**BHCShhhhhhh**

1. **Begin**

Good morning everyone, I am Pan an interns working in Andreas’s group. I am glad to present you some works about activity recognition on sensor level.

1. **Agenda**

Today I will introduce you

1. **motivation**

As you know, we are developing a tool to deploy the neuron network to the sensor level. Activity recognition on microcontroller is a very good application. But before that, We need to develop a functional network model.

The activity recognition here means we use the signal from the wearable sensors and the network to classify which activity or movement the subjects are doing. The sensors can be accelerometers or gyroscope.

In compare with the normal network used on PC or workstation, it has some constrains on sensor level. Computation should be fast to achieve a real-time response. The memory on the microcontroller is small so that a large network is not fit. Also the large net need more computation cycles.

1. **Goals**

Therefore, we want to train an activity recognition on the small network structure.

For the network training, we have 2 datasets. Our group has used PAMAP2 dataset trained a network under the condition of the target platforms.

But on the life test of the sensor, the results are not as good as expected.

Hence the second Dataset BHCS with larger samples and subjects are interested. We try to extend the dataset and get a better model.

In this presentation, we will show you the achieved results on Pamap dataset and the beginning work on BHCS dataset

1. **Pamap**

Pamap dataset is a opensource dataset provided. They recorded 18 kinds of activities of 8 subject around 40 min of each. We summarized them into 6 known activities classes:

And one class of others to contain all samples else.

For the network structure we choose a simple network with 2 convolutional layer and 2 dense layer. The parameters of the layer remain to be decided according to the hardware constrains.

The maxpooling layer is optional during our experiments

1. **Computational limits**

First of all we trained the network with this parameters. Which can achieve around 85% test accuracy. But this network is sure to be too large to fit in micro controller.

As we talked before, the computational limit is a main constrain in our scenario. So here we use the current model as baseline, run some test in different combinations of smaller parameters, including size of filters in the conv layer, number of neurons in dense layer and pooling stride.

Most of the them fit the 1M macs constrains.

1. **memory**

In addition, memory limits are also concerned. Here we have two different memory size for 2 different microcontrollers.

According to this two set of experiment, we could choose a best combination under the constrains for this network structure. Here we use 4 by 8 for convolutional filter, 32 for neuron number of dense layer and maxpooling with stride 3 as interim layer.

1. **Results and discussion**

This table lists the memory usage and computational power needed in the baseline model and the chosen model. As long as the results.

We could see that the number of parameters and MACs are cut down to around 1% of origin without the accuracy drop.

discussion

However, though we could achieve high accuracy inside the dataset itself, the real-time test show that the model is insufficient. We deploy the model on the smart watch but it only gets 60% of accuracy.

There are many reasons could lead to this performance, for example the sample size is not enough because it only has 8 subjects. They may do the movement slightly different than what we did on life-test.

Or we don’t know how the pamap sensor is mounted on when they recorded the data,

A larger dataset might help us with this problem. Here the BHCS dataset comes to our consideration. So we what to use the new dataset the validate the old one, and find a way to unify the two dataset and even record some new data.

1. **BHCS**

BHCS dataset comes from Bosch hospital, they follow 52 patients about 4 hours and recording sensor data from both left and right of waist, wrist and ankle. And we summarized them into 7 classes as the pamap.

this dataset is much larger than the pamap. However, this dataset is quite imbalance. As you can see, more than half of the samples consists of sitting and walking. And very less of the samples are from stairs and others. And no samples of sport. We assume it is because the subjects are all patients so the main purpose of establishing this dataset is quite different as the pamap.

This cause some problem of training this dataset and the cross-validation between 2 datasets.

1. **Initial results**

First, we use the new dataset to train the selected network. And the result is just as expected.

Because the dataset itself is dominated by 2 classes, sitting and walking. The result is also dominated by these two classes. the network can easily tell the difference between sitting and walking. But the performance in other classes are not good.

In this case, the normal accuracy of 83% is meaningless. Because the majority of the test data is also sitting and testing.

Therefore, the take f1 score and balanced accuracy into consideration.

Balanced accuracy calculates the average of the proportion corrects of each class individually. So, the imbalance of the dataset is considered.

The balanced accuracy in the initial test only achieved 64%. Therefore, we need some method to overcome the imbalanced of the dataset.

1. **over under sampling**

One approach to do it is undersampling and oversampling method. It randomly select some data from the majority class and copy the samples from the minority class to match their quantities.

We applied this method is resample each class into equal. However, network will become overfitting. Because the quantity of majority is hundreds or thousands of times more than minority classes like stairs and others, it leads to too many identity samples by using this method. So that the network easily remembers the feature of training data of stairs and others. It is so called overfitting.

1. **segment**

So we want to try in another way.

Formerly we segment the raw data by a constant stride. For example. We have a 10s data of walking. If we segment it with window of 1s and stride of 0.5 s. we could generate 19 samples.

The raw data of minority class in raw class are always short and less.

So we try to segment the raw data with different stride. If it is a long data of majority calss. We segment it with large stride. If it is a small data of minority class, we use small stirde to get more data. Although these data are partly similar, but the similar part is has different time location so it wont be regarded as same in neuron network.

As you can see, the imbalance is partly fixed. And the quantity of stairs and others has a definite growth.

1. **Results**

The confusion matrix shows that except the class others, all the others classes are recognized well. The class others is more difficult to deal with because it may has no same patterns.

We list the statistic of all three experiment here. During the suppress of imbalance, the balanced accuracy increases around 10%.

1. **Cross validation**

The next topic is cross validation. We haven’t finished this part of work so here we just have a glimpse at it.

What we want to do is using the BHCS dataset to test the pamap trained network and vice versa. The reason to do this is to check the consistency of 2 dataset, study the inner feature of activity recognition and prepare for merge the datasets together or create new activity dataset.

However, these two datasets has much difference.

1. **Raw data**

Here we print a raw data of 3 axis accelerometer form each dataset. as we can see, the activities in pamap are much more drastic than BHCS. And due to the gravity some ans static activity, we could see the coordinate system might different. We haven’t look in to it and do some math, but it will be my next task.

1. **Cross validation result**

At last let us have a simple look at the cross-validation results of what we currently achieved.

First, BHCS dataset has no sports data, so sports data will be considered as any. We shall not care about this part.

Then the sitting class from both side can not recognize each other. this means the coordinate system calibration is still wrong or then have different gesture during sitting in both sides.

The others may contain different activities.

1. **Outlook**

New Quantization

1. **begin**

Hallo, I am Qizhen pan,

Our team, leadered by Andreas frischen, has been committing ourselves to develop a Tool, which can help us to deploy neuron network models on the sensor level.

In this presentation, Marvin and I will show you the latest progress of our deployment tool

1. **introduction**

First, I will quickly remind you why we develop this Tool, and what our neuron network deployment tool capable of.

As we know, the neuron network has become a very powerful method in many fields, like image classification and signal processing. Therefore, it is prospective on the sensor level, if the sensor can provide processed data by the neuron network. Out main function of our tool is to transfer the python code of neuron network in to C/++ code and deploy it to the microcontroller.

However, there are several main obstacles. Normally, the neuron network needs large computing power and memory, and it always require a fast responds and lower energy consume it is a portable device.

Hence, our tools also contain a constrain finder, to locate the suitable network size according to the given memory and CPU cycles. In addition, we have pruning algorithm and quantization algorithm to cut the size of the network down without or with a little accuracy drop.

The Introduction of the tool itself is quick in this presentation, If you are not familiar with this tool and are interested in it, don’t worry, we have a live demonstrator of the tool later, or please contact us after the meeting, we will show you the details. Now the main purpose of this presentation is introduce you the new quantization method by me, and backend runtime benchmarking on microcontroller.

1. **trans**

Hallo,

1. **quantization introduction**

why we need quantization in the deployment tool, as we said just before, On microcontroller there are many constrains compare with the normal computer, So we want to quantize the floating point number into fixed point number in order to save memory and accelerate the compute procedure.

What is quantization. We take 8 bit as example. Quantization means we convert any floating point number into a 8 bit fixed point number multiply by a scaler. Considering the sign bit, the fixed point number can be integer between -128 to 127. The scaler is exponential and we take m as a scaler factor. the scaler will be 2 to the power of scaler factor.

To be notice, the choice of scaler is arbitrary. For example, the number 234.25 can be quantized in these two ways. As you can see, the accuracies are different.

1. **Single value quantization**

Let see how quantization works on the single value first.

Assume we have a floating point value, example 234.25. we convert it to binary, it has 8 bit before decimal point and 2 after. But after quantization only seven bits are allowed. So we count 7 front left. Which is the integer number after the quantization. The bits after that will be discarded. But to represent the same value, we need one more bits. So we determine the scaler factor as -1 and the scaler will be 2 to power of 1. In this way, the original number becomes 117 mutiply 2.

Another example is here, it could have 3 fractional bits and than the resolution is 2 to power of minus 3.

Totally, the larger the scale factor is, the high the resolution it has. But high-resolution leads to small interval between 2 values and small range. So that, in this way of quantization, we get the highest resolution.

1. **Network quantization**

Next, we will see how to quantization goes in the neuron network.

First, every tensor in the neuron network will be quantized by one scaler factor. it means we process the quantization tensor-wise, not value-wise. Take a single neuron as example, we have input x, parameters weights and bias and output, the input is exactly the output of previous layer. So, we should quantize the 2 parameters and the output. There are slightly difference. For parameters we quantize them into integer tensor and one scaler. But for output we only need to estimate the range of possible value and determine the scaler factor.

After the quantization, the neuron becomes integer computation and bit shifts.

For tensor quantization, we pick one value from it the determine the scaler factor. but which value? For example, here is 2 by 2 tensor. If we use 12.1875, it turns out scale factor should be -3, as I explained in last slide, to optimize this number itself. However, it gives a very small range. Every number greater than this will be clipped. If we use the largest number to determine the scaler factor, no one will be clipped but resolution is kind of low.

1. **Previous approach**

after the above preparation of quantization principle, we finally come to introduce our quantization method in the neuron network. We will remind you the current method and then

introduce you the new algorithm.

As we explained. The task is quantizing every parameter in the neuron network and get a suitable scalar factor for output of layers.

First the have to estimate the output distribution of each layer. What we do is the feed a bunch of test data in to the network and get the output of each layer. Then we have a tensor to be quantized.

For each tensor, we sort the values and pick one value according to a given percentile. For example, 99 percent. Then we pick the value greater than 99% of the values, to determine the scale factor. usually we take it 99 percent instead of 100%. Because there may have some outliers which locates far away from the most values. To cover this value costs the loss the resolution.

we use the chosen values to determine the scalar factor. for output it is enough. For parameters, furthermore, we use the scalar to quantize the tensor and store it.

1. **New**

Nevertheless, during the quantization we will loss some information. Most of the value will either be clipped or loss resolution. But in neuron network it is kind of black box and we don’t know which value has more impact to the results. Besides, the percentile is set manually, the best option for each neuron network may different and we have no idea about it.

Therefore, we made some improvement based on the current method.

The idea is quite simple. When we finished the current method, we will get a scalar factor for every tensor. Using this scalar factor as starting point. We test the other values next to it, to see whether it can get better results. Here we use L2 distance between the original output and quantized output as the measurement.

There are slightly different between quantizing output and quantizing parameters. For output. We calculate the output, and then quantize the output with scalar factor. and then calculate the L2 distance between these two tensors.

1. **Quantize parameter**

For parameters quantization, we compute the output with original weights and bias, then compute the output with quantized weights and bias, get the L2 distance between them and do the optimize algorithm,

Besides, the whole procedure is going layer by layer. So that it is more flexible to choose the quantization parameters.

1. **results**

Here, we produced some results on 3 dataset, MNIST and CIFAR dataset are simple image classification tasks. and PAMAP2 is activity recognition problem as we discussed in the morning. We quantized the trained network with different bitwidth, 4, 7, 15 bits and in different ways. Namely original method in different percentile to choose scalar factor, and the new quantization method with a improper percentile as starting point.

The original method could achieve a very close accuracy to the unquantized model. However, it quite depends on the choice of the percentile to determine scalar factor. If the percentile is not suitable, the accuracy drops a lot. The new method overcome this problem. Even start from an improper point, it will automatically find the scalar factor which has the smallest impact to the result.

1. **results**

in this part of the presentation, we provide you a improved method of quantization in neuron network, which is less sensitive to the parameters and can automatically get the best scalar factor.

In the future we will do some further work like make the choice of quantization method layer-wise, finish the 16bit on sensor level and do more test.

Thanks, in next part marvin will introduce you some progress on microcontroller runtime benchmarking.