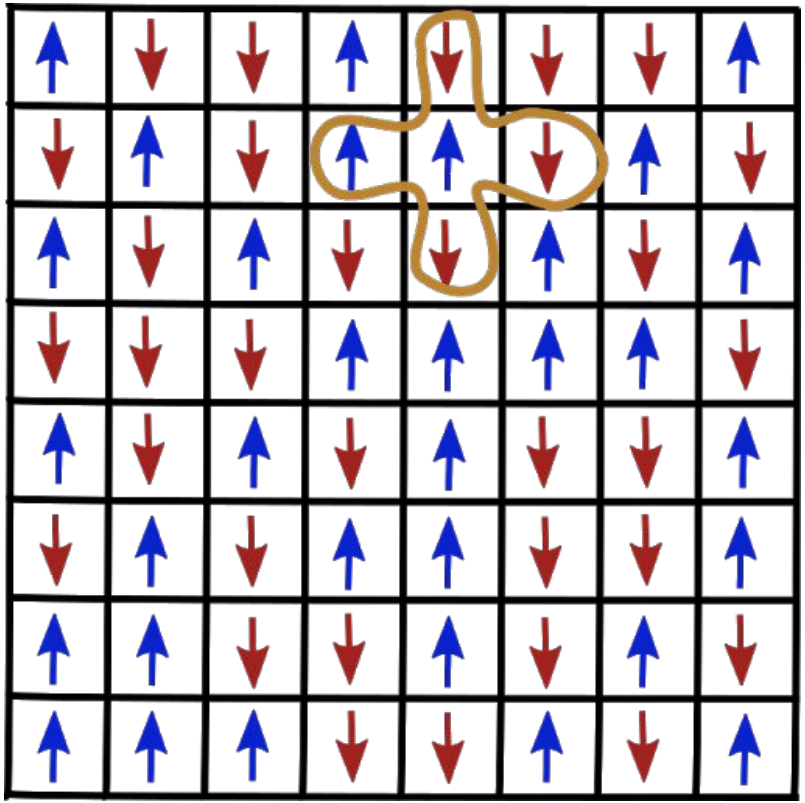
Ising model 3d with masks

Danylo Lykov

Ising model

Local decision:



Performance of naive loops is bad

```
Time for 1/9th of a sweep:

size 100  : 0.026321172714233 4
size 500  : 0.403746128082275 4
size 1000 : 1.470579862594604 5
size 5000 : 39.99836683273315 4
```

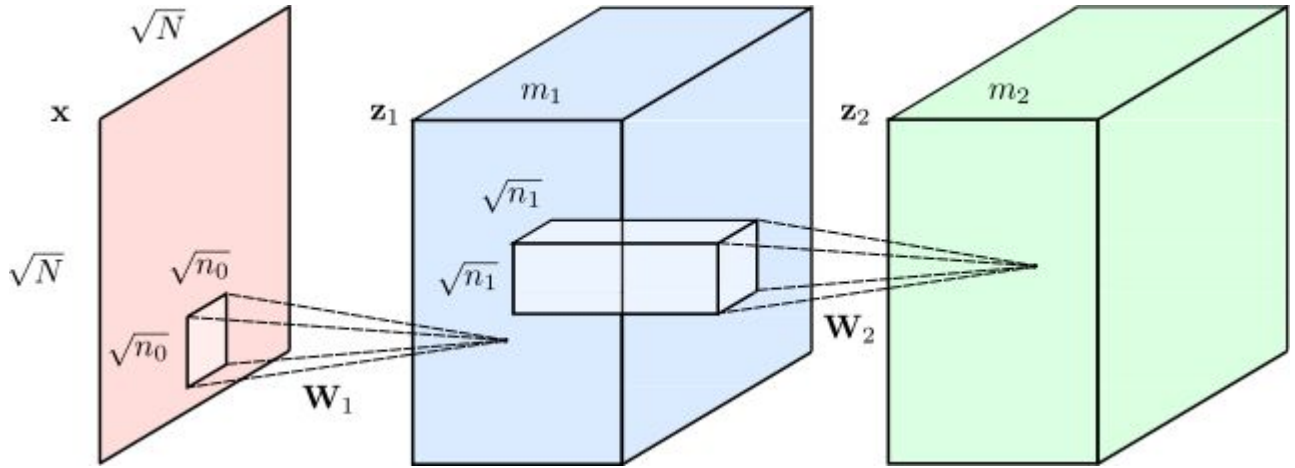
Performance of naive loops is bad

```
Time for 1/9th of a sweep:

size 100  : 0.0263211727142334
size 500  : 0.4037461280822754
size 1000 : 1.4705798625946045
size 5000 : 39.998366832733154
```

```
Time for 1/9th of a sweep:

size 100  : 0.0008029937744140625
size 500  : 0.007186412811279297
size 1000 : 0.05148077011083984
size 5000 : 0.8673443794250488
```

Convolutional neural networks

Local filters:



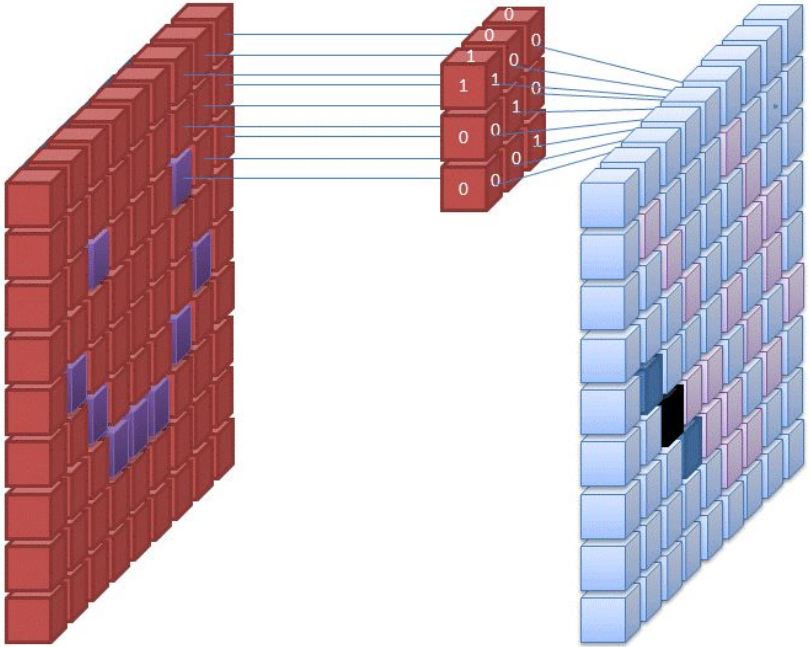
Convolutional neural networks

Local filters =>
Can compute in parallel

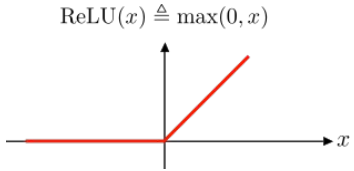
Have to set stride=3

Compute 1/9 of all spins at every step

Have to ‘roll’ array after every step

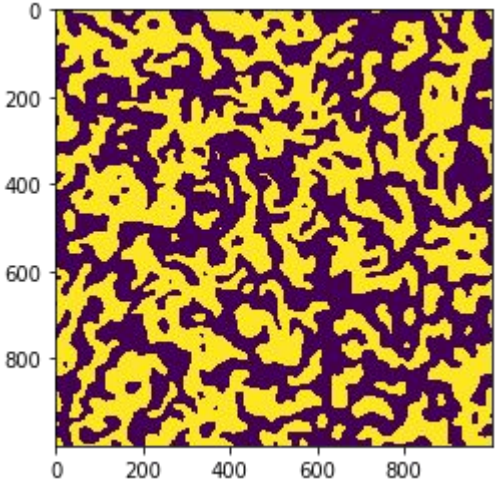


```
def metrop_step(grid, conv, beta):
    D = 3
    # Roll to be random
    rix = np.random.randint(0, high=3, size=D)
    grid = T.roll(grid, shifts=tuple(rix), dims=tuple(range(2,2+D)) )

    # Get neighbours contribution
    dE = 2*conv(grid)[0,0]

    # Get energy change
    scatter_ixs = [np.arange(1, d_-1, 3) for d_ in grid.shape[2:]]
    ixs = (0,0) + np.ix_(*scatter_ixs)
    sub = grid[ixs]
    dE = sub*(dE + 2*conv.mu)

    # Randomly flip spins
    acc_prob = T.exp(-beta*F.relu(dE))
    random = T.rand_like(acc_prob)
    sub[acc_prob > random] *= -1
    grid[ixs] = sub

    # Bookkeeping
    dE[acc_prob < random] *= 0
    sub[acc_prob < random] *= 0
    return grid, float(dE.sum().detach()), 2*float(sub.sum().detach())
```

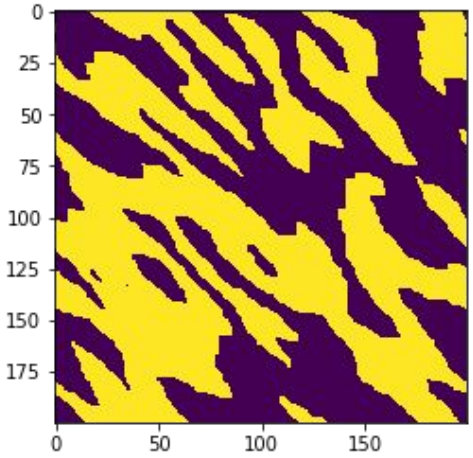


Masks

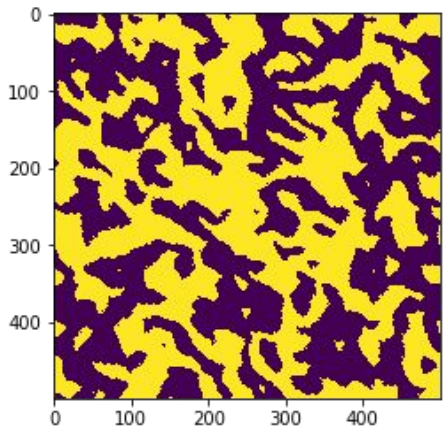
```
def get_nn_mask(J, h):
    return np.array([
          [0, J, 0]
        ,[J, h, J]
        ,[0, J, 0]
    ])
```



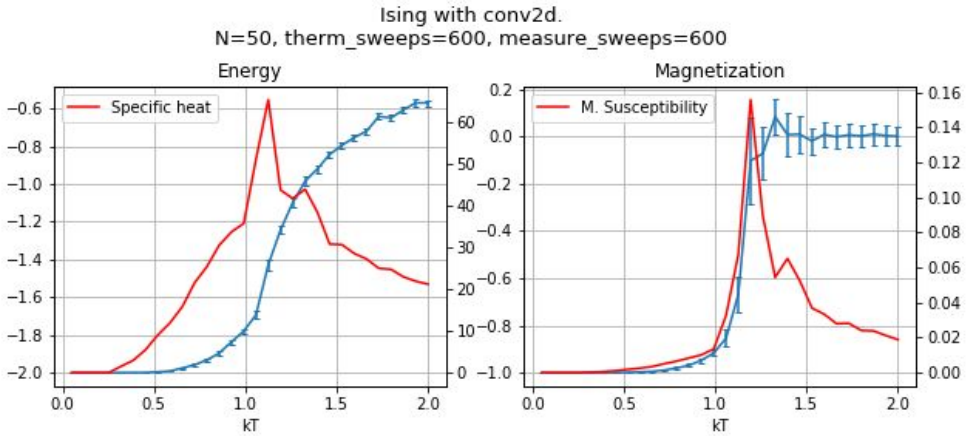

```
def get_diagonal_mask(J, h):
    a = 1/np.sqrt(2)
    return np.array([
          [J*a,  J, -J*a]
        ,[J,    h,  J ]
        ,[-J*a, J, J*a]
    ])
```

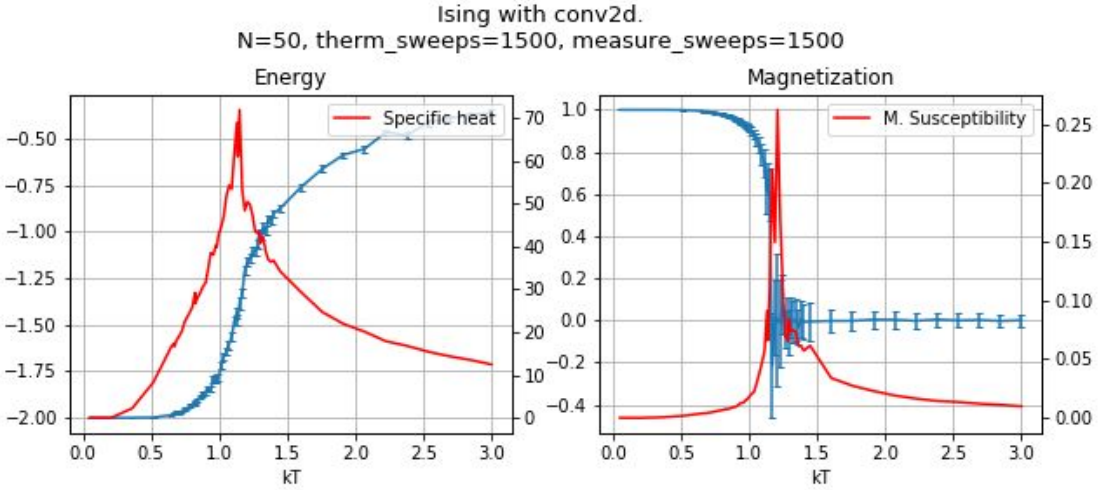



```
def get_funny_mask(J,
h):
    return np.array([
          [J*2, J, J/2]
        ,[J,   h, J  ]
        ,[J/2, J, -J/2]
    ])
```

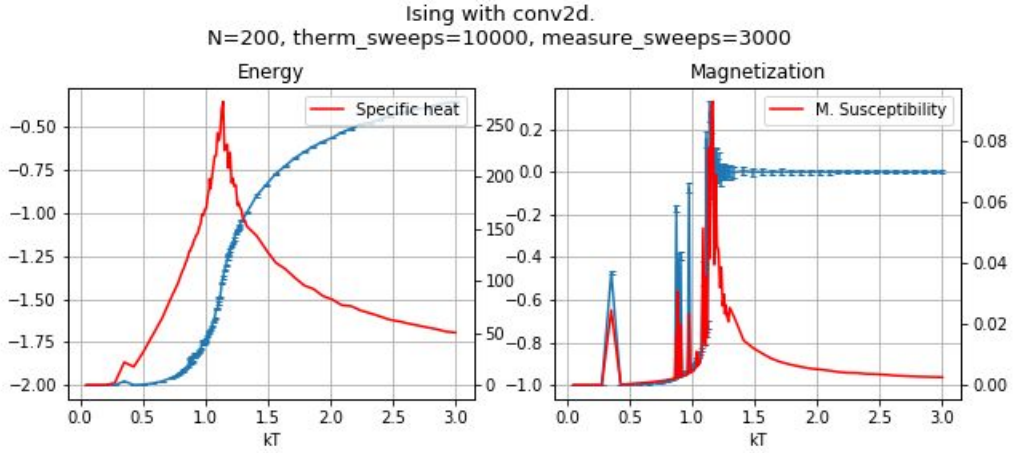


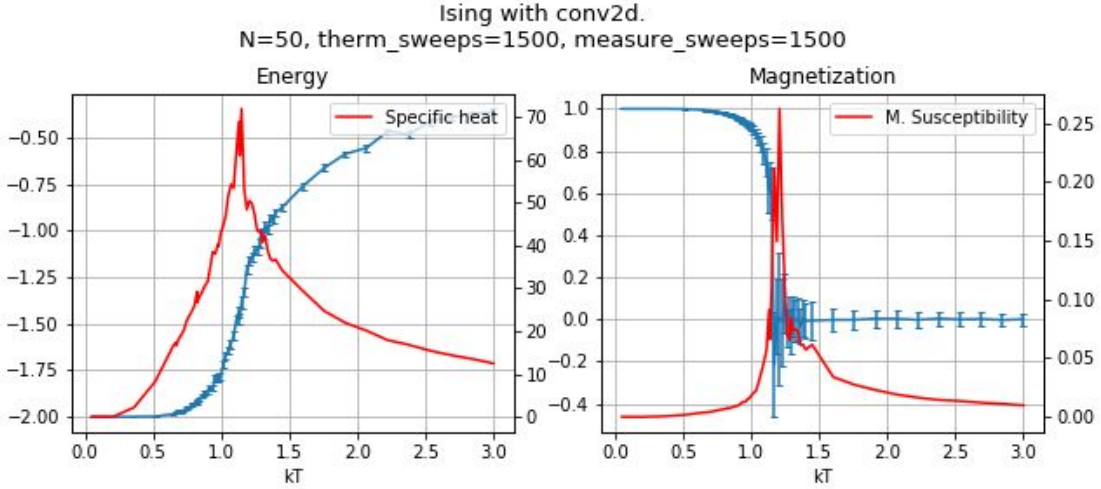
Results


Ising with conv2d.
N=50, therm_sweeps=600, measure_sweeps=600
Energy
Specific heat
kT
Magnetization
M. Susceptibility
kT

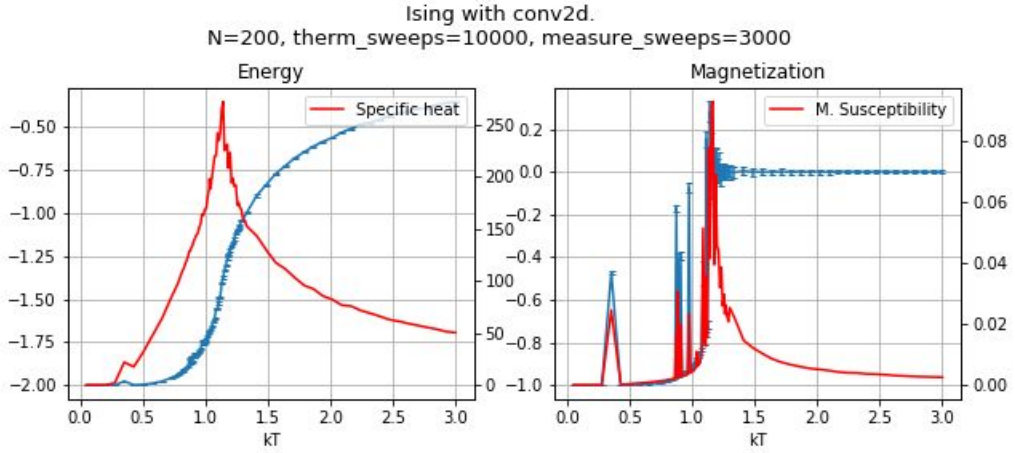

Ising with conv2d.
N=50, therm_sweeps=1500, measure_sweeps=1500
Energy
Specific heat
kT
Magnetization
M. Susceptibility
kT

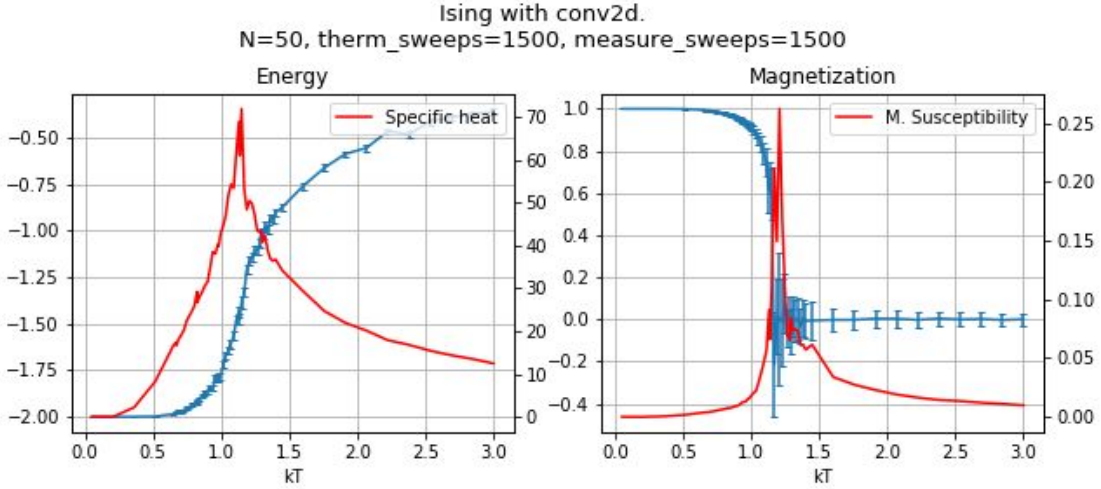
Results


Ising with conv2d.
N=200, therm_sweeps=10000, measure_sweeps=3000
Energy
Specific heat
Magnetization
M. Susceptibility
kT
kT


Ising with conv2d.
N=50, therm_sweeps=1500, measure_sweeps=1500
Energy
Specific heat
Magnetization
M. Susceptibility
kT
kT

Results


Ising with conv2d.
N=200, therm_sweeps=10000, measure_sweeps=3000
Energy
Specific heat
Magnetization
M. Susceptibility
kT
kT


Ising with conv2d.
N=50, therm_sweeps=1500, measure_sweeps=1500
Energy
Specific heat
Magnetization
M. Susceptibility
kT
kT

Anisotropy

```
def get_anisotropic_mask(J, H, ratio):
    h = 1+ratio
    v = 1-ratio
    return np.array([
         [0,  J*v, 0 ]
        ,[J*h, H, J*h]
        ,[0,  J*v, 0 ]
    ])
```

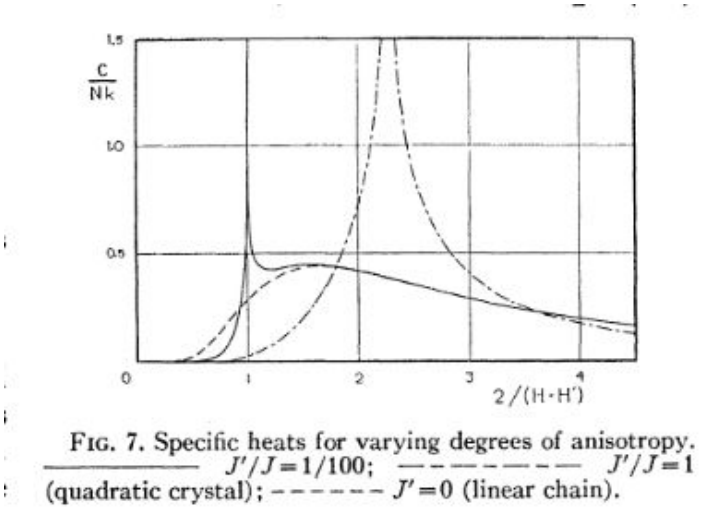
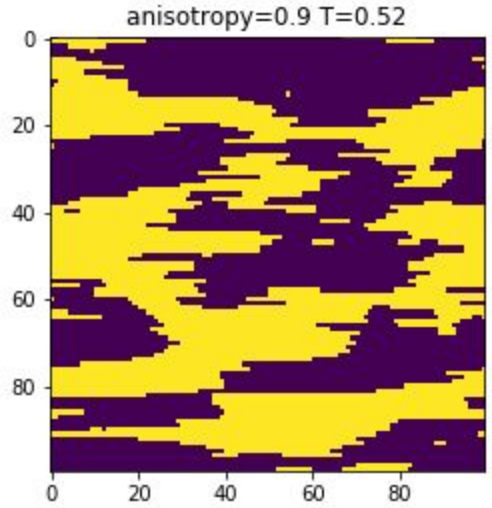


FIG. 7. Specific heats for varying degrees of anisotropy. ——— $J'/J=1/100$; – – – – – – – – $J'/J=1$ (quadratic crystal); – – – – – – $J'=0$ (linear chain).

Crystal Statistics. I. A Two-Dimensional Model with an Order-Disorder Transition

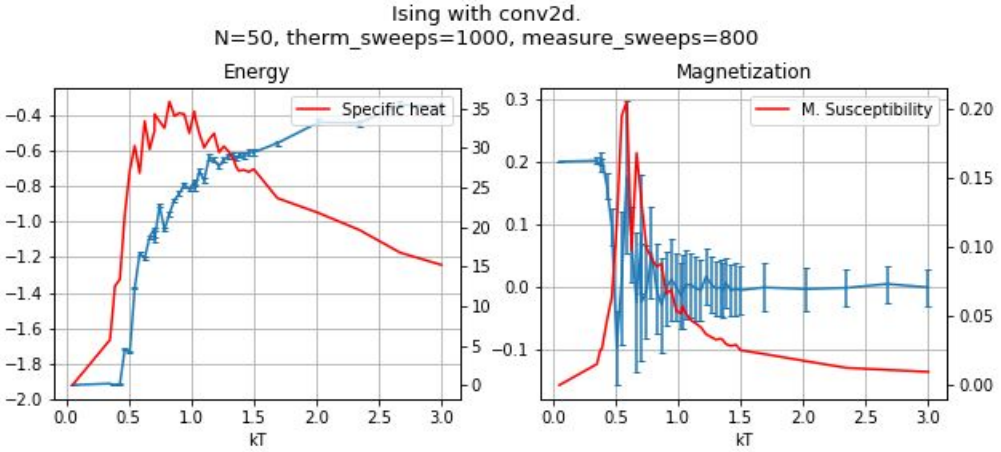
LARS ONSAGER
Sterling Chemistry Laboratory, Yale University, New Haven, Connecticut
(Received October 4, 1943)

Anisotropy

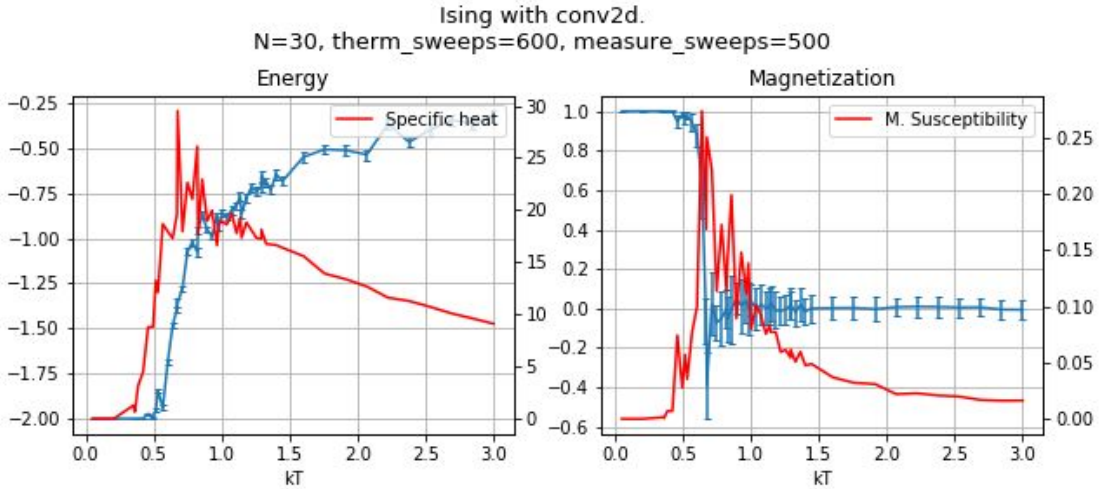


Anisotrop

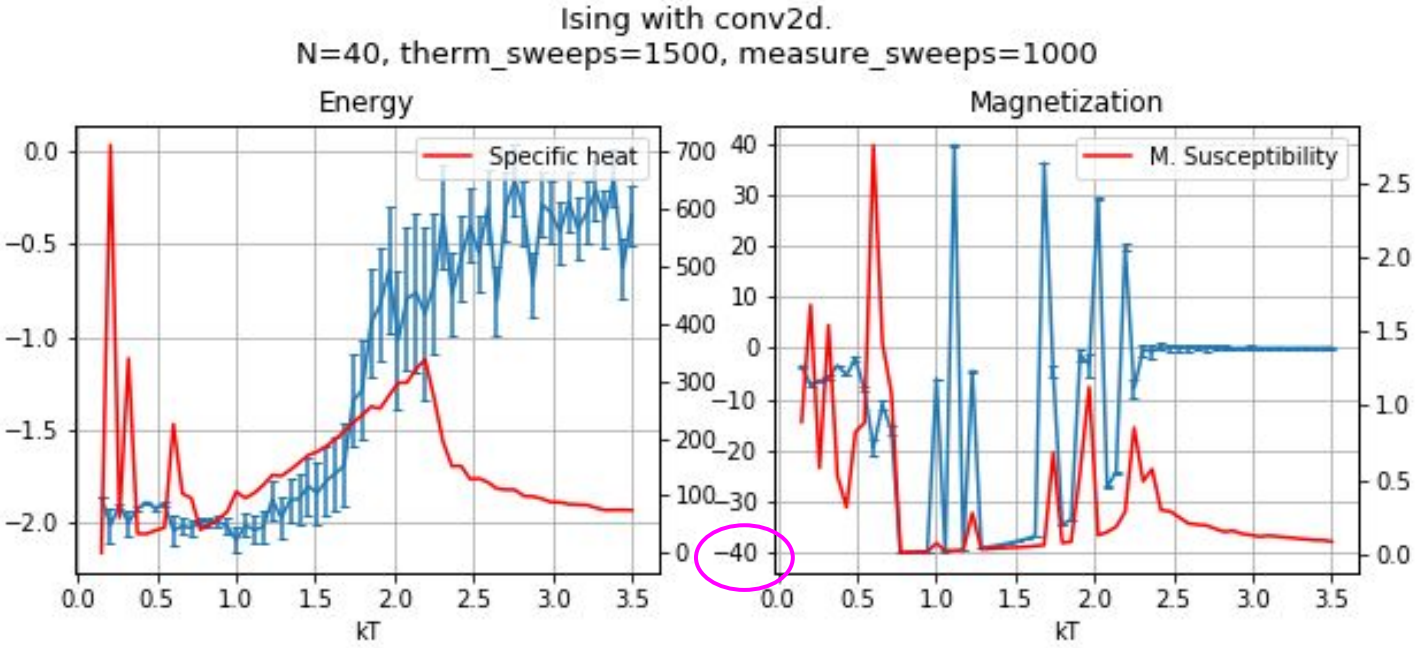
Anisotropy=0.98
J’/J = 100



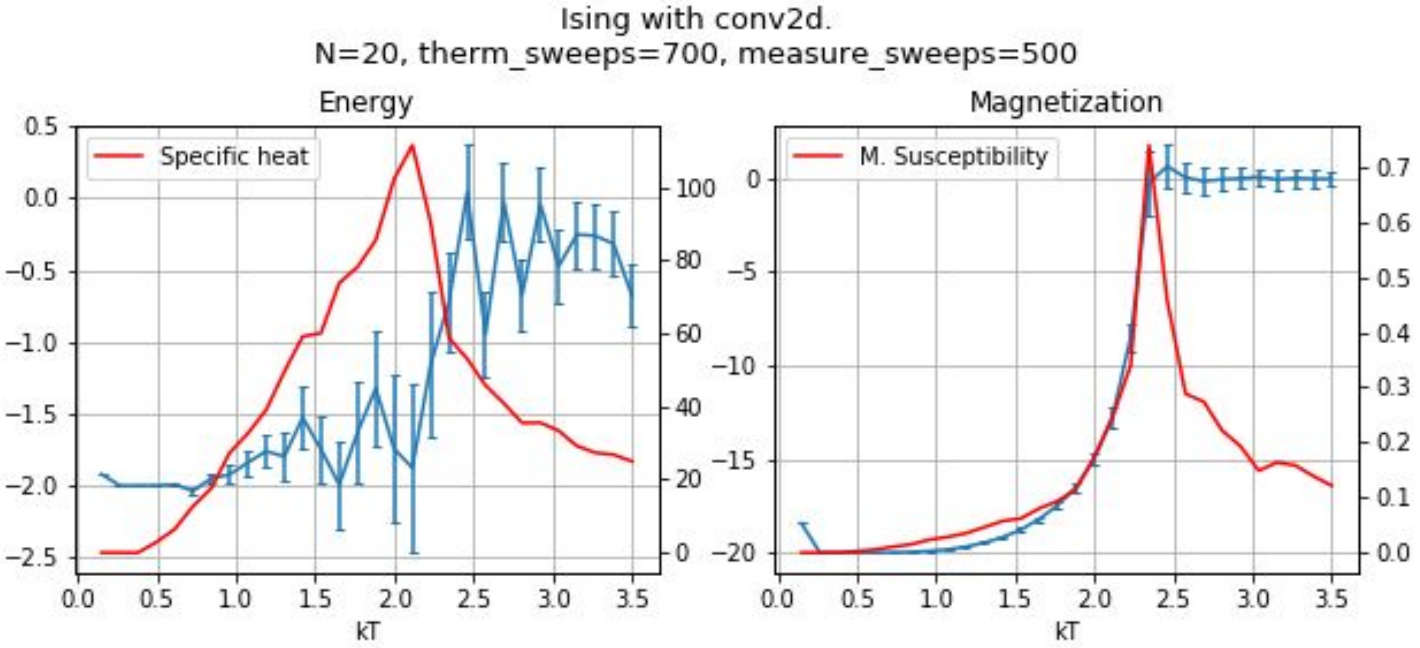
Anisotropy=0.95
J’/J = 39



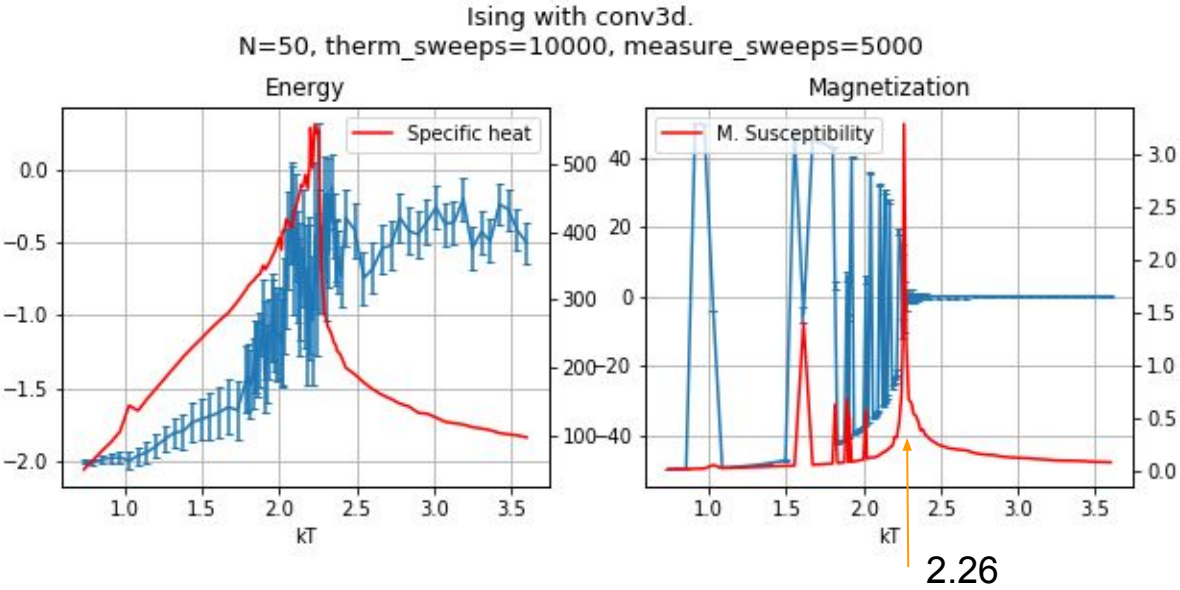
3D

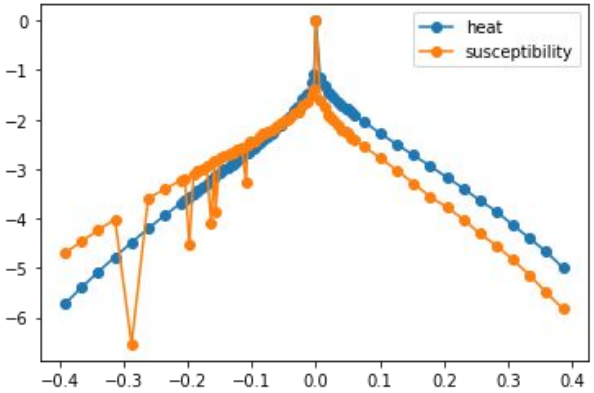

Ising with conv2d.
N=40, therm_sweeps=1500, measure_sweeps=1000
Energy
Specific heat
kT
Magnetization
M. Susceptibility
kT

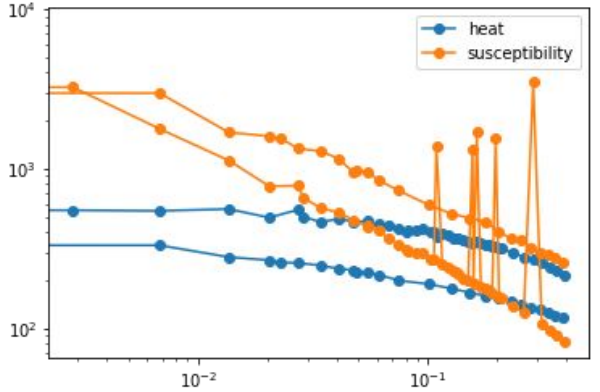
3D


Ising with conv2d.
N=20, therm_sweeps=700, measure_sweeps=500
Energy
Specific heat
kT
Magnetization
M. Susceptibility
kT

3D

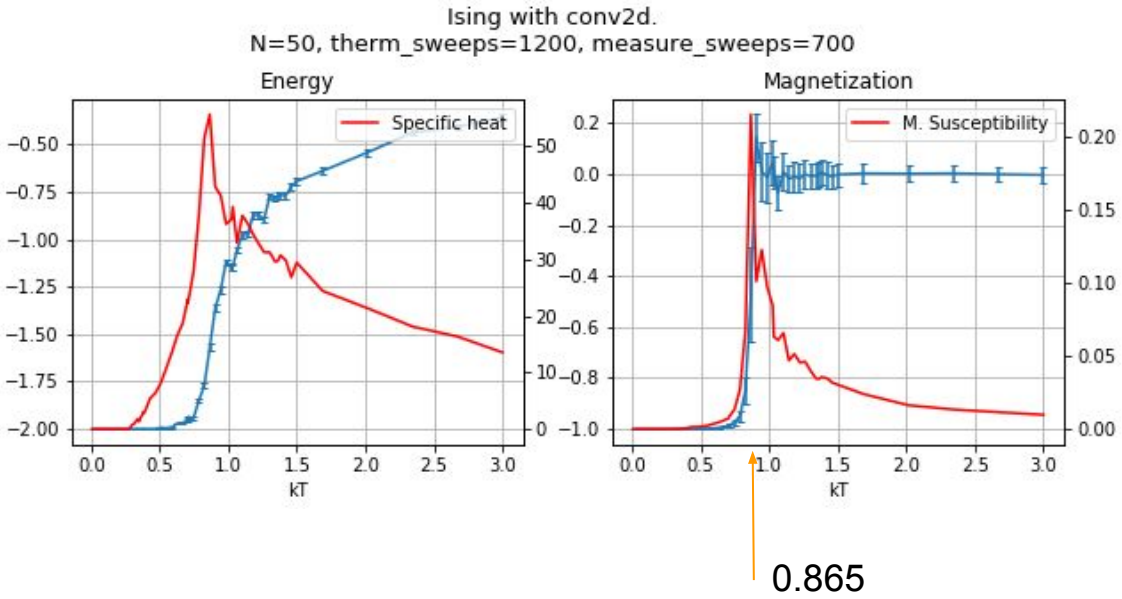

Ising with conv3d.
N=50, therm_sweeps=10000, measure_sweeps=5000
Energy
Specific heat
kT
Magnetization
M. Susceptibility
kT
2.26

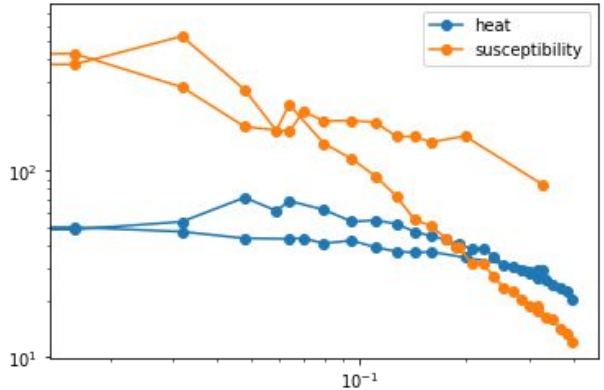

heat
susceptibility


heat
susceptibility

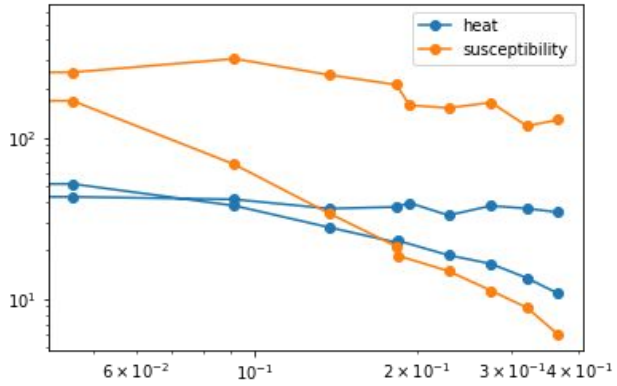
Diagonal mask

```
def get_diagonal_mask(J, h):
    a = 1/np.sqrt(2)
    return np.array([
          [J*a,  J, -J*a]
        ,[J,     h,  J ]
        ,[-J*a, J, J*a]
    ])
```



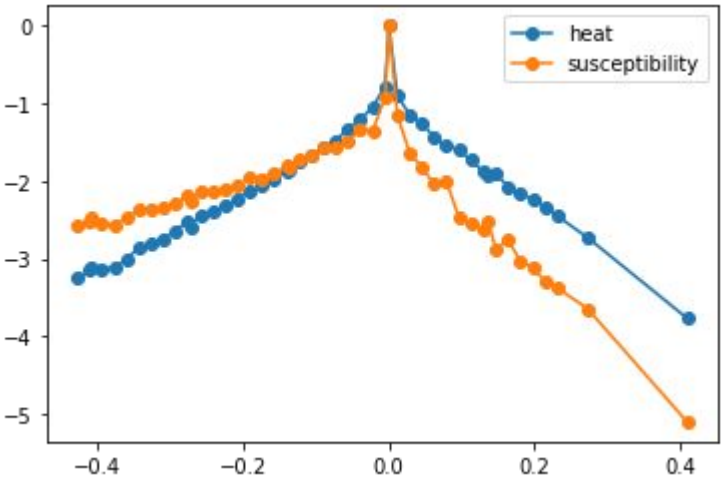


loglog

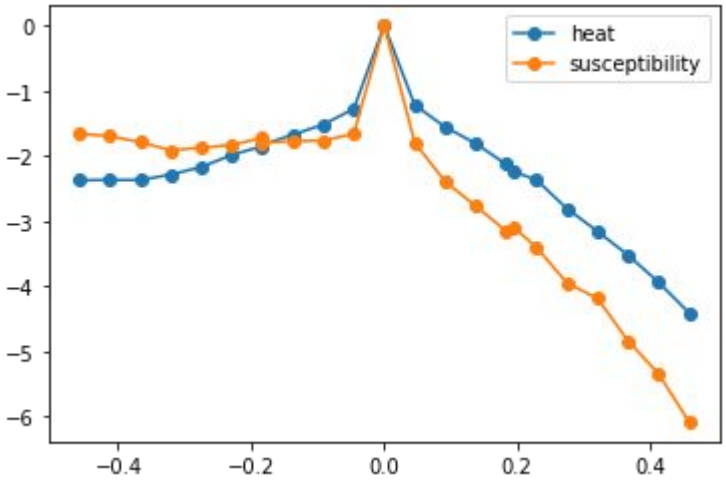


Ordinary mask N=50

Diagonal mask N=50



$$\frac{\log|f(\tau)|}{\log|\tau|}$$



Some simulation times

```
[N=400 sweeps=3500 measure_sweeps=1000 times=200]
221968.30s user 11780.24s system 9366% cpu 41:35.68 total

[N=50 sweeps=700 measure_sweeps=700 times=40]
17 minutes 1020s

[N=400 sweeps=10 000 measure_sweeps=3000 times=100]
316822.51s user 19133.65s system 9919% cpu 56:26.68 total
```