DTU Compute
Department of Applied Mathematics and Computer Science

Golf-ball velocity calculation through radar spectrography

Filip Roszkowski (s233421), Marcel Zelent (s233422), Karl Mihkel Seenmaa (s232512)

1 Introduction

In this project, the team was tasked with optimising a Machine Learning Model developed by Trackman for the purpose of interpolating a golf ball's velocity from a spectogram (Fig. 1). The data, derived from four radar receivers, includes power spectrograms for each receiver and phase spectrograms, resulting in six input channels to the ML model.

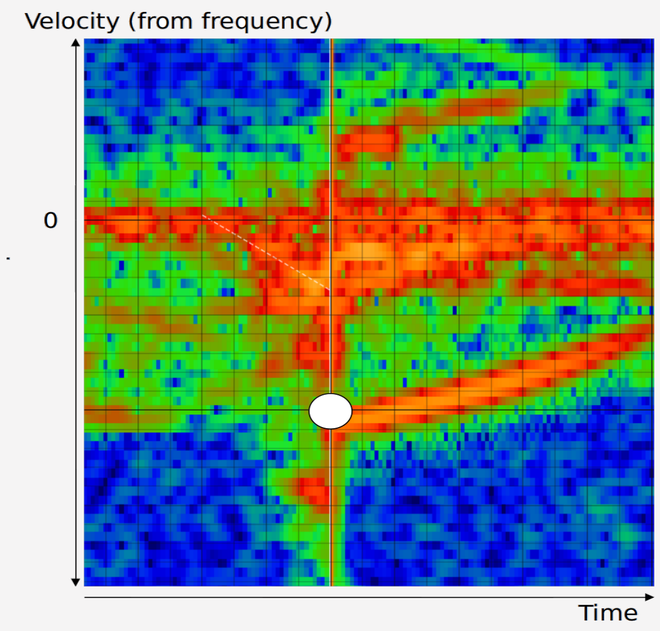

Figure 1: Spectogram containing velocity from frequency data.

The Trackman team has already found a suitable model with satisfactory performance for this task. However, since it is supposed to run on an embedded system, it is essential for it to be as computationally efficient as possible. Therefore the goal of the project is to find a method of downsizing it, while still keeping an error value the same or lower than the original. Furthermore, it is desirable if the smaller model can also operate with faster prediction speed.

The baseline model has **38M parameters**, with 4 convolutional (6, 16, 32, 64, 128) and 3 linear layers (1024, 256, 1). A visualisation is seen on Figure 2.

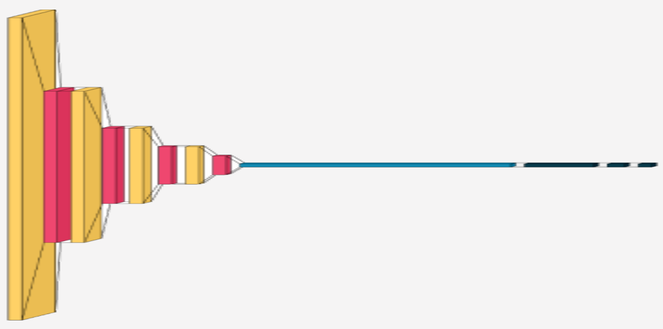

Figure 2: Baseline model architecture.

2 Methods

In order to achieve the desired goal, the group has decided on 4 approaches: parameter tuning, LSTM, Distilled Knowledge and Quantization.

2.1 Parameter tuning

The simplest approach to the problem is to reduce the number of weights by adjusting the sizes of each layer in the CNN and ANN. It was found that actually this can be utilized to great extent, where the CNN could be shrunk by a factor of 2 and the ANN by 4. At that point the model was still performing the task as well as the baseline, in some cases even improving upon it.

Finally, we combined these two methods together, that had **ca. 9 times** fewer parameters (4.5M) than the baseline model. It consists of 5 convolutional layers, and 3 linear layers.

	Test RMSE	Test Log RMSE	Model parameters	Size [MB]
Baseline	1.59	0.20	38M	153.67
CNN shrunk	1.69	0.23	20M	73.6
ANN shrunk	1.54	0.19	10M	36.8
Combined	1.29	0.11	4.5M	17.35

Table 1: Model metrics.

2.2 LSTM

LSTMs are good at handling sequential data, thus we decided to also try training LSTM models on sequential data, for which we implemented our own data loader, that loaded the sequences from the 4 different receivers. The models trained on the data did not perform as well as hoped. There may be a couple of different reasons for that:

- noise in the data
- small models (had to reduce the size of our LSTM models significantly since the memory required to train was too high)
- bad model architecture

2.3 Distilled Knowledge

Distilled Knowledge is a technique where two models are used: a teacher model that refers to an advanced and highly trained network, and a student model, which is a greatly simplified version of it. Then, during the process of learning of the student, the loss is defined as follows:

$$L_{total} = L_{hard} * \alpha + L_{soft} * (1 - \alpha) \quad (1)$$

The hard loss is the loss from normal training, while soft loss is defined as the difference between the prediction made by the teacher and the student. Furthermore, the α parameter can be used to adjust the ratio between them.

The team thus took the baseline model as the teacher, and then made an appropriately small student model, using the experience gained during parameter tuning. As a result, the team was able to achieve a model with only **6,651 parameters**. The layers in both the CNN and ANN were minimized as much as possible, which is shown on Figure 3. An important step here was to increase the kernel size of the pooling layers, which meant that the number of input neurons to ANN was down to meager 55 (37120 originally). As a result of all of these adjustments, the final size of the model was **32 KB**.

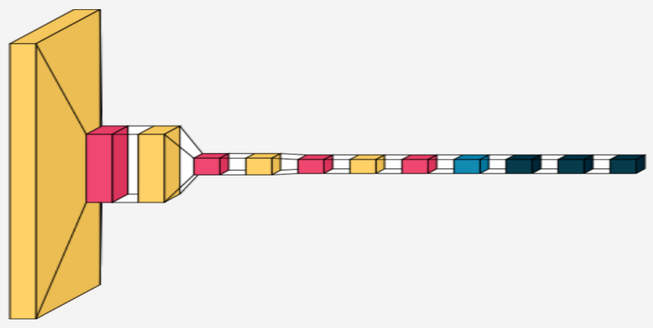

Figure 3: Student network architecture

The method does have a negative effect on the performance, as now the Test RMSE was around 1.54 in comparison to 1.29 of the *Combined Simplified* model. Nonetheless, that is within the range of the original baseline.

2.4 Quantization

Quantization is a process in which the 32-bit floating point numbers used inside the model are turned into 8-bit integers (16-bit floats for GPU). This considerably saves on inference time, as the Arithmetic Logic Units (ALU's) inside a modern chip are more optimized to work with these numbers. The process starts by teaching a Machine Learning model to a high degree using floating point numbers, and then transforming it into the simplified version by a process called calibration. During calibration, inference is preformed on the model while the quantization framework observes the minimum and maximum values of the each neuron input in order to properly rescale the weights and biases to fit the expected range of inputs - so that the impact of the lower precision is minimized.

There are various way to quantize a model.

- dynamic (runtime) quantization
- static (post training) quantization
- quantization aware training

Presented on this poster are the Static and Dynamic methods. As can be seen on Figure 4 and figure 6 where the static quantization results are shown. This is the main method used for CPU inference speed-up. The quantization is done after the model training and is done only once before the inference. As for the GPU as seen on Figure 5, Dynamic quantization is preferred. It is done just-in-time before the inference and during the runtime, this way it has an added benefit of being able to adjust the calibration to unseen data were as the static quantization can not be changed online. All of the quantized models preformed within 0.01% of RMSE error of the non-quantized models.

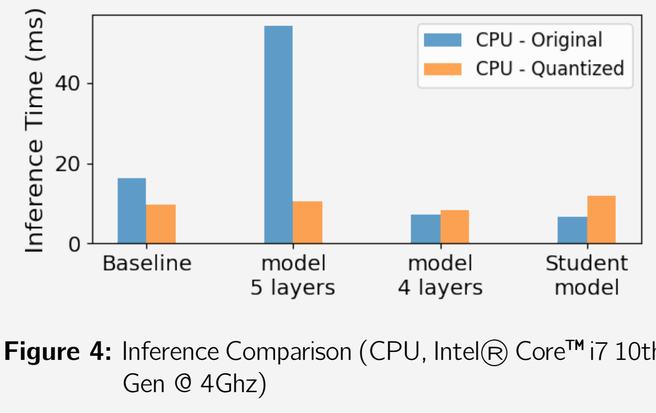

Figure 4: Inference Comparison (CPU, Intel® Core™ i7 10th Gen @ 4Ghz)

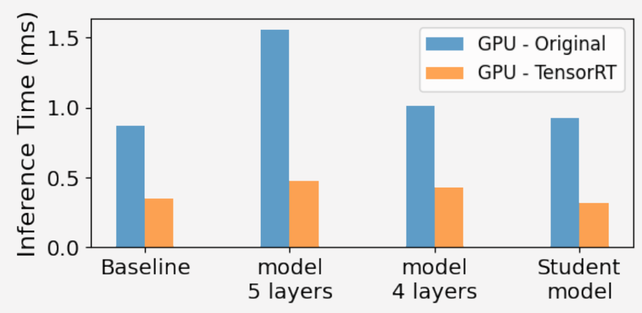

Figure 5: Inference Comparison (GPU, NVIDIA GTX 1650 Ti)

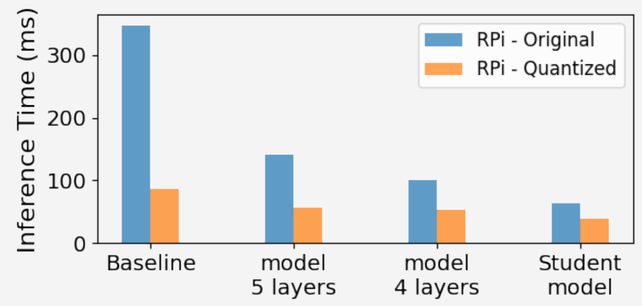

Figure 6: Inference Comparison (Raspberry Pi 4, Broadcom BCM2711 @ 1.8Ghz)

	Baseline	model 5 layers	model 4 layers	Student model
Original size [MB]	153.67	17.35	19.22	0.03
Quantized size [MB]	38.46	4.39	4.83	0.02

Table 2: Model file size comparison.

3 Summary

The team was tasked with optimizing the size and inference time of a baseline model developed to predict a velocity of a golfball. Four approaches were taken, namely parameter tuning, LSTM, Distilled Knowledge and Quantization. Apart from LSTM, all methods proved to drastically reduce the size of the model, improving inference time and accuracy of prediction.

DTU Compute
Department of Applied Mathematics and Computer Science
DTU Compute
Department of Applied Mathematics and Computer Science
DTU Compute
Department of Applied Mathematics and Computer Science
DTU Compute
Department of Applied Mathematics and Computer Science