Jayakrishnan K S 21112022 Quantization revision, Grouped convolution

bin the number of values in larger range mapped to each point in smaller
maping between differnt numberlines or ranges
larger range 1000.0 to -1000.0
smaller range -128 to 127
2001/256 values will be in a bin ther are 256 such bins

scale multiplication is the shirking of a number line & zero point adding is the shifting of the number line into desired range in our case -128 to 127
or 0 to 255

calibration retrivel of scale and zero point from feature maps. quantization requires calibration.

s=(float_max-float_min)/(int8_max-int8_min)
zp=int8_boundary-float_oundary/scale

float_x=scale*(quantized_x - zp)

in the model representation the 4 diemanerions are NCHW/NHWC for filter
FWHC/WHCF.

N is batch diamension one means one sample image HWC is the hight width and channel of that image feature map

for models we pass in a sample of the input space get output. then itterate over the set of samples to get a real max and real min using which we compute for the model scale and zero point.

convolution : need filter size specification as well as kernel size specification.
in it the size of each filter is kernel size spatial diamension with the depht as the input channel size. its filter count sepcifies the output channel depth.

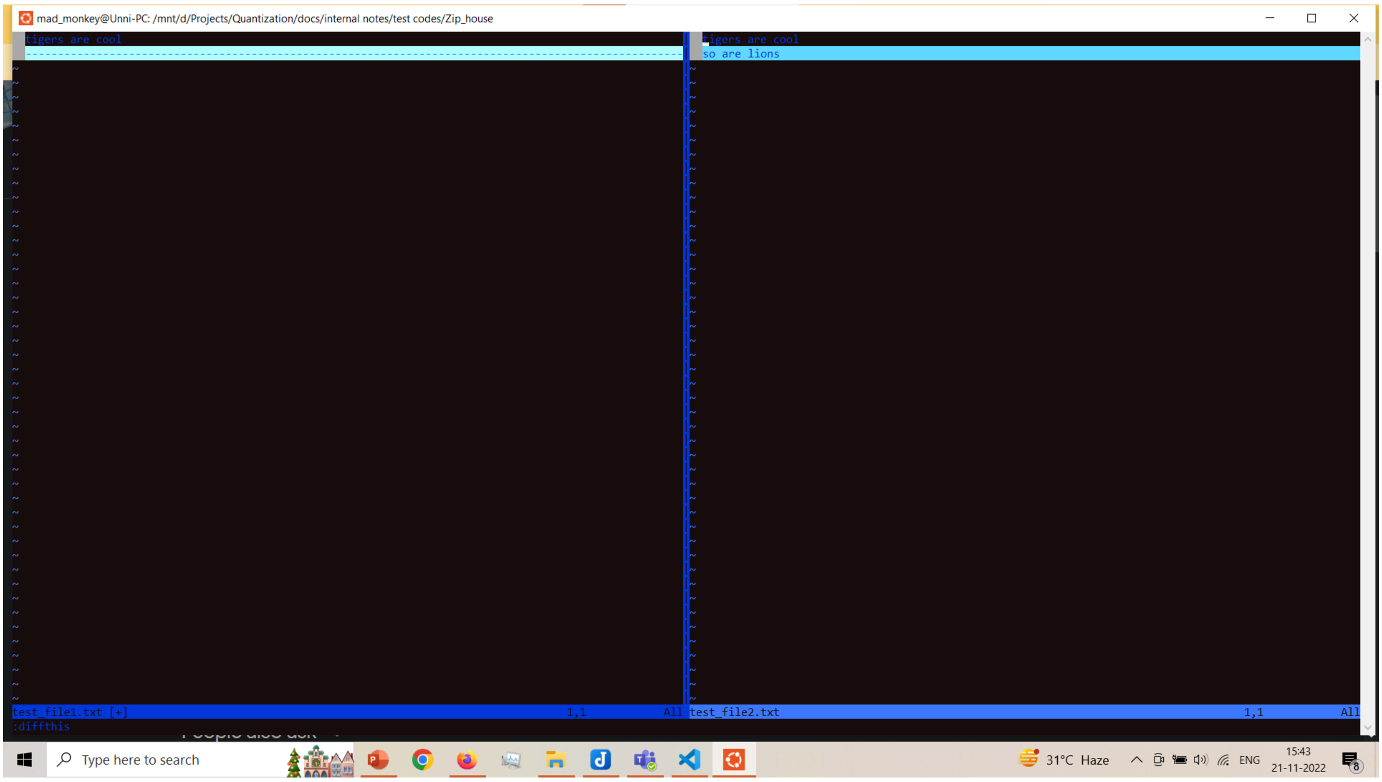
in depth wise the filter size is set as 1 (depth mutliplier?)

vim diff this command will give you a differnece between two files. open 2 files go to first file and type diff then go to second file the do diff this in cmd mode will give the difference between the files.

to creat a file
`cat > file_name.type`

vim command to open 2 files
`vim -O file_1.py file_2.py`

esc get you into command there `diffthis` in both parrallely open files to see the difference between the file.



grouped convolution

in it the channel of input is split according to the groups specifed. it is the widening of the network then concatenation of the results. to produce the same result. as a normal convolution.

```
def get_grouped_conv_models(input_filters,groups):
    inputs=tf.keras.Input(shape=(20,20,64))
    x = tf.keras.layers.Conv2D(filters=input_filters, kernel_size=(3,3), strides=(1, 1), padding='same', groups=groups, activation='relu')(inputs)

    group_conv_model=
tf.keras.Model(inputs,x,name="Group_conv_{}".format(input_filters))
    group_conv_model.save("./output/h5/{}.h5".format(group_conv_model.name))

    y=get_grouped_conv(x=inputs,input_filters=input_filters,groups=groups)

decomposed_grouped_conv_model=tf.keras.Model(inputs,y,name="Decomposed_Group_conv_{}".format(input_filters))

decomposed_grouped_conv_model.save("./output/h5/{}.h5".format(decomposed_grouped_conv_model.name))
```

here as one can see the grouped conv splits the input channelwise into specified number of groups. then each channel is convolved over with each filter also split in such a manner producing as many filters as inputs sub slices then these arte convolved each and the result is produced which are then concatenated to get the output of that filter this is repeated for each filter and the output feature map is made