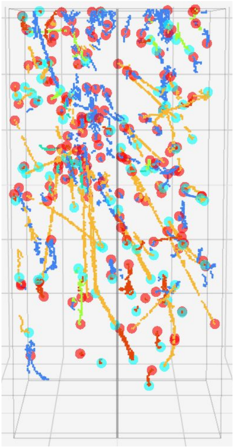
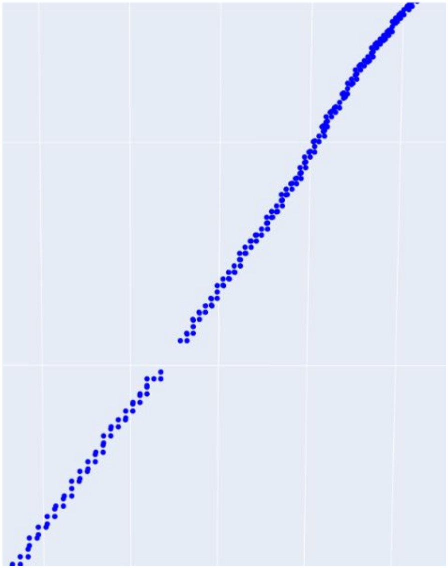
LArDRIP Update



Hilary Utaegbulam
University of Rochester

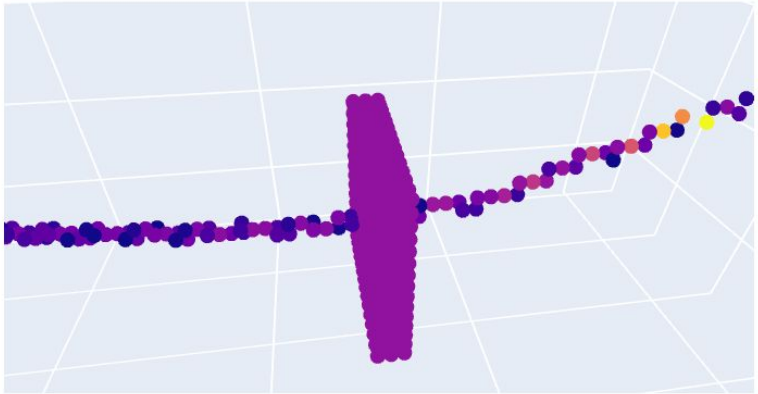
Problem Statement

- Can we train a deep learning model to infer missing regions of charged particle tracks, given XYZE information?
- Possibly add physics?

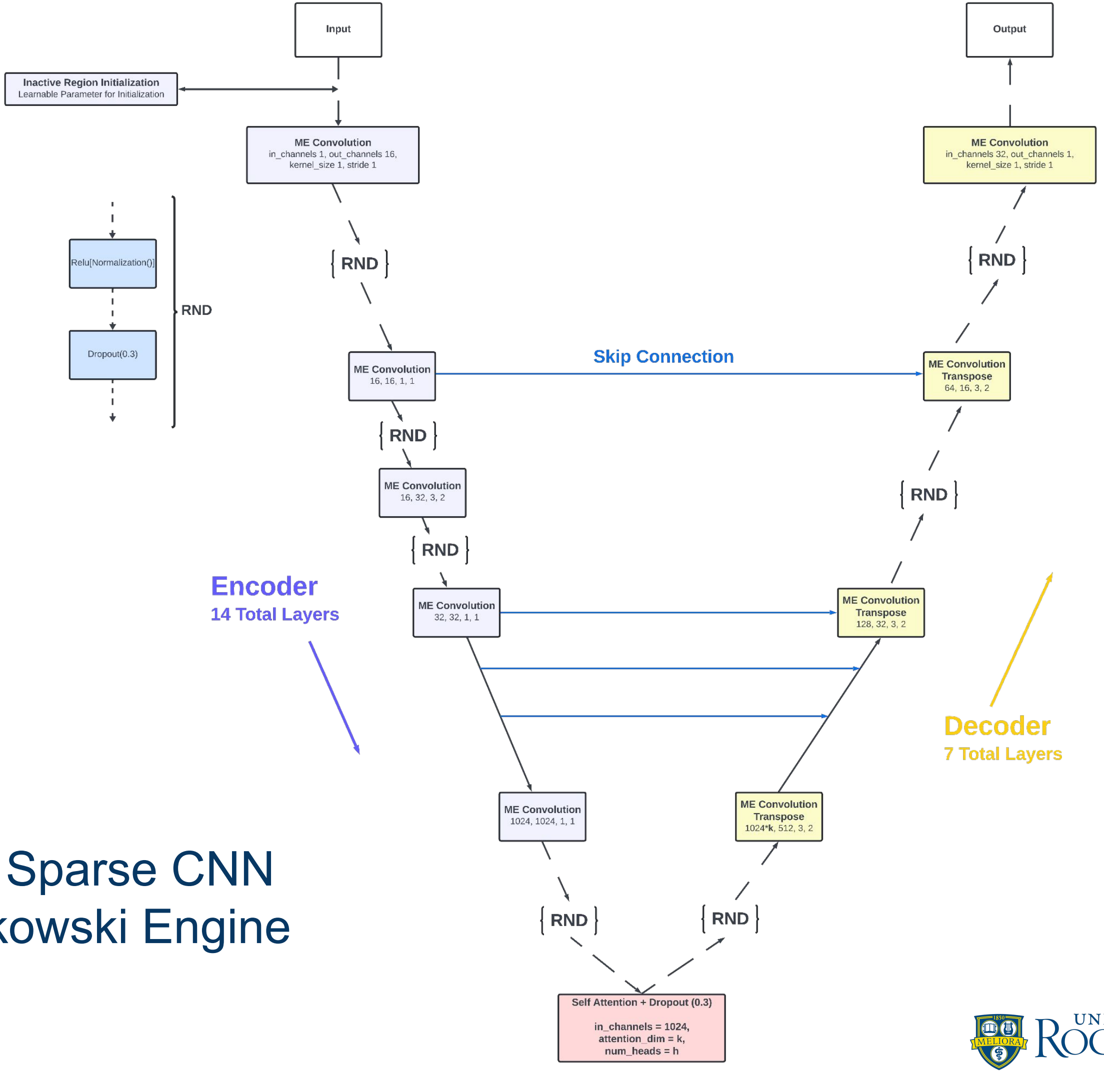


Method

- Voxelized tracks!
- Replace missing region with dense voxel grid
- Initialize voxel region with some energy value(s)
  - E = 0, random, -1
- Use a Sparse CNN to determine which voxels in the gird regions should activate—which voxels are non-zero?
  - Eventually predict what the non-zero voxel energies should be



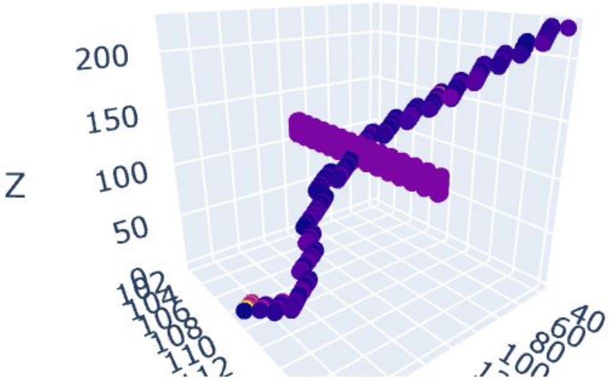
Model



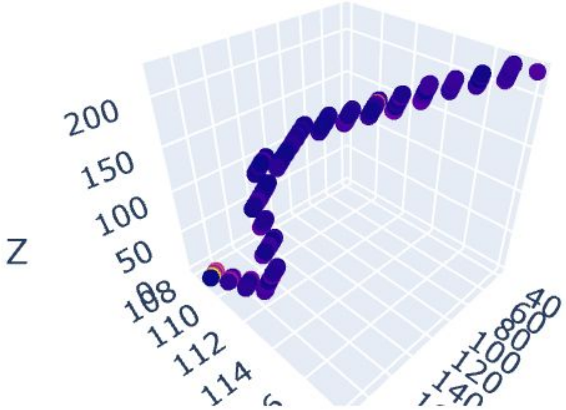
- UNet style Sparse CNN
- Using Minkowski Engine

Results

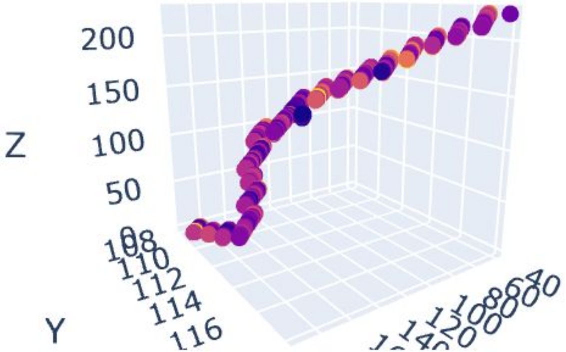
Input



Target

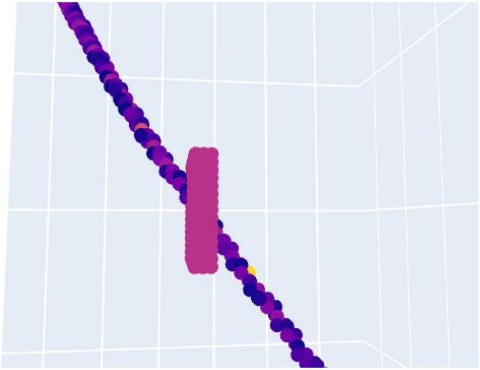


Prediction

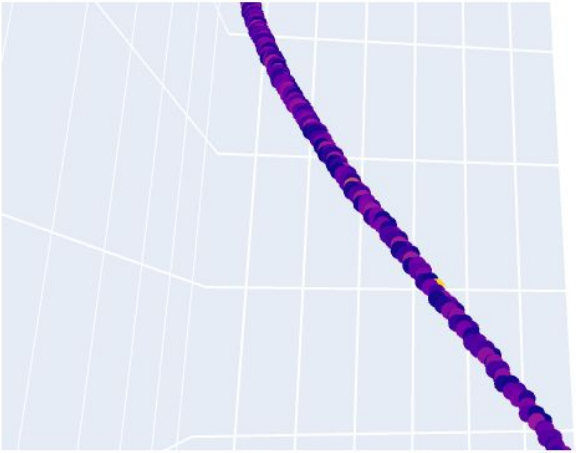


Results

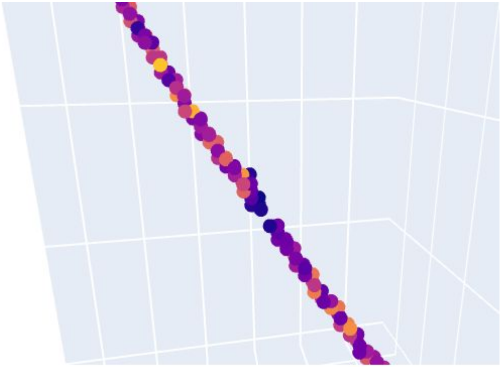
Input



Target

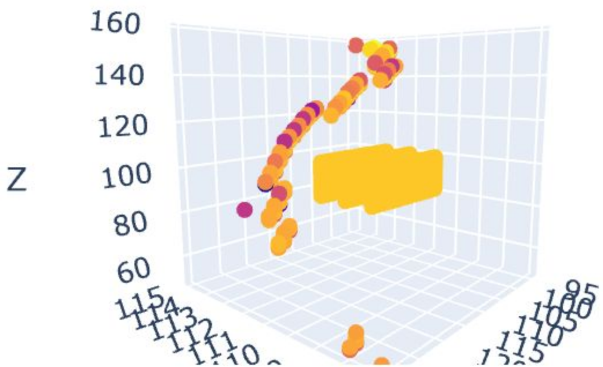


Prediction

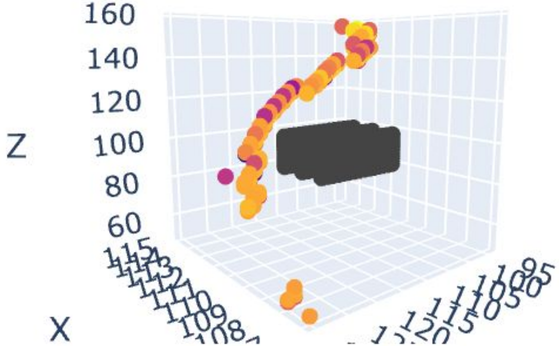


Results

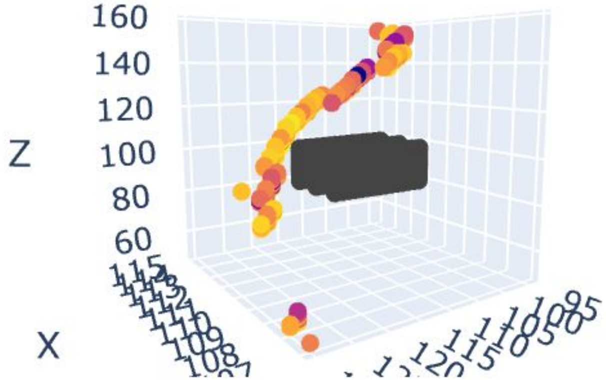
Input



Target

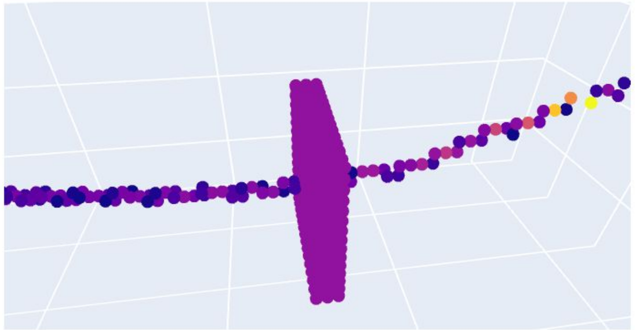


Prediction

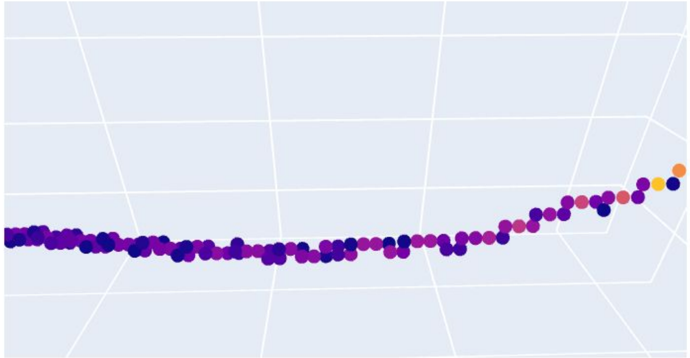


Results

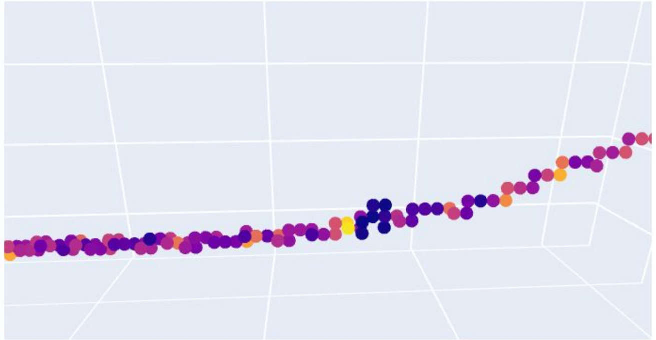
Input



Target

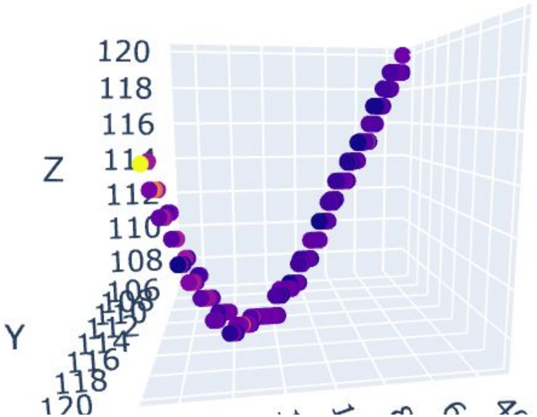


Prediction

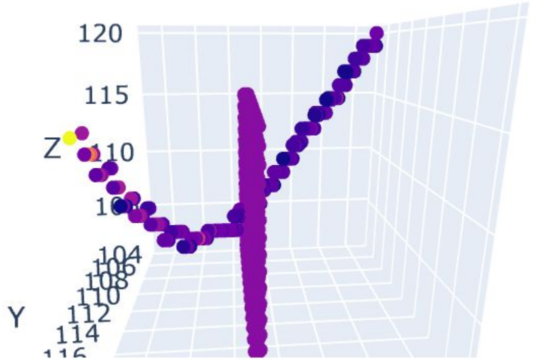


Results

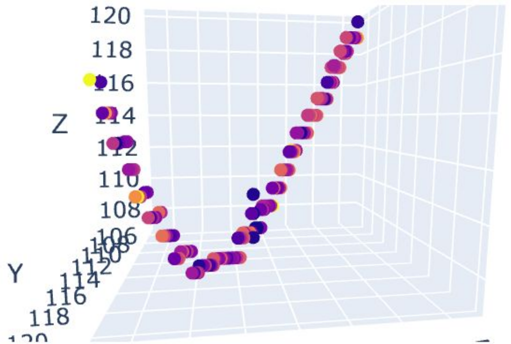
Target



Input



Prediction



Loss Function

- Loss metric:

$$\mathbf{mask}_{\text{track}} = \begin{cases} 1 & \text{if } \mathbf{target} \neq 0 \\ 0 & \text{if } \mathbf{target} = 0 \end{cases}$$

$$\mathbf{mask}_{\text{nontrack}} = \begin{cases} 1 & \text{if } \mathbf{target} = 0 \\ 0 & \text{if } \mathbf{target} \neq 0 \end{cases}$$

$$\text{loss}_{\text{track}} = (\mathbf{predicted} - \mathbf{target})^2 \cdot \mathbf{mask}_{\text{track}}$$

$$\text{loss}_{\text{nontrack}} = (\mathbf{predicted} - \mathbf{target})^2 \cdot \mathbf{mask}_{\text{nontrack}}$$

Weights are learnable parameters

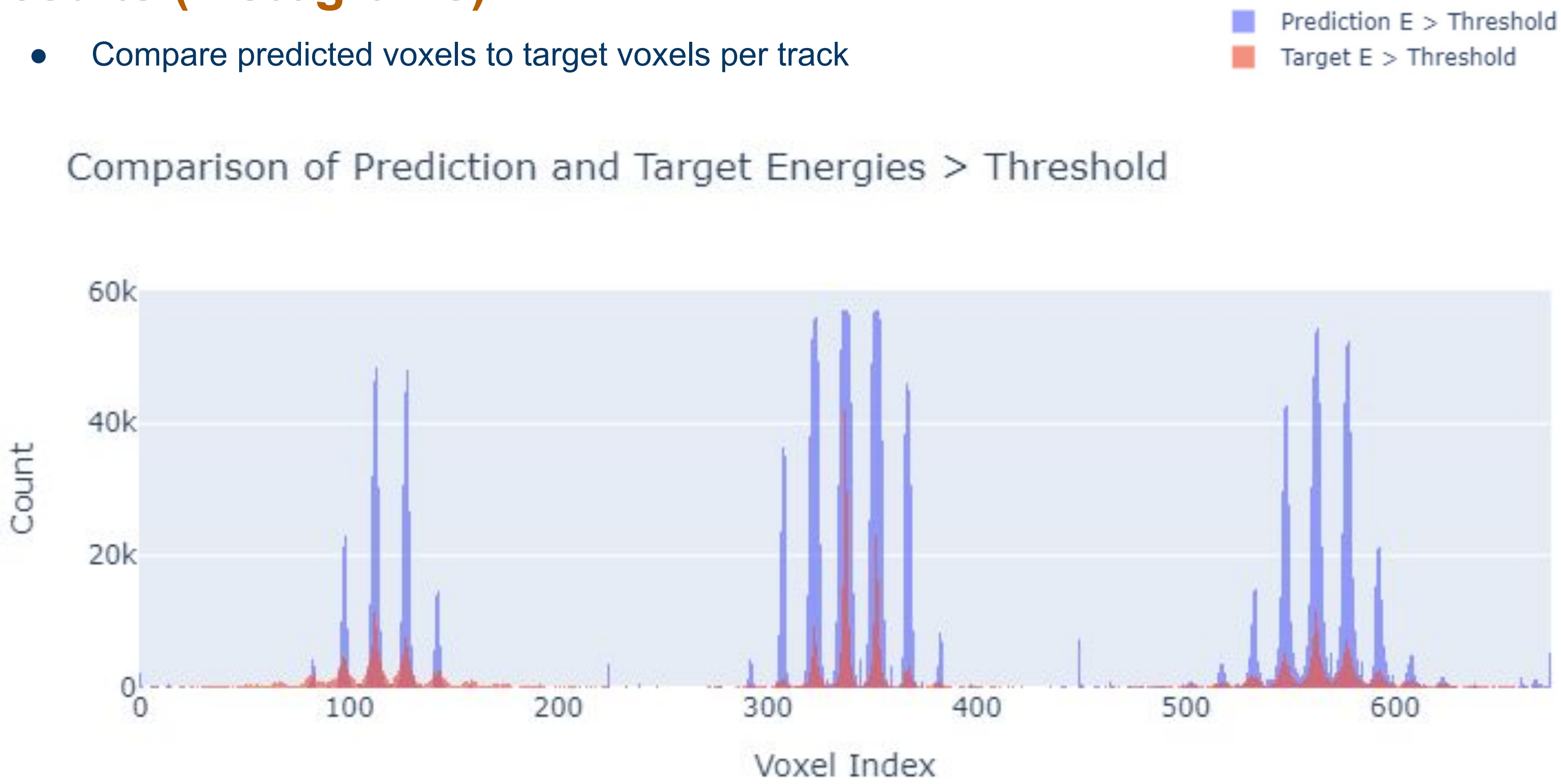
$$\text{loss} = weights_0 \cdot \text{loss}_{\text{track, scaled}} + weights_1 \cdot \text{loss}_{\text{nontrack, scaled}}$$

Results

train loss, val loss
— train loss
-- val loss
loss
0.2
0.15
0.1
0.05
0
epoch
0
200
400
600
800
1k
1.2k
1.4k

Results (histograms)

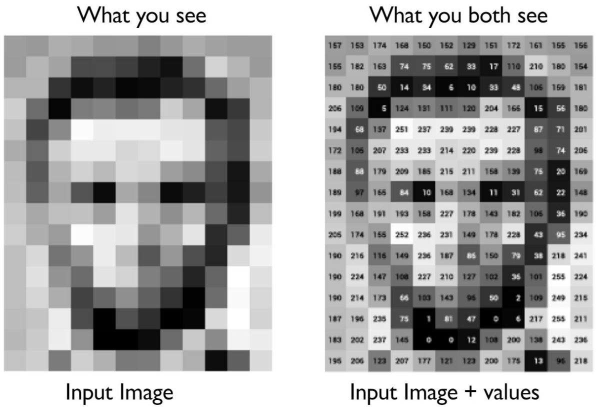
- Compare predicted voxels to target voxels per track



Backup Slides

Sparse CNNs

-



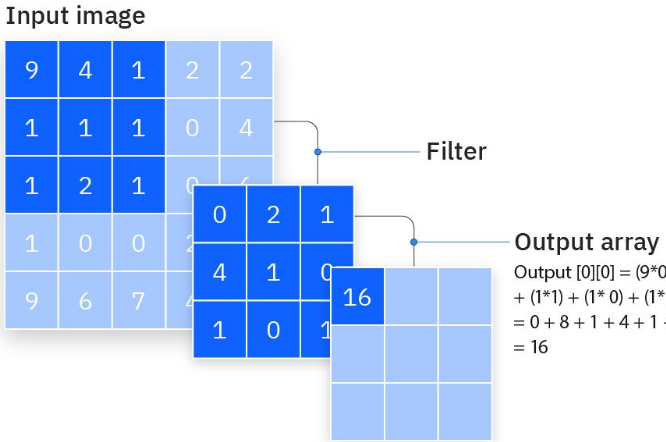
What the computer "sees"

157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218

Pixel intensity values
("pix-el"=picture-element)

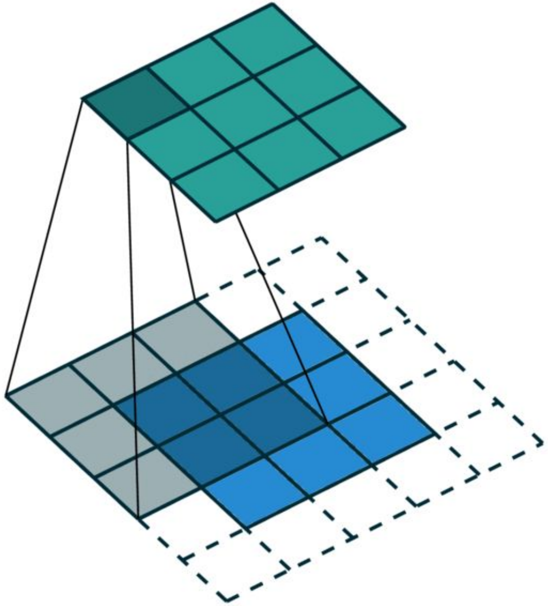
1.

2.



Output [0][0] = (9*0) + (4*2) + (1*4)
+ (1*1) + (1* 0) + (1*1) + (2* 0) + (1*1)
= 0 + 8 + 1 + 4 + 1 + 0 + 1 + 0 + 1
= 16

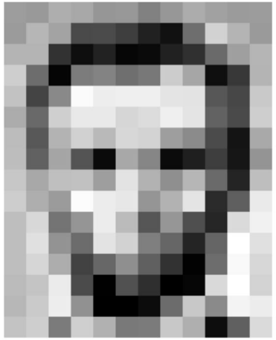
3.



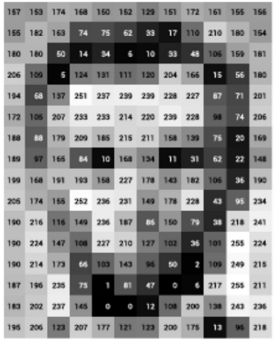
Sparse CNNs

-

What you see	What you both see	What the computer "sees"



Input Image



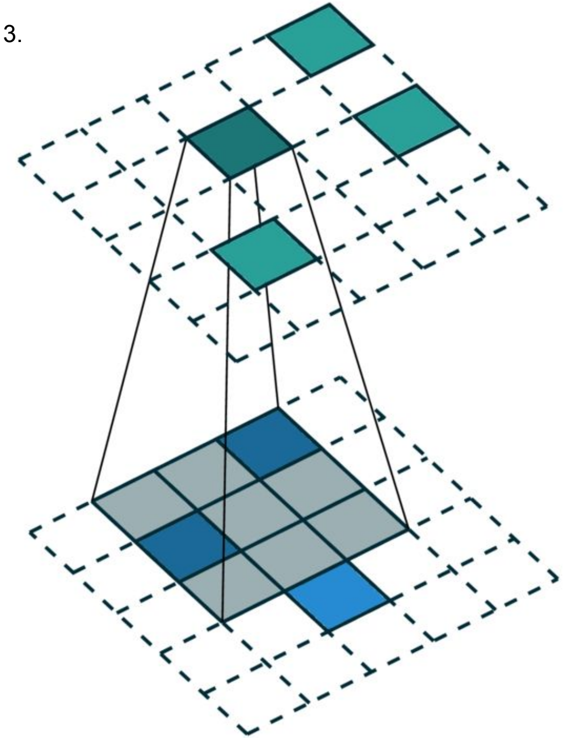
Input Image + values

157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218

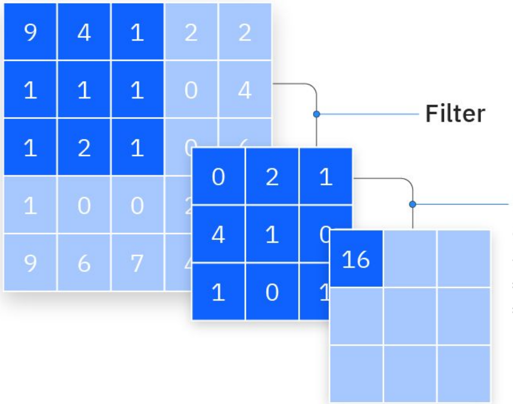
Pixel intensity values
(“pix-el”=picture-element)

1.

3.



2. Input image



Filter

Output array

Output [0][0] = (9*0) + (4*2) + (1*4)
+ (1*1) + (1* 0) + (1*1) + (2* 0) + (1*1)
= 0 + 8 + 1 + 4 + 1 + 0 + 1 + 0 + 1
= 16

1. https://mit6874.github.io/assets/sp2020/slides/L03_CNNs_MK2.pdf 2. https://shorturl.at/SbOyh 3. https://nvidia.github.io/MinkowskiEngine/_images/conv_dense.gif

Loss Function

- Loss metric:

$$\text{loss}_{\text{track}} = (\mathbf{predicted} - \mathbf{target})^2 \cdot \mathbf{mask}_{\text{track}}$$

$$\text{loss}_{\text{nontrack}} = (\mathbf{predicted} - \mathbf{target})^2 \cdot \mathbf{mask}_{\text{nontrack}}$$

$$\mathbf{mask}_{\text{track}} = \begin{cases} 1 & \text{if } \mathbf{target} \neq 0 \\ 0 & \text{if } \mathbf{target} = 0 \end{cases}$$

$$\mathbf{mask}_{\text{nontrack}} = \begin{cases} 1 & \text{if } \mathbf{target} = 0 \\ 0 & \text{if } \mathbf{target} \neq 0 \end{cases}$$

$$\text{loss}_{\text{track, scaled}} = \frac{\sum \text{loss}_{\text{track}}}{\text{numel}(\mathbf{mask}_{\text{track}})}$$

$$\text{loss}_{\text{nontrack, scaled}} = \frac{\sum \text{loss}_{\text{nontrack}}}{\text{numel}(\mathbf{mask}_{\text{nontrack}})}$$

$$\mathbf{weights} = \text{softmax}(\log \mathbf{weights}_{\text{param}})$$

$$\text{loss} = weights_0 \cdot \text{loss}_{\text{track, scaled}} + weights_1 \cdot \text{loss}_{\text{nontrack, scaled}}$$