厦门亿联网络技术股份有限公司
杭州分公司

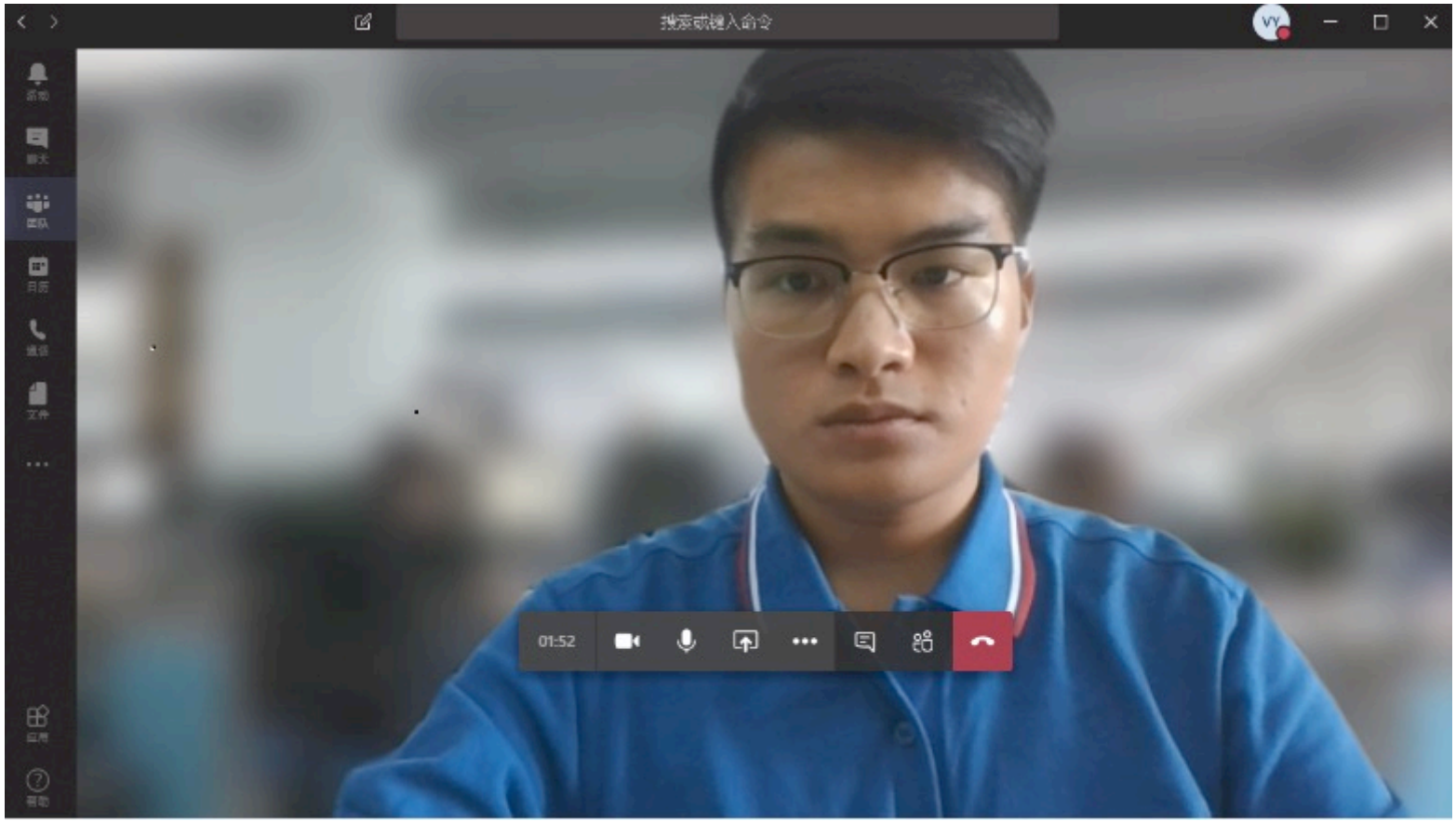
TVM for deloyment

dolphintear

1. Voice Communication

2. Video Conferencing

1. Face detection/recognition, background blur, human pose estimation



Why chosing TVM for our deployment?

1. OpenVino a black box, can not deploy our network(with depthwise conv2d,)

2. TVM can not only deploy our network, but also get a good performance gain by autotuning

3. TVM can support many kinds of hardware platform: Intel/arm CPU, Nividia/arm GPU, VTA…

TVM on windows：

1. Get a .log file from the autotvm on Ubuntu

2. Use the .log from step1 on Windows to generate the .dll for deployment

3. For application on 32bits, no support of 32bit tensorflow , a workround from **FrozenGene**

a. python/tvm/contrib/ndk.py
 options = options if options else [“-shared”, “-fPIC”, “-m32”]
b. python tensorflow_blur.py to get the .log
c. Use the .log, with target=“llvm –mcpu=i686 –mtriple=i686-linux-gnu”
then TVM_NDK_CC=“clang –m32” python tf_blur.py

Runtime problem about multi-threads ：

```
DWORD WINAPI TestThreadProc(LPVOID lpParameter)
{
    THREAD_DATA *pThreadData = reinterpret_cast<THREAD_DATA*>(lpParameter);

    DLTensor * inTensor = pThreadData->input;
    DLTensor *outTensor = pThreadData->output;

    cout << " Beginning to excute the thread ...." << endl;
    for (int i = 0; i < 2; i++)
    {
        pThreadData->nInfer->Infer_Work(inTensor, outTensor);
        cout << "time :" << i << endl;
    }


    return 0L;
}

void MultiThreadProcess(THREAD_DATA nThreadData)
{
    HANDLE nThread = CreateThread(NULL, 0, TestThreadProc, &nThreadData, 0, NULL);
    Sleep(3000);
    DWORD ret = WaitForSingleObjectEx(nThread, INFINITE, TRUE);

    if (ret == WAIT_OBJECT_0)
    {
        cout << " Thread " << GetCurrentThreadId() << "writing to database...\n" << endl;
    }
    else if (ret == WAIT_ABANDONED)
    {
        cout << "Thread failed ...\n" << endl;
    }

    CloseHandle(nThread);


}
```