Article

Deep Sparse Learning for Automatic Modulation Classification Using Recurrent Neural Networks

Ke Zang [1], Wenqi Wu [1] and Wei Luo [2,*,†]

[1] College of Biomedical Engineering and Instrument Science, Yuquan Campus, Zhejiang University, 38 Zheda Road, Hangzhou 310027, China; martinzang@zju.edu.cn (K.Z.); winkywow@zju.edu.cn (W.W.)
[2] Department of Biomedical Engineering, The Chinese University of Hong Kong, Shatin, New Territories, Hong Kong, China
* Correspondence: williamluo@cuhk.edu.hk
† Most of this work was done when he was at Zhejiang University, Hangzhou 310027, China.

Abstract: Deep learning models, especially recurrent neural networks (RNNs), have been successfully applied to automatic modulation classification (AMC) problems recently. However, deep neural networks are usually overparameterized, i.e., most of the connections between neurons are redundant. The large model size hinders the deployment of deep neural networks in applications such as Internet-of-Things (IoT) networks. Therefore, reducing parameters without compromising the network performance via sparse learning is often desirable since it can alleviates the computational and storage burdens of deep learning models. In this paper, we propose a sparse learning algorithm that can directly train a sparsely connected neural network based on the statistics of weight magnitude and gradient momentum. We first used the MNIST and CIFAR10 datasets to demonstrate the effectiveness of this method. Subsequently, we applied it to RNNs with different pruning strategies on recurrent and non-recurrent connections for AMC problems. Experimental results demonstrated that the proposed method can effectively reduce the parameters of the neural networks while maintaining model performance. Moreover, we show that appropriate sparsity can further improve network generalization ability.

Keywords: automatic modulation classification; deep sparse learning; recurrent neural networks

check for updates

Citation: Zang, K.; Wu, W.; Luo, W. Deep Sparse Learning for Automatic Modulation Classification Using Recurrent Neural Networks. *Sensors* **2021**, *21*, 6410. https://doi.org/10.3390/s21196410

Academic Editors: Alan Michaels and William Chris Headley

Received: 15 August 2021
Accepted: 23 September 2021
Published: 25 September 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Automatic modulation classification (AMC) refers to the automatic recognition of the modulation category of the received signal. This technology is widely used in spectrum management and interference recognition, etc. [1,2]. With the rapid development of wireless communication in recent years, the modulation types of signals has become more complex and diverse, which makes AMC a crucial technique in crowded radio environments.

Initially, the likelihood-based method was used to solve the modulation classification problem [3,4]. Although the methods based on likelihood can reduce the probability of mismatch, they usually suffer from high computational complexity, and their self-based theoretical system models are difficult to match with practical scenarios. To alleviate the computation overhead in practice, feature-based method [5–7] for AMC arose in response to the proper time and conditions. Traditional feature-based methods mainly consist of manually extracted features and appropriate classifiers. The feature-based approach became the mainstream because it is robust relative to different signals and has better generalization ability. With the development of Big Data and the improvement of computing power, deep neural networks has shown its powerful potential in many fields, such as computer vision [8] and natural language processing [9]. Among these models, recurrent neural networks (RNNs) [10] have been playing an essential role in tasks involving sequential data due to their ability to find the dependencies between data located in different parts of a sequence. For AMC problems, the manually designed features in traditional methods are usually extracted both locally and globally from the observed signal, which is exactly what RNNs are good at. Moreover, it is often observed that manually designing features may result in loss of information that is essential for accurate classification [11]. Therefore, researchers have attempted to use RNNs such as long short-term memory (LSTM) [12] for higher accuracy in AMC problems [13–15]. However, being overparameterized is a widely recognized property of deep neural networks [16,17]. It is difficult to apply deep neural networks into the edge devices [18,19], such as Internet-of-Thing(IoT) devices, which are usually equipped with limited device memories [20]. Therefore, removing the redundant connections of deep learning models while maintaining their performance is often desirable since it can alleviate computational and storage burdens.

Sparse learning is an efficient technique for training a sparsely connected neural network. Neural network pruning, the task of reducing the size of a network by removing parameters, has been the subject of numerous studies in recent years. Pruning a neural network amounts to removing its superfluous connections while maintaining model performance as much as possible [18,21]. The sparsity of a deep learning network is determined by the proportion of zeros in its trainable parameters $\boldsymbol{\theta}$. A sparse structure is usually obtained by multiplying the $\boldsymbol{\theta}$ element wisely by a binary mask $\mathbf{m}$ of the same size.

$$\boldsymbol{\theta}^{sparse} = \boldsymbol{\theta} \odot \mathbf{m}. \quad (1)$$

Here, $\odot$ stands for element-wise multiplication. For many years, people have generally believed that training a dense, overparameterized network in advance is the key to effective subsequent sparseness [22]. Accordingly, certain methods are needed to remove redundant parameters without significantly affecting the model performance. Most of the currently used algorithms for generating sparse masks begin with a dense model, and then they increase the sparsity of the model through the network pruning. Pruning methods can be divided into two categories: iterative pruning [23–26] and one-shot pruning [22,27,28]. As shown in Figure 1, an iterative pruning method first trains a model until it converges. Then, the sparse mask $\mathbf{m}$ will be updated based on specific criteria that vary across different methods. This process is repeated until the model sparsity meets the requirements. Han et al. introduced a method to reduce the storage and computation of neural networks by an order of magnitude via removing the unimportant connections [23]. Frankle et al. proposed the Lottery Ticket Hypothesis, that is, a dense, randomly initialized feedforward network with separate training containing subnetworks (winning tickets) that can achieve similar test accuracy as the original network under a similar number of iterations [26]. In addition, they presented an iterative pruning algorithm to identify winning tickets on the MNIST and CIFAR10 datasets. On the other hand, the model training process in one-shot pruning algorithms is interleaved with pruning steps. Thus, a sparsely connected model can be obtained within one training process. In [27], the authors prune the network through the L_0 norm regularization that allows for straightforward and efficient learning of model structures with stochastic gradient descent. The single-shot network pruning method [22] prunes a given network once at initialization prior to training. After pruning, a network is trained in the standard manner.

In existing methods, pruning is usually performed by using an iterative finetuning process, or with a pruning scheme designed heuristically, or with the addition of hyperparameters, thereby undermining their utility. In this work, we present a new approach that prunes a given network based on the statistics of the weight magnitude and gradient momentum without iterative training. Unlike other pruning algorithms that were applied only to feedforward neural networks (FNNs), such as multilayer perceptrons (MLPs) and convolutional neural networks (CNNs) on non-sequential data, the proposed method in this paper is validated on both feedforward and recurrent neural architectures. The main contributions of this paper are as follows:

(1) A novel one-shot neural network pruning algorithm based on weight magnitude and gradient momentum is proposed to produce sparse RNNs for solving AMC problems

without compromising model performance. Specifically, we demonstrate that it is crucial to retain non-recurrent connections while pruning RNNs.

(2) In addition to the sequential AMC problem, the efficiency of the proposed method is also validated on non-sequential dataset, including MNIST and CIFAR10, with feed-forward neural networks.

(3) The experimental results reveal that the proposed pruning method can serve as a regularization technique as the resulting sparse models can outperform their dense counterparts even with high-level sparsity.

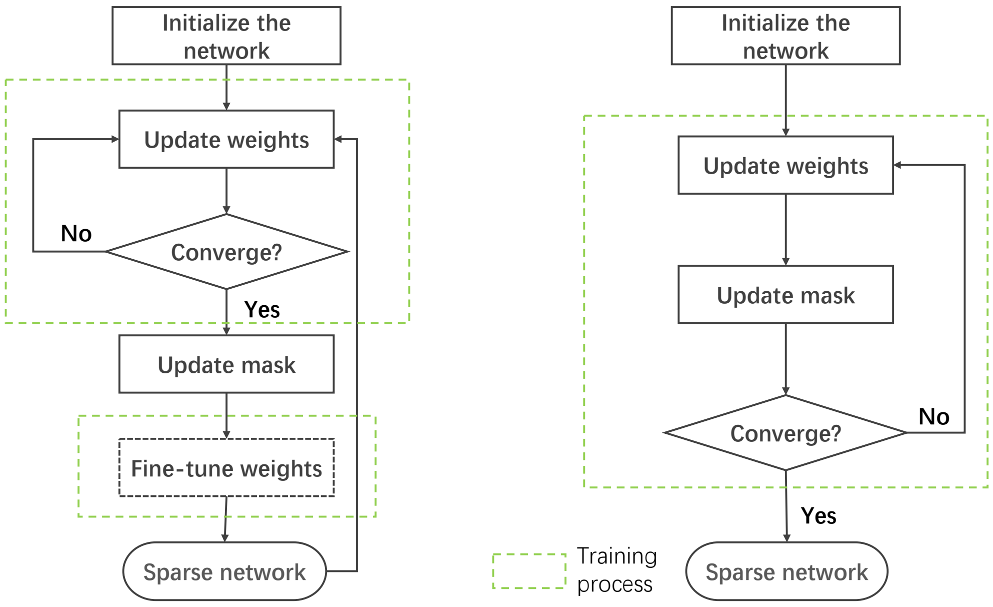

Figure 1. Iterative pruning (**left**) and one-shot pruning (**right**).

2. Methods

2.1. Notation

Bold numbers such as **1** and **0** denote vectors consist of the corresponding numbers. The element-wise multiplication is denoted by $\odot$, and the convolution operation is denoted by $*$. The sigmoid activation function σ used extensively in deep learning models is defined as $\sigma(x) = \frac{1}{1+e^{-x}}$, and the hyperbolic function tanh is defined as $\tanh(x) = 2\sigma(2x) - 1$.

2.2. Recurrent Neural Networks

Recurrent neural networks [29–31], particularly with gated cells such as LSTMs [12] and gated recurrent units (GRUs) [32], are perhaps the most popular architectures for modeling temporal sequences. The LSTM reads from and writes to its internal states by using a gating mechanism, which allows information to pass selectively. There are three different kinds of gating units inside an LSTM cell, namely the write, read, and forget gates. The write and read gates are used to filter out useless information flowing in and out of the recurrent cells, respectively, while the forget gates can selectively erase old memories.

These three gates are realized by the gating mechanism, as shown in Figure 2, and are formulated as follows.

$$\mathbf{i}_t = \sigma(\mathbf{W}_i\mathbf{h}_{t-1} + \mathbf{U}_i\mathbf{x}_t + \mathbf{b}_i), \tag{2a}$$

$$\mathbf{o}_t = \sigma(\mathbf{W}_o\mathbf{h}_{t-1} + \mathbf{U}_o\mathbf{x}_t + \mathbf{b}_o), \tag{2b}$$

$$\mathbf{f}_t = \sigma(\mathbf{W}_f\mathbf{h}_{t-1} + \mathbf{U}_f\mathbf{x}_t + \mathbf{b}_f). \tag{2c}$$

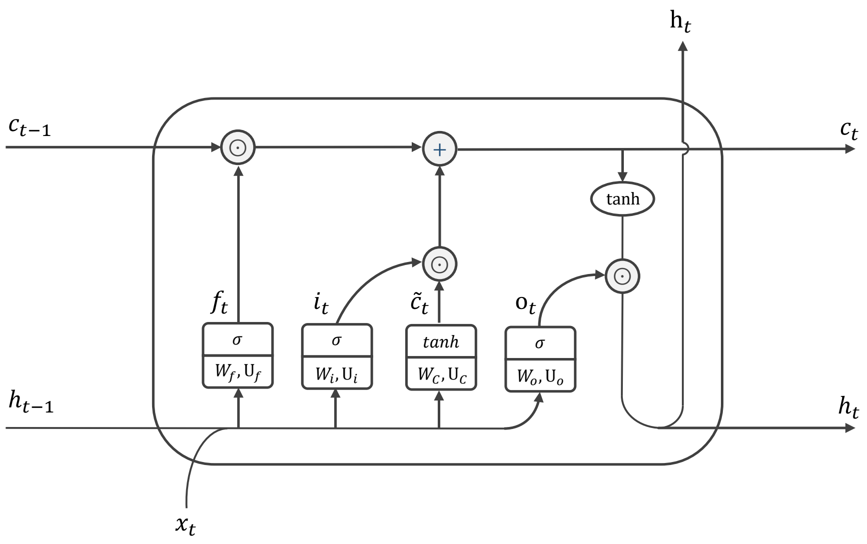

Figure 2. Long short-term memory [14].

The candidate memory $\tilde{\mathbf{c}}_t$ to be written is defined as the following.

$$\tilde{\mathbf{c}}_t = \tanh(\mathbf{W}_c\mathbf{h}_{t-1} + \mathbf{U}_c\mathbf{x}_t + \mathbf{b}_c). \tag{3}$$

The new memory $\mathbf{c}_t$ and the information flowing out the cell $\mathbf{h}_t$ are then given by the following.

$$\mathbf{c}_t = \mathbf{c}_{t-1} \odot \mathbf{f}_t + \tilde{\mathbf{c}}_t \odot \mathbf{i}_t, \tag{4}$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t). \tag{5}$$

GRU is another popular recurrent architecture based on gate units, which was first introduced by Chung et al. in 2014 [32]. It can be seen as a variation of LSTM that explicitly couples write and forget gates.

Another RNN we used in this paper is a hierarchical RNN with grouped auxiliary memory named GAM-HRNN [14]. As shown in Figure 3, the main framework of the network is a hierarchical structure with other RNNs as the kernel. At each time step, due to the group distributed mechanism, the corresponding part of the unit of state is overwritten, while the other parts change slightly or remain unchanged. In this manner, useful information can be saved without overwriting the long-term memory. After the auxiliary memory has been updated, the state of each layer in the hierarchical structure is updated sequentially by using the information selectively read from the auxiliary memory and the state passed from the previous time step. In this manner, the network provides a shortcut in time and space, which is good for confronting conflicts between short and long periods of information and preserving long-term information. Formulation details can be found in [14].

Note that the learnable weights in recurrent units are denoted by **W** and **U**, representing the recurrent and non-recurrent connections, respectively. We used different pruning strategies for these two kinds of weights in the experiments of this paper.

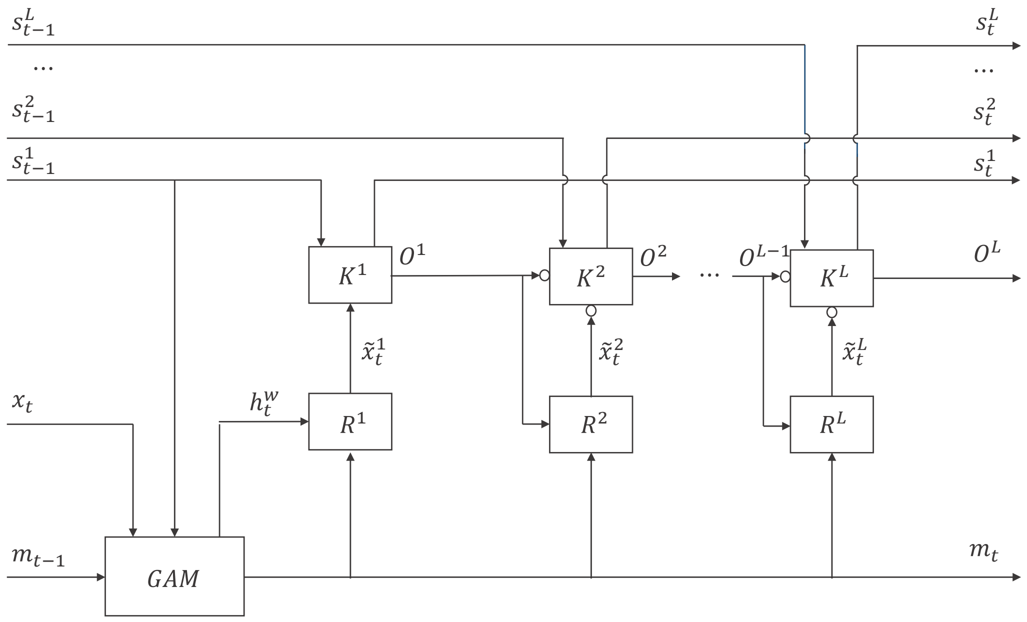

Figure 3. Diagram of the hierarchical recurrent neural network with grouped auxiliary memory architecture. Inputs conveyed by arrows with ∘ will be concatenated together [14].

2.3. Pruning Method

Given a dataset $\mathcal{D}^{train} = (x_i, y_i)_{i=1}^N$, the objective function can be formulated as follows.

$$\mathcal{E}(F(\cdot,\boldsymbol{\theta});\mathcal{D}^{train}) = \frac{1}{N}\sum_{i=1}^{N}\mathcal{L}(y_i, F(x_i,\boldsymbol{\theta}))). \tag{6}$$

Here, $\mathcal{L}$ is the loss function such as cross-entropy, and $\boldsymbol{\theta}$ denotes the parameters of network $F(\cdot,\boldsymbol{\theta})$.

The main hypothesis behind the neural network sparsity literature is that neural networks are usually overparameterized, implying that most elements in the weight vector $\boldsymbol{\theta}^*$ of a converged model $F(\cdot,\boldsymbol{\theta}^*)$ are redundant. Thus, comparable performance can be achieved by using a smaller network [33], e.g., the sparse version of this model $F(\cdot,\mathbf{m}\odot\boldsymbol{\theta}^*)$. Here, $\mathbf{m}$ is a sparse mask. To this end, the objective is to learn a sparse network while maintaining the accuracy of its dense counterpart as much as possible. The sparse mask can be generated via network pruning. A part of the pruning algorithms quantifies the importance of network connections based on their magnitude [23,26]. However, these methods will also remove connections that can greatly reduce the loss after being updated, yet they are not significant in magnitude. This can be circumvented by considering the weight gradient $\mathcal{G}_t = \nabla_{\boldsymbol{\theta}}\mathcal{E}$ as another factor to measure the connection sensitivity [22,28,34]. In practice, deep learning models are always trained using the stochastic gradient descend algorithm; thus, the gradient momentum is always used to estimate the global gradient calculated on the entire dataset.

$$v_t = \gamma v_{t-1} + (1-\gamma)\nabla_{\boldsymbol{\theta}}\mathcal{E}(\boldsymbol{\theta}). \tag{7}$$

Here, γ is usually set to a value that is less than but close to 1, e.g., 0.9.

In this paper, we propose a novel one-shot neural network pruning algorithm based on both magnitude and gradient momentum of learnable parameters. The algorithm is described in Algorithm 1. Previous investigation has already shown that removing the connections with small magnitude can yield sparse network without sacrificing performance. Thus, in the proposed method, we directly preserve the weights of large magnitudes (Algorithm 1 line 11). In order to preserve the weak connections that may potentially contribute to reducing training loss, we also used the gradient momentum to estimate the importance of each weight. Assume that N connections should be pruned in each iteration, we first select $\delta\cdot$ N connections with the smallest magnitude as candidates. Here, $\delta(>1)$ is a hyper-parameter. We then select N connections with the lowest score of importance and remove them from the network by updating the sparse mask. The score of importance for each parameter $\boldsymbol{\theta}_i$ is defined as follows:

$$\mathbf{s}_i = \alpha \cdot |\boldsymbol{\theta}_i| + (1-\alpha) \cdot |\mathbf{v}_i|, \tag{8}$$

where $\mathbf{v}_i$ is the gradient momentum of the i-th parameter $\boldsymbol{\theta}_i$, and α is another hyper-parameter.

Algorithm 1 The proposed method

```
Require: Training set $\mathcal{D}^{train} = (x_i, y_i)$
Require: Network F with parameters $\boldsymbol{\theta}$
Require: Pruning interval K, hyper-parameter for calculating momentum $\gamma$, hyper-
    parameter for pruning $\delta$ and $\alpha$, pruning rate $p$.
 1: Initialize the parameters $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta_0}$;
 2: Initialize the momentum $\mathbf{v} \leftarrow \mathbf{0}$;
 3: Initialize the parameter mask: $\mathbf{m} \leftarrow \mathbf{1}$;
 4: repeat
 5:     for n=1 to K do
 6:         Generate data batch: $D^{batch} \sim D^{train}$;
 7:         Update $\boldsymbol{\theta}$:$\boldsymbol{\theta} \leftarrow update(\boldsymbol{\theta} \odot \mathbf{m}, D^{batch})$;
 8:         Update momentum using Equation (7);
 9:     end for
10:     Get number of connections to be pruned $N = p \cdot sum(\mathbf{m})$;
11:     Get the candidate mask based on magnitude: $\mathbf{m_c} \leftarrow$ getCandidateMask($\delta \cdot N, \boldsymbol{\theta}$);
12:     Calculate score of importance $\mathbf{s}$ using Equation (8);
13:     Select connections to be pruned among candidates based on score $\mathbf{m_s} \leftarrow$
    getConnectionsToPrune($\mathbf{s}, \mathbf{m_c}$) ;
14:     Update sparse mask $\mathbf{m} \leftarrow$ update($\mathbf{m_s}$);
15: until meeting training termination condition
```

Note that in the early stage of the proposed algorithm, the network can be trained without pruning for several iterations for warming up, which may improve the final performance in some cases. In this paper, the training process is always terminated when a certain degree of sparsity has been met.

3. Experimental Results and Discussions

In this section, we evaluate the performance of the proposed method on the standard MNIST and CIFAR10 datasets that are always used as benchmark datasets for pruning tasks by comparing with other pruning methods. Then, we used the proposed method for AMC problems on the standard RadioML 2016.10a dataset via RNNs comparing with other classification methods.

3.1. Experimental Configuration

We used Xavier uniform initalizer [35] for all weights and Adam optimizer [36] for the training processes. The models were implemented using Tensorflow [37]. All experiments were repeated 10 times.

3.1.1. MNIST and CIFAR10 Datasets

The proposed method in this paper is first compared with other sparse learning techniques including a magnitude-based pruning algorithm proposed in [23] and a rewinding-after-pruning training scheme used to find the 'lottery ticket' presented in [26]. All pruning methods to be compared are performed on the Lenet-300-100 [38] model for the MNIST dataset and a two-layer CNN for the CIFAR10 dataset. The Lenet-300-100 model is a fully connected network with two hidden layers consisting of 300 and 100 neurons, respectively. The two-layer CNN includes two convolutional layers and a pooling layer followed by two fully connected layers and an output layer. We denote this model as Conv2 in this paper. Other details of these two feedforward neural networks can be found in [26]. We adopted the original setups described in the corresponding paper to configure the pruning algorithms to be compared. The hyperparameters used in the proposed method are listed in Table 1.

Table 1. Pruning hyper-parameters for Lenet-300-100, Conv2, and RNNs.

	Lenet-300-100 (MNIST)	**Conv2 (CIFAR10)**	**RNNs (RadioML)**
Batch size	100	64	400
Warm up epochs	1	5	10
Prune frequency	Once per epoch	Once per epoch	Once per epoch
Prune rate p	0.005	0.002	0.002
α	0.2	0.3	0.3
γ	0.4	0.3	0.3
δ	2	2	2

3.1.2. RadioML 2016.10a Dataset

For AMC problems, we verified the effectiveness of our method by mainly using an open modulated classification dataset named RadioML2016.10a [39]. There are 220,000 RF signals modulated by three analog and eight digital modulation types. Specifically, analog modulation methods include wide band FM (AM-FM), single-sideband AM (AM-SSB), and wideband FM (WB-FM) and digital modulation methods include quadrature phase-shift keying (QPSK), eight phase-shift keying (8PSK), quadrature amplitude modulation 16 (QAM16), quadrature amplitude modulation 64 (QAM64), cover binary phase-shift keying (BPSK), continuous phase frequency-shift keying (CPFSK), Gauss frequency-shift keying (GFSK), and pulse-amplitude modulation four (PAM4). Each signal is 128 in length and samples per symbol is eight. The signal-to-noise ratio (SNR) is evenly distributed from -20 dB to 18 dB at intervals of 2 dB. Radio channel including time varying multi-path fading, random walk drifting, and non-ideal effects covering carrier frequency offset oscillator drift, etc., are well-characterized. More details can be found in [39].

The proposed method was used to prune recurrent neural architectures including LSTM, GRU, and GAM-HRNN. Three pruning methods were also tested on GAM-HRNN model. All of these recurrent models have two hidden layers. The normalized amplitude and phase of the signal are obtained from the original IQ data. We set the forget gate bias to 1.0 for LSTM. Details regarding the model hyperparameters can be found in [14]. Performances of densely connected neural networks including sequential convolutional recurrent neural network (SCRNN) and GAM-HRNN are also reported. All models have roughly the same number of parameters. For RNNs, we only prune the recurrent connections mentioned in 2.1. Details regarding the prune hyperparameters are listed in Table 1.

3.2. Results on Standard MNIST and CIFAR10 Datasets

For the MNIST dataset, it can be observed from Figure 4 that for all the methods, as the percentage of weights remaining decreased, the accuracy first increased and then decreased. At a high percentage of weights remaining, the proposed method performed better than other methods. However, as the percentage of weights remaining decreased, the accuracy of the proposed method was observed to be lower than that of the lottery hypothesis. When the percentage of remaining weights further decreased, the proposed method was found to exhibit better performance than others again. As shown in Figure 5, the results of CIFAR10 are similar to those of MNIST. However, on the CIFAR10 dataset, the proposed method performed consistently better than other methods considering the percentage of remaining weights.

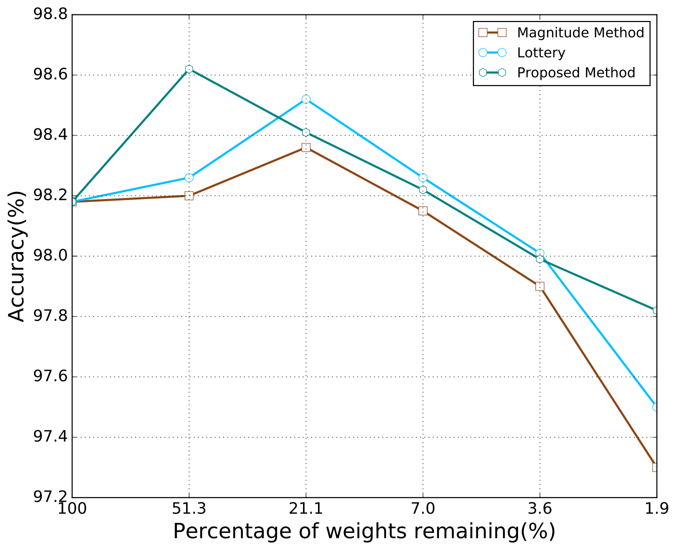

Figure 4. Classification accuracy for different percentages of weights remaining from Lenet-300-100 on MNIST.

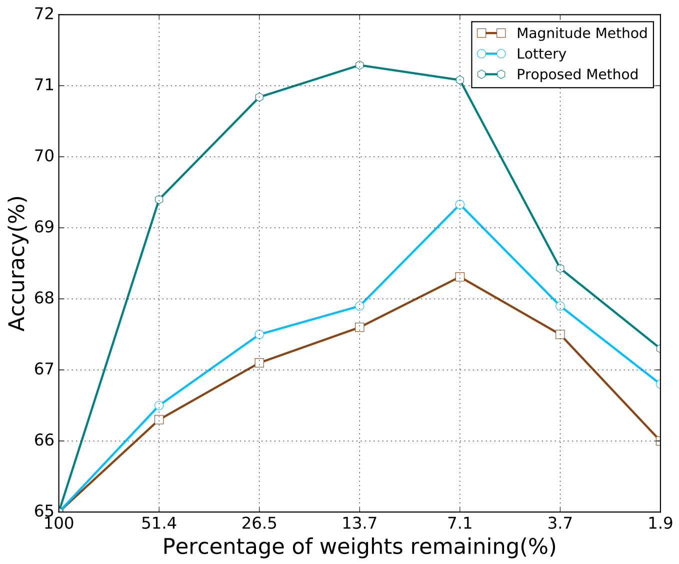

Figure 5. Classification accuracy for different percentages of weights remaining from Conv2 on CIFAR10.

The highest accuracy of each model achieved by each pruning method is listed in Table 2. The results demonstrate that a neural network with a certain degree of sparsity may outperform its dense counterpart, and the proposed method in this paper always yields a higher performance boost for each model on each task.

Table 2. Best accuracy of each model on MNIST and CIFAR10 (%).

	Lenet-300-100 on MNIST Accuracy (%)	**Conv2 on CIFAR10 Accuracy (%)**
Unpruned baseline	98.16 ± 0.06	65.09 ± 0.045
Magnitude-based [23]	98.40	68.31
Lottery [26]	98.52	69.33
Proposed method	98.62 ± 0.067	71.29 ± 0.059

3.3. Results on RadioML 2016.10a Dataset

In this section, we discuss the performance of the proposed method for recurrent neural architectures for AMC problems.

We first apply the proposed pruning method to remove the recurrent connections of RNNs used in [14], including LSTM, GRU, and GAM-HRNN. The classification accuracies with different percentages of remaining weights are presented in Figure 6. It can be observed that as the model parameters decreased, the accuracy of the model first increased and then decreased, which is similar to the feedforward cases. With a certain degree of sparsity, each model can outperform its dense counterpart. Note that the percentage of weights remaining corresponding to the best accuracy for each model is different. Moreover, all recurrent models are still able to beat their dense counterparts with most of their connections being pruned, especially for GAM-HRNN. Classification accuracy for different pruning methods on GAM-HRNN model is presented in Figure 7. We can find that the proposed method also achieve better performance on AMC problems compared to other pruning methods. The performances of other methods on AMC problems are also compared. Table 3 reports the corresponding average accuracy for all SNRs, and the SNR ranges from 0 to 20 dB. The dash symbol in Table 3 indicates that the metric was not reported in the corresponding paper. It can be observed that the sparse GAM-HRNN model produced by the proposed method outperforms the previous state-of-the-art model on both two metrics. Meanwhile, Figure 8 shows the classification accuracy for the proposed method on three RNN models at different SNRs. We can observe that all three models perform poorly at low SNR. However, the advantage of GAM-HRNN over the other two models becomes more obvious as SNR increases.

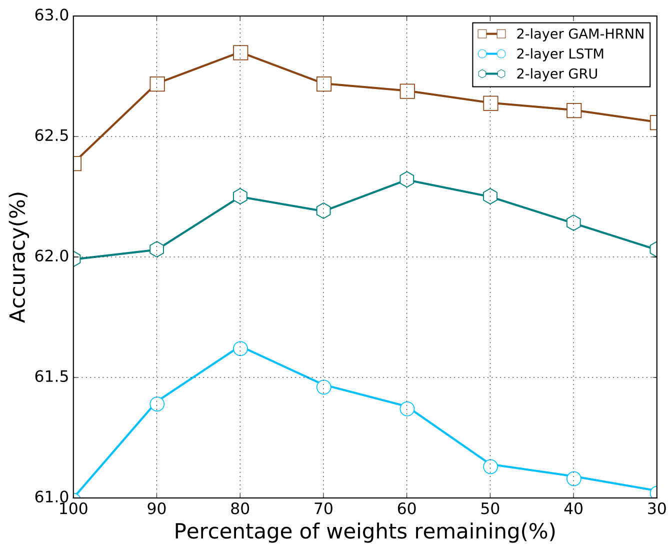

Figure 6. Classification accuracy for the proposed pruning method on three RNN models.

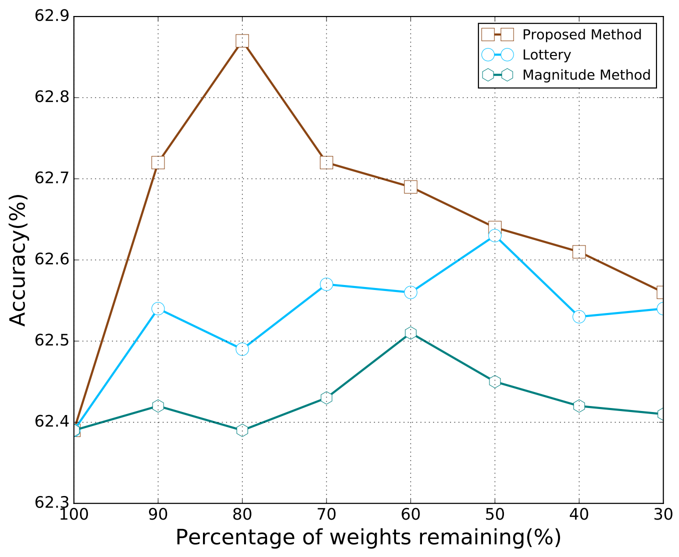

Figure 7. Classification accuracy for different pruning methods on GAM-HRNN model.

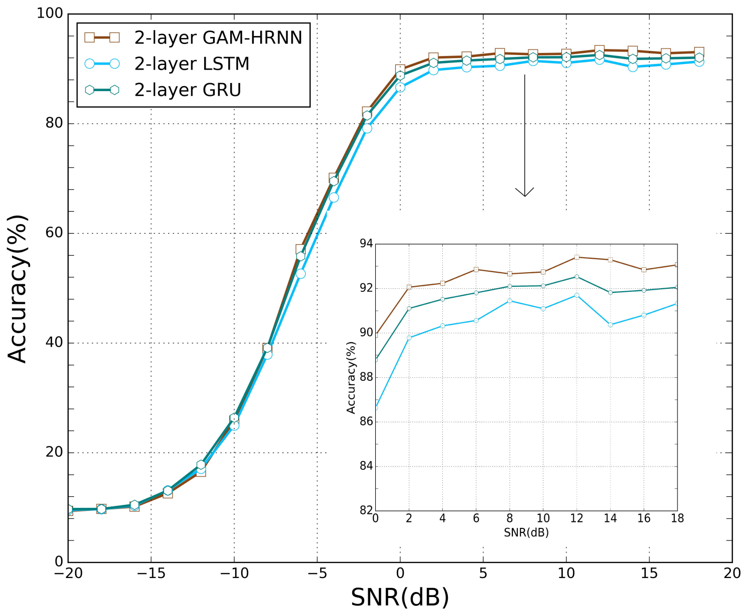

Figure 8. Classification accuracy for the proposed method on three RNN models at different SNRs.

Table 3. Average accuracy for all SNRs (AccAS) and for SNR ranges from 0 dB to 20 dB (AccASH) (%).

	AccAS		**AccASH**	
	Original	**Pruned**	**Original**	**Pruned**
2-layer GAM-HRNN	62.47 [14]	62.87 ± 0.077	92.2 [14]	92.45 ± 0.083
2-layer LSTM	60.8 ± 0.073	61.12 ± 0.11	90 [13]	90.81 ± 0.09
2-layer GRU	61.68 ± 0.087	62.06 ± 0.073	91.13 ± 0.069	91.59 ± 0.063
Cross model [40]	62.41	-	-	-
Multipath CNN [41]	-	-	90.7	-
Multitask CNN [42]	59.43	-	-	-
SCRNN [15]	-	-	92.1	-

The confusion matrices at three SNRs (18 dB, 0 dB, and −8 dB) for pruned GAM-HRNN with the best accuracy using the proposed method is shown in Figure 9. Even at high SNR, the network cannot distinguish am-DSB and WBFM signals well. This can be attributed to the small observation window (0.64 ms of modulated speech per example) and low information rate with frequent silence between words [43]. Meanwhile, the network also has a certain ambiguity to distinguish QAM16 and QAM64. However, this problem has been alleviated compared to [14] since the proposed method improves the network generalization ability.

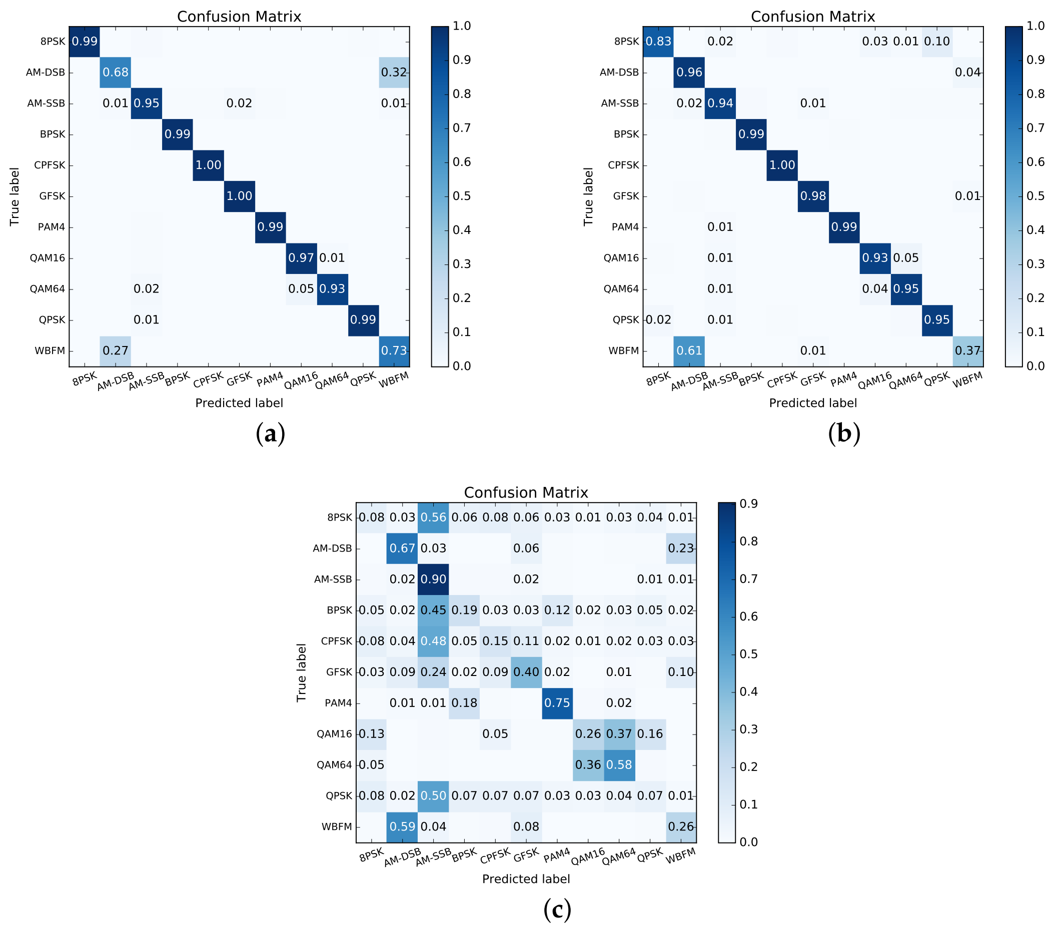

Figure 9. Confusion matrices of 2-layer pruned GAM-HRNN model on RadioML 2016.10a dataset at (**a**) 18 dB SNR, (**b**) 0 dB SNR, and (**c**) −8 dB SNR.

As mentioned in 2.1, recurrent models have recurrent connections and non-recurrent connections. Figure 6 shows the results of the proposed pruning algorithm by removing only the recurrent connections. The importance of performing this step is illustrated in Figure 10, which shows the consequences of pruning both the recurrent and non-recurrent connections. We can observe that LSTM suffers from an obvious performance decrease as the connections being pruned. As for GRU and GAM-HRNN, the performances are not as good as those shown in Figure 6. Therefore, we conjecture that removing non-recurrent weights that connect input neurons to recurrent units at each time step may result in inefficient feature extraction, resulting in worse model performance. On the other hand, pruning recurrent weights appropriately can facilitate RNN memory transmission.

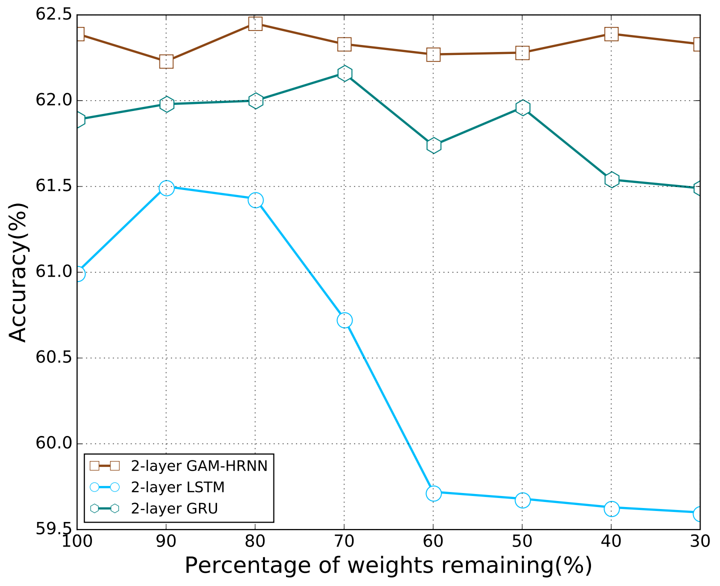

Figure 10. Classification accuracy for the proposed pruning method on three RNN models (both recurrent and non-recurrent connections are pruned).

4. Conclusions

In this paper, we present a sparse learning algorithm for RNNs on AMC problems based on the statistics of weight magnitude and gradient momentum. We demonstrate experimentally that non-recurrent connections should be retained during pruning. The proposed method can alleviate the computational and storage burden for recurrent models, facilitating their hardware implementations on devices with limited resources. Furthermore, experimental results also show that the proposed method can produce a neural model with a certain degree of sparsity that outperforms its dense counterpart. The efficiency of the proposed method is verified on both feedforward and recurrent neural architectures.

Our future work includes many aspects. For example, as the surviving neural connections are updated during training, the gradient of training loss with respect to the removed connections may become larger. Such connections can also contribute to reducing the training loss efficiently. Thus, mechanisms to restore the pruned weights can be considered. On the other hand, the proposed method produces unstructured sparsity other than structured sparsity [24]. Hence, the resulting sparse networks are currently not supported in terms of being accelerated by hardware. For this reason, the proposed method can be further extended to produce structured sparsity.

Author Contributions: Conceptualization, K.Z. and W.L.; methodology, K.Z. and W.L.; software, K.Z. and W.L.; validation, K.Z., W.W. and W.L.; formal analysis, K.Z., W.W. and W.L.; investigation, K.Z., W.W. and W.L.; data curation, K.Z.; writing—original draft preparation, K.Z.; writing—review and editing, K.Z., W.W. and W.L.; project administration, W.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Dobre, O.A.; Abdi, A.; Bar-Ness, Y.; Su, W. Survey of automatic modulation classification techniques: Classical approaches and new trends. *IET Commun.* **2007**, *1*, 137–156.:20050176. [CrossRef]
2. Huang, S.; Yao, Y.; Wei, Z.; Feng, Z.; Zhang, P. Automatic Modulation Classification of Overlapped Sources Using Multiple Cumulants. *IEEE Trans. Veh. Technol.* **2017**, *66*, 6089–6101. [CrossRef]

3. Dobre, O.A.; Hameed, F. Likelihood-based algorithms for linear digital modulation classification in fading CHANNELS. In Proceedings of the Canadian Conference on Electrical and Computer Engineering, Ottawa, ON, Canada, 7–10 May 2006; pp. 1347–1350. [CrossRef]
4. Chavali, V.G.; Da Silva, C.R. Classification of digital amplitude-phase modulated signals in time-correlated non-Gaussian channels. *IEEE Trans. Commun.* **2013**, *61*, 2408–2419. [CrossRef]
5. Swami, A.; Sadler, B.M. Hierarchical digital modulation classification using cumulants. *IEEE Trans. Commun.* **2000**, *48*, 416–429. [CrossRef]
6. Yuan, J.; Zhao-yang, Z.; Pei-liang, Q. Modulation classification of communication signals. In Proceedings of the MILCOM 2004 IEEE Military Communications Conference, Monterey, CA, USA, 31 October–3 November 2004. [CrossRef]
7. Lopatka, J.; Pedzisz, M. Automatic modulation classification using statistical moments and a fuzzy classifier. In Proceedings of the WCC 2000—ICSP 2000, 2000 5th International Conference on Signal Processing Proceedings, 16th World Computer Congress 2000, Beijing, China, 21–25 August 2000; pp. 1500–1506. [CrossRef]
8. Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; et al. ImageNet Large Scale Visual Recognition Challenge. *Int. J. Comput. Vis.* **2015**, *115*, 211–252.
9. Devlin, J.; Chang, M.W.; Lee, K.; Toutanova, K. BERT: Pre-training of deep bidirectional transformers for language understanding. In Proceedings of the 2019 Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies—Proceedings Conference, 2019; Minneapolis, MN, USA, 2 June–June 7 2019; Volume 1, pp. 4171–4186.
10. Lipton, Z.C.; Berkowitz, J.; Elkan, C. A Critical Review of Recurrent Neural Networks for Sequence Learning. *arXiv* **2015**, arXiv:1506.00019.
11. Zhang, D.; Ding, W.; Zhang, B.; Xie, C.; Li, H.; Liu, C.; Han, J. Automatic modulation classification based on deep learning for unmanned aerial vehicles. *Sensors* **2018**, *18*, 924. [CrossRef]
12. Hochreiter, S.; Schmidhuber, J. Long Short-Term Memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef]
13. Rajendran, S.; Member, S.; Meert, W.; Giustiniano, D.; Member, S.; Lenders, V.; Pollin, S.; Member, S. Classification with Distributed Low-Cost Spectrum Sensors. *IEEE Trans. Cogn. Commun. Netw.* **2018**, *4*, 433–445. [CrossRef]
14. Zang, K.; Ma, Z. Automatic Modulation Classification Based on Hierarchical Recurrent Neural Networks with Grouped Auxiliary Memory. *IEEE Access* **2020**, *8*, 213052–213061. [CrossRef]
15. Liao, K.; Zhao, Y.; Gu, J.; Zhang, Y.; Zhong, Y. Sequential Convolutional Recurrent Neural Networks for Fast Automatic Modulation Classification. *IEEE Access* **2021**, *9*, 27182–27188.
16. Denton, E.; Zaremba, W.; Bruna, J.; LeCun, Y.; Fergus, R. Exploiting linear structure within convolutional networks for efficient evaluation. *Adv. Neural Inf. Process. Syst.* **2014**, *2*, 1269–1277.
17. Ba, L.J.; Caruana, R. Do deep nets really need to be deep? *Adv. Neural Inf. Process. Syst.* **2014**, *3*, 2654–2662.
18. Luo, Y.; Hong, P.; Su, R.; Xue, K. Resource Allocation for Energy Harvesting-Powered D2D Communication Underlaying Cellular Networks. *IEEE Trans. Veh. Technol.* **2017**, *66*, 10486–10498. [CrossRef]
19. Liu, M.; Song, T.; Gui, G. Deep cognitive perspective: Resource allocation for noma-based heterogeneous IoT with imperfect SIC. *IEEE Internet Things J.* **2019**, *6*, 2885–2894. [CrossRef]
20. Wang, Y.; Yang, J.; Liu, M.; Gui, G. LightAMC: Lightweight Automatic Modulation Classification via Deep Learning and Compressive Sensing. *IEEE Trans. Veh. Technol.* **2020**, *69*, 3491–3495. [CrossRef]
21. Carreira-Perpiñán, M.A.; Idelbayev, Y. 'Learning-Compression' Algorithms for Neural Net Pruning. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 8532–8541. [CrossRef]
22. Lee, N.; Ajanthan, T.; Torr, P.H. SNIP: Single-shot network pruning based on connection sensitivity. *arXiv* **2018**, arXiv:1810.02340.
23. Han, S.; Pool, J.; Tran, J.; Dally, W.J. Learning both weights and connections for efficient neural networks. In Proceedings of the Advances in Neural Information Processing Systems, Montreal, QC, Canada, 7–12 December 2015; pp. 1135–1143.
24. He, Y.; Zhang, X.; Sun, J. Channel pruning for accelerating very deep neural networks. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 22–29 October 2017; pp. 1389–1397.
25. Li, H.; Samet, H.; Kadav, A.; Durdanovic, I.; Graf, H.P. Pruning filters for efficient convnets. In Proceedings of the 5th International Conference on Learning Representations (ICLR 2017)—Conference Track Proceedings, Toulon, France, 24–26 April 2017; pp. 1–13.
26. Frankle, J.; Carbin, M. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In Proceedings of the 7th International Conference on Learning Representations (ICLR 2019), New Orleans, LA, USA, 6–9 May 2019; pp. 1–42 .
27. Louizos, C.; Welling, M.; Kingma, D.P. Learning sparse neural networks through L0 regularization. In Proceedings of the 6th International Conference on Learning Representations (ICLR 2018)—Conference Track Proceedings, Vancouver, BC, Canada, 30 April–3 May 2018; pp. 1–13.
28. Dettmers, T.; Zettlemoyer, L. Sparse Networks from Scratch: Faster Training without Losing Performance. *arXiv* **2019**, arXiv:1907.04840.
29. Elman, J.L. Finding structure in time. *Cogn. Sci.* **1990**, *14*, 179–211. [CrossRef]
30. Werbos, P.J. Backpropagation Through Time: What It Does and How to Do It. *Proc. IEEE* **1990**, *78*, 1550–1560. [CrossRef]
31. Stanley, R.E.; Taraza, D. Bearing characteristic parameters to estimate the optimum counterweight mass of a 6-cylinder in-line engine. *Am. Soc. Mech. Eng. Intern. Combust. Engine Div. (Publ.) ICE* **2001**, *36*, 123–135. [CrossRef]

32. Chung, J.; Gulcehre, C.; Cho, K.; Bengio, Y. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. *arXiv* **2014**, arXiv:1412.3555.
33. Reed, R.; Member, S. Pruning Algorithms-A survey. *IEEE Trans. Neural Netw.* **1993**, *4*, 740–747. [CrossRef] [PubMed]
34. Dai, X.; Yin, H.; Jha, N.K. NeST: A Neural Network Synthesis Tool Based on a Grow-and-Prune Paradigm. *IEEE Trans. Comput.* **2019**, *68*, 1487–1497.
35. Glorot, X.; Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. *J. Mach. Learn. Res.* **2010**, *9*, 249–256.
36. Kingma, D.P.; Ba, J.L. Adam: A method for stochastic optimization. In Proceedings of the 3rd International Conference on Learning Representations (ICLR 2015)—Conference Track Proceedings, San Diego, CA, USA, 7–9 May 2015; pp. 1–15.
37. Abadi, M.; Agarwal, A.; Barham, P.; Brevdo, E.; Chen, Z.; Citro, C.; Corrado, G.S.; Davis, A.; Dean, J.; Devin, M.; et al. TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. *arXiv* **2016**, arXiv:1603.04467.
38. LeCun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. Gradient-based learning applied to document recognition. *Proc. IEEE* **1998**, *86*, 2278–2323. [CrossRef]
39. O'Shea, T.J.; Corgan, J.; Clancy, T.C. Convolutional radio modulation recognition networks. *Commun. Comput. Inf. Sci.* **2016**, *629*, 213–226.
40. Ma, H.; Xu, G.; Meng, H.; Wang, M.; Yang, S.; Wu, R.; Wang, W. Cross model deep learning scheme for automatic modulation classification. *IEEE Access* **2020**, *8*, 78923–78931. [CrossRef]
41. Tekbiyik, K.; Ekti, A.R.; Gorcin, A.; Kurt, G.K.; Kececi, C. Robust and Fast Automatic Modulation Classification with CNN under Multipath Fading Channels. In Proceedings of the IEEE Vehicular Technology Conference, Antwerp, Belgium, 25–28 May 2020;
42. Mossad, O.S.; Elnainay, M.; Torki, M. Deep convolutional neural network with multi-task learning scheme for modulations recognition. In Proceedings of the 2019 15th International Wireless Communications and Mobile Computing Conference (IWCMC 2019), Tangier, Morocco, 24–28 June 2019; pp. 1644–1649. [CrossRef]
43. O'Shea, T.; Hoydis, J. An Introduction to Deep Learning for the Physical Layer. *IEEE Trans. Cogn. Commun. Netw.* **2017**, *3*, 563–575.