=$ 10 and chunk size $=$ 20 seconds, grid search on $\mathit{architecture}\{FCN4, FCN5\} \times \mathit{weightdecay}\{10^{-9}, 3\cdot 10^{-9}, 10^{-8}, 3\cdot 10^{-8}, 10^{-7}, 3\cdot 10^{-7}, 10^{-6}, 3\cdot 10^{-6}, 10^{-5}, 3\cdot 10^{-4}\} \times \mathit{dropout}\{1, 0.95, 0.9, 0.85, 0.8, 0.75, 0.65, 0.5\}$
- Tested similar settings with *vanilla* SGD optimizer for $\mathit{learningrate}\{10^{-4}, 10^{-5}, ...10^{-9}\}$
- Same with momentum optimizer for $\mathit{momentum}\{0.1, 0.2, 0.5, 0.8, 0.9\}$
- Modified the FCN architectures to allow different convolution depths and tested with $\mathit{basedepth}\{8, 16, 35, 50, 60, 80\}$

Results

The general observed behaviour was that all the different variants overfitted with no regularization at all, but seemed unable to learn even the training data when the least amount of regularization of any kind was introduced. In any of the cases, the overall accuracy of the validation results wasn't better than random guessing.

Rāga Recognition Dataset – Approach

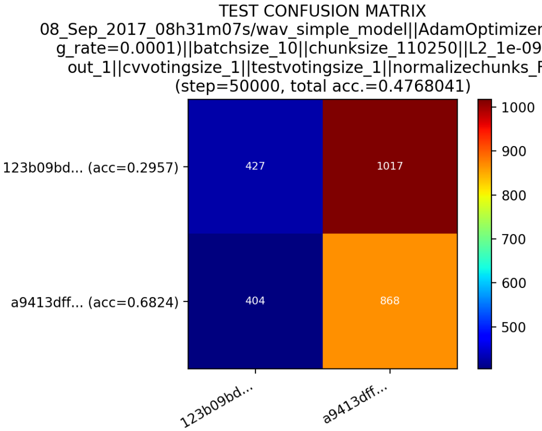
- Switched to the smaller MFCC representation to allow bigger batches, and also to the CQT to incorporate the logarithmic scalation of the frequency axis to the preprocessing. Also did some final tests on the plain wav files
- Reduced the problem space to the RRD_{CMD} (hoping for denser data), first with 5 arbitrary rāgas and then with 2 contrasting ones.
- Due to the observed contrast between the overfitting with no regularization and the rapid gaining of bias, I oriented my grid search to find a "sweet spot" between them.

For an example of the grid search performed, together with the thought lines behind them, see the next slide, extracted from the repository's *3_main_pipeline.py* file.

RRD_{CMD} – Finetuning Example

```
#  bsize, chsize,     optimizer                              opt args,                     l2,       dropout            model
#this tended to say always class1
l1 = [40, 20*43, tf.train.GradientDescentOptimizer, {"learning_rate":lambda x:1e-5},lambda x:0, lambda x: 1, carnatic_model_basic_base1
# this rises batch acc about 0.2 every 5k steps, but starts overfitting a lot after 7k
l2 = [40, 20*43, tf.train.AdamOptimizer, {"learning_rate":lambda x:1e-5},lambda x:1e-10, lambda x: 1, carnatic_model_basic_base16]
# this also overfits, more l2!
l3 = [40, 20*43, tf.train.AdamOptimizer, {"learning_rate":lambda x:1e-5},lambda x:1e-9, lambda x: 1, carnatic_model_basic_base16]
# this overtfits even more than l3: train goes up normally, but test cost skyrokets and acc. blocks bc it says that everything is class
l4 = [40, 20*43, tf.train.AdamOptimizer, {"learning_rate":lambda x:1e-5},lambda x:1e-8, lambda x: 1, carnatic_model_basic_base16]
# lets see with dropout
l5 = [40, 20*43, tf.train.AdamOptimizer, {"learning_rate":lambda x:1e-5},lambda x:0, lambda x: 0.9, carnatic_model_basic_base16]
l6 = [40, 20*43, tf.train.AdamOptimizer, {"learning_rate":lambda x:1e-5},lambda x:0, lambda x: 0.8, carnatic_model_basic_base16]
l7 = [40, 20*43, tf.train.AdamOptimizer, {"learning_rate":lambda x:1e-5},lambda x:0, lambda x: 0.7, carnatic_model_basic_base16]
l8 = [40, 20*43, tf.train.AdamOptimizer, {"learning_rate":lambda x:1e-5},lambda x:0, lambda x: 0.6, carnatic_model_basic_base16]
# actually, more dropout overfits too (l5-l8)... weird!!. The CV cost skyrockets and the batch acc rises fast too. So lets reduce the m
l9  = [60, 20*43, tf.train.AdamOptimizer, {"learning_rate":lambda x:1e-5},lambda x:1e-9, lambda x: 1, carnatic_model_basic_base2]
l10  = [60, 20*43, tf.train.AdamOptimizer, {"learning_rate":lambda x:1e-5},lambda x:1e-9, lambda x: 1, carnatic_model_basic_base4]
l11  = [60, 20*43, tf.train.AdamOptimizer, {"learning_rate":lambda x:1e-5},lambda x:1e-9, lambda x: 1, carnatic_model_basic_base6]
l12  = [60, 20*43, tf.train.AdamOptimizer, {"learning_rate":lambda x:1e-5},lambda x:1e-9, lambda x: 1, carnatic_model_basic_base8]
l13  = [60, 20*43, tf.train.AdamOptimizer, {"learning_rate":lambda x:1e-5},lambda x:1e-9, lambda x: 1, carnatic_model_basic_base10]
l14  = [60, 20*43, tf.train.AdamOptimizer, {"learning_rate":lambda x:1e-5},lambda x:1e-9, lambda x: 1, carnatic_model_basic_base12]
l15  = [60, 20*43, tf.train.AdamOptimizer, {"learning_rate":lambda x:1e-5},lambda x:1e-9, lambda x: 1, carnatic_model_basic_base14]
# confmatrix were more distributed, but still cv barely above random.
# At this point i would conclude that the mnist-based models arent good for this data.
# I will try to go for the CQT, bigger kernels (with valid padding) and less layers:
l16 = [10, 20*43, tf.train.AdamOptimizer, {"learning_rate":lambda x:1e-5},lambda x:0, lambda x: 0.5, fcn4]
l17 = [10, 20*43, tf.train.AdamOptimizer, {"learning_rate":lambda x:1e-5},lambda x:0, lambda x: 0.7, fcn4]
l18 = [10, 20*43, tf.train.AdamOptimizer, {"learning_rate":lambda x:1e-5},lambda x:0, lambda x: 0.9, fcn4]
l19 = [10, 20*43, tf.train.AdamOptimizer, {"learning_rate":lambda x:1e-5},lambda x:0, lambda x: 1, fcn4]
# all this networks cant even get the training data, even with basenum=128. So I removed dropout and printed now the alpha to see whats
l20= [2, 20*43, tf.train.AdamOptimizer, {"learning_rate":lambda x: expdecay(x, 1, 0.1)},lambda x:1e-5, lambda x: 1, fcn4]
```

RRD – Confusion Matrix Example



An examination of the confusion matrices shows that the model is learning orthogonally to the classification boundary, that is, features that don't play a role in rāga classification.

Section 6

Conclusions and Follow-Up

Conclusions and Follow-up

- Conclusions:
    1. The chosen time/frequency representations are probably too sparse to allow any kind of regularization to be effective
    2. basic preprocessing like cutting off non-melodic segments would have been instrumental
    3. Histograms of the network weights would have helped the diagnose
- Follow-up:
    1. Train similar networks on the TDMS to confirm data issue
    2. Incorporating the TDMS to other problem domains (f.e. western MGR)
    3. Implementing the auralization technique[7] (a sort of deconvolution for audio) as a diagnose tool to get a beter insight on the networks' problems

THANK YOU

Questions?

Bibliography

Ian Goodfellow Yoshua Bengio and Aaron Courville. "Deep Learning". Book in preparation for MIT Press. 2016. URL: http://www.deeplearningbook.org.

Convolutional Neural Networks – Model Training. Tutorial part of the TensorFlow API. 2017. URL: https://www.tensorflow.org/tutorials/deep_cnn#model_training.

Sankalp Gulati. "Computational Approaches for Melodic Description in Indian Art Music Corpora". PhD Thesis directed by Prof. Xavier Serra Casals at the Universitat Pompeu Fabra in Barcelona. PhD thesis. 2016.

Sankalp Gulati. *Computational Approaches for Melodic Description in Indian Art Music Corpora*. Slides for the PhD Dissertation the Universitat Pompeu Fabra in Barcelona. 2016. URL: mtg.upf.edu/system/files/publications/PhD_SankalpGulati.pdf.

Keunhyoung Luke Kim Jongpil Lee Jiyoung Park and Juhan Nam. "Sample-Level Deep Convolutional Neural Networks For Music Auto-Tagging Using Raw Waveforms". In: (2017). Korea Advanced Institute of Science and Technology (KAIST). URL: http://arxiv.org/abs/1703.01789.

György Fazekas Keunwoo Choi and Mark Sandler. "Automatic Tagging Using Deep Convolutional Neural Networks". In: (2016). 17th International Society for Music Information Retrieval Conference. URL: http://arxiv.org/abs/1606.00298.

Kyunghyun Cho Keunwoo Choi György Fazekas and Mark B. Sandler. "Explaining Deep Convolutional Neural Networks on Music Classification". In: *CoRR* (2016). URL: http://arxiv.org/abs/1607.02444.

Kyunghyun Cho Keunwoo Choi György Fazekas and Mark B. Sandler. "On the Robustness of Deep Convolutional Neural Networks for Music Classification". In: *CoRR* (2017). URL: http://arxiv.org/abs/1706.02361.

Mark B. Sandler Keunwoo Choi György Fazekas and Kyunghyun Cho. “Convolutional Recurrent Neural Networks for Music Classification”. In: *CoRR* (2016). URL: http://arxiv.org/abs/1609.04243.

Mark B. Sandler Keunwoo Choi György Fazekas and Kyunghyun Cho. “Transfer Learning for Music Classification and Regression Tasks”. In: *CoRR* (2017). URL: http://arxiv.org/abs/1703.09179.

Andrés Fernández Rodríguez. “Machine Learning for Music Classification”. Bachelor’s thesis supervised by Prof. V. Ramesh at the Goethe University Frankfurt a. M. 2016. URL: https://github.com/andres-fr/bachelor-thesis.

Xavier Serra. “A Multicultural Approach in Music Information Research”. In: (2011). 12th International Society for Music Information Retrieval Conference (ISMIR). URL: http://mtg.upf.edu/node/2276.

Xavier Serra. “Creating Research Corpora for the Computational Study of Music: the case of the CompMusic Project”. In: (2014). URL: http://mtg.upf.edu/node/2899.

Ajay Srinivasamurthy et al. “Corpora for Music Information Research in Indian Art Music”. In: (2014). International Computer Music Conference/Sound and Music Computing Conference (Athens, Greece). URL: http://mtg.upf.edu/node/2990.