LIF Project Specification

Simon Kaufmann
sikaufma@ucsd.edu

Owen Jow
owen@eng.ucsd.edu

April 2020

1 Overview

Radio communication plays an important role in modern day life and shapes the way people are able to communicate with each other through mobile devices. A growing number of communication services (like the 5G mobile phone network) need to share a very limited frequency spectrum. The ability to share the same frequency range for different services could therefore be useful in the future, but requires being able to classify and recognise received radio signals by the service they belong to. In this project, we aim to implement an efficient and low-power computing system to classify radio signals. Our method will be based on a learning system inspired by biological neurons and will be evaluated using RadioML, a publicly available dataset of radio signals.

2 Approach

Background

Traditional radio communication devices consist of dedicated hardware units for various signal processing tasks (e.g. amplifiers, filters, modulator). Cheaper and more powerful modern digital computing hardware enables us to perform many of these tasks in software instead of hardware.

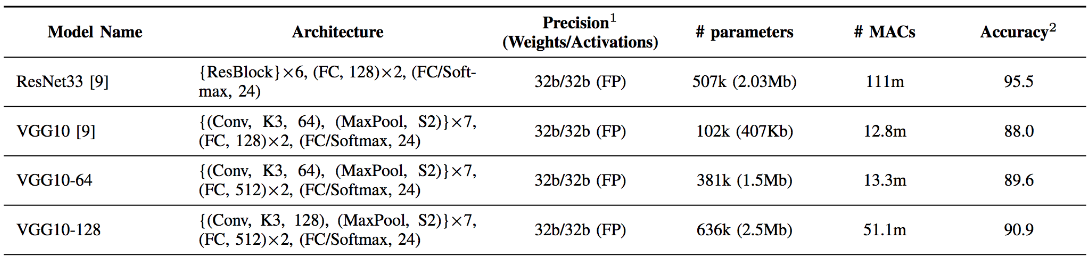
This approach has the advantage that besides performing traditional signal processing tasks, the computer is able to assist with additional tasks like classification of the received radio signal (modulation classification). Such classification can for example be useful in non-cooperative communication settings (e.g. military units trying to decode communication of hostile forces) or generally for detecting the type of service transmitted at specific frequencies.

Machine learning has proved to be an effective approach for performing complex classification tasks. As described in the following section, an implementation for this modulation classification task already exists using a traditional convolutional network. For this project, we will examine the use of a novel type of neural network, the so-called spiking neural network (SNN), which is inspired by biological neuronal processes in the brains of animals and humans.

Previous Work

In one previous effort ([1]), our project lead and several other collaborators performed a study of the efficacy and complexity of several deep learning models as applied to the RadioML automatic modulation classification task. In that project, the best performing models were convolutional networks (ResNet and VGG) with large numbers of parameters and operations. The networks were basically architected to perform a number of convolutions on the input radio signals before reducing the activations to per-class probabilities using a softmax layer. We plan to use this project as a baseline with which we can compare our own results.

Since spiking neural networks are much less developed than traditional neural networks, it is unlikely that our less complex model will perform up to par with the existing ResNet and VGG approaches. However, this is okay because our motivation for using SNNs comes from the fact that the associated computational complexity and power requirements can be less than those of traditional CNN-based counterparts. Given the highly real-time nature of the problem and the need for limited memory and power consumption on an FPGA, these aspects of the solution are more important to consider than they would usually be. For example, the ResNet, while it performed the best of all models, could not be implemented on an FPGA with high throughput because it was too large and required too much on-chip memory to store intermediate activations. Thus we consider it an important component of our project to quantify the performance of our models *relative to their parameter counts and computational requirements.*



Model Name	Architecture	Precision[1] (Weights/Activations)	# parameters	# MACs	Accuracy[2]
ResNet33 [9]	{ResBlock}×6, (FC, 128)×2, (FC/Softmax, 24)	32b/32b (FP)	507k (2.03Mb)	111m	95.5
VGG10 [9]	{(Conv, K3, 64), (MaxPool, S2)}×7, (FC, 128)×2, (FC/Softmax, 24)	32b/32b (FP)	102k (407Kb)	12.8m	88.0
VGG10-64	{(Conv, K3, 64), (MaxPool, S2)}×7, (FC, 512)×2, (FC/Softmax, 24)	32b/32b (FP)	381k (1.5Mb)	13.3m	89.6
VGG10-128	{(Conv, K3, 128), (MaxPool, S2)}×7, (FC, 512)×2, (FC/Softmax, 24)	32b/32b (FP)	636k (2.5Mb)	51.1m	90.9

Figure 1: These are the ResNet and VGG model sizes and accuracies from the project lead's previous paper ([1]). They were the best-performing and also largest models in the project. We will be aiming to get as close as possible to these accuracies with less computationally-expensive models and reduced precision.

SNN with Deep Continuous Local Learning

For our approach, we plan to follow the spiking neural network method called "Deep Continuous Local Learning" (DECOLLE) that is outlined in [2]. It provides a novel way to train SNNs, which are event or spike-based networks that are naturally equipped to process temporal data (which can be represented as spikes over time) and that are theoretically more powerful than artificial neural networks because of the differences in their neuron design. The problem with SNNs is that spikes over time are not differentiable, so they are difficult to train with the standard optimization methods of today (e.g. stochastic gradient descent) which require the ability to compute gradients of the loss function with respect to every parameter in the network.

DECOLLE solves that problem by using auxiliary classifiers on a per-layer basis to provide local classification signals for each linear integrate & fire (LIF) neuron level in the SNN. LIF neurons are modeled after biological neurons and are represented by a number of state variables such as membrane potential and resetting state.

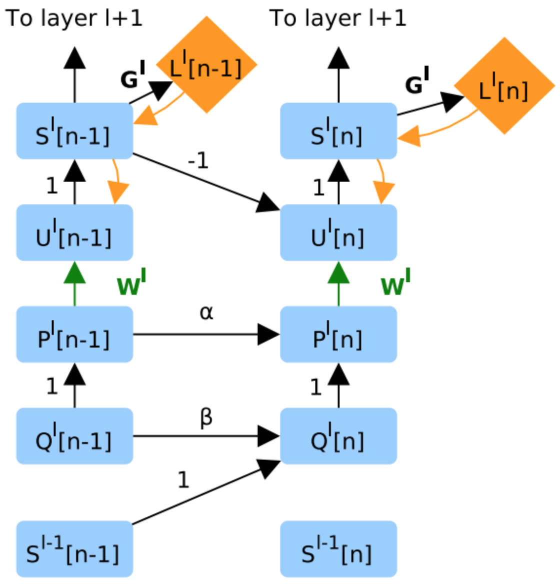


Figure 2: An illustration of the DECOLLE feedforward SNN architecture over two timesteps. Each blue block represents a state variable, each column represents a timestep, and the orange diamonds are auxiliary losses. The arrows represent the flow of information (orange arrows represent the flow of gradients). As you can see, information from one time step is fed into the next time step, which helps the network model the temporal aspect of the data.

Project Parts

We intend to build off of the code provided by the authors of the DECOLLE paper, extending their work according to the following sequence of steps:

- Implementing a larger spiking neural network for RadioML based on the dcll library, which is modeled after the VGG network architecture used in [1].
- Optimizing this network for RadioML classification accuracy by tuning various

hyperparameters like learning rate, momentum, etc.

- Extending the Brevitas quantization library to support spiking neural networks
- Training a quantized version of the RadioML SNN and optimizing it for model size, memory bandwidth, and speed while maintaining as high of an accuracy as possible

Further Extensions

Eventually, the model created for this project should be able to run on dedicated hardware (e.g. to be used in Sofware Defined Radios) which means that similarly to the VGG used in *Previous Work*, an FPGA implementation will be desirable.

FPGAs are customizable computer chips that can be programmed for specific purposes using a hardware description language. Compared to processors where the hardware structure is fixed and only software can be modified, an FPGA allows the programmer to adapt the configuration and wiring of the entire computer chip to the needs of the project. Due to this property, FPGAs deviate from the standard sequential processor model and can perform many separate operations in parallel which makes FPGAs well suited for the implementation of neural networks.

There is some uncertainty in this project around how easy it will be to adapt the Brevitas library for spiking neural networks. If it turns out that no considerable progress is made in this regard, we may revise the schedule and focus on the FPGA implementation of the RadioML SNN instead of its quantization.

3 Project Objectives

Our overall objective is to train an SNN model which approximates previous VGG/ResNet accuracies on RadioML modulation classification and train a quantized version of the same network to optimize for computing power and memory. We plan to work toward our large-scale goal by producing the following results:

- **An SNN network for RadioML trained using DECOLLE:**
  Based on the dcll Python library provided with [2], we plan to create a model trained using the RadioML dataset for modulation classification. This model will be evaluated on accuracy.

- **A quantized version of the SNN network for RadioML:**
  In order to prepare the SNN for effective and fast computation, we intend to create a quantized version of the neural network using the Python library Brevitas. This network will be evaluated on its required space, computational efficiency (timing) and accuracy.

For more details about the milestones, deliverables, and schedule, see section 7.

4 Feasibility and Risks

Performance

In one sense, there is a risk that the DECOLLE spiking neural network will not perform well on the RadioML dataset, as SNNs are difficult to train and do not currently produce results on par with traditional deep learning schemes. We would likely need to develop or utilize new SNN-based optimization methods in order to resolve this; however, that is not the focus of this project. Here we are primarily interested in seeing how well we can do with the current training methods, and potentially quantifying the performance gains (in terms of compute and memory) of DECOLLE over VGG, with and without quantization.

Training Time / Hyperparameter Tuning

Another difficulty and contributor to risk stems from the fact that deep learning methods require a significant amount of training time, meaning there is a long turnaround before we are able to see a result – and the result might not even be good or valid (due to implementation error, poorly-chosen hyperparameters, or the limitations of the method). Optimizing hyperparameters to get good training results may require many iterations of the same training process which can potentially delay project progress. We will try to alleviate this problem by making healthy use of the GPU cluster, where we have the ability to run multiple experiments in parallel for faster development, and by allocating sufficient time for the tedious process of hyperparameter tuning in the schedule.

5 Group Management

We plan to make decisions by consensus (taking into account the advice of our project lead) and to communicate via Slack channel, making Skype calls as necessary. We will report weekly progress to our project lead via Slack.

Progress will be monitored using the milestone schedule. If we experience significant delays of our milestones due to problems encountered, we may revise the schedule and focus our efforts on creating an optimized RadioML SNN (without quantization). In case the Brevitas implementation causes problems in particular, we may instead develop an FPGA implementation of the RadioML SNN and leave the quantization as a further extension to the project for teams later on.

Owen's main focus will be the implementation of the SNN in PyTorch using the DCLL library while Simon will work on the quantization of SNNs using Brevitas. We recognize that there is some uncertainty around the areas with the most time-consuming challenges for the project. Therefore, this division is tentative and we will stay in regular contact throughout the project to combine our efforts in overcoming the main challenges.

6 Project Development

Software Components

Development for this project is mostly software-based. These are the main software components that we will use:

- **dcll**, a Python library built on top of PyTorch implementing spiking neural networks and the DECOLLE learning rule
- **Brevitas**, a Python library based on PyTorch for quantization-aware training of neural networks
- **Vivado HLS**, a software package for high level synthesis and simulation of hardware designs for FPGAs

Owen will focus more on the adaptation of the dcll library, while Simon will be working with Brevitas. Nevertheless, since many of the software components are related, we will both be involved with all major parts of the project.

Resources

The training of neural networks can be computationally very expensive and works best on GPU clusters maintained for this purpose. For this project we will have access to the GPU cluster run by the university.

The codebase will be maintained on a shared GitHub repository to allow for effective collaboration and version control. Documentation for our codebase will be maintained using README files in markdown format which allows us to integrate code and documentation effectively within the same repository.

Additionally, a project website will be created containing a project description and links to the codebase. Reports documenting our project results will also be published on the project website.

7 Schedule

In this section we outline the schedule for the project. Table 1 provides a summary, while a more detailed description can be found below.

Week	Type	Description	
4	Milestone	Train SNN (DCLL lib) with MNIST dataset	Simon
	Milestone	Preprocess and load RadioML dataset for SNN	Owen
5	Milestone	Get familiar with Brevitas library	Simon
	Milestone	Get familiar with the DCLL codebase	Owen
6	Milestone	Extend Brevitas for usage with SNNs from dcll library	Simon
	Milestone	Adapt SNN for RadioML using DCLL PyTorch library	Owen
7	Deliverable	Hyperparameter tuning of SNN for RadioML	Simon
	Deliverable	Hyperparameter tuning of SNN for RadioML	Owen
8	Milestone	Adapt RadioML SNN for quantization in Brevitas	Simon
	Milestone	Tuning of architecture parameters for RadioML SNN	Owen
9	Deliverable	Optimize quantized version of RadioML SNN	Simon
	Deliverable	Optimize quantized version of RadioML SNN	Owen
10	Deliverable	Final report and video (focus: quantization)	Simon
	Deliverable	Final report and video (focus: SNN implementation)	Owen

Table 1: Summary of milestone/deliverable schedule

Week 4

Simon:

Description:	Train SNN (DCLL lib) with MNIST dataset.
Completion Criteria:	Have an SNN network that can be run on the MNIST dataset on the GPU cluster.

Owen:

Description:	Preprocess and load RadioML dataset for SNN.
Completion Criteria:	Have the RadioML data on the GPU cluster in a form that can be fed to the SNN.

Week 5

Simon:

Description:	Get familiar with Brevitas library.
Completion Criteria:	Written report including a description of which functionality Brevitas offers, how the codebase of Brevitas is organized, how it is used for regular neural networks and outline on where extensions for spiking neural networks will have to be made.

Owen:

Description:	Get familiar with the DCLL codebase, determine which parts of the code will need to be changed in order to create a VGG-like SNN architecture using DCLL modules.
Completion Criteria:	Written report which includes a high-level overview of the different parts of the code and a description of which specific files and functions will need to be changed or utilized in order to define a VGG-like architecture for use with RadioML.

Week 6

Simon:

Description:	Extend Brevitas for usage with spiking neural networks from dcll library.
Completion Criteria:	Have a trained quantized version of the MNIST network for different bit sizes (16-, 8- and 4-bit) and report on accuracy achieved.

Owen:

Description:	Adapt SNN for RadioML using DCLL PyTorch library.
Completion Criteria:	Have a first SNN modeled after VGG that can be trained on RadioML data (ignoring actual performance at this point).

Week 7 - Deliverable

Owen & Simon:

Description:	Hyperparameter tuning of SNN for RadioML Explore different versions for hyperparameters focusing on learning rate, learning rate decay schedule, optimizers, momentum, batch size, etc. to optimize achieved accuracy of the network.
Completion Criteria:	Classify RadioML data by modulation with at least, say, 60% accuracy on the test set (which corresponds to the worst performance reported in the previous work).
Deliver:	Report on architecture, implementation and hyperparameters for the model and discuss resulting accuracy.

Week 8

Simon:

Description:	Adapt RadioML SNN for quantization in Brevitas.
Completion Criteria:	Have a trained quantized version of RadioML SNN in Brevitas (without focusing on accuracy at this point).

Owen:

Description:	Tuning of architecture parameters for RadioML SNN Hyperparameter tuning can be a very time-consuming and tedious process. Therefore, we schedule this week to continue experimentation, focusing on modifications and optimizations of the model architecture.
Completion Criteria:	Have optimized RadioML SNN with highest achieved accuracy, update report with results.

Week 9 - Deliverable

Owen & Simon:

Description:	Optimize quantized version of RadioML SNN.
	Experiment with the Brevitas model for RadioML to reduce model size while maintaining high accuracy.
Completion Criteria:	Have optimized quantized RadioML SNN with highest achieved accuracy.
Deliver:	Report on architecture, implementation and hyperparameters for quantized RadioML SNN.

Week 10 - Deliverable

Owen & Simon:

Description:	Final report and video
Completion Criteria:	Submit final report and video addressing project progress, challenges, solutions and results.

References

[1] Stephen Tridgell, David Boland1, Philip H.W. Leong, Ryan Kastner, Alireza Khodamoradi, and Siddhartha. Real-time automatic modulation classification using rfsoc. In *RAW 2020*.

[2] Jacques Kaiser, Hesham Mostafa, and Emre Neftci. Synaptic plasticity dynamics for deep continuous local learning. *CoRR*, abs/1811.10766, 2018.

[3] A. Tsakmalis, S. Chatzinotas, and B. Ottersten. Modulation and coding classification for adaptive power control in 5g cognitive communications. In *2014 IEEE 15th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, pages 234–238, 2014.