DEEPER CONVOLUTIONAL NEURAL NETWORKS AND BROAD AUGMENTATION POLICIES IMPROVE PERFORMANCE IN MUSICAL KEY ESTIMATION

Stefan Andreas Baumann
stefan-baumann@outlook.com

ABSTRACT

In recent years, complex convolutional neural network architectures such as the Inception architecture have been shown to offer significant improvements over previous architectures in image classification. So far, little work has been done applying these architectures to music information retrieval tasks, with most models still relying on sequential neural network architectures. In this paper, we adapt the Inception architecture to the specific needs of harmonic music analysis and use it to create a model (InceptionKeyNet) for the task of key estimation. We then show that the resulting model can significantly outperform state-of-the-art single-task models when trained on the same datasets. Additionally, we evaluate a broad range of augmentation methods and find that extending augmentation policies to include a more diverse set of methods further improves accuracy. Finally, we train both the proposed and state-of-the-art single-task models on differently sized training datasets and different augmentation policies and compare the differences in generalization performance.

1. INTRODUCTION

Determining the key of a music piece is an essential step when analyzing its harmonic properties. Besides theoretical music analysis, this property is used for assessing the harmonic compatibility of music pieces [1], which is crucial when mixing multiple music pieces, as is often done by disc jockeys. As the determination of the key requires expert knowledge, systems for automatic key estimation are crucial for enabling large-scale analyses and enabling everyone, regardless of prior knowledge and skill, to harness key information. This can be especially useful for algorithmically generated mixes/playlists (e.g. Spotify's Daily Mix [1]): by optimizing the ordering of these playlists to maximize harmonic compatibility, the perceived quality of these mixes can potentially be improved. We believe that for informing these decisions, regardless of whether they are made algorithmically or by a disc-jockey, improving the performance of key estimation algorithms is crucial to enable improvements in the creation of those mixes.

[1] https://newsroom.spotify.com/2018-05-18/how-your-daily-mix-just-gets-you/

2. METHODS

2.1 Audio preprocessing

Before the audio data is put into the neural network model, it is preprocessed to generate a time-frequency domain representation. To obtain this representation, we use a Constant-Q transform as implemented in the librosa Python package [2] with 24 bands per octave distributed over the range from C_1 to C_8 and downsample to obtain 5 frames per second, the same framerate as used in [3].

2.2 Model

Many state-of-the-art key estimation models (e.g., [3–6]) use sequential convolutional neural networks which work on a time-frequency representation of the audio sample. Recent work has shown that more modern image classification architectures like Inception [7] and ResNet [8] are capable of outperforming at least some sequential architectures such as VGG [9] and AlexNet [10] on the task of audio classification [11]. Inspired by these findings, we present a model, which we call *InceptionKeyNet*. We base it on Inception V3 [7], as this architecture has shown the best performance on 2/3 metrics in an evaluation of audio classification models [11]. Compared to VGG-like state-of-the-art feature extractors in key estimation models, the Inception architecture is appealing as it promises much lower computational cost for similar performance [7]. Furthermore, it also promises to be especially useful in the context of localization [12], a property that could be expected to be advantageous for key estimation, as at least the determination of the key root requires precise localization capabilities. Like in the previously mentioned evaluation, we apply some modifications to the model to adapt it to the task: our primary modification is the removal of the local pooling layers from the model. This modification is necessary, as the exact "location" of features in the spectrogam on the frequency axis decides the corresponding pitch, and applying multiple steps of pooling there would prevent the model from being able to distinguish between single pitches. We also adjust the stride of all convolution layers except for the first one to be 1 for the same rea-

son. Furthermore, we also remove the auxiliary network as in [11]. Finally, we scale down the number of filters by 80% (rounding down to the next integer value) of those from the original model. We do this to reduce the capacity of the model to compensate for the significantly smaller number of output classes and training samples as compared to ImageNet [13], the dataset Inception V3 was optimized for [2].

2.3 Training method

For training, we split the training samples into 5 folds of equal size. We train 5 separate model instances for each round, where each instance uses a different fold for validation and trains on the samples from the remaining four, as is typical for cross-validation. To increase variety in the data, we only give a random 20s snippet of the whole audio sample to the model during training, as in [3]. We train the models with the Adam [15] optimizer with a learning rate of 0.001 and use a batch size of 32, optimizing the categorical cross-entropy between the 24 output classes – one for each combination of pitch and either major or minor mode – and the ground truth target. During training, we use a dropout rate of $p = 0.5$.

Furthermore, we use early stopping to determine the end of the training process, stopping the training when the loss does not improve over 50 epochs. For the resulting model, the weights from the epoch with minimal validation loss are used. Those instances are then combined into an ensemble, where the class probabilities are averaged to obtain a single prediction, to further improve performance.

2.4 Datasets

Similar to previous works like [3], we utilize various datasets spanning multiple genres, for both training and testing. For training, we use the following datasets:

GiantSteps MTG Key: A dataset of 1486 key annotations [16] generated from user corrections from the Beatport service in the same way as the GiantSteps Key dataset [17]. The music pieces are primarily focused on electronic dance music [3].

McGill Billboard: A dataset of 742 songs from the American Billboard charts between 1958 and 1991 [18] [19]. While the keys are not annotated in the original version, there exists a version with key annotations for a subset of 625 songs [20], which we use. Of these, we were able to obtain audio excerpts for 617 pieces.

For some trainings, we add an additional dataset, created using data mining. It consists of 3410 additional key annotations (overlaps with other datasets have been excluded) for which we have audio excerpts available, primarily focused on popular music pieces, but also including some classical pieces.

As only the GiantSteps datasets have audio available as a part of the dataset, we collect 30-second excerpts for all dataset entries, resulting in up to two such excerpts per entry. If multiple excerpts are available for a single music piece, we randomly choose the excerpt to use for each epoch when training. When validating and testing, we use the longer 120-second excerpt from the GiantSteps datasets if available, and choose randomly otherwise, ensuring that the same excerpts are used between tests.

[2] Multiple other approaches including modifying the stem and leaving out some mixed blocks were evaluated, too, but the best performance was achieved when just scaling down the number of filters.

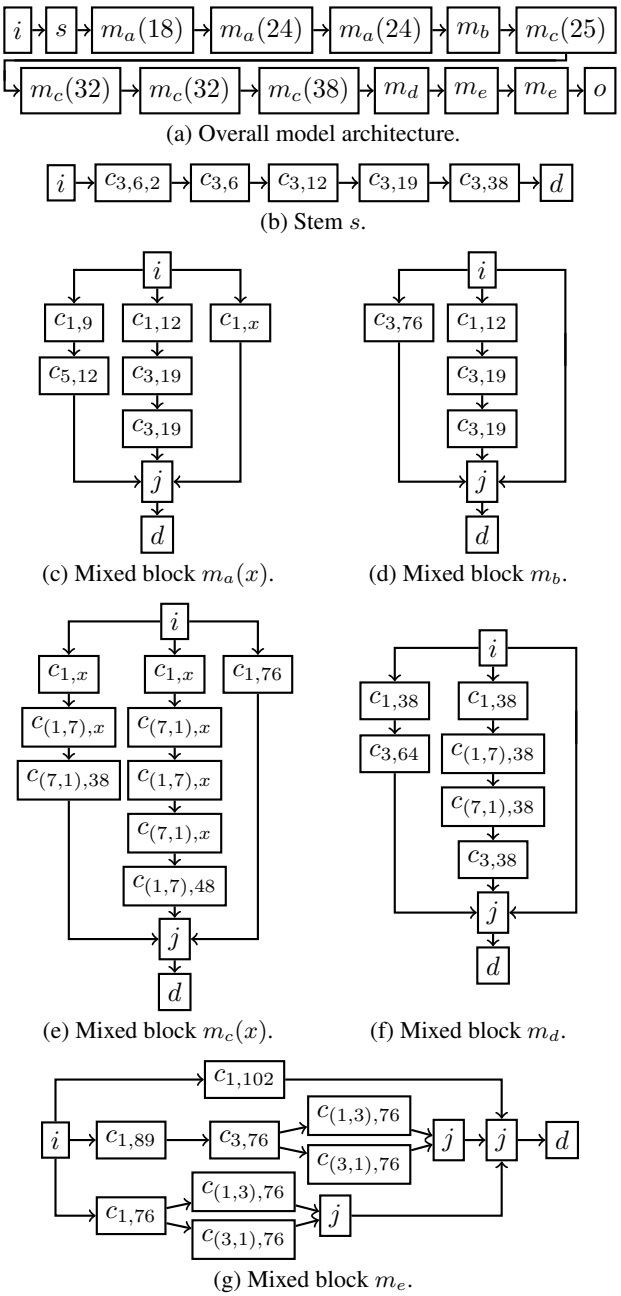

Figure 1: The architecture of the feature extractor of the InceptionKeyNet model. i denotes the input of the model or a section, $c_{k,n,s}$ (or $c_{(k_1,k_2),n,s}$) denotes a 2-dimensional convolution layer with a kernel size of $k \times k$ (or $k_1 \times k_2$), a stride of $s \times s$ and n filters, followed by batch normalization and ReLU activation. If s is not given, it defaults to 1. j denotes a layer that joins all preceding layers along their filter axis. d denotes a Spatial Dropout [14] layer with Dropout probability p. Different output structures o are presented and evaluated in 3.1.

2.5 Metrics

For evaluating the performance of key classification models, we use the same score as used in MIREX evaluations [21], which we refer to as the "MIREX score". For this, we use the mir_eval library [3] [22]. When computing the score, the predictions are divided into the following categories to consider how close the classification result is to the ground truth key:

Correct: Predictions where the root and mode are classified correctly. These predictions give a full point.

Perfect fifth: Predictions where the mode has been classified correctly, and the predicted root is either a fifth (7 semitones) higher or lower than the ground truth. These predictions give 0.5 points.

Relative major/minor: Predictions where the major or minor key relative to the ground truth key have been classified. These predictions give 0.3 points.

Parallel major/minor: Predictions where the root has been classified correctly while the mode differs. These predictions give 0.2 points.

Any predictions that do not fall under any of the previously listed categories, give no points. These point scores are then averaged to obtain the overall MIREX score.

3. EXPERIMENTS

3.1 Model Output Structures

While the original Inception architecture is already theoretically capable of processing inputs of arbitrary size when some minimum dimensions are met, we also evaluate the output structure used in the AllConv model and a modified version of it, to find out whether different output architectures affect the model performance. We evaluate the following three different output structures:

Original: The original output structure from the Inception v3 architecture, which consists of global average pooling, followed by a fully-connected layer with one neuron for each class and softmax activation [7].

AllConv-A: A 1×1 convolution layer with one filter for each class, followed by global average pooling and softmax activation as found in the AllConv model [3].

AllConv-B: Our modified variant of the AllConv-A structure, where the frequency dimension of the convolution layer is extended to cover the whole frequency axis from the previous layer.

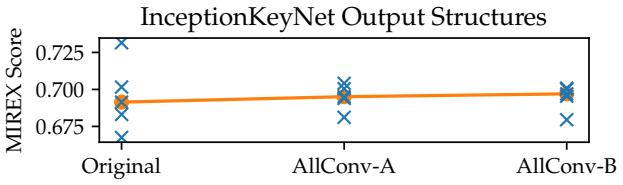

Figure 2: The validation MIREX score achieved with three different output structures. Blue crosses correspond with one model, orange dots represent the median scores.

[3] The scoring of fifth errors has to be changed to also accept descending ones to match the scoring used in the latest MIREX evaluations.

The results from the evaluation are shown in figure 2. It can be seen that while all three output structures show comparable average performance, the original output structure has a significantly higher variance in validation performance than the other two variants. Overall, the performance achieved by the different folds is the most consistent with the AllConv-B variant, so we choose to proceed using it as the model output structure.

3.2 Augmentation Methods

To artificially increase the diversity of the training dataset, augmentation can be applied to the input data. While previous works on key estimation models primarily relied on pitch-shifting as the sole augmentation method (e.g. [3–6]), we evaluate a range of different augmentation methods to see whether a more diverse set of augmentation methods can help improve generalization. We evaluate the following augmentation methods (see figure 3 for visualizations of some of these methods):

Pitch-Shifting: A method that has already been used in previous works on key classification models [3–6]. It works by shifting all of the sounds in a recording by a number of semitones and adjusting the target key root accordingly. For our evaluation, we test symmetric pitch shifting ranges, which are defined via the hyperparameter $\Delta f_{max} \in \mathbb{N}_0$, which controls the discrete uniform distribution $X_{\Delta f,\text{pitch shift}} \sim \mathcal{U}\{-\Delta f_{max}, \Delta f_{max}\}$ used to randomly determine the pitch shift $\Delta f_{\text{pitch shift}}$. We precompute all of the potential pitch-shifted versions for all training samples ahead-of-time and apply it in the time-domain using SOX [4] [24] before applying the Constant-Q transform.

Time-Warping: Based on the time-warping methodology used in the SpecAugment augmentation policy [25] for speech recognition models, we create a single-parameter version of their method: we choose six reference points along the border of the spectrogram, four at the corners and one in the center of the time axis on each side. The center points are then randomly moved along the time axis by a distance of w, with the spectrogram being warped accordingly. The augmentation hyperparameter w_{max} determines the bounds of the uniform distribution $X_{\text{w}} \sim \mathcal{U}(-w_{max}, w_{max})$, from which the random values for w are sampled.

Frequency and Time Masking: Two other augmentation methods used in the SpecAugment augmentation policy [25], where a part of the frequency spectrum and a part of the time axis are omitted. We use two hyperparameters, f_{max} and t_{max}, and choose the width of the range to be omitted randomly from the uniform distributions $X_{\text{f}} \sim \mathcal{U}(0, f_{max})$ and $X_{\text{t}} \sim \mathcal{U}(0, t_{max})$. We then choose two random starting points f_0 and t_0 and finally omit the frequency bands in the range $[f_0, f_0 + f]$ the time steps in $[t_0, t_0 + t]$ from the spectrogram.

[4] Pitch-shifting with Rubberband [23] and by shifting the spectrogram were also tested, but it was found that the models trained with these alternative methods showed significantly worse performance.

Loudness Augmentation: We implement loudness augmentation by generating a factor k, with which all of the magnitudes in the spectrogram are multiplied. This factor is sampled from a uniform distribution $X_{\mathrm{k}} \sim \mathcal{U}\left(k_{max}^{-1}, k_{max}\right)$, whose bounds are given via the hyperparameter k_{max}.

Additive Gaussian Noise: Adding noise with a mean of zero to the input samples is a very straightforward way of augmentation: we use a Gaussian distribution $X \sim \mathcal{N}(\mu = 0,\, \sigma^2)$ from which a separate value is sampled randomly and then added to each data point in the spectrogram, with the noise intensity being controlled via the hyperparameter σ.

Frequency Filtering: Random frequency filtering, as previously used in [26], is a method where a random filter is generated, which amplifies or dampens a range of frequencies. The range of affected frequencies is defined via a Gaussian function, resulting in the amplitude response

$$A(f_{\mathrm{st}}) = max\left(1 + \frac{s}{\sigma\sqrt{2\pi}} \times e^{\left(-\frac{(f_{\mathrm{st}} - f_{0,\mathrm{st}})^2}{2\sigma^2}\right)}, 0\right),$$

with which the values of the spectrogram are multiplied. We define it in semitone space for simplicity and clamp the values of the amplitude response at 0 to avoid negative amplitudes. Each filter is defined via three parameters σ, s, and f_0, which determine the width, amplification and location respectively. These parameters are randomly sampled from three separate uniform distributions $X_{\mathrm{s}} \sim \mathcal{U}\left(-s_{max}, s_{max}\right)$, $X_{\sigma} \sim \mathcal{U}\left(0, \sigma_{max}\right)$ and $X_{f_0} \sim \mathcal{U}\left(0, f_{max}\right)$. s_{max} and σ_{max} are given as hyperparameters, while f_{max} is defined as the maximum frequency represented in the spectrogram.

Generally, the mentioned augmentation methods are applied at runtime with random parameters, introducing random variations into the input data. The one exception to this is pitch-shifting, where $2\Delta f_{max}$ additional variations per sample are generated ahead-of-time. These also have different key root labels, which can help compensating imbalances in the datasets' root distribution.

We evaluate these methods by performing a Bayesian optimization on all of the augmentation hyperparameters, maximizing the median validation MIREX score over a 5-fold cross-validation for each evaluated set of hyperparameters. The resulting hyperparameters define our augmentation policy. Optimizing all of the augmentation method hyperparameters together as compared to doing so independently is important, as different hyperparameter values for one augmentation method might influence another. By evaluating all methods simultaneously, we can thus get more accurate results than with independently optimized hyperparameters. To get an insight into how each hyperparameter influences model performance, we look at the fitted Gaussian process from the Bayesian optimization and compute the conditional probability distribution for that hyperparameter given that the others are set to "good" values and did not fail to train. As our condition for a hyperparameter value being "good", we check whether it is

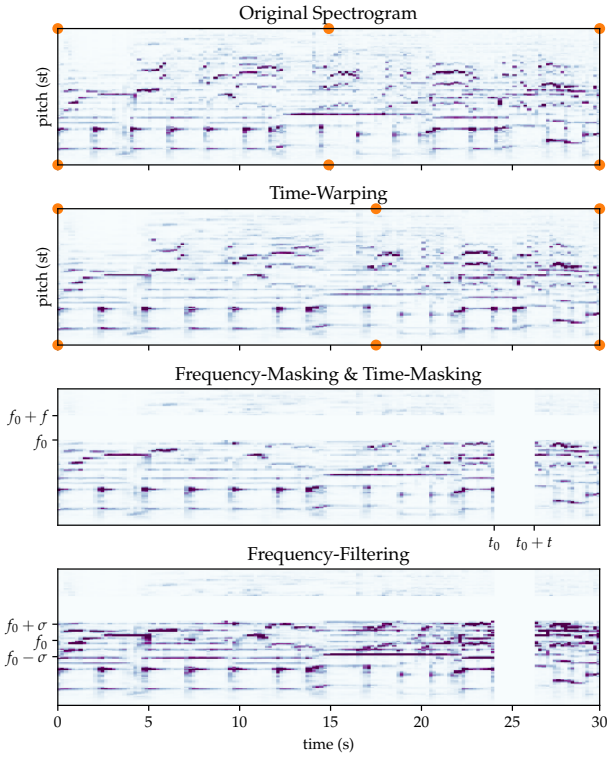

Figure 3: A visualization of some of the augmentation methods. The first row shows an entire unmodified spectrogram with the reference points for time-warping in their default position. In the second spectrogram, time-warping ($w = 2.6$s) has been applied, and the reference points have been moved accordingly. The third row shows the warped spectrogram after frequency-masking ($f = 15$st) and time-masking ($t = 2.2$s) have been applied at random positions, and the final spectrogram shows the result after applying frequency filtering ($\sigma = 10$st; $s = 90$).

at most 5% larger or smaller than the optimal value we obtained during our optimization. This way, we get evaluations of the performance of our hyperparameters that do not depend on exact values for the other hyperparameters.

A number of graphs visualizing the effect, which the hyperparameters corresponding to the different augmentation methods have on the validation performance, are shown in figure 4. From this, it is clear that loudness augmentation and additive Gaussian noise negatively affect the validation performance of our model, while pitch-shifting, time-warping, frequency-masking, and time-masking can all help improve performance on unseen samples.

For pitch-shifting, it seems that the chosen evaluation range from 0st to 12st was too small, and even more extreme values might prove beneficial to model performance. As we did not evaluate higher values in the full optimization process, we confine ourselves to the range as mentioned earlier and choose a value of 12st for the augmentation policy. For time-warping, frequency-masking, and time-masking, optimal values can be observed inside the evaluated range, although the performance improvement is significantly smaller than with pitch-shifting (especially so for frequency masking). For frequency filtering, there

seems to be a "sweet spot" near the middle of the evaluated range, where validation performance is increased slightly.

The final optimal values we obtained from the Bayesian optimization process are shown in table 1.

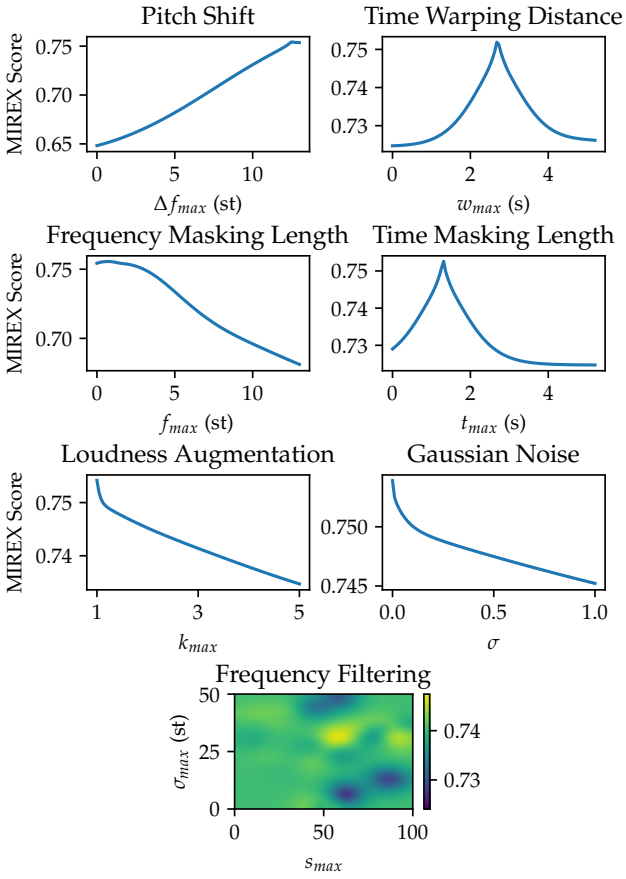

Figure 4: Estimated effect on the validation MIREX score for different augmentation methods. The blue line represents the mean validation performance for the line graphs. The frequency filtering parameters are presented as a 2-dimensional map, where the color corresponds to the mean validation performance.

Method	Parameter	Value
Pitch-Shifting	Δf_{max}	12st
Time-Warping	w_{max}	2.6s
Frequency-Masking	f_{max}	1st
Time-Masking	t_{max}	1.2s
Loudness-Augmentation	k_{max}	not applied
Additive Gaussian Noise	σ	not applied
Random Filtering	σ_{max}	36.25st
Random Filtering	s_{max}	30

Table 1: The optimal values for each augmentation hyperparameter as determined by a Bayesian optimization process, for which 178 hyperparameter combinations (resulting in 890 separate models with cross-validation) were evaluated.

4. EVALUATION

In this section, we evaluate the performance of the InceptionKeyNet model and compare it to two existing state-of-the-art single-task models. For these comparisons, we use the following datasets:

GiantSteps Key (GS): A dataset of 604 key annotations generated from user corrections from the Beatport service and a range of smaller datasets, with no overlap with the GiantSteps MTG Key dataset used for training [17].

KeyFinder (KF): A dataset of 1000 key annotations from multiple genres. [29]. We were able to obtain audio excerpts for 833 of the music pieces.

Isophonics (I): Four datasets containing songs by The Beatles, Queen, Zweieck, and Carole King. [30]. As we do not have access to the full recordings, we only use songs where a single key is annotated, resulting in 151 songs by The Beatles, 8 by Queen, 7 by Zweieck, and 2 by Carole King.

RockCorpus (RC): A dataset with annotations for 200 entries from the Rolling Stone "500 Greatest Hits of All Time" list [31]. As only the key root is given, we use a method proposed in [3] to obtain mode annotations: if at least 80% of the annotated tonic chords are of one mode, that one is selected for the overall key; otherwise, the sample is excluded. This results in 188 music pieces, of which we have audio excerpts for 186.

As we are only working with excerpts and do not have excerpts available for all entries of the various datasets, the test scores for the KeyFinder, Isophonics and RockCorpus datasets are not necessarily directly comparable to results obtained by other publications. To be able to give fair comparisons to other models on all datasets, we replicate or test them on our datasets when comparing different models. For the GiantSteps Key dataset, the scores are comparable with those in other publications, as we have the original audio files available.

We compare our model with two other single-task models: the AllConv [3] and the JXC1 [5] [5] models. To obtain fair comparisons, we replicate the models using their respective preprocessing methods as specified in the original publications [3, 5]. We then train each model on our two different training datasets, and using either only pitch-shifting or our augmentation policy. For more direct comparability, we used the same pitch-shifting range from -6st to $+6$st for all three models, which results in a range 1st wider than used in the original publications for the AllConv and JXC1 paper. We use the method of stopping the training described in subsection 2.3 for the AllConv model, as testing showed better performance than the default method. Finally, as we use cross-validation ensembles for our own model, we also use them for the AllConv and JXC1 models to make the comparison as fair as possible. Introducing ensembles leads to an average absolute increase in MIREX score of 3.1% and 3.6% for the AllConv and JXC1 models respectively. This is significantly

[5] We use JXC1 instead of the JXC2 model as the latter requires training in a multi-task setting, which would go beyond the scope of this paper.

Model Architecture	Augmentation	Test MIREX Scores [%] GS	KF	I	RC	Avg
Small Training Dataset (GiantSteps MTG Key and McGill Billboard)						
InceptionKeyNet (ours, ensemble, 1.7M parameters per model)	pitch-shift (−6st to +6st)	73.94	71.31	**79.23**	78.49	73.69
	our policy	**75.50**	**71.94**	78.93	78.28	**74.46**
AllConv [3] ($N_f = 20$, ensemble, 462k parameters per model)	pitch-shift (−6st to +6st)	74.27	67.78	75.83	77.58	71.74
	our policy	73.38	66.76	75.36	**80.70**	71.25
JXC1 [5] (ensemble, 12.5M parameters per model)	pitch-shift (−6st to +6st)	72.27	69.11	76.61	75.48	71.54
	our policy	73.66	68.03	72.80	78.49	71.46
Large Training Dataset (GiantSteps MTG Key, McGill Billboard, and our data mining dataset)						
InceptionKeyNet (ours, ensemble, 1.7M parameters per model)	pitch-shift (−6st to +6st)	74.35	**70.58**	80.24	83.92	74.14
	our policy	**75.68**	70.49	81.61	**84.62**	**74.75**
AllConv [3] ($N_f = 20$, ensemble, 462k parameters per model)	pitch-shift (−6st to +6st)	74.44	68.45	77.14	78.76	72.36
	our policy	73.06	66.15	74.88	80.70	70.81
JXC1 [5] (ensemble, 12.5M parameters per model)	pitch-shift (−6st to +6st)	74.44	69.27	**81.85**	81.34	73.45
	our policy	74.62	69.40	79.70	81.61	73.39
Reference Models (weights as in the original publication; trained on other datasets)						
AllConv [3, 27] (single model)	pitch-shift (−4st to +7st)	74.62	63.76	75.83	73.49	69.56
QM Key Detector v5 [28] (single model)	-	57.76 [6]	48.12	64.40	60.48	54.15

Table 2: The results of our evaluation. The models are separated into groups depending on the datasets they have been trained on. The GS, KF, I, and RC columns show the test MIREX score on each test dataset; the rightmost column shows the average of those scores, weighted by dataset size. The best test scores in each group of models is marked in bold font.

larger than the increase of 2.0% that the InceptionKeyNet model shows, which means that any performance lead of the InceptionKeyNet model would be larger when comparing single models. The results from this evaluation are shown in table 2. As an additional reference, we also include the performance of the original trained version [27] of the AllConv model, which shows comparable average performance to our reproduction when considering the previously mentioned gain from using ensembles. Furthermore, we also include the performance of the QM Key Detector v5 [28] model on our versions of the test datasets, even though this model shows significantly worse performance across all of our test datasets.

It can be seen that when trained on the same datasets, the InceptionKeyNet model is able to outperform the AllConv and JXC1 models, with the difference between it and the next best model being 2.72% for the small and 1.30% for the large training dataset. While all models show improved performance when trained on a larger dataset, only InceptionKeyNet sees increased performance when our augmentation policy is applied. This suggests that, while broader augmentation can improve performance in at least some cases, the parameters possibly have to be tailored to each model. It can also be seen that the performance advantage given by the broad augmentation policy decreases slightly when the amount of training data increases. This likely means that the broad augmentation policy is particularly useful when little data is available.

There also seems to be no clear correlation between model parameter count and performance, which suggests that the architecture choice probably plays an important role in deciding a models performance.

5. CONCLUSION

This paper presented an adaptation of the Inception V3 [7] architecture for harmonic music analysis tasks and used it to create a model for the key estimation task. We proceeded to evaluate a broad range of augmentation methods to find a tailored augmentation policy, and finally evaluated our model and compared it to two state-of-the-art single-task models, training all of them on the same datasets to obtain fair comparisons. These comparisons showed that the InceptionKeyNet model is capable of significantly outperforming state-of-the-art single-task models when trained on the same data with the same augmentation methods, and that further improvements are possible when applying a broad augmentation policy.

Ultimately, we believe that the main contribution of this work is showing that there is still potential to outperform state-of-the-art models in common harmonic music analysis tasks when deeper neural network architectures and matching extensive augmentation schemes are used. We hope for the presented findings to inspire further research applying similar approaches to other related MIR tasks, potentially even combining them into multi-task models. These have shown promising performance in the past, outperforming their single-task variant in at least one case [5].

The source code to train and run our model, and the subsets of public datasets where we had audio excerpts available for training, are available online [7]

[6] We assume that the very slight deviation compared to the value in [32] is due to different versions of dependencies of sonic-annotator.

[7] https://github.com/stefan-baumann/inceptionkeynet.

6. REFERENCES

[1] I. Sha'ath, "Estimation of Key in Digital Music Recordings," Master's thesis, Birkbeck College, University of London, 2011.

[2] B. McFee, C. Raffel, D. Liang, D. 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, pp. 18–25.

[3] F. Korzeniowski and G. Widmer, "Genre-Agnostic Key Classification With Convolutional Neural Networks," in *Proceedings of the 19th International Society for Music Information Retrieval Conference*. ISMIR, Sep. 2018, pp. 264–270.

[4] ——, "End-to-end musical key estimation using a convolutional neural network," in *2017 25th European Signal Processing Conference (EUSIPCO)*. IEEE, 2017, pp. 966–970.

[5] J. Jiang, G. G. Xia, and D. B. Carlton, "MIREX 2019 Submission: Crowd Annotation for Audio Key Estimation," *Music Information Retrieval Evaluation eXchange*, 2019.

[6] H. Schreiber and M. Müller, "Musical Tempo and Key Estimation using Convolutional Neural Networks with Directional Filters," in *Proceedings of the 16th Sound & Music Computing Conference*, 2019, pp. 47–54.

[7] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the Inception Architecture for Computer Vision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2818–2826.

[8] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[9] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," in *3rd International Conference on Learning Representations, ICLR 2015, Conference Track Proceedings*, 2015.

[10] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, vol. 25. Curran Associates, Inc., 2012.

[11] S. Hershey, S. Chaudhuri, D. P. Ellis, J. F. Gemmeke, A. Jansen, R. C. Moore, M. Plakal, D. Platt, R. A. Saurous, B. Seybold *et al.*, "CNN architectures for large-scale audio classification," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 131–135.

[12] "Going deeper with convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2015, pp. 1–9.

[13] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2009, pp. 248–255.

[14] J. Tompson, R. Goroshin, A. Jain, Y. Lecun, and C. Bregler, "Efficient Object Localization Using Convolutional Networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, jun 2015, pp. 648–656.

[15] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, 2015.

[16] Á. Faraldo, "giantsteps-mtg-key-dataset," GitHub, last accessed on 2021-01-23. [Online]. Available: https://github.com/GiantSteps/giantsteps-mtg-key-dataset

[17] P. Knees, Á. Faraldo Pérez, H. Boyer, R. Vogl, S. Böck, F. Hörschläger, M. Le Goff *et al.*, "Two data sets for tempo estimation and key detection in electronic dance music annotated from user corrections," in *Proceedings of the 16th International Society for Music Information Retrieval Conference (ISMIR)*, 2015, pp. 364–70.

[18] K. Shaffer, E. Vasiete, B. Jacquez, A. Davis, D. Escalante, C. Hicks, J. McCann, C. Noufi, and P. Salminen, "A cluster analysis of harmony in the McGill Billboard dataset," *Empirical Musicology Review*, vol. 14, no. 3-4, p. 146, 2020.

[19] "The McGill Billboard Project," last accessed on 2020-10-28. [Online]. Available: https://ddmal.music.mcgill.ca/research/The_McGill_Billboard_Project_(Chord_Analysis_Dataset)/

[20] F. Korzeniowski, "bb.zip," last accessed on 2021-01-23. [Online]. Available: http://www.cp.jku.at/people/korzeniowski/bb.zip

[21] "2020 Audio Key Detection," MIREX Wiki, 2020, last accessed on 2020-10-26. [Online]. Available: https://www.music-ir.org/mirex/wiki/2020:Audio_Key_Detection

[22] C. Raffel, B. McFee, E. J. Humphrey, J. Salamon, O. Nieto, D. Liang, and D. P. W. Ellis, "MIR_EVAL: A Transparent Implementation of Common MIR Metrics." in *Proceedings of the 15th International Society for Music Information Retrieval Conference*. ISMIR, 2014, pp. 367–372.

[23] C. Cannam, "Rubberband," sourcehut, last accessed on 2021-01-31. [Online]. Available: https://hg.sr.ht/~breakfastquay/rubberband/

[24] C. Bagwell, "SoX - Sound eXchange," SourceForge, last accessed on 2021-01-31. [Online]. Available: http://sox.sourceforge.net/

[25] D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le, "SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition," in *Proc. Interspeech 2019*, 2019, pp. 2613–2617.

[26] J. Schlüter and T. Grill, "Exploring Data Augmentation for Improved Singing Voice Detection with Neural Networks," in *Proceedings of the 16th International Society for Music Information Retrieval Conference*. ISMIR, 2015, pp. 121–126.

[27] "madmom_models/key/2018," GitHub, last accessed on 2021-02-17. [Online]. Available: https://github.com/CPJKU/madmom_models/tree/master/key/2018

[28] C. Cannam, E. Benetos, M. E. P. Davies, S. Dixon, C. Landone, M. Levy, M. Mauch, K. Noland, and D. Stowell, "MIREX 2019: Vamp plugins from the Centre for Digital Music," *Music Information Retrieval Evaluation eXchange*, 2019.

[29] I. Sha'ath, "KeyFinder v2 Dataset," 2014, last accessed on 2021-01-15. [Online]. Available: http://ibrahimshaath.co.uk/keyfinder/KeyFinderV2Dataset.pdf

[30] "Isophonics Reference Annotations," last accessed on 2021-02-26. [Online]. Available: http://isophonics.net/content/reference-annotations

[31] T. De Clercq and D. Temperley, "A corpus analysis of rock harmony," *Popular Music*, vol. 30, no. 1, pp. 47–70, jan 2011.

[32] "2019 Audio Key Detection Results," MIREX Wiki, 2019, last accessed on 2020-10-26. [Online]. Available: https://www.music-ir.org/mirex/wiki/2019:Audio_Key_Detection_Results