Effective WiTech Identification Using Deep Transfer Learning with SNR as an Additional Feature

Sachin Nayak[1(✉)], Amitesh Singh Sisodia[2], and Subrahamanya Swamy Peruru[2]

[1] University of Washington, Seattle, WA 98195, USA
sachinn@uw.edu
[2] Indian Institute of Technology Kanpur, Kanpur 208016, Uttar Pradesh, India
{amiteshs,swamyp}@iitk.ac.in

Abstract. Efficient sensing of wireless technology is critical in today's congested wireless ecosystem for effective utilization of limited resources. This paper presents a novel approach to wireless technology identification using deep transfer learning techniques, which are known to outperform conventional methods. A thorough study is required to understand the format of the data which is best suited for the task of technology identification, since wireless technology data is available in multiple formats like time-domain and frequency-domain representation. More importantly, which neural network architecture works best for each of these representations is to be studied. In this work, we show that fully connected neural networks and convolutional neural networks work best for classifying frequency-domain data and long short-term memory networks for time-domain data. Further, wireless signals with different signal-to-noise ratios (SNRs) may require a different strategy for efficient classification. In particular, a model that works well with signals having a high SNR may not perform well on signals that have low SNR. In this work, we study how the concept of transfer learning can be leveraged to design neural networks that work across different SNRs. In particular, we study questions like whether a neural network pre-trained on high-SNR data can improve the performance on low-SNR data? Our experimental results show that leveraging transfer learning can give additional gains of 4 to 20% in accuracy, depending on the SNR of the signal.

Keywords: Deep learning · Wireless technology identification · Signal to noise ratio · CNNs · LSTMs · FCNs

1 Introduction

In today's world, the spectrum is congested due to the presence of many devices communicating simultaneously [15]. This happens at both public locations like

Supported by University of Washington and the Indian Institute of Technology, Kanpur.

D. Jiang and H. Song (Eds.): SIMUtools 2021, LNICST 424, pp. 375–389, 2022.
https://doi.org/10.1007/978-3-030-97124-3_28

metro stations, streets and industrial locations like factories. Spectrum is sensed by both user devices such as mobile phones & laptops and industry devices, as well as mobile network operators (MNOs) at base stations that communicate to them. Mobile phones sense spectrum as belonging to 2G/3G/4G/5G on the setup as part of a process called initial access and are always on the lookout for Wi-Fi and Bluetooth channels. Wi-Fi devices and industrial devices sense signals in the unlicensed spectrum in order to communicate in the unused white spaces in between occupied spectrum and maximize spectrum usage. MNOs at base stations sense the licensed spectrum to allocate the spectrum with the least interference to its users, as well as to dynamically share spectrum with each other using a switching process.

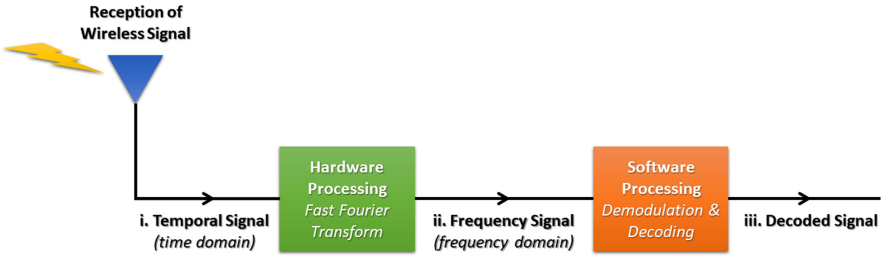

Fig. 1. Down link data processing after signal reception

Dynamic spectrum sharing is another application envisioned by the new 5G standard that requires sensing and switching spectrum [6,16]. In the process of mobile network operators switching from 4G to standalone 5G networks, 5G networks should be operated in non-standalone mode over existing 4G networks over the same frequency band to meet the requirements of spectrum sensing. Rapid spectrum sensing for demand-driven spectrum allocation between 4G and 5G is an essential part of this rollout process [5].

Spectrum sensing could be performed at various stages of the down link data processing, as shown in Fig. 1. Radio signal sensing for technology identification can be done either at acquisition as a temporal signal, after hardware processing as a frequency domain signal, or after software processing as the decoded signal. This is done conventionally at the end of the pipeline by decoding the signal but could be sped up significantly using deep learning algorithms, saving a large amount of energy. Another important aspect of sensing radio signals is that they appear at different signal-to-noise ratios (SNRs) based on the noise level and device density. A model designed for a particular SNR may not work for a different SNR. Hence, SNR is an important feature to consider while designing a model for wireless technology identification.

To meet the need for spectrum sensing for various applications, this paper describes an efficient and effective method to sense spectrum using deep learning. We focus on the unlicensed spectrum consisting of Wi-Fi, Bluetooth, and Zigbee signals since real signals are easy to capture for those technologies.

1.1 Related Work

Apart from detecting the wireless technology from the decoded signal as in Fig. 1, conventional methods for detecting wireless signals include energy detection [2] and standard machine learning [11] approaches. Most classification tasks focus on data for modulation schemes [8,12] rather than complex wireless technology standards like 4G, Wi-Fi, Bluetooth etc. Most data sets for classification purposes contain data obtained by simulations in MATLAB [13] or GNU Radio [9,17] due to the unavailability of real wireless signal data. Whenever real data was used or experimental studies were performed for data of wireless standards, the focus is one detecting one specific type of signal as in the following applications: Wi-Fi signal detection for LTE-WiFi Coexistence [4], detection of Bluetooth signals for frequency hopping [1] and detection of radar signals [7]. To the best of our knowledge, a combined study involving joint sensing of different wireless technologies by a single neural network has not been attempted in the past. Moreover, SNR is never considered as an input to any of the existing spectrum sensing algorithms. Hence, this study is unique as we trained a single neural network to sense a variety of wireless technology signals as well as adapting it across SNRs.

1.2 Our Contributions

In this work, we study the application of deep neural networks for wireless technology classification. We make the below contributions.

Matching Neural Networks to Wireless Signal Formats. Wireless technology data is available in both time domain and frequency domain, and can be represented in a variety of formats such as real & imaginary, phase & linear magnitude etc., for the same snapshot. The complexity and the accuracy of the classification task critically depends on the particular format of the signal and the architecture of the neural network that we choose, and therefore a thorough study is required to understand these dependencies. To that end, we identify the neural network architecture that works best for each format of wireless signals by studying their various characteristics. In particular, we observe that LSTMs work best for time domain data, FCNs & CNNs for frequency domain data. We also notice that the depth of the neural networks used for obtaining good accuracy on wireless signals is limited to 5–6 layers.

Using the Additional Feature of SNR in Wireless Signals. Wireless signals can appear at different SNRs based on the noise level and device congestion. We investigate whether the knowledge of this detected SNR of a signal can be effectively utilized to improve the performance. To that end, we propose a transfer learning based approach that improves the classification accuracy at certain SNRs, for which the neural network was performing poorly. In particular, we observe that transfer learning results in gains of 4 to 5% in accuracy for SNRs

above −12 dB, and 20% improvement for SNRs below −12 dB. This shows the adaptivity of neural networks for wireless technology classification across SNRs.

1.3 Organization

The rest of the paper is organized as follows. We start with Sect. 2 on understanding the nature of wireless technology data. Section 3 discusses our process of selecting neural network architectures for different wireless signal formats. Section 4 discusses simulation results across specific SNRs and the use of transfer learning to improve performance. Section 5 highlights the unique insights derived from our simulation study for the purpose of applying neural networks to wireless signals sensing, followed by a short conclusion. The appendix describes the experimental environment used for the simulation study in detail.

2 Nature of Wireless Technology Data

Wireless technology data is available as time domain representation and frequency domain representation for the same snapshot. Conversion to the frequency domain signal requires additional processing at the hardware side and hence is more time-consuming. In addition, the wireless signal in time/frequency domain can be represented as real, imaginary, phase, linear magnitude, decibel magnitude data series or a combination of them, so the best format needs to be selected for classification purposes. The SNR is a time varying feature in wireless technology data that provides an additional degree of freedom.

This section starts with a short description of the dataset, followed by the possible format of wireless signal data and finally a short section on the unique feature of SNR in the wireless signal classification.

2.1 Description of Dataset Used

We focus on radio signal classification of real wireless signal data for the unlicensed spectrum belonging to the Wi-Fi, Bluetooth, and Zigbee standards using the 'owl/interference' dataset [14] from the CRAWDAD database [3] that was established to share data sets collected from real networks. The 'owl/interference' dataset contains traces of IEEE 802.11b/g (Wi-Fi), IEEE 802.15.4 (Zigbee) and Bluetooth packet transmission with varying SNRs in the baseband as shown in Fig. 2. Additionally, different frequency offsets were added in the baseband to reflect different channels of the wireless technologies. The data is in the form of 128 sample IQ snapshots. 15 labels (Bluetooth CH20-29, Wi-Fi CH3-5 and ZigBee CH15-16) and 21 SNRs (−20 dB, −18 dB, . . . 0 dB, 2 dB, 4dB, . . . 20 dB) are possible as indicated by Fig. 2 and Fig. 4 respectively. The dataset is equally distributed in terms of labels and SNRs with 715 snapshots for each label and SNR combination.

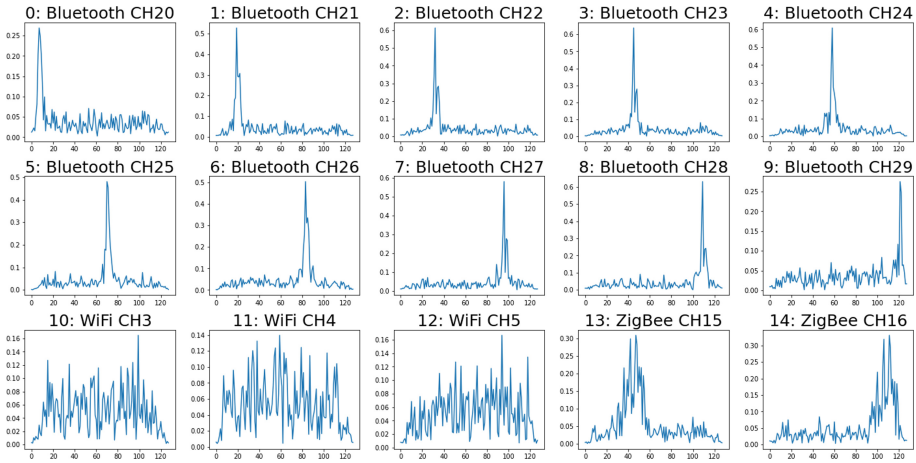

Fig. 2. FFT magnitude snapshots for each label (128 samples across X-axis at 0 dB SNR)

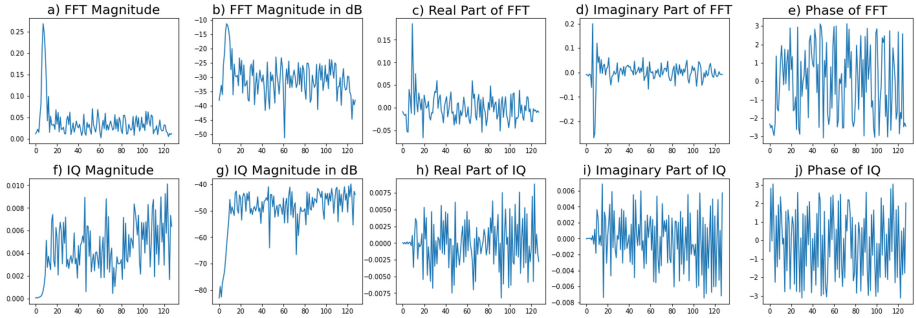

Fig. 3. 10 possible representations for the same wireless signal (128 samples across X-axis for snapshot of Bluetooth CH20 at 0 dB SNR)

2.2 Format of Wireless Signal Data

The time domain IQ representation of a signal can be converted to its frequency domain representation by performing an FFT. Both the time and frequency domain representations are complex vectors with imaginary and real parts that can be converted to phase/magnitude representations with the appropriate transforms. Thus, wireless signal data can be represented by 10 different data series for the same snapshot, as enumerated in Fig. 3.

A combination of these can also be used for any classification purposes, especially with CNNs and LSTMs that can extract features from multidimensional inputs. Popular combinations include the following: i. FFT IQ data in 2D form and ii. IQ Time Series in 2D form.

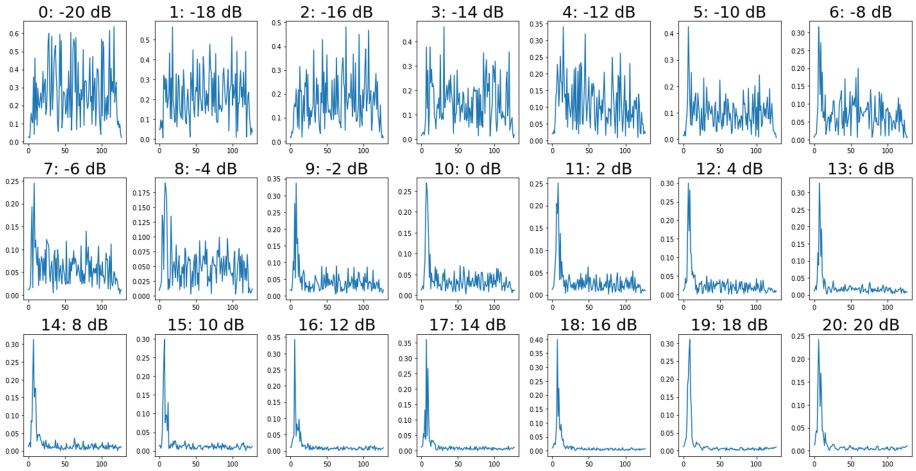

Fig. 4. Bin features become across SNR for the same frequency domain signal (128 samples across X-axis for snapshot of Bluetooth CH20)

2.3 SNR as an Additional Feature

SNR is an additional degree of freedom in wireless signals. This is because the SNR varies both as the hardware device moves spatially and temporally. As shown in Fig. 4, higher SNRs are easily classifiable having sharp features but low SNRs do not have sharp features that can be applied across SNRs. In the frequency domain, these features appears across frequency bins whereas in the time domain, they appear as time bursts. This fact plays an important part in understanding transfer learning to adapt neural network weights across SNRs.

This section discussed the real wireless signal dataset in use and its various representations, and finally the important additional dimension of SNR in wireless signals. The next section details the process of selecting neural network architectures for different wireless signal formats through a systematic simulation methodology.

3 Selecting Neural Network Architectures for Wireless Data Formats

The type of neural network that would suit each data type varies from the ones available in literature. We begin our simulation study by understanding the various formats of wireless signal data in relation to neural networks by classifying them with a simple baseline FCN. The details of the simulation environment used is described in the Appendix. Based on the results obtained regarding the nature of wireless data series, we expand into multi-layered FCNs, LSTMs and then CNNs.

Table 1. Baseline FCN structure

Layer type	Input size	Parameters	Activation function
Fully connected layer	128	15 neurons	SoftMax

3.1 Types of Neural Networks

Neural networks are used to learn complex non-linear functions from inputs to outputs for performing powerful tasks like classification or regression. The neural network architecture that would work best for each wireless signal format needs to be matched from the popular types of long-short term memory (LSTM), convolutional neural network (CNN) and fully connected network (FCN). LSTMs are well-suited for classifying, processing and making predictions based on time series data, since there can be lags of unknown duration between important events in a time series. FCNs would work well for frequency domain representations, as the data is compartmentalized into bins. CNNs are expected to work the best with IQ signal data series in the frequency domain, as they expect 2 dimensional inputs. CNN have been used rigorously for image classification tasks and are being recently extended to the wireless signal domain [10].

The complexity in terms of number of layers and neurons needs to determined as well. The higher the complexity of the neural network, the higher the accuracy, but the time required for classification also increases. A balance of these needs to be achieved to ensure the practical applicability of the neural networks that are trained.

3.2 Selecting Neural Network Architectures for Wireless Data Formats

We identify which format of data works with FCNs best by training the baseline FCN in Table 1 to classify the data formats mentioned in Sect. 2.3. The best classification score is for the 'FFT Magnitude' since most wireless technology features are evidenced by the frequency bins. Since 'FFT Magnitude' works best, we continue our exploration with that format for FCNs by varying the number of hidden layers & neurons and training to obtain the results in Table 3 and the final 2-layer FCN in Table 4.

Moving down the rows of Table 2, we observe that the real and imaginary parts of the FFT lead to very low accuracies of $\approx 20\%$ implying they don't contain sufficient information for classification by themselves. The accuracy for the phase part of FFT is almost 1/15 (remember our dataset has 15 labels) implying that it is uniformly distributed over labels and is least useful for classification out of the 5 FFT data series. We can infer that the information in a frequency domain signal is mainly represented by its magnitude and not its phase.

Looking at the second half of Table 2 containing the time domain data series representations, we notice that the accuracies are extremely low. These are better classified with LSTMs and not FCNs as LSTMs can model temporal correlations

Table 2. Accuracies across wireless signal data series for baseline FCN

Data Type	Accuracy (%)	Data type	Accuracy (%)
FFT Magnitude in dB	85.90	IQ Magnitude in dB	9.85
FFT Magnitude	88.90	IQ Magnitude	7.63
Real Part of FFT	20.58	Real Part of IQ	16.48
Imaginary Part of FFT	20.20	Imaginary Part of IQ	16.04
Phase Part of FFT	7.78	Phase Part of IQ	14.31

Table 3. Accuracies across different FCNs for 'FFT Magnitude' data series

Model	Accuracy (in %)
Baseline Model (No hidden layer but trained for 500 epochs)	85.90
One Hidden Layer with 60 neurons	88.53
One Hidden Layer with 80 neurons	88.90
Two Hidden Layers with 80 & 48 neurons	88.99
One Hidden Layer with 80 neurons, dropout = 0.2	88.90
One Hidden Layer with 80 neurons, dropout = 0.2	88.90
Two Hidden Layers with 80 & 48 neurons, dropout = 0.2 on first layer	89.12
Two Hidden Layers with 80 & 48 neurons, dropout = 0.2 on second layer	88.64
Two Hidden Layers with 80 & 48 neurons, dropout = 0.2 on both layers	89.20

Table 4. Proposed FCN structure

Layer type	Input size	Parameters	Activation function
Fully connected layer	128	80 neurons, dropout = 0.2	LeakyReLU
Fully connected layer	80	48 neurons, dropout = 0.2	LeakyReLU
Fully connected layer	48	15 neurons, dropout = 0.2	SoftMax

better. This also indicates that when a 2D convolutional neural network is to be used, the FFT data in the IQ form or phase/magnitude form would work best over IQ time series data as CNNs are not be able to capture temporal relations well. Another important observation we make is that now the magnitude part of the IQ data (as compared to the phase part) is uniformly distributed, implying that the information in a time domain signal is mostly represented by its phase and not magnitude.

Table 5. Proposed LSTM model

Layer type	Input size	Parameters	Activation function
Unidirectional LSTM Layer	128×2	256 neurons, dropout = 0.6	Tanh
Unidirectional LSTM Layer	128×256	256 neurons, dropout = 0.6	Tanh
Fully connected layer	256	15 neurons	SoftMax

Table 6. Structure of CNN2 [10]

Layer type	Input size	Parameters	Activation function
Convolutional layer	2×128	256 filters, filter size 1×3 dropout = 0.2	ReLU
Convolutional layer	$256 \times 2 \times 128$	80 filters, filter size 2×3 dropout = 0.6	ReLU
Fully connected layer	10240×1	256 neurons, dropout = 0.6	ReLU
Fully connected layer	256×1	15 neurons	SoftMax

Table 7. Performance Comparison for LSTM, CNN, and FCN models for different SNR scenarios [P: Precision; R: Recall; F1: F1-Score; High SNR: ≥0 dB; Low SNR: <0 dB]

Model	All SNR			High SNR			Low SNR		
	P	R	F1	P	R	F1	P	R	F1
LSTM	0.96	0.96	0.96	0.96	0.92	0.94	0.83	0.83	0.81
CNN	0.96	0.96	0.96	0.99	0.97	0.98	0.84	0.86	0.84
FCN	0.94	0.94	0.94	0.95	0.94	0.94	0.88	0.87	0.87

On experimenting with LSTMs of different lengths for our dataset, we arrive at the LSTM in Table 5 with 2 layers. The input to this LSTM is the IQ time series with both the real and imaginary components. In addition, we just choose CNN2, reproduced in Table 6, that was shown to give very high accuracies for modulation recognition [10] with input as the FFT IQ data format. On fully training these three models for the entire dataset for High (≥ 0 dB) and Low SNR (< 0 dB) splits of the dataset, we summarize the results as in Table 7. We note that the CNN and LSTM models perform slightly better than the FCN model due to the larger size of their inputs, enabling them to use all the information about the snapshot. Note that for each neural network type, i.e., LSTM, CNN and FCN, a different model (with different weights) is learned for each of the 3 splits of the dataset, i.e., i. All SNR, ii. Low SNR (< 0 dB) and iii. High SNR (≥ 0 dB), resulting in 3 models for each type.

The results from Table 7 shows that accuracies for the low SNR split differ from that of high SNR split by over 10% implying that SNR has an influence on wireless signal classification. For example, the accuracy changes from 83% for Low SNR to 96% for All SNR. The next section showcases simulation results for each SNR to understand this more deeply, and then elucidates the innovation of applying transfer learning for wireless signal classification.

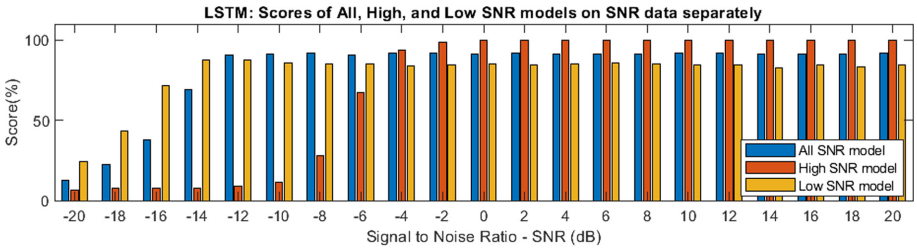

Fig. 5. Accuracies across SNR for LSTM models trained with different splits

4 Simulation Results on Specific SNRs and Use of Transfer Learning

In depth investigations are performed at each SNR specifically to truly understand the nature of wireless signal data and the adaptability of neural networks across this new dimension of wireless signals. This results in an understanding that transfer learning across SNRs can give large gains, as the accuracies across SNRs vary by large margins. Hence, our last subsection showcases the utility of transfer learning to adapt neural network weights across SNRs.

4.1 Investigating the Performance on Specific SNRs

SNR has a significant impact on the performance of the deep learning models, as models trained at low SNRs would not work as well as those trained at high SNRs. Moreover, a given trained deep learning model is more likely to perform better for high SNR inputs over low SNR inputs. To further analyze the variability of SNR influence on models trained with the three different splits: i. All SNR, ii. High SNR and iii. Low SNR, we visualize the accuracies for each SNR separately for LSTM as in Fig. 5 and make the below observations.

- As expected, the accuracies at lower SNR is lower with the values starting to drop beginning with 0 dB, i.e., for negative values. There is a drastic drop in accuracy at -12 dB.
- Models trained on the low SNR split seem to work reasonably at the higher SNRs, but models trained on the high SNR split work poorly at low SNRs and break down below -12 dB.
- For the lowest SNRs of ≤ -12 dB, the models trained on the low SNR split seem to work better than the models trained over all SNRs implying the need to train models specifically for these lowest SNRs.

The reason behind these observations is that deep learning models can learn more generalized patterns at low SNRs but only specific patterns from high SNR data. The fact that specific models need to be trained to detect lowest SNRs of ≤ -12 dB means that these SNRs do not contain enough information, and most of the information in the lower SNR split comes from the range of -8 dB to

0 dB. This calls for the need to learn specific weights or models to improve the performance of neural networks for specific SNRs which we do through the process of transfer learning described in the next section.

4.2 Leveraging Transfer Learning across the SNR Feature

Motivated by the need to improve deep learning performance for specific SNRs and inspired by the fact that neural networks pretrained at low SNRs can learn generalized features, we turn to transfer learning. Transfer learning is a machine learning method where a model learned for a given task is reused for a second task, using it as the starting or base model. It enables one to reuse or transfer information from previously learned tasks for the learning of new tasks.

We use transfer learning to improve the performance of deep learning models in the previous section in relation to specific SNRs. For a given neural network type, say LSTM, we have 3 models trained for the 3 splits of All SNR, Low SNR and High SNR respectively. We perform transfer learning to each SNR for each of the 3 models for the given network type. We perform the same for all 3 network types to result in the accuracies illustrated by Fig. 6. The results for SNR >-2 dB are not shown, as their accuracies without transfer learning are almost 100%. We make the below observations regarding applying transfer learning to improve the performance of our models for specific SNRs.

- Transfer learning improves accuracies across all the SNRs in Fig. 6 for all pretrained models by modest values of 4 to 5% showing that transfer learning is effective in improving the performance of deep learning for wireless signals.
- Transfer learning seems to be most effective with LSTMs followed by CNNs and FCNs.
- The gains increase to 10 to 20% for the lowest SNRs for all the three models. In particular, when pretrained LSTMs are used, additional gains of 10% are witnessed.
- In general, neural networks pre-trained on the low SNR split give the best performance with transfer learning.
- However, FCNs are different in the manner that pretraining on the all SNR split gives larger gains over the low SNR split, showing that LSTM and CNN can learn generalized patterns due to their more complex structure.

Transfer learning works well with wireless signals for specific SNRs because any given wireless signal (say Bluetooth CH20) has the same features across SNRs. For example, we can look back at Fig. 4 to notice that the frequency peak appears at the same location across SNRs. Pretraining on the low SNR splits gives the best gains as neural networks can learn the most generalized features from them due to the presence of noise, which makes it robust to overfitting. Additional experiments were tried regarding transfer learning from the low SNR split to the high SNR split (or between any two splits) rather than specific SNRs, but they did not show good results, showing that transfer learning works only for a narrow SNR range.

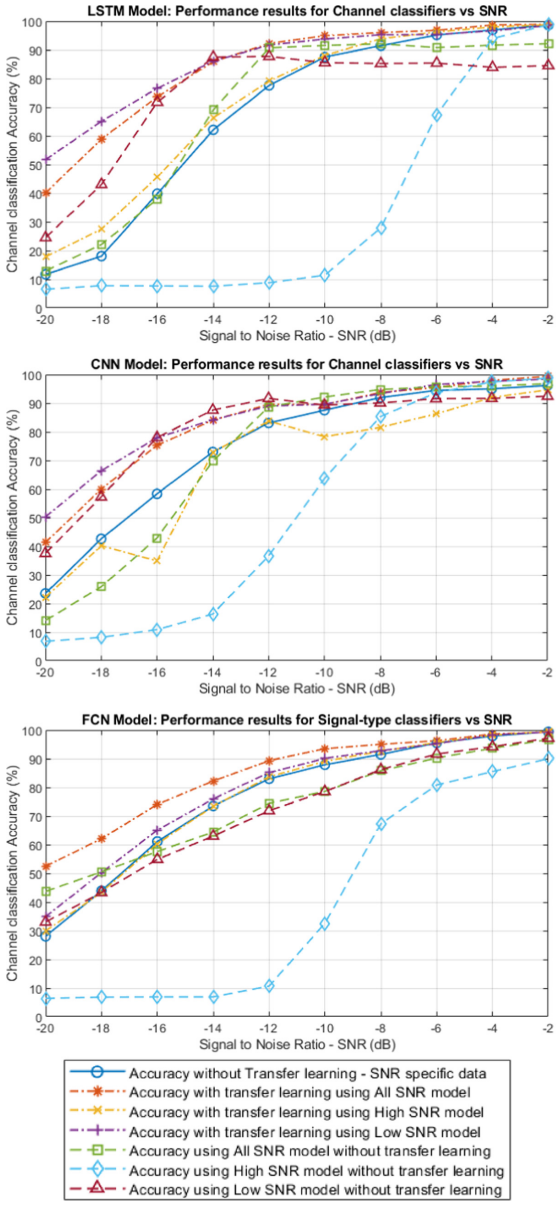

Fig. 6. Performance of Models on SNR specific data with and without transfer learning with Accuracy as performance parameter [Detailed information is present in the Legend]

This subsection emphasized the innovation of transfer learning as applied to wireless signal classification using deep learning. The next section summarizes all such insights obtained along this study for future research purposes.

5 Summary of Insights

This section discusses the unique insights derived from our simulation study across different types of neural networks, data formats and SNRs for the purpose of deployment of neural networks for wireless signal sensing, whether in consumer devices or industrial devices that control the network themselves like base stations. We make the below findings.

Information in Various Wireless Signal Formats. In wireless signals, the information in the frequency domain is represented mainly by the magnitude different from the phase in the time domain. The real and imaginary parts by themselves do not carry enough information and need to be input together for any deep learning applications. However, training neural networks solely using the 'FFT Magnitude' (as we did for the FCN we proposed) or 'IQ Phase' is less effective than using both the real and imaginary parts as input (as we did for the proposed LSTM and CNN2).

Neural Networks Selected for Various Wireless Signal Formats. One requires LSTMs or other forms of RNNs to represent the temporal structure of time domain signals in order to classify them as compared to FCNs or CNNs for frequency domain signals. As indicated by Fig. 1, frequency domain signals are obtained at a later stage in the down link processing pipeline and would require higher latencies to process. The neural networks we ended up choosing after thorough experimentation had reasonable length of a couple of layers, implying that neural networks can be feasibly implemented for hardware applications.

Insights on Using SNR as an Additional Feature. Neural networks are able to learn more generalized patterns at low SNRs and specific patterns from higher SNRs. However, the lowest SNRs do not have enough information in them, implying the need to train neural networks specifically for them. When applied to a narrow SNR range, transfer learning improves the performance of neural networks. The gains obtained are in the range of 4 to 5%. The gains increase to $> 20\%$ for the lowest SNRs.

6 Conclusion

Application of deep learning to wireless technology identification is not well researched due to the unavailability of real wireless signal data. This study started with understanding the nature of real wireless signal data using deep learning to arrive at the point that using FCNs/CNNs for FFT data and LSTMs for IQ time-series data is suitable. Further investigation showed that the performance of the neural network varies across the SNR of the wireless signal in question, calling for the need to study this additional dimension of wireless signals. We propose a transfer learning based approach to improve the accuracies at

SNRs for which the models were performing poorly. For signals with poor SNRs (<-12 dB), the proposed transfer learning approach improves the accuracies by 20%. In general, for other SNRs, we observe gains of 4 to 5%. The gains in speed and accuracy would make deep learning a strong choice for wireless technology identification.

Appendix- Details of Training and Analysis

Implementation Details. The neural networks are implemented using the Keras framework, an open-source software library that provides a Python interface for building artificial neural networks. It has a simple and highly modular interface, which makes it easier to create complex neural network models. Colab, a Google's free notebook environment, is used for training the models using GPUs.

Training Details. The whole dataset is partitioned into training, validation, and test splits in [60:20:20] ratio The labels y_data is converted into One-hot vectors for classification using Keras library. The input X_data for each model is normalized to have a train_mean of 0 and a standard deviation of 1. This removes bias in the data for better training, improves the stability of the model, and speeds up the training process. All the models are trained for a maximum of 200 epochs and the batch_size is 1024 for all the models, unless mentioned. Categorical loss is used along with Adam optimizer. EarlyStopping Technique is used for checking the problem of overfitting using validation accuracy as the monitor (parameters: min_delta = 0.001, patience = [LSTM:15; CNN:60; FCN:60]).

Analysis Details. Test accuracy is used as the main indicator for model performance in this paper because it is the accuracy achieved by the model on unseen data and hence is the optimal choice for checking robustness for practical applications. Performance of a classification model is quantified in terms of the precision (Π), recall (Ψ) and F1-score performance metrics. The precision metric quantifies how many positive results are actually positive, the recall provides information on how much true positives are identified correctly as positive, and F1-score gives an overall measure for the accuracy of a classifier model since it is the harmonic average of precision and recall. These metrics are calculated as below.

$$\Pi = \frac{\xi}{\xi + \nu}, \Psi = \frac{\xi}{\xi + \mu}, F_1 - Score = 2 \times \frac{\Pi \times \Psi}{\Pi + \Psi},$$

where ξ, ν, and μ denote the numbers of true positive, false positive, and false negative, respectively.

References

1. Bitar, N., Muhammad, S., Refai, H.H.: Wireless technology identification using deep convolutional neural networks. In: 2017 IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC), pp. 1–6 (2017). https://doi.org/10.1109/PIMRC.2017.8292183

2. Cabric, D., Tkachenko, A., Brodersen, R.W.: Experimental study of spectrum sensing based on energy detection and network cooperation. In: Proceedings of the First International Workshop on Technology and Policy for Accessing Spectrum, TAPAS '06, p. 12-es. Association for Computing Machinery, New York (2006). https://doi.org/10.1145/1234388.1234400
3. CRAWDAD a community resource for archiving wireless data at dartmouth. https://crawdad.org/about.html. Accessed 13 June 2021
4. Dziedzic, A., Sathya, V., Rochman, M.I., Ghosh, M., Krishnan, S.: Machine learning enabled spectrum sharing in dense LTE-U/Wi-Fi coexistence scenarios. IEEE Open J. Veh. Technol. **1**, 173–189 (2020)
5. A new standard for dynamic spectrum sharing. https://www.ericsson.com/en/blog/2019/6/dynamic-spectrum-sharing-standardization. Accessed 15 June 2021
6. Kaltenberger, F., Knopp, R., Danneberg, M., Festag, A.: Experimental analysis and simulative validation of dynamic spectrum access for coexistence of 4g and future 5g systems. In: 2015 European Conference on Networks and Communications (EuCNC), pp. 497–501 (2015). https://doi.org/10.1109/EuCNC.2015.7194125
7. Lees, W.M., Wunderlich, A., Jeavons, P., Hale, P., Souryal, M.: Deep learning classification of 3.5-GHZ band spectrograms with applications to spectrum sensing. IEEE Trans. Cogn. Commun. Netw. **5**, 224–236 (2019)
8. Meng, F., Chen, P., Wu, L., Wang, X.: Automatic modulation classification: a deep learning enabled approach. IEEE Trans. Veh. Technol. **67**(11), 10760–10772 (2018). https://doi.org/10.1109/TVT.2018.2868698
9. O'Shea, T., West, N.: Radio machine learning dataset generation with gnu radio. In: Proceedings of the GNU Radio Conference, vol. 1, no. 1 (2016). https://pubs.gnuradio.org/index.php/grcon/article/view/11
10. O'Shea, T.J., Corgan, J.: Convolutional radio modulation recognition networks. CoRR (2016). http://arxiv.org/abs/1602.04105
11. O'Mahony, G.D., Harris, P.J., Murphy, C.C.: Detecting interference in wireless sensor network received samples: a machine learning approach. In: 2020 IEEE 6th World Forum on Internet of Things (WF-IoT), pp. 1–6 (2020). https://doi.org/10.1109/WF-IoT48130.2020.9221332
12. Peng, S., Jiang, H., Wang, H., Alwageed, H., Yao, Y.D.: Modulation classification using convolutional neural network based deep learning model. In: 2017 26th Wireless and Optical Communication Conference (WOCC), pp. 1–5 (2017). https://doi.org/10.1109/WOCC.2017.7929000
13. Riyaz, S., Sankhe, K., Ioannidis, S., Chowdhury, K.: Deep learning convolutional neural networks for radio identification. IEEE Commun. Mag. **56**(9), 146–152 (2018). https://doi.org/10.1109/MCOM.2018.1800153
14. Schmidt, M., Block, D., Meier, U.: CRAWDAD dataset owl/interference (v. 2019–02-12) (2019). https://crawdad.org/owl/interference/20190212
15. Taylor, G., Middleton, C., Fernando, X.: A question of scarcity: spectrum and Canada's urban core. J. Inf. Pol. **7**, 120–163 (2017). http://www.jstor.org/stable/10.5325/jinfopoli.7.2017.0120
16. Wan, L., Guo, Z., Chen, X.: Enabling efficient 5G NR and 4G LTE coexistence. IEEE Wirel. Commun. **26**, 6–8 (2019). https://doi.org/10.1109/MWC.2019.8641417
17. Ziegler, J.L., Arn, R.T., Chambers, W.: Modulation recognition with GNU Radio, Keras, and HackRF. In: 2017 IEEE International Symposium on Dynamic Spectrum Access Networks (DySPAN), pp. 1–3 (2017). https://doi.org/10.1109/DySPAN.2017.7920747