ULTRA-LIGHT DEEP MIR BY TRIMMING LOTTERY TICKETS

Philippe Esling, Theis Bazin, Adrien Bitton, Tristan Carsault, Ninon Devis
IRCAM - Sorbonne Université, CNRS UMR 9912 - 1, place Igor Stravinsky, Paris, France
esling@ircam.fr

ABSTRACT

Current state-of-the-art results in Music Information Retrieval are largely dominated by deep learning approaches. These provide unprecedented accuracy across all tasks. However, the consistently overlooked downside of these models is their stunningly massive complexity, which seems concomitantly crucial to their success.

In this paper, we address this issue by proposing a model pruning method based on the *lottery ticket* hypothesis. We modify the original approach to allow for explicitly removing parameters, through *structured trimming* of entire units, instead of simply masking individual weights. This leads to models which are effectively lighter in terms of size, memory and number of operations.

We show that our proposal can remove up to 90% of the model parameters without loss of accuracy, leading to ultra-light deep MIR models. We confirm the surprising result that, at smaller compression ratios (removing up to 85% of a network), lighter models consistently outperform their heavier counterparts. We exhibit these results on a large array of MIR tasks including *audio classification*, *pitch recognition*, *chord extraction*, *drum transcription* and *onset estimation*. The resulting ultra-light deep learning models for MIR can run on CPU, and can even fit on embedded devices with minimal degradation of accuracy. [1]

1. INTRODUCTION

Over the past decades, Music Information Retrieval (MIR) has witnessed a growing interest, with a wide variety of tasks such as *genre classification*, *chord extraction* and *music recommendation* [1] being increasingly implemented in end-user products. Recently, MIR has predominantly improved with machine learning, and almost all state-of-art accuracies are obtained by deep learning models [2]. Although these approaches provide unprecedented results, the major issue in modern deep learning lies in the tremendous complexity and immense size of the models employed. Indeed, deep networks for images can reach up to billions of parameters and new leaps in accuracy seem to only come by worsening this situation. As a showering example of this complexity [3], the inference on a single image in the pervasive ResNet model [4] requires 7.7 GFLOPS [2] . This exploding size leads to profound issues in both the use and understanding of these models. As they are extremely demanding in computation and memory, it precludes their implementation in end-user embedded systems which prevails in audio applications, and also raises some serious environmental issues. Finally, such complexity decreases the potential interpretability of these models.

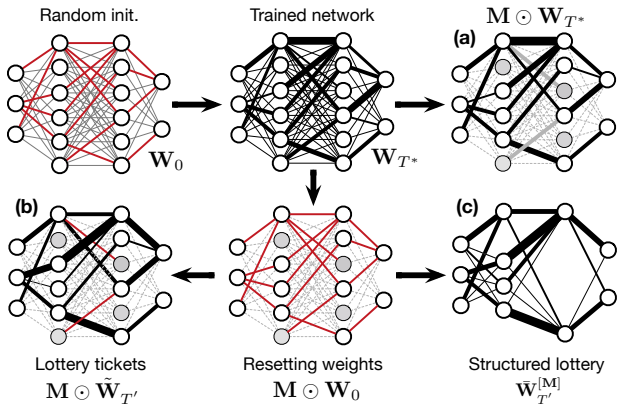


Figure 1. Comparing (a) traditional pruning with (b) the lottery ticket hypothesis, and (c) our structured lottery approach to obtain ultra-light deep networks.

The idea of eliminating unnecessary weights (*pruning*) was proposed early for neural networks [5]. Most methods are based on masking the smallest-amplitude weights from a large network, as depicted in Figure 1. Other approaches such as *quantization* [6] or *knowledge distillation* [7] have been proposed to decrease the size and energy consumption of trained models with equivalent accuracy. However, keeping the original accuracy of complex large models seems only possible at low compression rates [8]. Furthermore, recent benchmark studies [9] pointed out that most of the proposed methods seems to achieve a similar efficiency in both accuracy and model size.

Recently, the *lottery ticket hypothesis* [10] suggested that randomly-initialized neural networks already contain powerful subnetworks (called *winning tickets*) that could reach the same or higher accuracy than the original networks if they were trained in isolation. Hence, by finding these subnetworks, we could drastically prune most of the

[1] Supplementary results and code to reproduce experiments are available at https://github.com/acids-ircam/lottery_mir

[2] FLOPS: floating point operations

weights in large networks and still obtain the same level of accuracy. This implies that the same task could be solved in a very lightweight, memory and energy-efficient way. Furthermore, these subnetworks could be easier to analyze, which could simplify further works towards explainability [11]. Several studies have analyzed different properties of this hypothesis [12–14], as it raises the exciting prospect to obtain much smaller networks that provide a similar accuracy compared to the typically larger state-of-art models. However, this method has two major flaws. First, it has a large training cost, as finding winning tickets seems to only be stable when the training is repeated multiple times over iteratively smaller networks [15]. Second, pruning is done by *masking* the weights, which means that the resulting networks retain the size and computation cost of the original ones, even if most of the weights are unused.

In this paper, we extend the lottery approach to effectively remove weights, obtaining models with a lower size and inference time, while still maintaining a commensurate accuracy. To do so, we introduce a method based on the lottery ticket hypothesis, and we replace the masking operation with a structured pruning operation (termed *trimming* here). The original network capacity is reduced by removing entire computation units (or convolutional channels). This alleviates issues of the original lottery ticket method as, although we still need to repeat the training, it becomes faster at each iteration. We discuss different criteria for selecting the units and their differences to the original lottery ticket hypothesis. Notably, unstructured masking allows to work on local connectivity patterns, whereas trimming can only impact this aspect if we perform *global* selection (ranking units across the network). We show that this approach can be successfully applied across MIR tasks, leading to *ultra-light* deep MIR models. We evaluate the efficiency of replacing the masking operation by our trimming criterion and show that we still obtain commensurate accuracy when removing up to 90% of the model parameters. We also maintain the surprising result [10] that lighter models (removing up to 85% of the network) obtain higher accuracy, while we effectively reduce the model size. We evaluate these results on a large array of MIR tasks including *instrument* [16] and *singing voice classification* [17], *pitch recognition* [18], *automatic chord extraction* [19], *drum transcription* [20] and *onset estimation* [21].

2. STATE-OF-ART

2.1 Model compression and pruning

Various approaches have been proposed for reducing the size of neural network models, while trying to maintain accuracy [5]. These approaches can be globally divided between *pruning* or *compressing* networks. We group in the *compression* category the *distillation* [7] (training a smaller model to fit the internal representations of a larger one) and *quantization* [6] (reducing the size of networks by using lower-resolution weights or binary numbers) approaches. Here, we focus on *pruning*, but note that compression and quantization can be further applied on pruned models.

The goal of *pruning* [5] is to identify and remove weights of a network that are not critical to its accuracy. The original approach to pruning starts by fitting a large and overparametrized network to completion. Then, we aim to mask the less relevant weights in this trained model based on a given selection method. This criterion tries to analyze the usefulness of different parameters, commonly based on their magnitude [22]. Finally, the resulting masked network is *fine-tuned*, trying to restore the accuracy of the original network [9]. Hence, the critical aspect in this approach lies in the method of weight selection. This criterion can perform either a *structured* or *unstructured* and *local* or *global* selection. *Unstructured* pruning acts on individual parameters separately, *structured* pruning aims to effectively remove parts of the networks. Hence, unstructured methods are mostly based on *masking* the weights based on their magnitude [5, 22]. Oppositely, structured pruning aims to remove entire *hidden units* or *convolutional channels* from a network [8, 23].

However, recent studies showed that most pruning methods are mostly equivalent [8]. These approaches usually lead to smaller accuracy than the large network and at low pruning rates, with performance degrading with the amount of weights removed [9], although some are able to maintain (but not outperform) the original accuracy [8].

2.2 Lottery ticket hypothesis

The recently proposed *lottery ticket hypothesis* [10] states that inside a randomly-initialized network, there already exist some considerably smaller subnetworks which would be extremely efficient if trained in isolation. Hence, parts of the weights drawn by random initialization before training already provide a specific topology and parameter configuration that make training particularly effective. The major difference between this approach and the previous magnitude-based selection from which it is inspired [22] is that the selected weights are *reset* to their initialization value before retraining the smaller architecture. Doing so, very small subnetworks (less than 1% of the original network size) could be found across several architectures, even outperforming the larger networks at smaller pruning ratios. For deeper architectures, they further showed [12] that winning tickets should be *rewound* to a given iteration, rather than to initialization values. Interestingly, this seems to confirm that overparameterization is needed to find an optimal solution, but that a lighter solution exists, which is optimized in the compression phase of the training [13,24].

2.2.1 Formalization

We consider a network as a parametric function $f(\boldsymbol{x}; \boldsymbol{W})$, with a set of weights $\boldsymbol{W} \in \mathbb{R}^D$ that are first initialized through sampling $\boldsymbol{W}_0 \sim p(\boldsymbol{W})$. The weights are updated by using a training *algorithm* $\mathcal{A}(i, \boldsymbol{W}_0)$ which maps initial weights $\boldsymbol{W}_0$ to weights $\boldsymbol{W}_i$ at iteration $i \in \{1, .., T\}$, by performing successive updates similar to

$$\boldsymbol{W}_{i+1} = \boldsymbol{W}_i - \eta \nabla_{\boldsymbol{W}} \mathcal{L} \tag{1}$$

with a given loss function $\mathcal{L}$ and learning rate η.

A *subnetwork* of the original network $f(\boldsymbol{x}; \boldsymbol{W})$ can be defined as a tuple $(\boldsymbol{W}, \boldsymbol{M})$ of the original weights $\boldsymbol{W} \in \mathbb{R}^D$ and a mask $\mathbf{M} : \{0,1\}^D$. Hence, the subnetwork computes the function $f(\boldsymbol{x}; \boldsymbol{M} \odot \boldsymbol{W})$, where $\odot$ denotes the element-wise product.

Lottery Ticket Hypothesis. Given a randomly initialized network $f(\boldsymbol{x}; \boldsymbol{W}_0)$ with $\boldsymbol{W}_0 \sim p(\boldsymbol{W})$, that is trained to reach accuracy a^* in T^* iterations, with final weights $\boldsymbol{W}_{T^*}$, there exists a subnetwork $(\boldsymbol{W}_k, \boldsymbol{M})$ with a given mask $\boldsymbol{M} \in \{0,1\}^{|\boldsymbol{W}|}$ and iteration $k \ll T^*$, such that if we retrain this subnetwork, it will reach a *commensurate accuracy* $a \geq a^*$ in *commensurate iterations* $T \leq T^* - k$ and *fewer parameters* $\|\boldsymbol{M}\|_0 \ll |\boldsymbol{W}|$.

These highly efficient subnetworks (called *winning tickets*) depend on the original initialization, and can only be identified after full training [12]. Thus, the selected weights and remaining topology form the architecture of the winning ticket. These weights are *reset* to their initialization values *before* the network was trained or *rewound* at an early iteration. The resulting architecture is then retrained until completion, and the whole process is repeated, as described in Algorithm 1.

Algorithm 1 Lottery ticket training with rewinding

1: $\boldsymbol{W}_0 \sim p(\boldsymbol{W})$ ▷ Random initialization
2: $\boldsymbol{M} = \mathbf{1}_{|\boldsymbol{W}|}$ ▷ Initial mask
3: $\boldsymbol{W}_k = \mathcal{A}(k, \boldsymbol{W}_0 \odot \boldsymbol{M})$ ▷ Training for k iterations
4: **while** $\mathcal{C}(\boldsymbol{M}, a, \boldsymbol{W})$ **do** ▷ Stopping criterion $\mathcal{C}$
5: $\quad \boldsymbol{W}_T = \mathcal{A}(T, \boldsymbol{W}_k \odot \boldsymbol{M})$ ▷ Train until completion
6: $\quad r = \mathcal{R}(\{\boldsymbol{W}_{T^*}\})$ ▷ Ranking criterion $\mathcal{R}$
7: $\quad \boldsymbol{M} = \mathcal{M}(r, \{\boldsymbol{W}_{T^*}\})$ ▷ Masking update $\mathcal{M}$

In their original paper [10], the authors underline the difference between *one-shot* pruning (masking is applied all at once) and *iterative pruning* (repeatedly pruning small parts of the network). They demonstrated that iterative pruning finds smaller architectures that reach higher accuracy than the original network and converge at earlier iterations. They showed on the MNIST dataset, that it was possible to keep the accuracy of large networks, even when masking up to 96.5% of the weights. Their most intriguing result is that smaller networks consistently reach *higher accuracy* than the original ones, even while removing up to 80% of the weights. In a follow-up study [12], they showed that these results could be obtained for deeper architectures, but only through the *rewinding* operation. Another exciting prospect of this hypothesis, is that the resulting subnetworks might encode implicit *inductive biases* for a given task or type of data. In that case, winning tickets could be *transferred* and trained on new tasks, even directly from their extremely lightweight versions [14].

2.3 Music Information Retrieval

Music Information Retrieval (MIR) encompasses all tasks aimed at extracting high-level knowledge from music data. This field has witnessed a flourishing interest, with multiple tasks being increasingly tackled such as *chord extraction*, *drum transcription* and *musical audio classification* [1]. Originally, most MIR researches revolved around the idea of extracting a set of hand-crafted features from the signal (such as the Mel-Frequency Cepstral Coefficients), in order to use these as input to machine learning algorithms [25]. Feature-based techniques have been challenged by the advent of deep learning approaches [26], which have shown impressive capacities to learn high-level features on complex data. They simultaneously set new state-of-art results across a wide range of MIR tasks, while opening the path towards unprecedented applications [27].

In this work, we consider a rather broad spectrum of MIR tasks where deep learning approaches are applied. Specifically, we address (i) *audio classification* [17] (finding the class label of audio signals inside a predefined set), (ii) *pitch recognition* [18] (extracting the fundamental frequency of a monophonic audio recording), (iii) *chord extraction* [19] (annotating audio with a given vocabulary of chords), (iv) *onset estimation* [21] (finding events in an audio stream) and (v) *drum transcription* [20] (transforming drums audio signal into a score). We redirect interested readers to [28] for a comprehensive review.

One of the common denominator in deep learning methods applied across all MIR tasks is that their unprecedented accuracy comes at the expense of an increasing size and complexity. Indeed, deep networks for images now reach billions of parameters and leaps in accuracy seem to only come by worsening this situation. An example of this trend in MIR can be seen in the recently proposed CREPE model [18] for pitch extraction. This task was largely handled through the YIN algorithm [29], an extremely simple algorithm, with few parameters and running with very low latency on CPU. For a modest gain in accuracy on the same task, CREPE requires *22 million parameters*, 2.82 GFLOPS and 2.36 seconds on CPU to compute the pitch of a single 4-seconds sample. This exploding size leads to profound issues in both the use and understanding of these models. First, they are extremely demanding in energy consumption and memory, which precludes their implementation in end-user interfaces and also raises serious environmental issues. Furthermore, this complexity stands in the way of any potential interpretability of such models.

3. METHODOLOGY

Here, we first discuss different selection criteria for structured network *trimming*. Then, we discuss different normalization strategies that can allow to perform global selection of units across layers.

3.1 Trimming criteria

In order to perform structured pruning, we need to evaluate the efficiency of *entire units* of computation, rather than individual weights. In the case of convolutional networks, this would amount to analyze the *channels* of each layer. Indeed, channel pruning appears more hardware friendly, and also allows to truly reduce the size of the final model. In the following definitions, we consider that any computation layer can be seen as a weighted transform $f(\boldsymbol{x}, \boldsymbol{W}^{(l)})$,

with a matrix $\boldsymbol{W}^{(l)} \in \mathbb{R}^{n_{out} \times n_{in}}$. Note that we intentionally simplify the notation for more complex layers (*convolutional* or *recurrent*), which embed more complicated matrices. However, we consider in the following that the selection criteria $\mathcal{C}(\boldsymbol{W}^{(l)})$ is computed across the n_{in} dimensions, and that it should produce a vector of n_{out} dimensions. This vector is used to rank the usefulness of different computation units. Hence, after each training iteration, we replace the masking criterion by directly removing parts of the weight matrix for each layer

$$\boldsymbol{W}^{(l)} = \boldsymbol{W}^{(l)}_{[C(\boldsymbol{W}^{(l)}),C(\boldsymbol{W}^{(l-1)})]}. \tag{2}$$

Note that we need to carry the pruning criterion from the preceding layer $C(\boldsymbol{W}^{(l-1)})$ in order to reflect potential changes in the structure of the network. All layers in the network that must maintain a given output dimensionality (such as the last layer) are defined as *unprunable*. Following a similar approach than the lottery ticket hypothesis, the remaining weights in the resulting matrix $\boldsymbol{W}^l$ are *rewound* to their values from an earlier iteration [13].

Magnitude. We define a *magnitude-based* criterion, similar to the original lottery [10]. However, we evaluate the overall magnitude of the weights for a complete unit as

$$\mathcal{C}_{mag}(\boldsymbol{W}^{(l)}_i) = \sum_{j=1}^{N_{in}} \left| W^{(l)}_{i,j} \right|. \tag{3}$$

Activation. We can rely on the activation statistics of each unit to analyze their importance. Hence, akin to the previous criterion, we perform a cumulative forward pass through the network after training the model and compute

$$\mathcal{C}_{act}(\boldsymbol{W}^{(l)}_i) = \sum_{k=1}^{\mathcal{D}_v} \left| f(\boldsymbol{x}_k, \boldsymbol{W}^{(l)})_i \right| \tag{4}$$

where we sum across examples of the validation set $\mathcal{D}_v$.

Normalization. An interesting direction proposed in [15] is to consider the *scaling* factor γ in batch normalization layers to evaluate the significance of each layer output. In this criteria, we rely on this *scaling* coefficient as a proxy to the importance of each unit

$$\mathcal{C}_{norm}(\boldsymbol{W}^{(l)}_i) = \left| \gamma^{(l)}_i \right|. \tag{5}$$

Note that this criterion forces each layer to be followed by a normalization layer, from which it can be computed.

4. EXPERIMENTS

We briefly detail the tasks on which we evaluate our method for ultra-light deep MIR. As we address a wide variety of models and datasets, we only provide essential explanations for each. However, unless stated, we follow all implementation details presented in the original papers.

4.1 Tasks

4.1.1 Audio (instrument and voice) classification

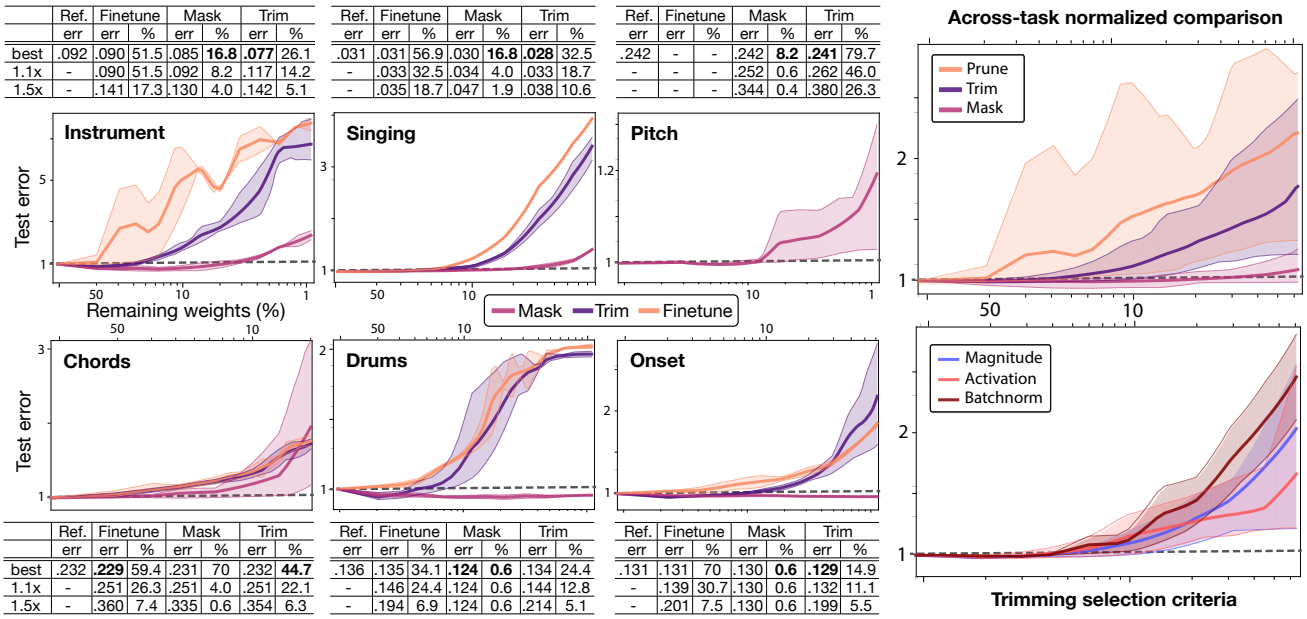
Audio classification is one of the seminal and most studied task in MIR [30]. We separate the evaluation into two independent sub-tasks of *singing voice* and *instrument* classification. For both tasks, the model is adapted from the baseline proposed in [17]. The raw input waveform is processed with a stack of 4 dilated 1-dimensional convolutions with batch normalization, ReLU and dropout, followed by 4 fully-connected layers that map to a *softmax*, which outputs a vector of class probabilities. The ground-truth label prediction is optimized with a *cross-entropy* loss. Singing voice classification is performed on mono audio inputs of 3 seconds at 44,100Hz for 10 vocal techniques and a given train/test split ratio [17]. For instrument classification, we rely on the 13 orchestral instruments from URMP [31] and the corresponding recordings from MedleyDB [32]. After silence removal, the combined datasets amount to a total of about 8h30 of isolated instrument recordings. Classification is done on audio inputs of 1.5 seconds at 22,050Hz extracted from the isolated tracks with a single label corresponding to the instrument played.

4.1.2 Pitch estimation

The goal of *pitch estimation* is to extract the fundamental frequency of an input audio. For this task, the recently proposed CREPE model [18] requires several large datasets, some of which are not publicly available. However, we only rely here on the open source *NSynth* dataset, which contains single note samples from a range of acoustic and electronic instruments [33]. This leads to 1006 instruments, with different pitches at various velocities available as raw waveforms. All samples last 4 seconds with a sampling-rate of 16kHz. As this incurs an extremely large training time, we use a subsampled dataset, randomly picking 10060 samples (ten notes per instrument). Finally, we trim all samples to their first two seconds to remove silent note tails, ensuring that most inputs to the model are voiced. CREPE is a 6-layer CNN operating directly on waveforms, followed by a single fully connected layer. The model is trained via binary cross-entropy to perform classification over a 360 bin logarithmic frequency scale spanning six octaves from pitch $C1$ to $B7$. The model operates on frames of 1024 samples, which we individually label with the note pitch. We use the *medium* architecture from the CREPE repository [18].

4.1.3 Chord extraction

Automatic chord extraction is defined as labeling segments of an audio signal using an alphabet of musical chords. We perform our experiments based on the model and datasets detailed in [19]. We use the *Beatles* dataset, which contains 180 songs annotated by hand. We rely on a CQT input with hop size 2048, mapped to a scale of 2 bins per semi-tone over 5 octaves starting from $C1$ and containing a total of 105 bins. As input we take 15 successive frames, corresponding to a temporal horizon of approximately 0.7 seconds. We augment the available data by performing all transpositions from -6 to +6 semi-tones. As baseline model, we rely on the CNN architecture described in [19], and evaluate the global accuracy measure.



	Ref.	Finetune		Mask		Trim	
	err	err	%	err	%	err	%
best	.092	.090	51.5	.085	**16.8**	**.077**	26.1
1.1x	-	.090	51.5	.092	8.2	.117	14.2
1.5x	-	.141	17.3	.130	4.0	.142	5.1

	Ref.	Finetune		Mask		Trim	
	err	err	%	err	%	err	%
best	.031	.031	56.9	.030	**16.8**	**.028**	32.5
1.1x	-	.033	32.5	.034	4.0	.033	18.7
1.5x	-	.035	18.7	.047	1.9	.038	10.6

	Ref.	Finetune		Mask		Trim	
	err	err	%	err	%	err	%
best	.242	-	-	.242	**8.2**	**.241**	79.7
1.1x	-	-	-	.252	0.6	.262	46.0
1.5x	-	-	-	.344	0.4	.380	26.3

	Ref.	Finetune		Mask		Trim	
	err	err	%	err	%	err	%
best	.232	**.229**	59.4	.231	70	.232	**44.7**
1.1x	-	.251	26.3	.251	4.0	.251	22.1
1.5x	-	.360	7.4	.335	0.6	.354	6.3

	Ref.	Finetune		Mask		Trim	
	err	err	%	err	%	err	%
best	.136	.135	34.1	**.124**	**0.6**	.134	24.4
1.1x	-	.146	24.4	.124	0.6	.144	12.8
1.5x	-	.194	6.9	.124	0.6	.214	5.1

	Ref.	Finetune		Mask		Trim	
	err	err	%	err	%	err	%
best	.131	.131	70	.130	**0.6**	**.129**	14.9
1.1x	-	.139	30.7	.130	0.6	.132	11.1
1.5x	-	.201	7.5	.130	0.6	.199	5.5

Figure 2. Comparing traditional *fine-tuning*, the lottery ticket *masking*, and our structured lottery *trimming* on each MIR task separately for all criteria (left), and across tasks (right, up) and across criteria in trimming (right, down).

4.1.4 Drum transcription

The *drum transcription* task aims at labelling an input audio with onsets of different drum sounds. We rely on an architecture inspired from [21]. The network takes mel-spectrogram inputs processed by 4 layers of 256 padded 2D convolutions of kernel size 5, with unit stride, followed by batch normalization and ReLU. We apply max-pooling at each layer along the frequency dimension. The resulting vector is processed by two linear layers with batch normalization and dropout. Finally, a specific output network of 3 linear layers for each drum sound produces onset probabilities, trained on binary vectors for each drum activation.

To train the network, we use the approach proposed by [34], using a subset of 5000 MIDI drum tracks that we map to random drum sounds to generate waveform recordings. We further rely on the SMT-Drums dataset [35], which provides 104 supplementary polyphonic drum set recordings. For both datasets, we compute a mel-spectrogram of 64 bins, ranging from 20 to 11025 Hz, based on a FFT of window size 2048 and hop size 512.

4.1.5 Onset estimation

Onset estimation [21] aims to detect events in a given audio input. In order to evaluate this task, we rely on the same network presented in the previous section for drum transcription. However, for this task, the last part of the network maps to a single detection subnetwork. We rely on the same drums dataset, but merge all labels to detect event onsets, rather than specific elements of the drumkit.

4.2 Training

All models are trained following their respective procedure and hyperparameters. However, we use a common mini-batch size of 64, the ADAM optimizer with a weight decay penalty of $2e^{-4}$, and an initial learning rate of $1e^{-3}$, which is halved every 10 non-decreasing epochs. We train the models for a number of epochs that is fixed for each task (following the original papers) and keep the model with the best validation score. For the lottery training, we perform *masking* or *trimming* of 30% of the weights at each pruning iteration. We rewind the weights to their values at half of the training epochs. We repeat this process 15 times, leading to models with up to 99.5% of weights removed. This whole lottery training is repeated 5 times, providing the variance and impact of the initialization on the results.

5. RESULTS

5.1 Global evaluation

First, we provide a global evaluation across different tasks, by plotting the respective evolution of the test error as we iteratively remove weights either by classical *fine-tuning*, using the original lottery with *masking*, or our proposed *trimming*. We report for each task the *best* model (lowest test error), *smallest* model (test error at most 1.5 times the original one), and *optimal* model (error at most 1.1 times the original). Results are displayed in Figure 2.

As we can see, classical *fine-tuning* is mostly unable to find more efficient lighter networks and only works at very low pruning rates. Oppositely, our *trimming* approach is able to consistently find networks that are both much smaller and more accurate than reference models. In this regard, the best performances are obtained for *onset detection*, where we find a network with only **14.9**% of the original weights (removing **85.1**% of the weights), while having an error rate of **0.129** (compared to 0.131 for the original). These results hold for almost all tasks: most networks where we trim up to **75**% of the weights pro-

duce lower errors, and we can remove up to **85**% of the weights with minor damage to the test error. Interestingly, the results of *chord extraction* seems to produce the smallest enhancement. This could be explained by the fact that the model has the lowest original number of parameters. Hence, this underlines the crucial need to rely on *largely overparametrized* models to find efficient subnetworks. Regarding the *smallest* models, we are able to remove on average up to 95%, while having a reasonable test error. When comparing our approach to the original lottery *masking*, it seems that masking consistently produces better performances at higher pruning rates, confirming the original results [10] for MIR tasks. However, note that the weights in the masking approach are not removed (however, a fraction of these weights could be removed in a post-processing step). We hypothesize that this resilience to larger pruning ratios stems from the fact that *masking* is able to work on local connectivity patterns, whereas our approach cannot.

5.2 Across-task comparison

5.2.1 Pruning approaches

The lottery ticket hypothesis crucially depends on initialization values for training efficient subnetworks. To evaluate this property across different tasks, we perform the normalized comparison shown in Figure 2 (right, up).

Here, we normalize the error of each task by dividing it by the error of the reference large model, so that its test error is 1. As we can see, using fine-tuning, the approach is unable to obtain subnetworks with higher accuracy, and the error quickly degrades as we remove more weights. Furthermore, it appears that the results are rather unstable, producing large variations in the final test error. Instead, by *rewinding* the weights and *trimming* we consistently obtain smaller subnetworks (up to 75% of the weights removed) that outperform the original models. We are able to apply extensive trimming before the error starts to degrade, globally around 90% across tasks. Hence, it appears that efficient subnetworks can be found solely through the correct combination of connection topology and weights.

5.2.2 Selection criteria

The success of pruning methods depends on the criterion selecting *which* weights should be kept or pruned. Hence, we perform a normalized comparison of different criteria for *trimming*, and display results in Figure 2 (right, down).

Although the global trend seems to be equivalent for most criteria at low pruning ratios, their differences amplify as we remove an increasing amount of weights. Overall, it seems that the *activation* criterion provides the most stable results. Furthermore, it allows to maintain lower error rates, even at higher pruning ratios. However, at lower pruning ratios, it seems that the *magnitude* criterion produces slightly better and more stable results. Finally, the *batchnorm* criterion seems to provide an interesting alternative at low pruning ratios. However, its performance degrades faster than other criteria at very high pruning rates.

task	mod.	error	param	size	FLOPS	mem
inst.	ref	0.092	797 K	10 M	572 M	190 M
	trim	0.117	93.4 K	2.3 M	38.3 M	41.9 M
sing.	ref	0.031	1.4 M	19 M	663 M	194 M
	trim	0.038	144 K	2.7 M	94.4 M	53.2 M
pitch	ref	0.242	5.9 M	49 M	2.8 G	256 M
	trim	0.262	224 K	1.0 M	2.8 M	9.6 M
chord	ref	0.232	416 K	1.4 M	27.2 M	22.1 M
	trim	0.251	91.9 K	0.2 M	1.72 M	589 K
drum	ref	0.136	8.1 M	22 M	3.54 G	667 M
	trim	0.144	1.0 M	3.7 M	87.5 M	10.2 M
onset	ref	0.131	4.7 M	21 M	2.66 G	532 M
	trim	0.132	522 K	3.7 M	87.1 M	8.2 M

Table 1. Comparison between reference models and our trimmed models on *test error*, *number of parameters*, *disk size*, *inference FLOPS* and *memory used* across tasks.

5.3 Resulting model properties

We provide a detailed analysis of the gains provided by our *trimming lottery* for each task. We compare the reference model to the *optimal* one (smallest model within 1.1 times the original test error) found by trimming. We evaluate their *test error*, *number of parameters*, *disk size*, *FLOPS* (required to infer from a single input example) and *memory used* for different MIR tasks, as detailed in Table 1. As discussed previously, we are able to obtain models maintaining the error rates, while having only a small portion of the capacity of the very large models. This can be witnessed in the final properties of the trimmed models. A very interesting observation is that this decrease in parameters amounts to an even larger decrease in the *memory* and *computation power* required. Indeed, while most trimmed models are 10 times smaller than original large models, they use 20 to 50 times less computation power and memory requirements. This can be explained by the fact that most operations are processed across the dimensions of the previous layer. Hence, even small gains in number of parameters can lead to dramatic gains in computation.

6. CONCLUSION

In this paper, we presented a method to obtain ultra-light deep models for MIR, by extending the lottery ticket hypothesis to effectively trim the networks. We have shown that these efficient trimmed subnetworks, removing up to 85% of the weights in deep models, could be found across several MIR tasks. We have also shown that extremely small networks could be found by relying on *masking*, but these do not provide actual enhancement in terms of computation or memory requirements. Oppositely, we have shown that given the non-linear relationship between the number of parameters and computation required, we could find extremely light networks through trimming. These results encourage the crucial implementation of MIR models in embedded audio platforms, which would allow broader end-user applications. The major downside of this approach is its training time, which we partly address by decreasing the cost of each pruning iteration. However, the intriguing prospect of *ticket transfer* [14] could provide such initializations right *at the onset* of training.

7. ACKNOWLEDGEMENTS

This work is supported by the ANR:17-CE38-0015-01 MAKIMOno project, the SSHRC:895-2018-1023 ACTOR Partnership and Emergence(s) ACIDITEAM project from Ville de Paris and ACIMO projet of Sorbonne Université.

8. REFERENCES

[1] M. A. Casey, R. Veltkamp, M. Goto, M. Leman, C. Rhodes, and M. Slaney, "Content-based music information retrieval: Current directions and future challenges," *Proceedings of the IEEE*, vol. 96, no. 4, pp. 668–696, 2008.

[2] E. J. Humphrey, J. P. Bello, and Y. LeCun, "Moving beyond feature design: Deep architectures and automatic feature learning in music informatics." in *12th International Society for Music Information Retrieval Conference (ISMIR)*, 2012, pp. 403–408.

[3] Y. You, Z. Zhang, C.-J. Hsieh, J. Demmel, and K. Keutzer, "ImageNet training in minutes," in *Proceedings of the 47th International Conference on Parallel Processing*, 2018, pp. 1–10.

[4] 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)*, 2016, pp. 770–778.

[5] Y. LeCun, J. S. Denker, and S. A. Solla, "Optimal brain damage," in *Advances in Neural Information Processing Systems (NIPS)*, 1990, pp. 598–605.

[6] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *arXiv preprint arXiv:1510.00149*, 2015.

[7] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," in *NIPS Deep Learning and Representation Learning Workshop*, 2015.

[8] Z. Liu, M. Sun, T. Zhou, G. Huang, and T. Darrell, "Rethinking the value of network pruning," in *International Conference on Learning Representations*, 2019.

[9] T. Gale, E. Elsen, and S. Hooker, "The state of sparsity in deep neural networks," *arXiv preprint arXiv:1902.09574*, 2019.

[10] J. Frankle and M. Carbin, "The lottery ticket hypothesis: Finding sparse, trainable neural networks," in *International Conference on Learning Representations (ICLR)*, 2019.

[11] J. Frankle, G. K. Dziugaite, D. M. Roy, and M. Carbin, "Linear mode connectivity and the lottery ticket hypothesis," *arXiv preprint arXiv:1912.05671*, 2019.

[12] ——, "Stabilizing the lottery ticket hypothesis," *arXiv preprint arXiv:1903.01611*, 2019.

[13] J. Frankle, D. J. Schwab, and A. S. Morcos, "The early phase of neural network training," in *International Conference on Learning Representations*, 2020.

[14] A. Morcos, H. Yu, M. Paganini, and Y. Tian, "One ticket to win them all: generalizing lottery ticket initializations across datasets and optimizers," in *Advances in Neural Information Processing Systems (NIPS)*, 2019, pp. 4933–4943.

[15] H. You, C. Li, P. Xu, Y. Fu, Y. Wang, X. Chen, R. G. Baraniuk, Z. Wang, and Y. Lin, "Drawing early-bird tickets: Toward more efficient training of deep networks," in *International Conference on Learning Representations (ICLR)*, 2019.

[16] P. Esling, A. Chemla-Romeu-Santos, and A. Bitton, "Bridging audio analysis, perception and synthesis with perceptually-regularized variational timbre spaces." in *18th International Society for Music Information Retrieval Conference (ISMIR)*, 2018.

[17] J. Wilkins, P. Seetharaman, A. Wahl, and B. A. Pardo, "VocalSet: A singing voice dataset," in *Proceedings of the 19th International Society for Music Information Retrieval (ISMIR) Conference*, 2018, pp. 468–474.

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

[19] T. Carsault, J. Nika, and P. Esling, "Using musical relationships between chord labels in automatic chord extraction tasks," in *International Society for Music Information Retrieval (ISMIR) Conference*, 2018.

[20] K. Choi and K. Cho, "Deep unsupervised drum transcription," in *20th International Society for Music Information Retrieval (IISMIR) Conference*, 2019.

[21] J. Schlüter and S. Böck, "Improved musical onset detection with convolutional neural networks," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2014, pp. 6979–6983.

[22] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," in *Advances in Neural Information Processing Systems (NIPS)*, 2015, pp. 1135–1143.

[23] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient ConvNets," in *International Conference on Learning Representations (ICLR)*, 2017.

[24] R. Shwartz-Ziv and N. Tishby, "Opening the black box of deep neural networks via information," *arXiv preprint arXiv:1703.00810*, 2017.

[25] B. McFee, L. Barrington, and G. Lanckriet, "Learning content similarity for music recommendation," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 8, pp. 2207–2218, 2012.

[26] E. J. Humphrey, J. P. Bello, and Y. LeCun, "Feature learning and deep architectures: New directions for music informatics," *Journal of Intelligent Information Systems*, vol. 41, no. 3, pp. 461–481, 2013.

[27] P. Esling, N. Masuda, A. Bardet, R. Despres, and A. Chemla-Romeu-Santos, "Flow synthesizer: Universal audio synthesizer control with normalizing flows," *Applied Sciences*, vol. 10, no. 1, p. 302, 2020.

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

[29] A. De Cheveigné and H. Kawahara, "Yin, a fundamental frequency estimator for speech and music," *The Journal of the Acoustical Society of America*, vol. 111, no. 4, pp. 1917–1930, 2002.

[30] G. Tzanetakis and P. Cook, "Musical genre classification of audio signals," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 10, no. 5, pp. 293–302, 2002.

[31] B. Li, X. Liu, K. Dinesh, Z. Duan, and G. Sharma, "Creating a multitrack classical music performance dataset for multimodal music analysis: Challenges, insights, and applications," *IEEE Transactions on Multimedia*, vol. 21, no. 2, pp. 522–535, 2019.

[32] R. Bittner, J. Salamon, M. Tierney, M. Mauch, C. Cannam, and J. Bello, "Medleydb: A multitrack dataset for annotation-intensive mir research," in *Proceedings of the 15th International Society for Music Information Retrieval (ISMIR) Conference*, 2014.

[33] J. Engel, C. Resnick, A. Roberts, S. Dieleman, D. Eck, K. Simonyan, and M. Norouzi, "Neural audio synthesis of musical notes with WaveNet autoencoders," *International Conference on Machine Learning*, vol. 70, pp. 1068–1077, 2017.

[34] M. Cartwright and J. P. Bello, "Increasing drum transcription vocabulary using data synthesis," in *Proc. International Conference on Digital Audio Effects (DAFx)*, 2018, pp. 72–79.

[35] C. Dittmar and D. Gärtner, "Real-time transcription and separation of drum recordings based on NMF decomposition." in *Proc. International Conference on Digital Audio Effects (DAFx)*, 2014, pp. 187–194.