```
from fastai.vision.all import *
```

Problem setup

```
path = untar_data(URLs.IMAGEWOOF_320)
```

```
model = resnet50(pretrained=True).cuda()
```

```
/home/molly/miniconda3/envs/fastai/lib/python3.10/site-packages/torchvision/models/_utils.py
  warnings.warn(
/home/molly/miniconda3/envs/fastai/lib/python3.10/site-packages/torchvision/models/_utils.py
  warnings.warn(msg)
```

```
dls = DataBlock((ImageBlock,CategoryBlock),
                splitter=IndexSplitter([0,2]),
                item_tfms=Resize(320),
                batch_tfms=aug_transforms(),
                get_items=get_image_files).dataloaders(path)
```

```
def model_call(model,x):
    #same as model.forward
        x = model.conv1(x)
        x = model.bn1(x)
        x = model.relu(x)
        x = model.maxpool(x)

        x = model.layer1(x)
        x = model.layer2(x)
        x = model.layer3(x)
        x = model.layer4(x)

        #x = model.avgpool(x)
        #x = torch.flatten(x, 1)
        #x = model.fc(x)
        return x
```

```
model.fc
```

```
Linear(in_features=2048, out_features=1000, bias=True)
```

```
#model_call(model,dls.one_batch()[0])
```

Going to use a relatively small number of buckets, could use much more if we wanted to have less total collisions.

```
nbits=8
buckets=2**nbits
bits=torch.arange(0,8,device='cuda')
nbits,buckets,2.**bits
```

```
(8,
 256,
 tensor([  1.,   2.,   4.,   8.,  16.,  32.,  64., 128.], device='cuda:0'))
```

```
bits
```

```
tensor([0, 1, 2, 3, 4, 5, 6, 7], device='cuda:0')
```

```
torch.sum(2**bits)
```

```
tensor(255, device='cuda:0')
```

How would we calculate the maximum value that we can represent?

```
[64, 2048, 10, 10]
```

```
[64, 2048, 10, 10]
```

```
[64, 8, 10, 10]
```

```
[64, 8, 10, 10]
```

```
(2.**bits).sum()
```

```
tensor(255., device='cuda:0')
```

```
#dividing by nbits gets us close to 1
projector=torch.randn(2048,nbits).cuda()/nbits**0.5
torch.linalg.norm(projector,dim=1)
```

```
tensor([1.0337, 1.0348, 1.0620,  ..., 1.0139, 1.1947, 1.1155], device='cuda:0')
```

```
nbits
```

```
8
```

```
[0,1,0,0,0], [0,0,1,0,0],[1,0,0,0,0]
```

```
([0, 1, 0, 0, 0], [0, 0, 1, 0, 0], [1, 0, 0, 0, 0])
```

```
b=dls.one_batch()[0]
```

```
dls.show_batch()
```

```
ys=model_call(model,b)
```

```
(ys[:,:,None]*projector[None,...,None,None]).shape
```

```
torch.Size([64, 2048, 8, 10, 10])
```

```
ys.shape,projector.shape
```

```
(torch.Size([64, 2048, 10, 10]), torch.Size([2048, 8]))
```

```
torch.einsum('ijkl,jm->imkl',ys,projector).shape
```

```
torch.Size([64, 8, 10, 10])
```

```
(torch.einsum('ijkl,jm->ijmkl',ys,projector)==(ys[:,:,None]*projector[None,...,None,None])
```

```
TensorImage(True, device='cuda:0')
```

How would you calculate the dot product for the second dim and first dim? (2048,10,10)@(2048,8)=(8,10,10)

Using Einstein notation?

```
projection=torch.einsum('bcxy,cz->bzxy',ys,projector)
```

```
projection
```

```
TensorImage([[[[-6.2792e+00,  1.3167e-01, -9.4779e-02,  ..., -2.4656e+00,
                -5.0584e+00, -1.3296e+00],
               [ 8.1754e+00,  9.6036e+00, -4.4357e+00,  ..., -4.6396e+00,
                -7.8996e+00, -2.8259e+00],
               [ 4.4467e+00, -2.1758e+00, -6.7790e+00,  ...,  4.6684e+00,
                 1.3650e+01,  1.0253e+01],
               ...,
               [ 6.3236e-01, -3.8342e+00, -1.3020e+01,  ..., -4.5836e+00,
```

```
  -3.6719e+00,  4.8875e+00],
 [-9.5392e+00, -4.3794e+00, -4.9069e+00,  ...,  4.2450e+00,
  -7.0259e+00, -6.6205e+00],
 [-2.8886e+00, -2.2777e+00, -3.5640e+00,  ..., -1.4655e+00,
  -5.7932e+00, -2.0072e+00]],

[[ 9.2835e+00,  7.7462e+00,  6.1908e+00,  ...,  2.9067e+00,
   8.0073e+00,  1.1350e+01],
 [ 1.1556e+01,  1.3565e+01,  9.3358e+00,  ...,  3.8952e+00,
   8.9265e+00,  1.1539e+01],
 [ 9.3274e+00,  7.3811e+00,  1.1513e+01,  ...,  1.6263e+01,
   2.4063e+01,  1.9001e+01],
 ...,
 [ 6.8415e+00,  5.9298e+00,  7.7832e+00,  ...,  2.8651e+01,
   4.1376e+00, -4.5388e+00],
 [ 5.3508e+00,  1.1588e+01,  1.2607e+01,  ...,  1.5089e+01,
   7.7975e+00,  5.8521e+00],
 [ 9.5215e+00,  6.9118e+00,  1.0100e+01,  ...,  8.9737e+00,
   6.7271e+00,  8.5211e+00]],

[[-3.1688e+00, -6.4222e+00, -2.8567e+00,  ..., -3.6577e+00,
  -5.9065e+00, -2.9208e+00],
 [-6.0068e+00, -9.4032e+00, -2.8770e+00,  ..., -6.6112e-01,
  -5.0611e+00, -8.4874e+00],
 [-4.4515e+00, -1.0154e+01, -1.2417e+00,  ..., -1.5574e-01,
  -1.1492e+00,  7.6063e+00],
 ...,
 [ 1.5988e+01,  1.3046e+01,  6.4920e+00,  ..., -4.4126e+00,
   1.4175e+00,  1.2447e+01],
 [ 1.9409e+01,  1.7188e+01,  6.8644e+00,  ..., -3.7321e+00,
  -2.8769e+00,  1.6388e+00],
 [ 1.6584e+01,  1.4844e+01,  9.3730e+00,  ..., -8.5079e-01,
  -6.0974e+00, -3.7285e+00]],

...,

[[-7.1310e+00,  6.5308e-01,  2.6390e+00,  ..., -4.4481e+00,
  -7.0668e+00, -9.1038e+00],
 [ 2.2496e+00,  5.2074e+00,  1.0821e+01,  ..., -3.1807e+00,
   8.5318e-01, -1.1862e+00],
 [-5.3933e+00,  3.0677e+00,  6.6602e+00,  ..., -4.9860e+00,
   1.2720e+00,  1.7180e+01],
 ...,
```

```
  [-3.8841e+00, -1.0031e+00,  1.4554e+01,  ...,  1.7657e+01,
    8.1946e+00,  8.0810e+00],
  [-7.9001e+00, -9.9230e+00,  1.9062e+00,  ...,  2.0380e+00,
   -1.1232e+00,  2.4761e+00],
  [-1.1327e+01, -1.5734e+01, -1.2547e+01,  ..., -1.1905e+00,
    3.4819e+00,  1.7690e+00]],

 [[-5.7783e+00,  1.5963e-01,  3.8980e+00,  ...,  1.9761e+00,
    1.8847e+00,  2.4780e-01],
  [-1.7248e+00, -3.0383e+00, -6.9456e+00,  ...,  8.2464e+00,
   -2.3109e+00, -2.6400e+00],
  [ 3.1076e-01,  1.4344e+00, -1.1225e+01,  ...,  5.7751e+00,
    1.9494e+00,  3.9672e+00],
  ...,
  [-1.1729e+01, -1.0031e+01, -2.1186e+01,  ..., -7.7848e+00,
   -1.1644e+01, -1.2071e+01],
  [-1.5569e+01, -2.1363e+01, -2.6422e+01,  ..., -1.6087e+01,
   -1.9381e+01, -2.5736e+01],
  [-2.8877e+01, -2.3298e+01, -1.9927e+01,  ..., -1.1429e+01,
   -1.3280e+01, -2.2741e+01]],

 [[-9.2639e+00, -3.3929e+00,  9.7048e+00,  ..., -3.8852e+00,
    8.4170e-01,  1.7012e+00],
  [-2.6492e+00,  3.2599e+00,  3.2146e+01,  ...,  2.9210e+00,
    2.3555e-01,  3.9248e-01],
  [ 1.1340e+01,  1.2774e+01,  2.0421e+01,  ..., -7.9307e+00,
   -1.6897e+01, -4.7462e+00],
  ...,
  [ 9.6164e+00,  1.3184e+01,  1.3542e+01,  ...,  7.6629e+00,
    1.3888e+01,  1.5293e+01],
  [ 1.0481e+01,  1.4385e+01,  1.1060e+01,  ...,  4.2098e+00,
    1.1673e+01,  1.5041e+01],
  [ 4.5687e+00,  1.4530e+01,  1.3015e+01,  ...,  9.6600e+00,
    1.2996e+01,  1.3536e+01]]],


[[[ 9.5124e+00,  7.0540e+00,  1.0731e+01,  ..., -1.1898e-01,
   -4.8526e+00, -1.6273e+01],
  [ 1.6990e-01, -1.3911e-01,  6.2190e+00,  ..., -6.3489e-01,
   -1.0237e+01, -8.5378e+00],
  [-3.7613e+00, -4.9234e+00, -2.4806e+00,  ..., -2.5140e+00,
   -7.3274e+00, -7.1875e+00],
  ...,
```

```
 [-6.5095e+00,  1.7462e+00, -1.9302e-01,  ..., -9.5685e+00,
  -7.6806e+00, -8.3060e-01],
 [-9.2985e-01, -6.4332e+00,  6.1271e+00,  ..., -1.7758e+01,
  -1.5903e+01, -1.8803e+01],
 [ 9.8972e+00,  1.4579e+01,  8.9024e+00,  ..., -2.1343e+01,
  -3.3487e+01, -2.7978e+01]],

[[ 9.7840e+00,  1.3461e+01,  2.7833e+01,  ...,  5.9922e+00,
    1.1757e+01,  2.2186e+01],
 [ 6.5224e+00,  6.7404e+00,  1.3893e+01,  ...,  9.1900e+00,
    1.8095e+01,  1.9578e+01],
 [ 8.7155e-01,  1.9546e+00,  8.3535e+00,  ...,  1.5843e+01,
    1.7649e+01,  1.1515e+01],
 ...,
 [-1.0384e+01, -1.4986e+01,  5.2173e+00,  ...,  3.0871e+01,
    4.3548e+01,  4.0713e+01],
 [ 1.8919e+01,  1.2526e+01,  2.4128e+00,  ...,  2.9831e+01,
    4.2634e+01,  3.9587e+01],
 [ 3.8178e+01,  3.2932e+01,  4.6763e+00,  ...,  3.2836e+01,
    3.7258e+01,  4.4807e+01]],

[[-6.7811e+00, -1.3298e+01, -2.0355e+00,  ..., -4.4833e+00,
  -6.0731e+00, -5.2660e+00],
 [-4.5190e+00, -4.7397e+00, -4.3055e+00,  ..., -8.3795e+00,
  -8.6657e+00, -3.1813e+00],
 [-4.0083e+00, -5.4050e+00, -8.3738e+00,  ..., -8.1771e+00,
  -5.7712e+00,  3.3252e+00],
 ...,
 [-2.0961e+01, -1.7522e+01, -1.9436e+01,  ..., -3.9490e+00,
    3.5565e+00,  4.4478e+00],
 [-2.4070e+01, -2.2667e+01, -7.1590e+00,  ..., -3.1348e+00,
    5.0762e+00,  1.2960e+01],
 [-7.6899e+00, -8.6212e+00,  6.7536e-01,  ..., -2.6530e-01,
    1.1745e+01,  1.5273e+01]],

...,

[[ 9.2374e-01,  3.2793e+00, -6.5183e+00,  ...,  4.7281e-01,
  -2.2870e+00,  1.3027e+00],
 [ 4.0763e+00,  3.0901e+00,  4.1876e+00,  ...,  2.0954e+00,
  -9.8625e-01, -3.1275e+00],
 [ 9.6910e-01,  2.8828e-01,  7.4595e-01,  ...,  1.0479e+01,
    8.2900e+00,  4.0798e+00],
```

```
 ...,
 [ 1.2553e+01,  1.4372e+01,  1.3448e+01,  ...,  1.2941e+01,
   7.6967e+00,  1.1426e+01],
 [ 1.1466e+01,  1.4855e+01,  2.0851e+01,  ...,  3.2110e+01,
   4.5123e+01,  4.3206e+01],
 [ 1.0600e+01,  7.3636e+00,  1.0523e+01,  ...,  1.6592e+01,
   2.7320e+01,  5.5086e+01]],

[[ 2.4515e+00,  1.0092e+01,  1.2336e+01,  ...,  6.2933e+00,
   4.9184e+00,  9.3574e+00],
 [ 1.2444e+00,  3.0530e+00,  7.1421e+00,  ...,  5.0260e+00,
   6.4202e+00, -1.7965e+00],
 [-2.0972e+00,  3.1371e-01,  8.9068e-01,  ...,  2.6291e+00,
   3.4665e+00, -1.9193e+00],
 ...,
 [ 1.5958e+00, -6.7022e+00, -5.6058e+00,  ...,  6.6774e+00,
   4.8079e+00,  3.4565e+00],
 [-4.5736e-02, -1.8359e+01, -7.8809e+00,  ...,  6.7820e+00,
   3.7120e+00,  4.0987e+00],
 [ 1.4360e+00, -5.5607e+00,  6.2634e-01,  ..., -1.5635e+00,
  -1.0811e+01, -2.0711e+00]],

[[ 2.8068e+00,  3.3115e+00,  7.0982e+00,  ...,  2.5611e+00,
  -1.5446e+00, -7.0554e+00],
 [ 3.2848e+00, -2.6821e+00,  4.3762e+00,  ...,  1.6894e+00,
  -1.6904e+00, -6.2212e+00],
 [-4.2415e-01, -3.8746e+00, -4.9796e-01,  ..., -3.5530e-01,
   2.7174e+00,  1.3833e+01],
 ...,
 [ 6.1781e+00,  3.1638e+00,  7.9841e+00,  ..., -9.4526e+00,
  -3.6841e+00, -1.4651e+01],
 [-6.8839e+00, -3.6912e+00, -3.6544e+00,  ..., -1.6174e+01,
  -2.0845e+01, -2.3711e+01],
 [-3.4264e+00, -5.4144e+00, -1.1610e-01,  ..., -1.2922e+01,
  -1.8294e+01, -2.1884e+01]]],


[[[ 1.6726e+01,  1.7428e+01,  4.2822e+00,  ...,  1.3308e+01,
    7.7680e+00,  1.4643e-01],
  [ 1.0972e+01,  1.4134e+01,  4.4387e+00,  ...,  1.5983e+01,
    7.5606e+00,  2.1118e+00],
  [ 8.8190e+00,  7.4110e+00,  1.1246e+01,  ...,  1.3696e+01,
    3.8566e+00,  6.2899e+00],
```

```
 ...,
 [ 1.2071e+01,  1.0601e+01,  1.6731e+00,  ..., -1.3564e+01,
  -1.3250e+01,  3.3810e-01],
 [ 1.8464e+00,  7.9721e+00,  3.5149e+00,  ..., -1.6235e+01,
  -1.0623e+01,  8.7157e+00],
 [ 8.7276e+00, -9.2056e-01,  7.5868e-01,  ..., -1.1893e+01,
    2.7289e+00,  2.7667e+01]],

[[ 6.6531e+00,  1.0010e+01,  2.2753e+01,  ...,  1.4925e+01,
    1.3476e+01,  1.6543e+01],
 [ 4.5088e+00,  1.9536e+01,  3.0127e+01,  ...,  1.1684e+01,
    9.8105e+00,  3.2875e+00],
 [ 4.2340e+00,  5.5834e+00,  1.8127e+01,  ..., -3.0256e+00,
    2.6275e+00, -7.1714e-01],
 ...,
 [ 1.6779e+01,  2.9478e+00,  1.5335e+01,  ...,  3.1999e+01,
    1.1919e+01,  8.4015e+00],
 [ 1.5926e+01,  2.0441e+00,  7.7098e+00,  ...,  1.2695e+01,
    1.0315e+01,  5.9256e+00],
 [ 4.5396e+00, -3.9510e+00,  3.0082e+00,  ...,  4.2423e+00,
    1.4620e+00, -8.1835e-01]],

[[-1.7676e+01, -1.7258e+01, -1.1684e+01,  ..., -1.0038e+01,
  -7.8505e-02,  5.6030e-01],
 [-1.2038e+01, -1.0079e+01,  3.9319e-01,  ..., -1.3313e+00,
  -2.6005e+00,  6.5192e+00],
 [-5.7970e+00, -1.0169e+01, -1.4580e+01,  ...,  3.7920e+00,
    2.1885e+00, -2.9092e+00],
 ...,
 [-9.1141e+00, -1.7944e+01, -2.5657e+00,  ..., -2.0504e+01,
  -8.4545e+00,  3.7092e+00],
 [-4.9958e+00, -9.3027e+00, -7.6127e+00,  ..., -1.3733e+01,
  -7.3914e+00,  1.0248e+01],
 [-2.4956e+00, -4.2638e+00, -3.6001e+00,  ..., -5.5006e-01,
    2.5708e+00,  5.7056e+00]],

...,

[[ 4.2114e-01, -7.9980e+00, -8.6686e+00,  ..., -4.2628e+00,
    8.8063e+00, -9.2941e-02],
 [-8.4785e+00, -1.7714e+01, -5.5977e-01,  ...,  9.8289e-01,
    1.2428e+01,  9.1162e+00],
 [-9.5216e+00, -2.6300e+01, -1.7297e+01,  ...,  4.7637e+00,
```

```
    6.2845e+00,  4.5398e+00],
   ...,
   [ 1.2768e+00, -6.3151e+00, -1.0408e+00,  ...,  7.4808e-01,
    -1.0534e+00,  4.4392e+00],
   [ 3.0409e+00, -1.3620e+00,  3.0350e+00,  ..., -9.6885e+00,
    -2.5018e+00, -3.8327e+00],
   [-1.6695e+01, -1.1960e+01, -6.9895e+00,  ..., -8.2594e+00,
    -1.4068e+01, -2.0667e+01]],

  [[ 5.1652e+00,  1.0395e+01,  7.4535e+00,  ...,  8.6804e+00,
     4.5424e+00,  1.4589e+01],
   [ 4.9551e+00,  1.8278e+01,  2.8135e+01,  ...,  2.0059e+01,
     1.2688e+01,  1.1655e+01],
   [ 8.6485e+00,  2.4904e+01,  4.3337e+01,  ...,  1.4686e+01,
     1.6079e+01,  2.0526e+01],
   ...,
   [ 3.6101e+00, -3.5364e+00,  7.2125e+00,  ..., -8.2165e-01,
    -7.4522e+00,  6.1776e+00],
   [-3.6109e+00, -6.7193e+00,  1.2664e+01,  ..., -5.5758e+00,
    -7.2992e+00,  3.7080e+00],
   [-9.0229e-01, -2.1786e+00,  7.6342e+00,  ..., -7.0221e+00,
    -9.6212e+00,  5.1720e+00]],

  [[-1.1960e+01, -1.1723e+01, -2.1839e+01,  ..., -7.9723e+00,
     9.2299e+00,  5.2175e+00],
   [-1.4355e+01, -2.1408e+01, -3.0463e+01,  ...,  3.5703e+00,
     4.7135e+00,  7.2011e+00],
   [-1.3361e+01, -2.6255e+01, -3.8794e+01,  ...,  2.9431e+00,
    -4.4089e+00,  1.5832e-01],
   ...,
   [-4.0332e+00, -1.0877e+01, -1.0693e+01,  ..., -7.4630e+00,
    -1.3602e-01,  1.2742e+00],
   [-3.7040e+00, -2.8553e+00, -8.7539e+00,  ...,  6.7736e-01,
     2.3848e+00, -1.8044e-01],
   [-8.3037e+00, -1.0215e+01, -8.9308e+00,  ...,  1.6606e+01,
     1.4776e+00, -7.1054e+00]]],


 ...,


 [[[ 3.0742e+01,  2.0617e+01,  3.4883e+00,  ..., -9.0017e+00,
    -1.6916e+00,  3.9473e-01],
```

```
 [ 3.5029e+01,  2.8505e+01,  1.3211e+01,  ..., -5.5471e+00,
  -6.9257e+00, -2.2475e+00],
 [ 1.4213e+01,  2.3229e+01,  1.6051e+01,  ..., -1.8002e-01,
  -3.7035e-01,  6.3750e+00],
 ...,
 [ 8.4374e-01,  4.3312e+00, -3.1439e-01,  ..., -4.4065e+00,
  -1.1842e+00,  3.8927e+00],
 [-1.3196e+00, -4.4678e-01, -2.1221e+00,  ..., -2.8409e+00,
  -4.0868e+00,  2.5064e+00],
 [-3.6652e-01, -9.4469e-01, -9.2601e+00,  ..., -3.7837e+00,
  -1.7367e+00,  8.7911e+00]],

[[ 2.1170e+01,  1.9605e+01,  8.4797e+00,  ...,  1.7911e+01,
    8.9692e+00,  6.8814e+00],
 [ 1.5301e+01,  1.3120e+01,  1.9084e+00,  ...,  1.5672e+01,
    1.0492e+01,  9.9011e+00],
 [ 4.6515e+00, -1.0032e+00,  1.7698e+00,  ...,  9.0264e+00,
    7.3372e+00,  2.9377e+00],
 ...,
 [ 9.4733e+00,  1.7173e+01,  1.2714e+01,  ...,  2.1929e+01,
    1.6661e+01,  1.2190e+01],
 [ 4.9162e+00,  1.5067e+01,  1.7427e+01,  ...,  1.6764e+01,
    1.1263e+01,  1.2861e+01],
 [ 3.4370e+00,  1.0124e+01,  1.2676e+01,  ...,  2.0788e+01,
    1.6364e+01,  2.2770e+01]],

[[ 1.7073e+01,  1.2411e+01,  3.9940e+00,  ..., -2.5094e-01,
    1.0745e+00,  2.0860e+00],
 [ 9.7372e+00,  2.9299e+00, -5.5394e+00,  ...,  1.9889e+00,
    3.3939e+00, -1.5617e+00],
 [-2.5642e-01, -1.1352e+00, -8.8535e+00,  ..., -4.5452e+00,
  -1.9397e+00, -7.2823e+00],
 ...,
 [-2.0316e+00, -2.9228e+00, -4.4708e+00,  ..., -3.9330e+00,
  -1.1184e+01, -9.1596e+00],
 [ 4.0511e-01, -5.1108e-01, -5.8130e+00,  ..., -3.4118e+00,
  -4.2866e+00, -2.3001e+00],
 [-3.2900e+00, -7.9968e+00, -4.4886e+00,  ..., -4.5846e+00,
  -5.1580e+00, -5.1870e+00]],

...,

[[-1.0441e+01,  3.6222e-01,  5.7083e+00,  ..., -8.7860e-01,
```

```
  -5.9626e+00, -2.4280e+00],
 [-3.2910e-01,  3.1008e+00,  1.1946e+01,  ...,  2.7110e+00,
  -5.6768e+00, -6.1909e+00],
 [-5.4021e+00,  1.0762e+01,  2.7438e+01,  ...,  5.8677e+00,
  -1.0364e-01, -2.4942e+00],
 ...,
 [ 7.0567e-01, -3.1743e+00, -1.8512e+00,  ...,  4.5055e+00,
   1.3033e+00, -8.2645e+00],
 [-3.0940e+00,  2.9549e+00,  1.0386e+01,  ..., -4.2258e+00,
  -8.3496e+00, -6.9864e+00],
 [-5.8167e+00,  5.5948e+00,  2.3419e+01,  ..., -6.6175e+00,
  -1.2134e+01, -7.1784e+00]],

[[-1.5039e+01, -5.6994e+00,  6.0646e+00,  ...,  4.8761e+00,
  -1.6647e+00, -3.3653e+00],
 [-1.2232e+01,  3.2679e-01,  2.1025e+00,  ...,  4.5207e+00,
  -6.3115e-01,  4.4318e-01],
 [-1.6360e+01, -4.2183e-01, -2.9451e+00,  ..., -3.2819e+00,
  -7.8153e+00, -3.9869e+00],
 ...,
 [-9.9416e+00,  1.5021e+01,  1.2003e+01,  ..., -5.2349e+00,
   2.6565e+00, -2.1975e+00],
 [-1.3382e+01,  2.7150e+00,  7.4891e+00,  ..., -5.7495e+00,
   8.4802e+00,  6.6035e+00],
 [-1.3810e+01,  7.6381e+00,  5.1958e+00,  ..., -1.8646e+00,
   3.0005e+00,  9.3771e-01]],

[[ 1.3810e+01,  1.7808e+01,  8.6061e+00,  ...,  1.1546e+01,
   8.8377e+00,  8.1417e+00],
 [ 1.1785e+01,  1.7363e+01, -4.3577e+00,  ...,  3.6378e+00,
   1.6090e+01,  1.5553e+01],
 [ 1.0965e+01,  2.3075e+00, -2.3853e+00,  ..., -8.7184e+00,
   1.7130e+01,  1.8968e+01],
 ...,
 [ 2.2361e+00,  2.7280e+00,  6.5013e+00,  ...,  5.6273e+00,
   8.9686e+00,  9.0201e+00],
 [ 5.5273e+00,  3.4220e+00, -9.4277e+00,  ...,  8.6742e-01,
   7.3521e+00,  1.1186e+01],
 [ 4.3832e+00, -6.7473e-01, -1.1554e+01,  ...,  6.7014e+00,
   8.0967e+00,  7.0652e+00]]],


[[[-2.6913e+01, -2.9568e+01, -8.9120e+00,  ..., -3.6777e+00,
```

```
  -9.5993e+00, -1.3907e+01],
 [-3.3114e+01, -4.0868e+01, -3.1856e+01,  ..., -2.1573e+00,
  -7.5814e+00, -1.1474e+01],
 [-2.7869e+01, -3.9983e+01, -2.6464e+01,  ...,  6.0545e+00,
  -6.4322e+00, -6.3355e+00],
 ...,
 [-1.6259e+01, -1.7102e+01, -9.8119e+00,  ..., -2.6094e+00,
  -9.1046e+00, -4.5698e+00],
 [-4.2792e+00, -1.4512e+01, -7.5014e+00,  ..., -3.2948e-01,
  -8.0459e+00, -5.2389e+00],
 [-2.2665e-03, -1.0965e+01, -7.7879e+00,  ..., -9.6266e+00,
  -1.0545e+01, -7.5697e-01]],

[[ 1.5671e+00,  1.2056e+01,  1.2069e+01,  ...,  2.4591e+01,
   3.2034e+01,  3.6939e+01],
 [-3.8125e+00,  3.2298e+00,  1.2571e+01,  ...,  2.2087e+01,
   2.9005e+01,  1.8920e+01],
 [ 7.9042e-01,  2.6120e+01,  3.5221e+01,  ...,  3.8826e+00,
   1.6314e+01,  2.7712e+01],
 ...,
 [-2.7849e+00, -1.0391e+01, -5.8262e+00,  ...,  6.0313e+00,
   6.1854e+00,  7.3942e+00],
 [ 8.9106e+00,  6.6072e+00,  3.7478e+00,  ...,  4.4233e+00,
  -1.8716e-01, -3.8388e-01],
 [ 2.1626e+01,  1.3852e+01,  1.0201e+01,  ...,  3.2672e+00,
  -1.1369e+00,  3.3530e+00]],

[[ 8.4911e+00,  8.4341e+00,  4.6288e+00,  ...,  1.3576e+01,
  -7.2182e+00, -1.2758e+01],
 [ 2.1120e+01,  1.2484e+01,  8.6090e-02,  ..., -1.1351e+00,
  -1.4208e+01, -1.1975e+01],
 [-3.1700e+00, -4.0954e+00, -1.6492e+00,  ..., -1.0604e+01,
  -9.4700e+00,  6.6750e-01],
 ...,
 [ 2.9052e+00,  6.3958e+00,  1.3706e-01,  ...,  5.7390e+00,
   3.1581e+00, -6.0765e-01],
 [-3.8417e+00, -4.2816e+00, -5.9917e-01,  ...,  6.7638e+00,
   5.0139e+00,  5.6882e+00],
 [-1.8144e+01, -1.2465e+01, -4.2393e+00,  ...,  4.2922e+00,
   6.0590e+00,  9.7644e+00]],

...,
```

```
[[-7.3576e+00,  1.0356e+01,  3.1724e-01,  ...,  1.8843e+01,
   1.8283e+01,  1.0810e+01],
 [-7.3645e+00,  5.2899e+00,  1.3292e+00,  ...,  1.3994e+01,
   8.5479e+00, -4.9072e+00],
 [-9.9293e+00,  2.7770e+00, -8.8405e+00,  ...,  3.1427e+00,
   1.9661e+00,  8.4572e+00],
 ...,
 [-8.4440e+00, -5.7747e+00, -7.7229e+00,  ...,  8.1245e+00,
  -3.8143e+00, -9.9999e+00],
 [ 6.4093e-01,  5.9969e+00, -6.3158e+00,  ...,  1.4593e+00,
  -5.8362e+00, -1.3630e+01],
 [ 4.7849e+00,  4.6974e+00,  3.1298e+00,  ..., -4.9041e+00,
  -1.0216e+01, -1.2580e+01]],

[[-5.8014e+00, -1.0372e+01,  1.0254e+01,  ..., -4.6294e-01,
   1.8217e+01,  1.0038e+01],
 [ 5.7797e+00,  9.2750e+00,  2.1096e+01,  ...,  8.8768e+00,
   2.2236e+01,  2.0851e+01],
 [ 1.4236e+01,  3.3656e+01,  3.3081e+01,  ...,  2.2807e+01,
   3.4478e+01,  1.7018e+01],
 ...,
 [ 1.9373e+01,  1.7202e+01,  7.0146e-01,  ...,  9.4769e+00,
   5.3170e+00,  7.5832e+00],
 [ 2.3346e+01,  2.6482e+01,  1.0779e+01,  ...,  1.0170e+00,
   9.9077e+00,  6.9110e+00],
 [ 1.7564e+01,  2.1025e+01,  1.2639e+01,  ...,  4.6908e+00,
   5.8073e+00,  3.1757e+00]],

[[-5.2906e+00, -2.7127e+00,  1.5336e+01,  ..., -3.3235e+00,
   3.0431e+00,  4.4982e+00],
 [-3.8740e+00, -2.7825e-01,  1.8444e+01,  ..., -6.8304e+00,
  -9.7458e+00, -2.2539e+00],
 [-2.3341e+00,  1.1108e+00,  9.1996e+00,  ...,  1.4541e+00,
   4.0044e+00,  5.3693e+00],
 ...,
 [-5.5670e+00, -3.2424e+00, -1.0724e+01,  ..., -2.3431e+00,
   3.1732e+00,  2.2250e+00],
 [ 4.7924e+00,  2.9135e+00,  1.0695e+00,  ..., -3.1337e+00,
   3.6863e+00,  9.4232e+00],
 [ 1.3064e+01,  1.4679e+00,  4.2512e+00,  ...,  2.0431e+00,
   5.5197e+00,  9.0792e+00]]],
```

```
[[[-4.1837e+00,  4.4228e+00,  7.5466e+00,  ...,  9.0294e+00,
    1.8985e+00,  1.3472e+00],
  [ 3.4190e+00,  7.7751e+00,  1.4765e+01,  ...,  1.0512e+01,
    1.2134e+01,  1.3603e+01],
  [ 1.1120e+01,  1.9633e+01,  1.5074e+01,  ...,  7.3059e+00,
    5.4364e+00,  1.3671e+01],
  ...,
  [ 7.7573e+00,  6.5959e+00,  2.4744e+00,  ...,  1.3797e+01,
    2.7515e+00,  2.0304e+00],
  [ 5.0850e+00,  9.8228e+00,  7.2073e+00,  ...,  3.0873e-01,
   -2.5682e+01, -1.1913e+01],
  [-1.9177e+00,  1.0239e+00,  2.1603e+00,  ...,  7.6998e-02,
   -1.0977e+01, -1.0928e+01]],

 [[ 9.6061e+00,  1.5370e+00,  2.8735e+00,  ...,  5.0786e+00,
    1.1083e+01,  9.2034e+00],
  [ 3.0860e+00,  7.1579e+00,  6.0781e+00,  ...,  7.9867e+00,
    3.1649e+00,  5.9685e+00],
  [ 2.0483e+00,  1.4068e+01,  9.0135e+00,  ...,  4.9285e+00,
    6.5073e+00, -3.5669e+00],
  ...,
  [-1.0659e+00,  4.2513e+00,  3.0158e+00,  ...,  2.1836e+00,
   -7.4681e+00, -1.0047e+01],
  [-6.4479e+00, -1.6330e+00,  7.5435e+00,  ...,  1.3588e+01,
    1.1536e+00, -1.1312e+01],
  [-4.2848e+00,  4.4101e+00,  9.2249e+00,  ...,  1.7328e+01,
    1.4831e+01, -7.3449e-01]],

 [[-1.3682e+00, -5.1753e+00, -4.2238e+00,  ..., -1.3184e+01,
   -1.0707e+01, -9.5371e+00],
  [-1.6860e+00, -3.3730e+00,  3.3757e+00,  ..., -9.6586e+00,
   -4.3802e+00, -4.1719e-02],
  [-4.7046e-01, -4.2123e-01, -3.8035e+00,  ..., -9.6757e+00,
   -1.5742e+01, -1.8032e+00],
  ...,
  [-4.2253e+00, -3.7856e+00, -4.7338e+00,  ...,  6.0688e+00,
   -2.2821e+00,  6.7935e+00],
  [-8.7673e+00, -2.8977e-01, -1.8588e+00,  ...,  7.1490e+00,
   -4.2642e+00, -4.3403e-01],
  [-1.3027e+01, -9.8119e+00, -4.1386e+00,  ..., -3.9124e+00,
   -1.8475e+01, -5.6218e+00]],

 ...,
```

```
  [[-1.3048e+01, -6.4990e+00,  3.7771e+00,  ..., -3.5662e-01,
    -2.1649e+01, -1.9933e+01],
   [-1.3308e+01,  1.0137e+00,  4.0163e+00,  ...,  1.5178e+01,
    -1.8908e+01, -2.6649e+01],
   [-6.0838e+00,  4.4509e+00,  9.2461e+00,  ...,  1.2517e+01,
    -6.9830e+00, -1.0261e+01],
   ...,
   [ 3.8149e+00,  1.0815e+01,  3.1839e+00,  ...,  9.5991e+00,
     3.8434e+00,  2.1540e+00],
   [-2.1296e+00, -1.8112e+00, -2.8965e+00,  ..., -6.4432e+00,
    -6.9710e+00, -5.0703e-01],
   [ 7.2261e+00, -2.4904e+00, -6.8260e+00,  ..., -8.1620e+00,
    -4.7305e+00, -1.3990e+00]],

  [[ 6.4019e+00,  1.4235e+01,  1.1706e+01,  ...,  8.1696e+00,
     1.3590e+01,  1.1112e+01],
   [ 9.0817e+00,  1.8450e+01,  1.8520e+00,  ...,  2.9886e+00,
     1.0353e+01,  1.9502e+01],
   [ 1.2688e+01,  1.2391e+01,  7.0384e+00,  ..., -4.8993e+00,
     4.6831e+00,  1.3654e+01],
   ...,
   [ 6.8772e+00,  4.8467e+00,  3.6678e+00,  ..., -3.3305e+00,
     2.7749e+01,  3.7785e+01],
   [ 1.4010e+01,  1.1408e+01,  5.3943e+00,  ..., -2.1537e+01,
     2.1972e+00,  3.4320e+01],
   [ 1.4877e+01,  1.7063e+01,  6.9275e+00,  ..., -1.4477e+01,
    -3.4017e+00,  3.0101e+01]],

  [[-3.0831e+00,  3.0010e+00,  6.9832e+00,  ..., -1.8217e+00,
    -7.1681e+00, -3.4423e+00],
   [ 8.3796e+00,  3.6513e+00, -2.5611e+00,  ..., -6.8060e+00,
    -1.4611e+01, -8.5084e+00],
   [-6.5332e-01, -1.1911e+01, -1.2325e+01,  ..., -4.0894e+00,
    -1.0735e+01,  1.7551e+00],
   ...,
   [ 3.4406e+00,  1.4224e+00, -3.6680e+00,  ..., -1.0623e+01,
    -2.8591e+01, -1.4722e+01],
   [ 3.2616e+00,  2.6747e+00, -1.9402e+00,  ..., -6.7384e+00,
    -1.2895e+01, -1.5971e+01],
   [ 1.2498e+01,  2.2431e+00, -3.8494e+00,  ..., -9.0071e+00,
    -2.1719e+01, -1.8405e+01]]]], device='cuda:0',
grad_fn=<AliasBackward0>)
```

```
  2**bits, 0-255
```

```
(tensor([  1,   2,   4,   8,  16,  32,  64, 128], device='cuda:0'), -255)
```

```
  (1*(projection>0)).shape,(2**bits[...,None,None]).shape
```

```
(torch.Size([64, 8, 10, 10]), torch.Size([8, 1, 1]))
```

```

```

```
torch.Size([64, 10, 10])
```

```
  projection.shape,(2.**bits).shape
```

```
(torch.Size([64, 8, 10, 10]), torch.Size([8]))
```

```
  (1*(projection>0)).shape,(2**bits).shape
```

```
(torch.Size([64, 8, 10, 10]), torch.Size([8]))
```

```
  2**bits,(1*(projection>0)).shape
```

```
(tensor([  1,   2,   4,   8,  16,  32,  64, 128], device='cuda:0'),
 torch.Size([64, 8, 10, 10]))
```

```
  ((1*(projection>0)[:,:,0,0])*(2**bits))
```

```
TensorImage([[  0,   2,   0,   8,   0,   0,   0,   0],
             [  1,   2,   0,   0,   0,  32,  64, 128],
             [  1,   2,   0,   0,   0,  32,  64,   0],
             [  0,   2,   0,   0,   0,   0,   0,   0],
             [  0,   2,   4,   8,  16,   0,   0, 128],
             [  0,   2,   0,   8,   0,   0,   0, 128],
             [  0,   2,   4,   0,  16,   0,  64, 128],
```

```
[  0,   2,   4,   0,   0,  32,  64, 128],
[  0,   2,   0,   0,   0,  32,   0,   0],
[  1,   2,   0,   0,   0,   0,  64, 128],
[  1,   2,   0,   0,   0,  32,   0, 128],
[  0,   2,   0,   0,   0,   0,  64, 128],
[  1,   2,   0,   0,   0,   0,  64,   0],
[  0,   2,   0,   0,   0,  32,  64, 128],
[  1,   2,   0,   0,   0,  32,   0,   0],
[  0,   2,   4,   8,   0,   0,   0,   0],
[  1,   2,   0,   0,   0,  32,  64, 128],
[  0,   2,   4,   8,  16,   0,  64, 128],
[  0,   2,   4,   0,   0,  32,   0,   0],
[  0,   2,   0,   8,   0,   0,   0, 128],
[  1,   2,   4,   0,   0,  32,  64,   0],
[  0,   2,   0,   0,   0,   0,  64,   0],
[  1,   2,   4,   0,   0,  32,   0,   0],
[  0,   2,   0,   0,  16,   0,   0, 128],
[  1,   2,   0,   0,  16,   0,   0, 128],
[  0,   2,   4,   0,  16,   0,  64, 128],
[  0,   2,   4,   8,  16,   0,  64, 128],
[  0,   2,   4,   8,  16,   0,  64, 128],
[  0,   2,   0,   0,  16,  32,   0,   0],
[  1,   2,   0,   0,   0,   0,  64, 128],
[  1,   2,   0,   0,   0,   0,   0,   0],
[  0,   2,   4,   8,   0,   0,   0, 128],
[  1,   2,   0,   0,   0,  32,   0, 128],
[  1,   2,   0,   0,   0,   0,   0, 128],
[  0,   2,   4,   8,   0,   0,  64,   0],
[  0,   2,   4,   8,   0,   0,  64,   0],
[  0,   2,   0,   8,   0,   0,   0, 128],
[  1,   2,   0,   8,   0,  32,  64,   0],
[  0,   0,   4,   8,  16,   0,  64, 128],
[  0,   0,   4,   8,  16,  32,   0, 128],
[  1,   0,   0,   0,   0,  32,  64,   0],
[  0,   2,   4,   8,  16,   0,  64, 128],
[  0,   2,   0,   0,  16,   0,   0, 128],
[  1,   2,   4,   8,   0,   0,   0, 128],
[  1,   2,   0,   0,   0,  32,  64,   0],
[  0,   2,   0,   8,  16,   0,  64, 128],
[  0,   2,   0,   0,   0,  32,   0,   0],
[  0,   2,   4,   0,   0,   0,   0, 128],
[  1,   0,   0,   0,   0,  32,  64,   0],
[  1,   2,   0,   0,   0,  32,   0, 128],
```

```
[  0,   2,   4,   0,  16,  32,   0, 128],
[  1,   2,   0,   0,  16,   0,  64, 128],
[  0,   0,   0,   0,  16,  32,  64,   0],
[  1,   2,   4,   0,  16,   0,  64, 128],
[  0,   2,   0,   0,   0,   0,  64,   0],
[  0,   2,   0,   8,   0,   0,  64,   0],
[  0,   2,   0,   0,   0,   0,   0, 128],
[  0,   2,   4,   8,   0,   0,  64,   0],
[  1,   2,   0,   8,   0,   0,  64,   0],
[  1,   2,   0,   0,   0,  32,  64,   0],
[  0,   2,   4,   8,  16,   0,  64,   0],
[  1,   2,   4,   8,  16,   0,   0, 128],
[  0,   2,   4,   0,  16,   0,   0,   0],
[  0,   2,   0,   8,  16,   0,  64,   0]], device='cuda:0')
```

```
torch.einsum('ijkl,j->ikl',1.*(projection>0),(2.**bits))
```

```
TensorImage([[[ 10.,  99., 226.,  ...,  66., 194., 194.],
              [ 35., 163., 162.,  ..., 194., 162., 130.],
              [211., 242., 178.,  ...,  67., 115., 119.],
              ...,
              [151., 134., 166.,  ..., 162., 166., 165.],
              [134., 134., 166.,  ..., 163., 146., 182.],
              [150., 150., 150.,  ..., 146., 178., 178.]],

             [[227., 243., 211.,  ..., 226.,  66., 106.],
              [227.,  98., 227.,  ..., 226.,  82.,  18.],
              [ 34.,  98.,  98.,  ..., 114., 242., 166.],
              ...,
              [232., 169., 170.,  ...,  98., 102., 110.],
              [ 42.,  42.,  43.,  ...,  98., 102., 102.],
              [ 99.,  35., 111.,  ...,  42.,  38.,  38.]],

             [[ 99.,  83.,  67.,  ...,  67., 227., 215.],
              [ 67.,  67.,  87.,  ..., 227., 243., 231.],
              [ 67.,  67.,  83.,  ..., 229., 119., 241.],
              ...,
              [ 99.,   3.,  67.,  ...,  34.,   2., 231.],
              [ 35.,   3., 107.,  ..., 130., 130.,  71.],
              [  3.,   8.,  75.,  ..., 130., 135.,  69.]],
```

```
          ...,

          [[159., 183., 231.,  ..., 194., 134., 151.],
           [151., 247.,  99.,  ..., 246., 150., 210.],
           [147., 161.,  35.,  ...,  34., 146., 147.],
           ...,
           [179., 195., 194.,  ..., 162., 226., 131.],
           [150., 226.,  98.,  ..., 130., 194., 195.],
           [146.,  98.,  98.,  ..., 130., 194., 195.]],

          [[ 22.,  54., 246.,  ...,  38., 226., 226.],
           [ 68., 102., 246.,  ...,  98.,  98.,  66.],
           [ 82., 226., 210.,  ..., 227., 226., 230.],
           ...,
           [ 68.,  68.,  68.,  ..., 102., 198., 202.],
           [226., 226., 194.,  ..., 102., 196., 204.],
           [226., 226., 226.,  ..., 222., 212., 214.]],

          [[ 90., 195., 227.,  ...,  91.,  83.,  83.],
           [203., 235., 119.,  ..., 115.,  75.,  75.],
           [ 75., 115., 115.,  ...,  51.,  83., 201.],
           ...,
           [249., 243., 115.,  ...,  63., 105., 109.],
           [193., 193.,  91.,  ...,  31.,  90.,  64.],
           [232., 203.,  91.,  ...,  27.,  18.,  64.]]], device='cuda:0')
```

How would we calculate the bucket for each x,y location?

```
hash_loc=torch.einsum('bchw,c->bhw',(projection>0).float(),2.**bits).int()
hash_loc.shape,hash_loc
```

```
(torch.Size([64, 10, 10]),
 TensorImage([[[ 10,  99, 226,  ...,  66, 194, 194],
               [ 35, 163, 162,  ..., 194, 162, 130],
               [211, 242, 178,  ...,  67, 115, 119],
               ...,
               [151, 134, 166,  ..., 162, 166, 165],
               [134, 134, 166,  ..., 163, 146, 182],
               [150, 150, 150,  ..., 146, 178, 178]],

              [[227, 243, 211,  ..., 226,  66, 106],
               [227,  98, 227,  ..., 226,  82,  18],
```

```
  [ 34,  98,  98,  ..., 114, 242, 166],
  ...,
  [232, 169, 170,  ...,  98, 102, 110],
  [ 42,  42,  43,  ...,  98, 102, 102],
  [ 99,  35, 111,  ...,  42,  38,  38]],

 [[ 99,  83,  67,  ...,  67, 227, 215],
  [ 67,  67,  87,  ..., 227, 243, 231],
  [ 67,  67,  83,  ..., 229, 119, 241],
  ...,
  [ 99,   3,  67,  ...,  34,   2, 231],
  [ 35,   3, 107,  ..., 130, 130,  71],
  [  3,   8,  75,  ..., 130, 135,  69]],

 ...,

 [[159, 183, 231,  ..., 194, 134, 151],
  [151, 247,  99,  ..., 246, 150, 210],
  [147, 161,  35,  ...,  34, 146, 147],
  ...,
  [179, 195, 194,  ..., 162, 226, 131],
  [150, 226,  98,  ..., 130, 194, 195],
  [146,  98,  98,  ..., 130, 194, 195]],

 [[ 22,  54, 246,  ...,  38, 226, 226],
  [ 68, 102, 246,  ...,  98,  98,  66],
  [ 82, 226, 210,  ..., 227, 226, 230],
  ...,
  [ 68,  68,  68,  ..., 102, 198, 202],
  [226, 226, 194,  ..., 102, 196, 204],
  [226, 226, 226,  ..., 222, 212, 214]],

 [[ 90, 195, 227,  ...,  91,  83,  83],
  [203, 235, 119,  ..., 115,  75,  75],
  [ 75, 115, 115,  ...,  51,  83, 201],
  ...,
  [249, 243, 115,  ...,  63, 105, 109],
  [193, 193,  91,  ...,  31,  90,  64],
  [232, 203,  91,  ...,  27,  18,  64]]], device='cuda:0',
dtype=torch.int32))
```

```
hash_loc.flatten().mode()
```

```
torch.return_types.mode(
values=TensorImage(99, device='cuda:0', dtype=torch.int32),
indices=TensorImage(1, device='cuda:0'))
```

```
hash_loc==99
```

```
TensorImage([[[False,  True, False,  ..., False, False, False],
              [False, False, False,  ..., False, False, False],
              [False, False, False,  ..., False, False, False],
              ...,
              [False, False, False,  ..., False, False, False],
              [False, False, False,  ..., False, False, False],
              [False, False, False,  ..., False, False, False]],

             [[False, False, False,  ..., False, False, False],
              [False, False, False,  ..., False, False, False],
              [False, False, False,  ..., False, False, False],
              ...,
              [False, False, False,  ..., False, False, False],
              [False, False, False,  ..., False, False, False],
              [ True, False, False,  ..., False, False, False]],

             [[ True, False, False,  ..., False, False, False],
              [False, False, False,  ..., False, False, False],
              [False, False, False,  ..., False, False, False],
              ...,
              [ True, False, False,  ..., False, False, False],
              [False, False, False,  ..., False, False, False],
              [False, False, False,  ..., False, False, False]],

             ...,

             [[False, False, False,  ..., False, False, False],
              [False, False,  True,  ..., False, False, False],
              [False, False, False,  ..., False, False, False],
              ...,
              [False, False, False,  ..., False, False, False],
              [False, False, False,  ..., False, False, False],
```

```
          [False, False, False,  ..., False, False, False]],

         [[False, False, False,  ..., False, False, False],
          [False, False, False,  ..., False, False, False],
          [False, False, False,  ..., False, False, False],
          ...,
          [False, False, False,  ..., False, False, False],
          [False, False, False,  ..., False, False, False],
          [False, False, False,  ..., False, False, False]],

         [[False, False, False,  ..., False, False, False],
          [False, False, False,  ..., False, False, False],
          [False, False, False,  ..., False, False, False],
          ...,
          [False, False, False,  ..., False, False, False],
          [False, False, False,  ..., False, False, False],
          [False, False, False,  ..., False, False, False]]],
        device='cuda:0')
```

How would we generate a mask for the locations that equal the mode(199)?

```
(hash_loc==199).shape,(hash_loc==199)
```

```
(torch.Size([64, 10, 10]),
 TensorImage([[[False, False, False,  ..., False, False, False],
               [False, False, False,  ..., False, False, False],
               [False, False, False,  ..., False, False, False],
               ...,
               [False, False, False,  ..., False, False, False],
               [False, False, False,  ...,  True,  True,  True],
               [False, False, False,  ...,  True, False, False]],

              [[False, False, False,  ..., False, False, False],
               [False, False, False,  ..., False, False, False],
               [False, False, False,  ..., False, False, False],
               ...,
               [False, False, False,  ..., False, False, False],
               [False, False, False,  ..., False, False, False],
               [False, False, False,  ..., False, False, False]],

              [[False, False, False,  ...,  True,  True,  True],
               [False, False, False,  ..., False, False, False],
```

```
          [False, False, False,  ..., False, False, False],
          ...,
          [False, False, False,  ..., False,  True,  True],
          [False, False, False,  ..., False, False,  True],
          [False, False, False,  ...,  True, False, False]],

         ...,

         [[False, False, False,  ..., False, False, False],
          [False, False, False,  ...,  True, False, False],
          [ True, False, False,  ..., False, False, False],
          ...,
          [False, False, False,  ..., False, False, False],
          [False, False, False,  ..., False, False, False],
          [False, False, False,  ..., False, False, False]],

         [[False, False, False,  ...,  True, False, False],
          [False, False, False,  ...,  True, False, False],
          [False, False, False,  ...,  True, False, False],
          ...,
          [False, False,  True,  ..., False, False, False],
          [False, False,  True,  ..., False, False, False],
          [False, False,  True,  ..., False, False, False]],

         [[False, False, False,  ..., False, False, False],
          [False, False, False,  ..., False, False, False],
          [False, False, False,  ..., False, False, False],
          ...,
          [False, False, False,  ..., False, False, False],
          [False, False, False,  ..., False, False, False],
          [False, False, False,  ..., False, False, False]]],
        device='cuda:0'))
```

```
b.shape,hash_loc.shape
```

```
(torch.Size([64, 3, 320, 320]), torch.Size([64, 10, 10]))
```

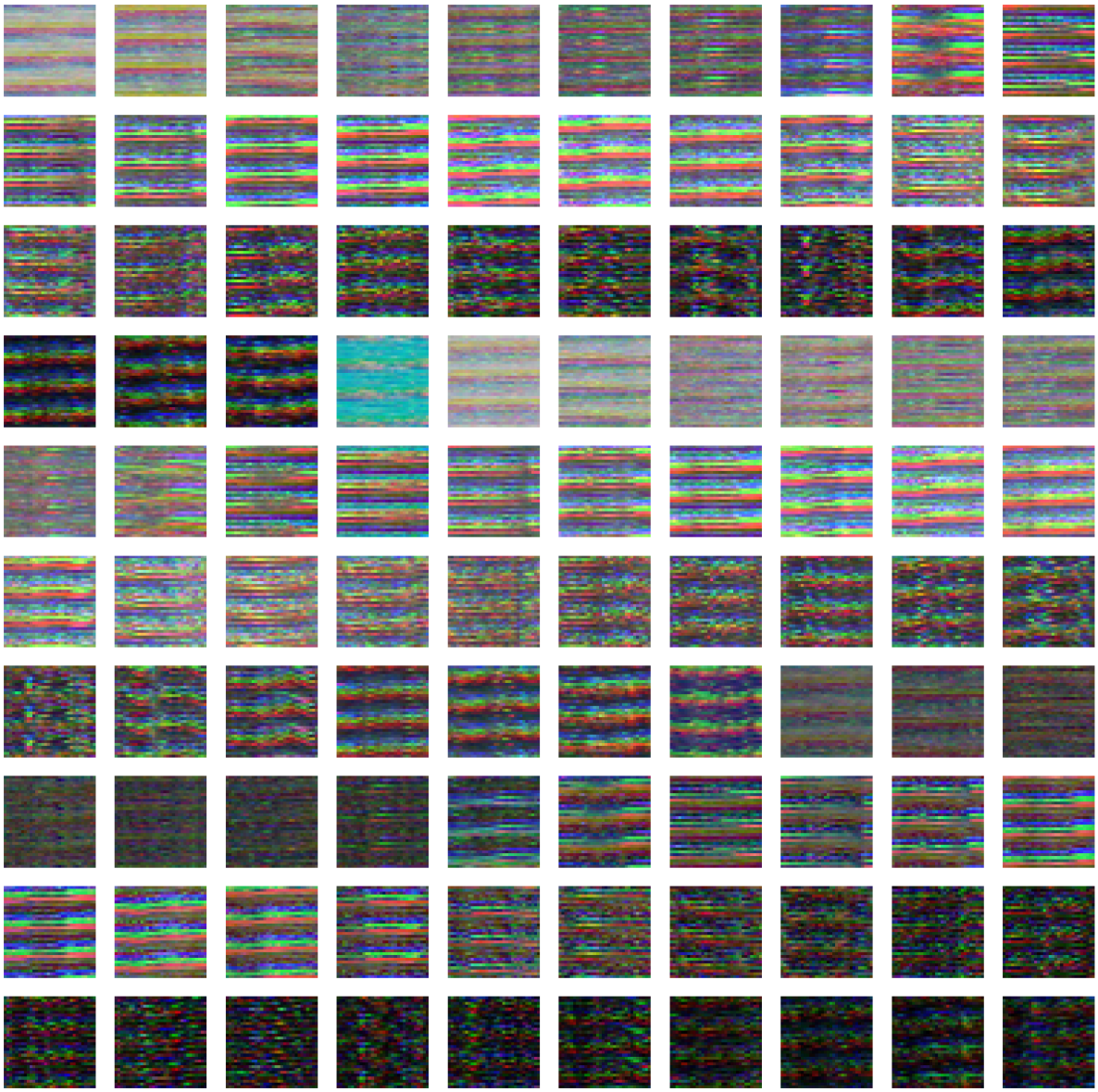
```
%timeit b.view(64, 3, 10,320//10, 10,320//10)
```

```
15.4 µs ± 2.08 µs per loop (mean ± std. dev. of 7 runs, 10,000 loops each)
```

```
tiled_img=b.view(64, 3, 10,320//10, 10,320//10).permute(0,2,4,1,3,5)
b.shape,tiled_img.shape
```

```
(torch.Size([64, 3, 320, 320]), torch.Size([64, 10, 10, 3, 32, 32]))
```

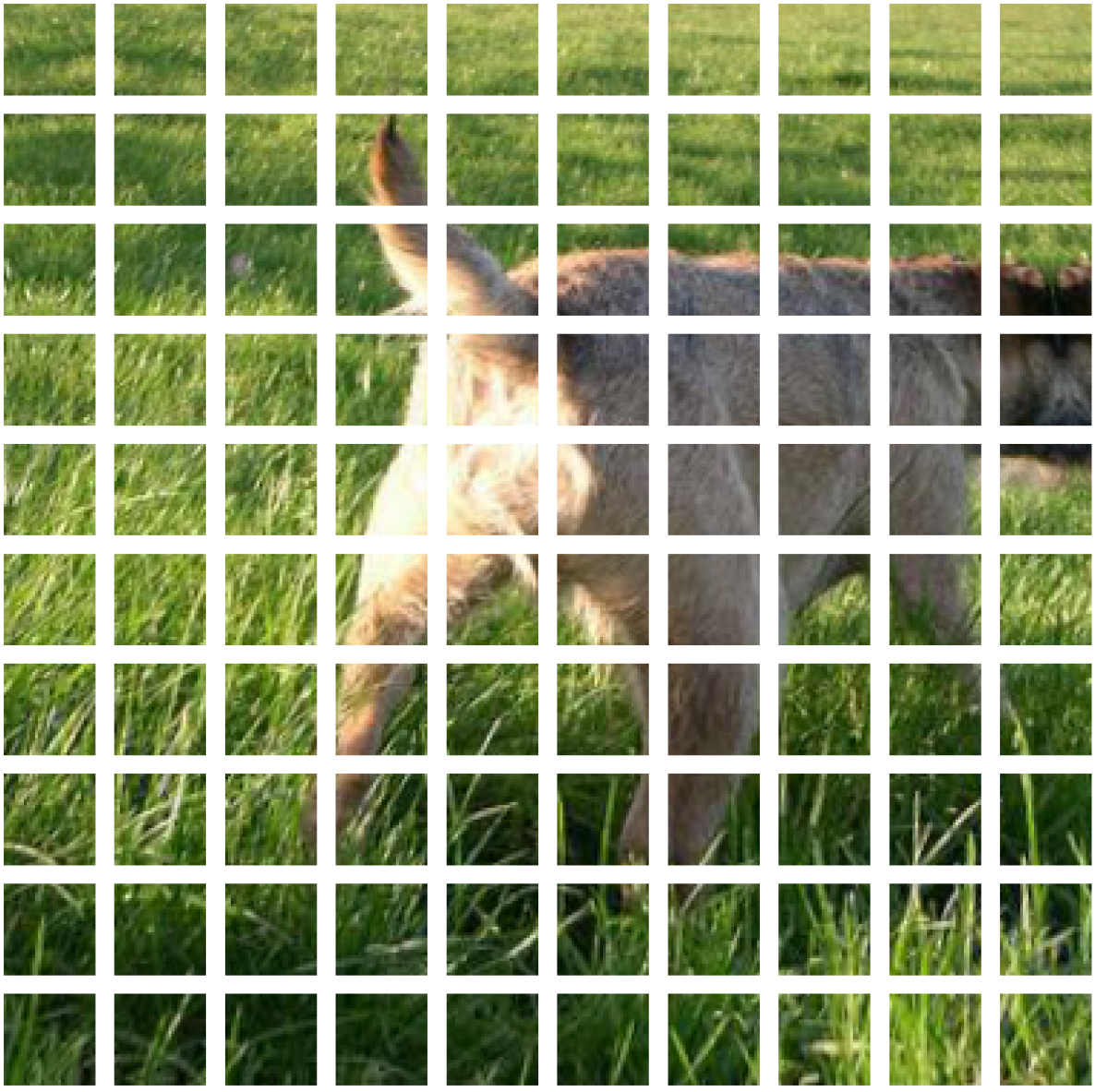
```
show_images(b.view(64, 10, 10, 3, 32, 32)[0].flatten(end_dim=1),nrows=10)
```



```
b.view(64, 3, 10,320//10, 10,320//10).shape
```

```
torch.Size([64, 3, 10, 32, 10, 32])
```

```
show_images(tiled_img[0].flatten(end_dim=1),nrows=10)
```



```
  tiled_img[1].shape,(hash_loc==99)[0].shape
```

```
(torch.Size([10, 10, 3, 32, 32]), torch.Size([10, 10]))
```

```
  tiled_img[8][(hash_loc==99)[8]].shape
```

```
torch.Size([15, 3, 32, 32])
```

How would select select the patches of the image that fall into our mode bucket?

```
  tiled_img[1][hash_loc[1]==199].shape
```

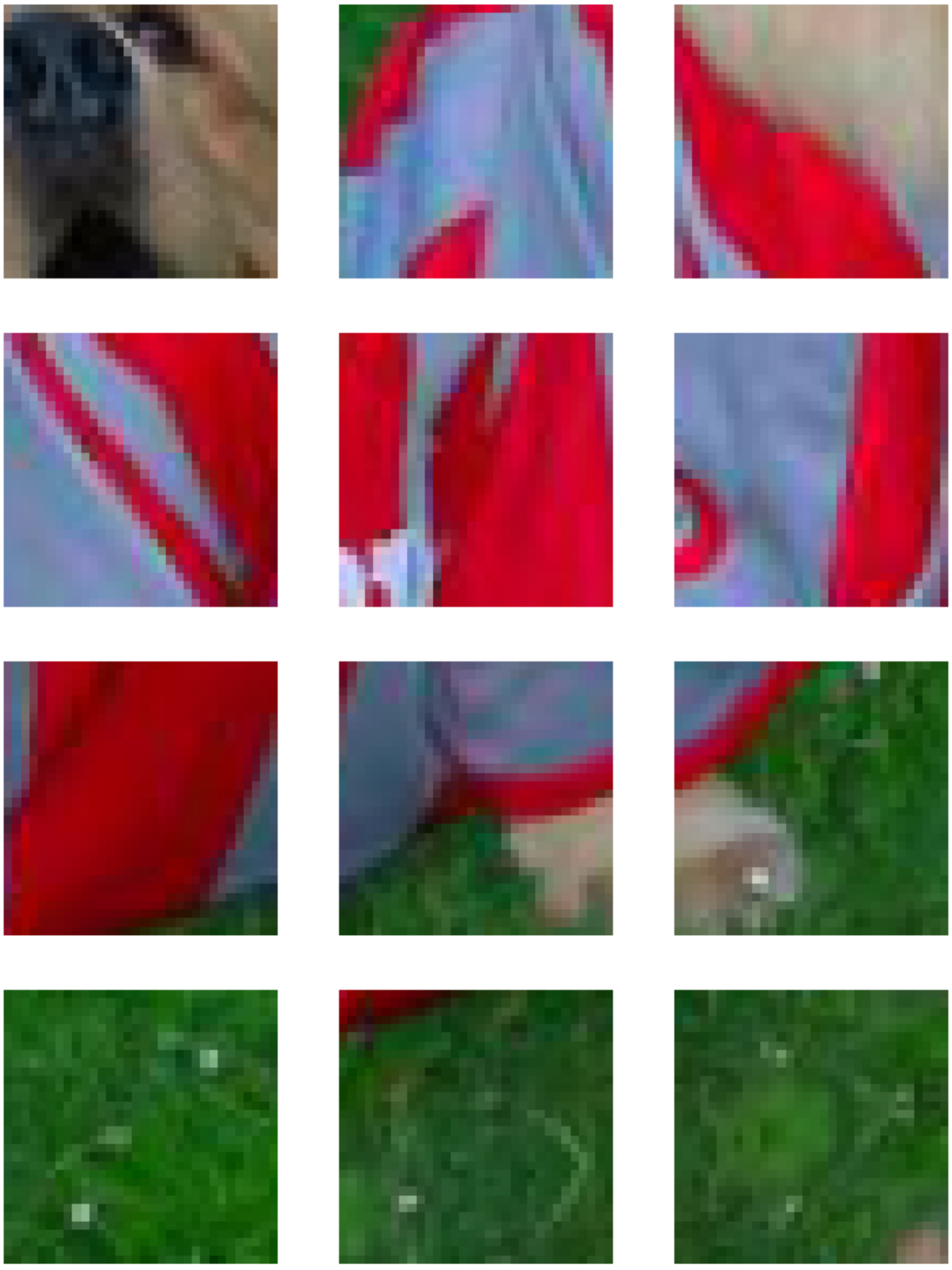
```
torch.Size([11, 3, 32, 32])
```

```
  show_image(b[0])
```

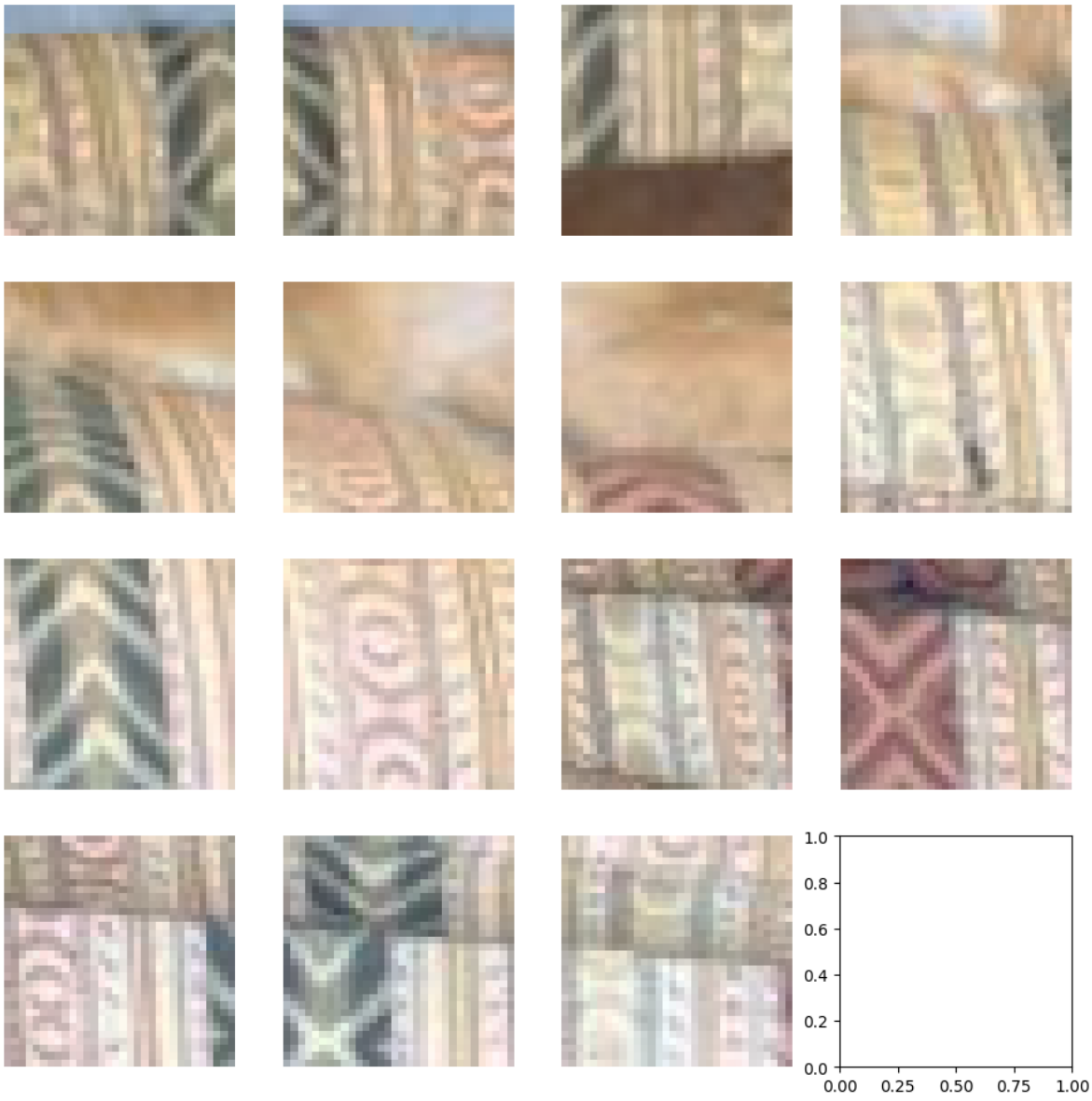
```
<AxesSubplot:>
```

Seems to detect “objects.” Other images included scales for weight things, and sea shells. Dog nose, might be due to that space being close.

```
show_images(tiled_img[0][hash_loc[0]==199][:64],nrows=4)
```



```
show_images(tiled_img[8][(hash_loc==99)[8]],nrows=4)
```




```
show_image(b[8])
```

```
<AxesSubplot:>
```

```
len(dls.dataset)*10*10
```

```
1295200
```

```
1295200/256
```

```
5059.375
```