Table of Contents

1 Measurements and results

1.1 Model comparisons

As mentioned in section **??** we used Keras Tuner model to find hyperparameters that would yield the highest accuracy. Instead of hard-coding hyperparameters when building a model with Keras API, we defined a search space of possible values with `HyperParameter` class and used that as a hyperparameter.

We then passed the created model to a `RandomSearch` class, with few other parameters such as batch size, number of epochs and maximum number of trials. We then started hyperparameter search, which means that Keras Tuner was randomly picking a set of hyperparameters and training a model with them. This process was repeated for a trial number of times. Accuracy and hyperparameters that were used while training every module were saved to a log file for later analysis.

After training a number of different models we handpicked a few of them and compared them. Comparison of equivalent model trained in Edge Impulse studio was also done.

1.1.1 Hyperparameter search space and results analysis

General structure of CNN model was already described in section **??** and in Figure **??**. We decided to search for following hyperparameters: number of filters in all three convolutional layers (can be different for each layer), filter size in all three convolutional layers (same for all layers), size of dense layer, dropout rate and learning rate. Possible values of hyperparameters (also known as hyperparameter search space) are specified in table 1.2.

Hyperparameter	**Set of values**
`FilterNum1`	From 16 to 80, with a step of 8
`FilterNum2`	From 16 to 80, with a step of 8
`FilterNum3`	From 16 to 80, with a step of 8
`FilterSize`	3 x 3 or 3 x 4
`DenseSize`	From 16 to 96, with a step of 8
`DropoutRate`	From 0.2 to 0.5, with a step of 0.05
`LearningRate`	0.0001 or 0.0003
Random search variable	**value**
`EPOCHS`	25
`BATCH_SIZE`	100
`MAX_TRIALS`	300

Table 1.1: Hyperparameter search space

Search space of `filter_numX`, `dense_size` and `dropout_rate` hyperparameters was chosen based on initial training tests and various models that were trained on similar data. Value of `filter_size` is usually 3 x 3, however all example ML projects were training on image date of same dimensions. We wanted to test how would a filter with same ratio of dimensions as image data (3 x 4 and 60 x 80 respectively) perform. Hyperparameter `learning_rate` was chosen heuristically, we saw that higher values, such as 0.001 or 0.003, would leave model's accuracy stuck at suboptimal optima, from where it could not be improved anymore.

We also had to set 3 variables that directly affected how long will random search last. From initial tests we saw that models usually reached maximum possible accuracy around 20th epoch, to give some headroom we set the number of epochs to 25. We kept batch size relatively small, at 100, which meant that weights would get updated regularly. Hyperparameter `MAX_TRIALS` had the biggest impact on the training time, we set it to 300.

Training lasted for about 12 hours. After it was done we compiled a list of 100 best performing models, part of it can be seen in Table **??**.

By providing accuracy metric, Keras tuner automatically sorts trained models in descending accuracy. This is in many use cases sufficient, however in our case, we

Hyperparameter	FilterNum1	FilterNum2	FilterNum3	DenseSize	DropoutRate	LearningRate	Number of parameters	Accuracy
Model ID								

Table 1.2: Hyperparameter search space

were also interested in best performing models that had small size.

- [ ] Do model comparisons, analyze them and present them. This means that you need to describe the setup, what you want to get from this subchapter, hyperparameter search space, list of models,

1.1.2 Comparison of selected, re-trained models

how you will select them, an image of history (accu and loss), confusion matrix, precision and recall

1.1.3 Comparison with Edge Impulse model

Describe how did you build Edge Impulse model, maybe mention transfer learning (might be extra work, you did not write about theory)

1.2 On device performance testing

Describe setup how are you timing,

1.2.1 Comparison of different optimization options

1.2.2 Comparison of performance of selected models

Transfer learning could be interesting here, bigger number of parameters and takes less time.

1.3 Power profiling of an embedded early warning system

- [ ] Power consumption test of whole setup, PIR wakes up wisent, wisent turns on stm32f7 and flir, which makes a picture, does inference, reports result and wisent sends the result. Otii image of consumption with marked sections.(shouldnt be hard)

1.3.1 Battery life estimations

Based on numbers and different scenarios estimate how long would this last with different batteries.

Bibliography