BacalhauNet: A Tiny CNN for Lightning-Fast Modulation Classification

José Rosa
School of Technology and Management
Polytechnic of Leiria
Leiria, Portugal
2190383@my.ipleiria.pt

Guilherme Carvalho
INESC TEC
Faculty of Engineering - University of Porto
Porto, Portugal
gcarvalho@fe.up.pt

Daniel Granhão
INESC TEC
Faculty of Engineering - University of Porto
Porto, Portugal
granhao@fe.up.pt

Tiago Gonçalves
INESC TEC
Faculty of Engineering - University of Porto
Porto, Portugal
tiago.f.goncalves@inesctec.pt

1 Introduction

The growing demand for wireless data is driving a need for improved radio efficiency, hence, being able to rapidly understand and label radio spectrum in an automatic manner will be of utmost importance to address several open problems such as spectrum interference monitoring, radio fault detection, dynamic spectrum access and opportunistic mesh networking. These challenges are under the scope of the concept of Automatic Modulation Classification (AMC), where the main goal is to monitor the radio-frequency spectrum and determine the different modulations [1, 2] to subsequently reach transmission decisions that transmit information more efficiently. The first approaches to AMC consisted of the handcraft of specialized feature extractors for specific signal types and properties, and the derivation of compact decision bounds from them using either analytically derived decision boundaries or statistical learned boundaries within low-dimensional feature spaces [1]. On the other hand, the growing success of Deep Learning (DL) is also playing a role in this field [3, 4, 5]. Deep learning algorithms have been achieving high predictive performances due to the increased access to massive datasets and computational power. These complex Artificial Neural Network (ANN) architectures usually have hard space, computation, and power requirements (e.g., these models may require several Graphics Processing Unit (GPU) devices to train and evaluate) [6]. On the other hand, if we aim to leverage the potential of deep learning for AMC in a real-world context, we must be able to achieve low latency and high throughput to accurately reflect the current status of the transmissions. Interestingly, the research and industry communities are joining efforts towards the transition of machine learning models into embedded systems, which may present several advantages: less dependency on the cloud, since there is no need to transfer large amounts of data to the cloud, thus allowing to economize on bandwidth and network resources; power efficiency, since many embedded systems are power-efficient and can operate for a long time without being charged, thus leading to a lower carbon footprint, and, consequently, much better sustainability. Besides, several studies suggest that while there is an inherent tension between complexity and predictive performance, there is no direct dependence between model complexity and good performance [7] (see Figure 1). The field of *model compression* plays a key role in the study of the practicality of the usage of such complex models (common Deep Neural Network (DNN) can involve hundreds of millions of operations and many megabytes of parameters) [8]: if large models are only needed for robustness during training, then significant compression of these models should be achievable, without impacting accuracy [6]. There are several strategies to achieve compact models: *knowledge distillation*, *low-rank factorization*, *pruning*, *quantization*. Besides, this can also be done in a co-designed fashion with hardware to provide the required system-level throughput [9, 10]. For the sake of comprehension, we decided to explain only the methods that we used in our proposal (pruning and quantization).

Pruning Pruning consists of the removal of the least important weights and/or activations of a neural network [11]. Pruning was first pursued due to the advantages that it can provide: better generalization, i.e., the likelihood of over-fitting the training data is diminished; less training data required; decreased times of training and inference. The pruning process might decrease the predictive performance of the ANN, hence, it requires the retraining or fine-tuning, iteratively, to regain its previous performance. Pruning methods can be roughly categorized as unstructured (i.e., removing individual parameters) or structured (e.g., removing complete layers). Usually, unstructured pruning methods generally allow a higher compression ratio as they remove the least important network connections without any restrictions, however, they make developing an efficient custom hardware architecture more difficult due to the irregular patterns in weights.

Quantization Quantization is the process of mapping values from a large set to values in a smaller set (i.e., the output contains a smaller range of values compared to the input without losing much information in the process). The quantization of neural network parameters and intermediate number representations (i.e., the activations) is a type of compression strategy that is usually applied to simultaneously reduce the model size, computation complexity, and memory access intensity. Besides its use for compression purposes, quantization is also very useful when targeting Field-Programmable Gate Array (FPGA) as they do not have native support for floating-point operations. Quantizing the floating-point values to a fixed-point representation is the traditional option as neural networks are known to be error-tolerant. In practice, quantization can be very straightforward to implement if performed at the end of training as a simple rounding process to the desired bit-width. However, this usually results in a significant performance reduction due to the loss associated with the quantization process. Quantization Aware Training (QAT) is a process that can be used to mitigate this loss. In QAT, training is modified such that the forward pass is performed at the desired precision while the backward pass retains always full precision, thus drastically reducing quantization loss [12, 13].

This report presents the methodologies and results achieved by our team "BacalhauNet" on the problem statement 7 (PS-007), "Hardware-Efficient Modulation Classification with RadioML", of the "ITU AI/ML in 5G Challenge", which challenged participants to design ANN with an awareness of both inference computation cost and accuracy to explore the landscape of compact models for AMC on the Deepsig RadioML 2018.01A dataset [1]. The main goal of this challenge was to design a neural network that achieves, at least, 56% accuracy on the dataset while minimizing its *inference cost*. The inference cost measures how computationally complex the forward phase of a neural network is. In this challenge, the inference cost score implementation was provided by the organizers. Submissions were then ranked according to their inference cost score, which compares the complexity of submitted solutions with the complexity of a provided baseline VGG-based model. Specifically, the score is given by Equation 1:

$$\text{Inference Cost Score} = \frac{\text{Submission Inference Cost}}{\text{Baseline Inference Cost}} \tag{1}$$

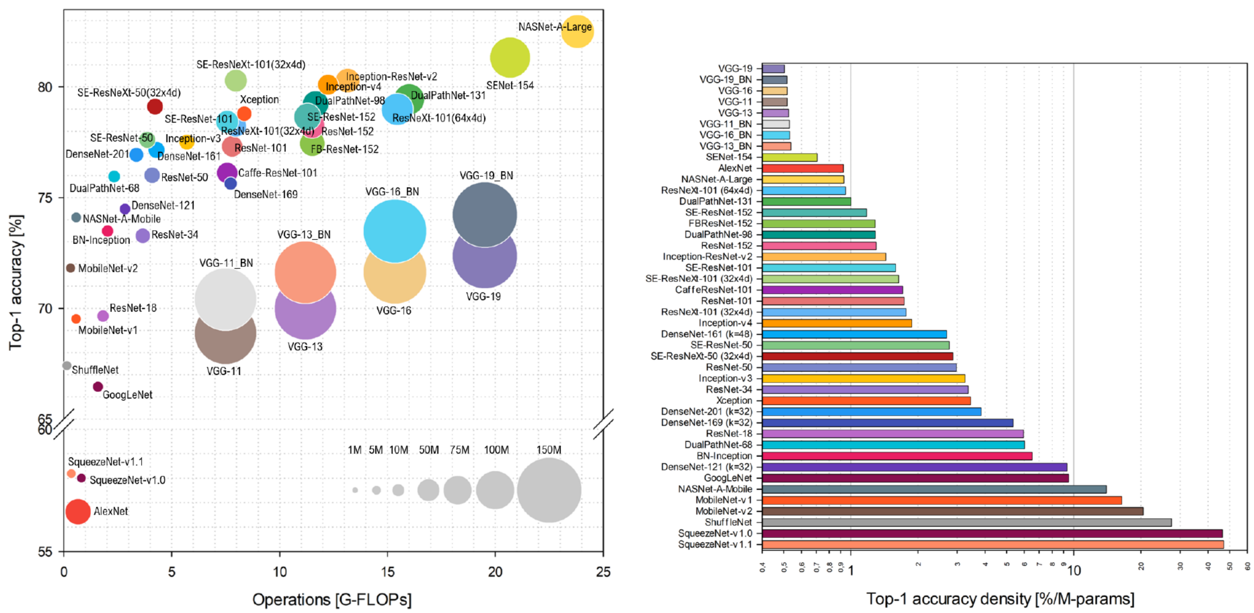
The remainder of the report is organized as follows: Section 2 presents our proposed model, BacalhauNet, and describes its training and hyper-parameter optimization methodologies; Section 3 presents and discusses the results obtained during the challenge; Section 4 concludes the report and provides possible lines of future work.

2 Methods

2.1 Initial experiments

Our initial approach was based on the MobileNetV3 [14]. MobileNetV3 is a low complexity Convolutional Neural Network (CNN) designed for the Central Processing Unit (CPU) of mobile phones. It is supported by the PyTorch [15] framework, so it was relatively simple to adapt the original implementation to allow the processing of 1D data. We trained a modified version of MobileNetV3-Small on the Deepsig RadioML 2018.01A dataset and we got an inference cost score (see Equation 1) of 30.42 and test accuracy of 61.3%. To fairly compare our version of the MobileNetV3-Small model and the baseline model we applied quantization to all weights and activations using a bit-width of 8 with the Brevitas [16] library. After the quantization, the test accuracy of the model dropped to 60.2% and the inference cost score was reduced to 7.25. The results achieved by our version of MobileNetV3-Small were worst than the baseline implementation provided by the challenge starting code. Therefore, we decided to implement a custom ANN that consists of a single depthwise separable convolution, with a kernel length of 5 and a stride of 2, followed by a pooling layer and a fully-connected layer. This simple model achieved 18% test accuracy in only 5 epochs and got an inference cost score of approximately 0.03, which means we need to improve its learning capacity. It is important to take into account that we started our experiments already using depthwise separable convolutions instead of standard convolutions since they are known for their reduced computational complexity [17].

[1] `https://www.deepsig.ai/datasets`

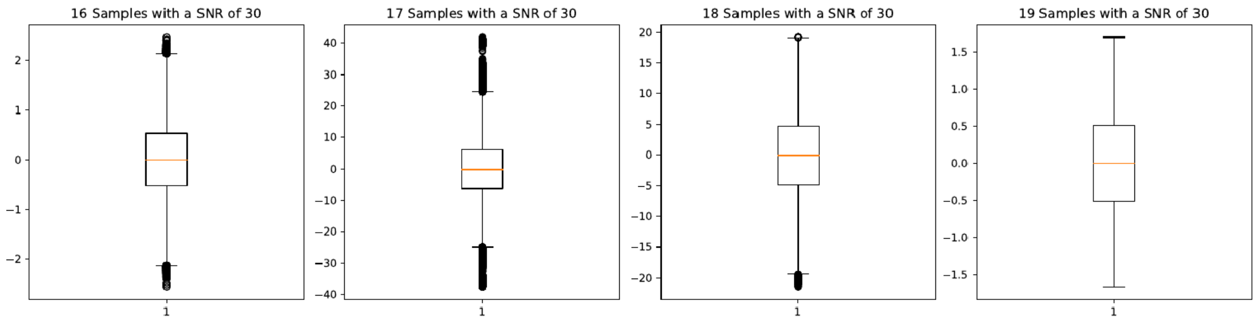


(a) (b)

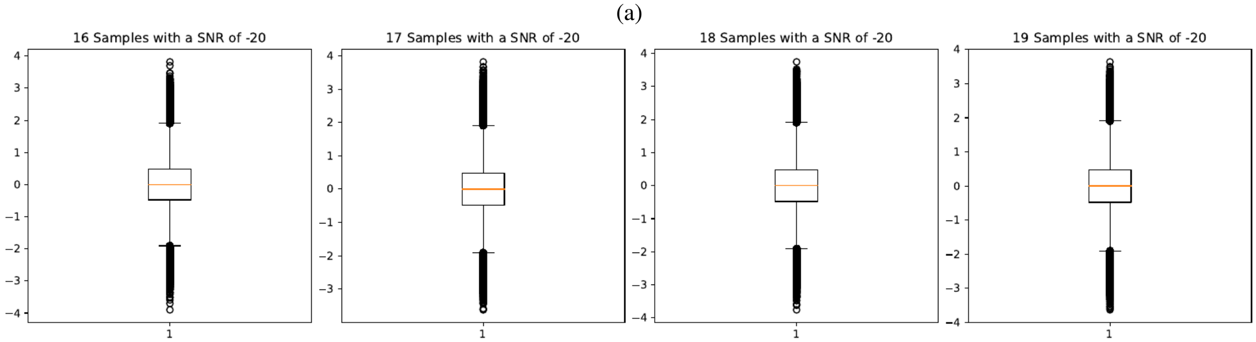
Figure 1: (a) Ball chart reporting the Top-1 accuracy vs. computational complexity: Top-1 accuracy using only the center crop versus Floating-Point OPerations (FLOPs) required for a single forward pass are reported. The size of each ball corresponds to the model complexity; (b) Top-1 accuracy density: The accuracy density measures how efficiently each model uses its parameters. Images from [7].

Inspired by the baseline implementation and based on the observations of the dataset samples per modulation we added a *hardtanh* activation layer as the first layer of this custom neural network. An attentive look to Figure 2a shows that the amplitude span of both I and Q channels of AM-SSB-WC and AM-SSB-SC modulations is bigger than the rest of the modulations for high Signal to Noise Ratio (SNR) levels. Our intuition is that clipping the AM-SSB-WC to a minimum and maximum value for both I and Q channels and clipping AM-SSB-SC I channel to a maximum value will still allow the model to perform classification while reducing the inter-class variance. We set the minimum and maximum *hardtanh* clipping values to -2 and 3 respectively. We can also notice that for lower SNR levels the noise impact is too high, destroying the majority of the information carried by the signal. In Figure 2b we can assess the impact of the noise for low SNR levels since the samples of all modulations are now contained in the range between -4 and 4. The destructive impact of the noise led us to train the neural network only with samples with an SNR level greater than or equal to -6 dB. The next step consisted of the fine-tuning of the kernel length of the depthwise convolutional layer. Inspired by the results described in [18], we tested large kernel lengths and we found out that the model achieved better accuracy until a certain threshold. Please note that the computational complexity linearly increases when increasing the kernel length. This exploratory analysis is illustrated in Figure 3. Based on this analysis, we decided to pick up a kernel length of 41 for further experiments. Regarding the stride, we started with a length of 2 since this allows us to downsample the feature maps, thus resulting in a reduction in computational complexity. We also used residual connections [19] in depthwise convolutional layers whenever the number of input and output channels is the same and the stride length is unitary. Our intuition is that these residual connections allow the processing of the features with different kernel lengths, and prevent the vanishing gradient problem.

At this point we had a strong prototype to work on, so we started to test multiple configurations of the network sequentially appending other depthwise convolutional layers to this prototype model. We found out that we needed a minimum of 4 stacked depthwise convolutional layers to reach the 56% accuracy threshold. We then started to perform fine adjustments of the parameters to find out the best combination. Figure 4a illustrates the parameters and metrics of the most efficient architectures and Figure 4b shows a plot comparing the accuracy against the inference cost score of some test scenarios. We picked the configuration depicted in Table 1 that achieved a 58.3% accuracy and a 1.158 inference cost score without any compression. At a later stage of our work, after quantization and pruning, we found out that if we added more layers, the network could be compressed further without compromising the accuracy and thus minimizing the inference cost score. This final architecture with a extra depthwise convolutional layer achieves 59.09% accuracy and a inference cost score of 1.416. Despite being more complex than the baseline network (i.e., the inference cost score is bigger than 1.0), this model is not compressed, making it an unfair comparison. If we apply the



(a)



(b)

Figure 2: (a) Samples distribution per class for an SNR level of 30dB; (b) Samples distribution per class for an SNR level of -20dB.

Table 1: Configuration of the prototype neural network.

Layer Type	Kernel Length	Residual Connection	Input Size
HardTanh	-	-	1024×2
DW Conv1D	27	-	1024×2
DW Conv1D	21	✓	512×24
DW Conv1D	15	-	512×24
DW Conv1D	9	✓	256×48
Global MaxPool1D	-	-	256×48
Fully Connected	-	-	1×48

same compression method of the baseline network to our model, we get approximately the same accuracy as its full precision counterpart while reducing the inference cost score to 0.146. This final architecture is called **BacalhauNet** to honor our country (Portugal) since it is known for the massive consumption of codfish (the English translation of the Portuguese name, “bacalhau”).

2.2 BacalhauNet

2.2.1 Architecture

BacalhauNet is strongly inspired in MobileNetv1 [20] and ResNet [19]. It uses depthwise separable convolutions instead of standard convolutions to reduce model complexity with a negligible impact on the model accuracy. To reduce the vanishing gradient problem and to allow the processing of the same features by different kernel sizes, a residual connection is used in the layers that share the same number of input and output channels and the with a stride equal to 1. The first layer is an *hardtanh* activation layer used to clip the samples from the dataset between -2 and 3 to reduce the inter-class variation. All the layers of the final architecture are described in Table 2. As mentioned before, the BacalhauNet configuration was found during an exhaustive sequential search for the best parameters for the first four convolutional layers. However, during pruning, we lost more accuracy than we tough so we experimented with making the model deeper, adding one more depthwise convolutional layer.

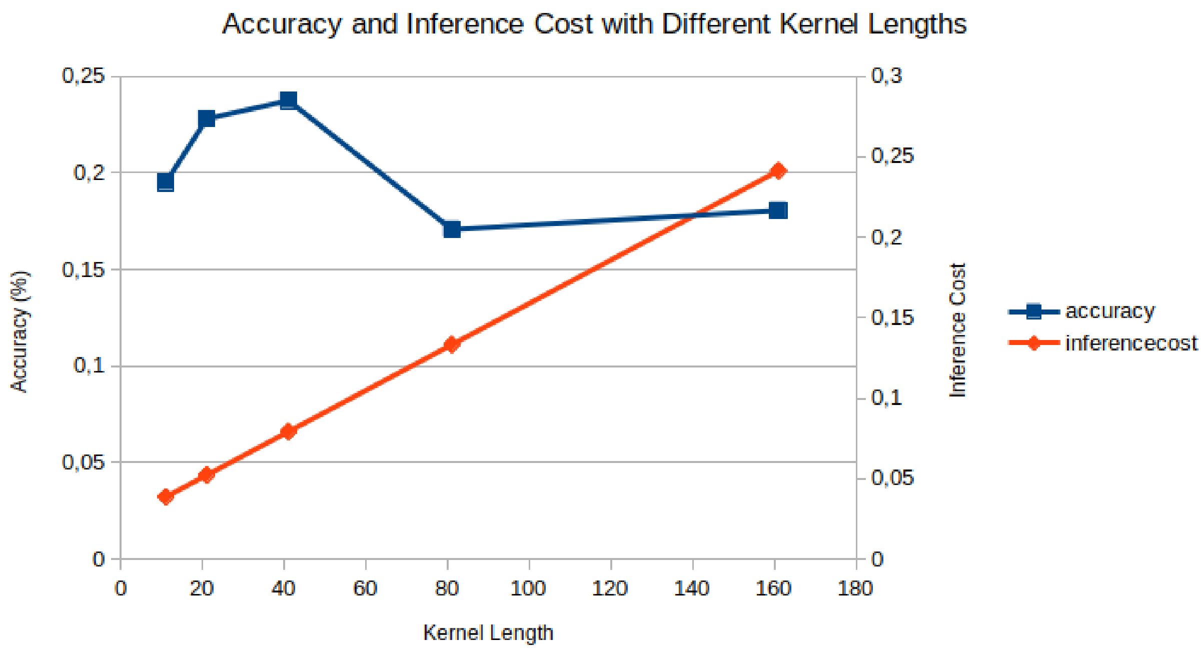
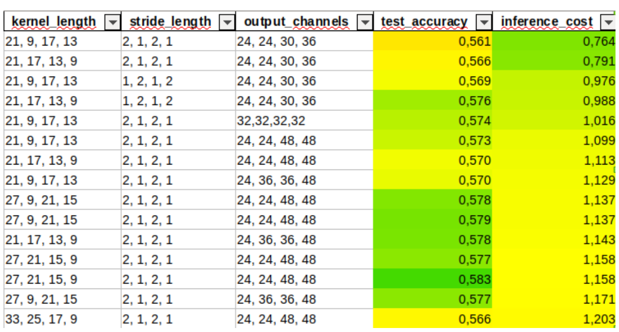
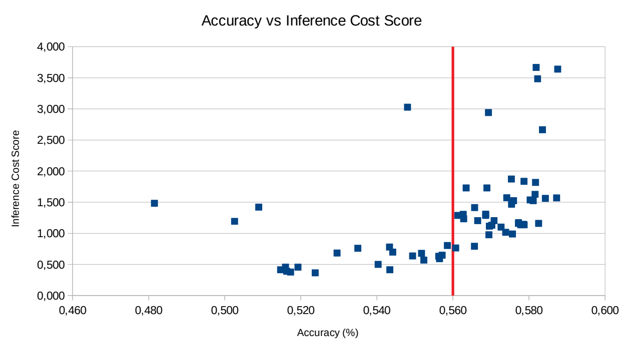


Figure 3: Impact of kernel length on accuracy and computational complexity (inference cost score).



kernel_length	stride_length	output_channels	test_accuracy	inference_cost
21, 9, 17, 13	2, 1, 2, 1	24, 24, 30, 36	0,561	0,764
21, 17, 13, 9	2, 1, 2, 1	24, 24, 30, 36	0,566	0,791
21, 9, 17, 13	1, 2, 1, 2	24, 24, 30, 36	0,569	0,976
21, 17, 13, 9	1, 2, 1, 2	24, 24, 30, 36	0,576	0,988
21, 9, 17, 13	2, 1, 2, 1	32,32,32,32	0,574	1,016
21, 9, 17, 13	2, 1, 2, 1	24, 24, 48, 48	0,573	1,099
21, 17, 13, 9	2, 1, 2, 1	24, 24, 48, 48	0,570	1,113
21, 9, 17, 13	2, 1, 2, 1	24, 36, 36, 48	0,570	1,129
27, 9, 21, 15	2, 1, 2, 1	24, 24, 48, 48	0,578	1,137
27, 9, 21, 15	2, 1, 2, 1	24, 24, 48, 48	0,579	1,137
21, 17, 13, 9	2, 1, 2, 1	24, 36, 36, 48	0,578	1,143
27, 21, 15, 9	2, 1, 2, 1	24, 24, 48, 48	0,577	1,158
27, 21, 15, 9	2, 1, 2, 1	24, 24, 48, 48	0,583	1,158
27, 9, 21, 15	2, 1, 2, 1	24, 36, 36, 48	0,577	1,171
33, 25, 17, 9	2, 1, 2, 1	24, 24, 48, 48	0,566	1,203

(a)



(b)

Figure 4: (a) Parameters and metrics of architectures that showed a good performance; (b) Accuracy vs inference cost score plot of other test cases.

Table 2: Configuration of the BacalhauNet architecture.

Layer Type	Kernel Length	Residual Connection	Input Size
HardTanh	-	-	1024×2
DW Conv1D	27	-	1024×2
DW Conv1D	21	✓	512×24
DW Conv1D	15	-	512×24
DW Conv1D	9	✓	256×48
DW Conv1D	9	-	256×48
Global MaxPool1D	-	-	128×48
Fully Connected	-	-	1×48

Table 3: Inference cost score and accuracy reached for BacalhauNet with different representations. Row in bold points to the better compromise between accuracy and inference cost score.

Representation Type	Test Accuracy	Inference Cost Score
Float	59.09%	1.4155
Quant - 8 bits	59.06%	0.1461
Quant - 7 bits	58.35%	0.1002
Quant - 6 bits	**58.67%**	**0.0781**
Quant - 5 bits	55.89%	0.0562

2.2.2 Network Compression: Pruning

In our initial experiments, we tried a structured pruning approach called *slimming* [21], but quickly concluded that the benefits of structured pruning, namely allowing more efficient hardware implementations, would not be captured by the challenge evaluation method while only handicapping the achievable compression. The inference cost formula takes pruning into account by ignoring every model parameter that is set to zero. It does not take into account the structure of parameters in any way, and, for that reason, we relied entirely on unstructured pruning.

A total of 3 pruning iterations were performed. More iterations could have been pursued, potentially leading to an even smaller model that achieved the minimum accuracy threshold, but due to time constraints, we had to stop this process to submit to the challenge on time. Each iteration was comprised not only of the prune itself but also a fine-tuning step in which the model is retrained to recover lost accuracy. Both the initial training before pruning and the retraining performed in fine-tuning steps were performed with a weight decay λ, thus promoting smaller weights [22]. The pruning was performed by setting and sticking to zero all weights with an absolute value less than ϵ. In each of the 3 pruning iterations, design space exploration of λ and ϵ was performed.

2.2.3 Network Compression: Quantization

To verify the impact that quantization would have in the proposed architecture, several iterations were trained with different bit-widths. In order to quantize the network we used quantized layers from Brevitas [16]. The quantized layers that were used are *QuantConv1d*, *QuantReLU*, *QuantHardTanh*, *QuantMaxPool1d* and *QuantLinear*. We fixed the input quantization to 8-bits and quantized both weights and activation from 8-bits down to 5-bits using Brevitas.

3 Results and Discussion

3.1 Pruning

Figure 5 presents the results from the design space exploration for each pruning iteration (models that result from the initial training and from each pruning iteration are represented with a different color). From each phase, one of the models is represented with a triangle. That is the one that was selected as the base for the following phase. The selection was performed by excluding models that were already too close to the minimum accuracy threshold (e.g. only consider model with accuracy above 0.57) and also by experimentation. That is, we also tried to use different models from the ones selected in the figure but concluded these were the ones the lead to best results in this specific case. From the figure, we can also visualize that, in each prune iteration, the value of the ϵ significantly impacts both the inference cost score and accuracy, decreasing both as ϵ goes higher. On the other hand, λ has almost no effect on the inference cost score but has a significant impact on accuracy. This makes sense, as it is the pruning that reduces model complexity. The amount of weight decay that is used in the fine-tune can reduce the inference cost score but only marginally. Despite not having much effect on the inference cost score of the output model in the step in which it is applied, as discussed previously, an higher λ will allow for a higher ϵ to be used in the following prune iteration.

3.2 Quantization

Table 3 displays the obtained accuracy and inference cost score of the network in both floating-point representation and several fixed-point representations. The presented results of quantization are for a fixed bit-width on the weights and activations from 8-bit down to 5-bit, while quantizing input values to 8-bits. As expected, quantization reveals a drastic reduction in the inference cost score with decreasing bit-width. However, as the representation goes below 6-bit, accuracy starts to be heavily affected and stops being above the required 56% threshold. In the end, 6-bit quantization was chosen due to its results bring a good compromise between accuracy and inference cost score.

4 Conclusions and Future Work

Our model (BacalhauNet) achieved an inference cost score of 0.0162 ($\approx 61.73\times$ compression) and was the winning submission of the problem statement 7 (PS-007), "Hardware-Efficient Modulation Classification with RadioML". Therefore, we proposed a methodology that enables the implementation of the proposed DNN in resource-constrained devices. Further work should be devoted to the optimization of the last depthwise separable convolutional layer since it was not optimized due to time constraints; the testing of different levels of quantization per layer, since it can increase even more the compression achieved; the exploration of different feature engineering approaches, to assess if we can further reduce the model's inference cost score while maintaining acceptable accuracy.

References

[1] Timothy James O'Shea, Tamoghna Roy, and T Charles Clancy. Over-the-air deep learning based radio signal classification. *IEEE Journal of Selected Topics in Signal Processing*, 12(1):168–179, 2018.

[2] Timothy J O'Shea, Johnathan Corgan, and T Charles Clancy. Convolutional radio modulation recognition networks. In *International conference on engineering applications of neural networks*, pages 213–226. Springer, 2016.

[3] Charles Clancy, Joe Hecker, Erich Stuntebeck, and Tim O'Shea. Applications of machine learning to cognitive radio networks. *IEEE Wireless Communications*, 14(4):47–52, 2007.

[4] Kyouwoong Kim, Ihsan A Akbar, Kyung K Bae, Jung-Sun Um, Chad M Spooner, and Jeffrey H Reed. Cyclostationary approaches to signal detection and classification in cognitive radio. In *2007 2nd ieee international symposium on new frontiers in dynamic spectrum access networks*, pages 212–215. IEEE, 2007.

[5] Thomas Warren Rondeau. *Application of artificial intelligence to wireless communications*. PhD thesis, Virginia Tech, 2007.

[6] Antonio Polino, Razvan Pascanu, and Dan Alistarh. Model compression via distillation and quantization. *arXiv preprint arXiv:1802.05668*, 2018.

[7] Simone Bianco, Remi Cadene, Luigi Celona, and Paolo Napoletano. Benchmark analysis of representative deep neural network architectures. *IEEE Access*, 6:64270–64277, 2018.

[8] Vivienne Sze, Yu-Hsin Chen, Tien-Ju Yang, and Joel S Emer. Efficient processing of deep neural networks: A tutorial and survey. *Proceedings of the IEEE*, 105(12):2295–2329, 2017.

[9] Yaman Umuroglu, Nicholas J Fraser, Giulio Gambardella, Michaela Blott, Philip Leong, Magnus Jahre, and Kees Vissers. Finn: A framework for fast, scalable binarized neural network inference. In *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pages 65–74, 2017.

[10] Michaela Blott, Thomas B Preußer, Nicholas J Fraser, Giulio Gambardella, Kenneth O'brien, Yaman Umuroglu, Miriam Leeser, and Kees Vissers. Finn-r: An end-to-end deep-learning framework for fast exploration of quantized neural networks. *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, 11(3):1–23, 2018.

[11] Yann LeCun, John S Denker, and Sara A. Solla. Optimal Brain Damage (Pruning). *Advances in neural information processing systems*, pages 598–605, 1990.

[12] Eunhyeok Park, Sungjoo Yoo, and Peter Vajda. Value-aware quantization for training and inference of neural networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 580–595, 2018.

[13] Amir Gholami, Sehoon Kim, Zhen Dong, Zhewei Yao, Michael W Mahoney, and Kurt Keutzer. A survey of quantization methods for efficient neural network inference. *arXiv preprint arXiv:2103.13630*, 2021.

[14] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, Quoc V. Le, and Hartwig Adam. Searching for mobilenetv3. *CoRR*, abs/1905.02244, 2019.

[15] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[16] Alessandro Pappalardo. Xilinx/brevitas, 2021.

[17] Yunhui Guo, Yandong Li, Liqiang Wang, and Tajana Rosing. Depthwise convolution is all you need for learning multiple visual domains. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 8368–8375, 2019.

[18] Hao Zhang, Fuhui Zhou, Qihui Wu, Wei Wu, and Rose Qingyang Hu. A novel automatic modulation classification scheme based on multi-scale networks. *arXiv preprint arXiv:2105.15037*, 2021.

[19] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[20] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[21] Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. Learning Efficient Convolutional Networks through Network Slimming. *Proceedings of the IEEE International Conference on Computer Vision*, 2017-Octob:2755–2763, 2017.

[22] Vinay Jethava, Anders Martinsson, Chiranjib Bhattacharyya, and Devdatt Dubhashi. The lovász ϑ function, svms and finding large dense subgraphs. *Advances in Neural Information Processing Systems*, 25:1160–1168, 2012.

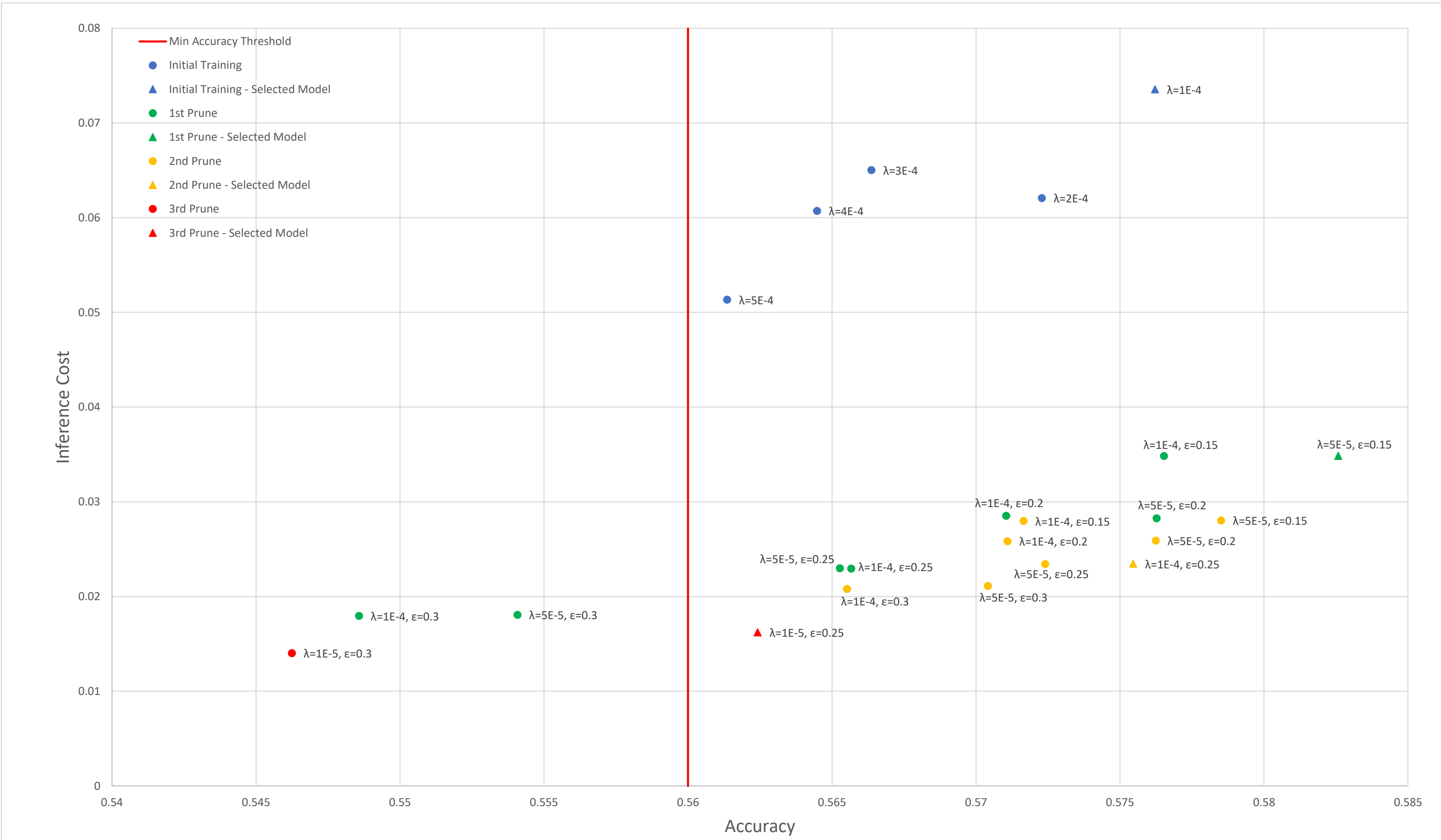


Figure 5: Results for the pruning design space exploration. Models that result from the initial training and from each pruning iteration are represented with a different color.