Characterizing Charge Defects in a Double Quantum Dot Device using Machine Learning

Presented by Zeest Fatima, University of Waterloo

Table of Contents

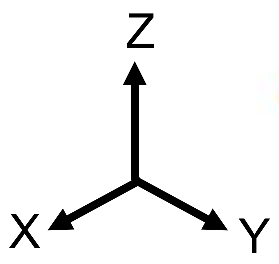
What are spin qubits

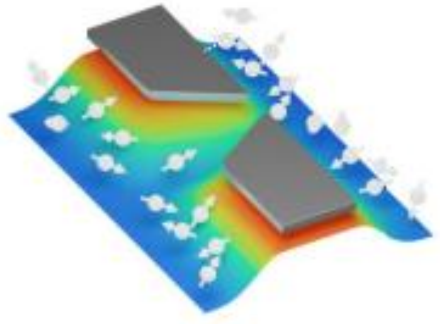
- Spin of an electron or a nucleus functions as an excellent qubit, as it provides a natural two-level system that is insensitive to electric fields, leading to long quantum coherence times
- There are 4 main kinds of spin-qubits:
    - Loss-DiVincenzo spin qubit
    - Singlet-triplet qubits
    - Donor spin qubits
    - Exchange-only and resonant-exchange qubits

We use this kind of qubits, specifically looking at the electron spin

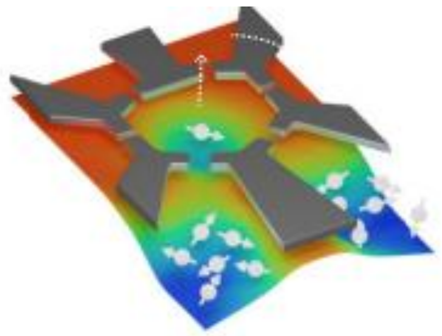
Electrostatically defined quantum dots

- Spin qubits in semiconductor devices rely on the 3D confinement of electrons. Electrons are confined in these potential wells, using voltage gates.

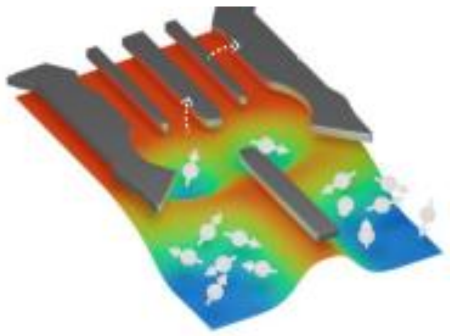




1) Free flowing singular electrons in the device



2) A single electron trapped between voltage gates. The voltage gates are just electrodes, and by applying voltages to different gates, they can manipulate the electric potential to create potential wells that then trap single electrons. Called a **single dot**.

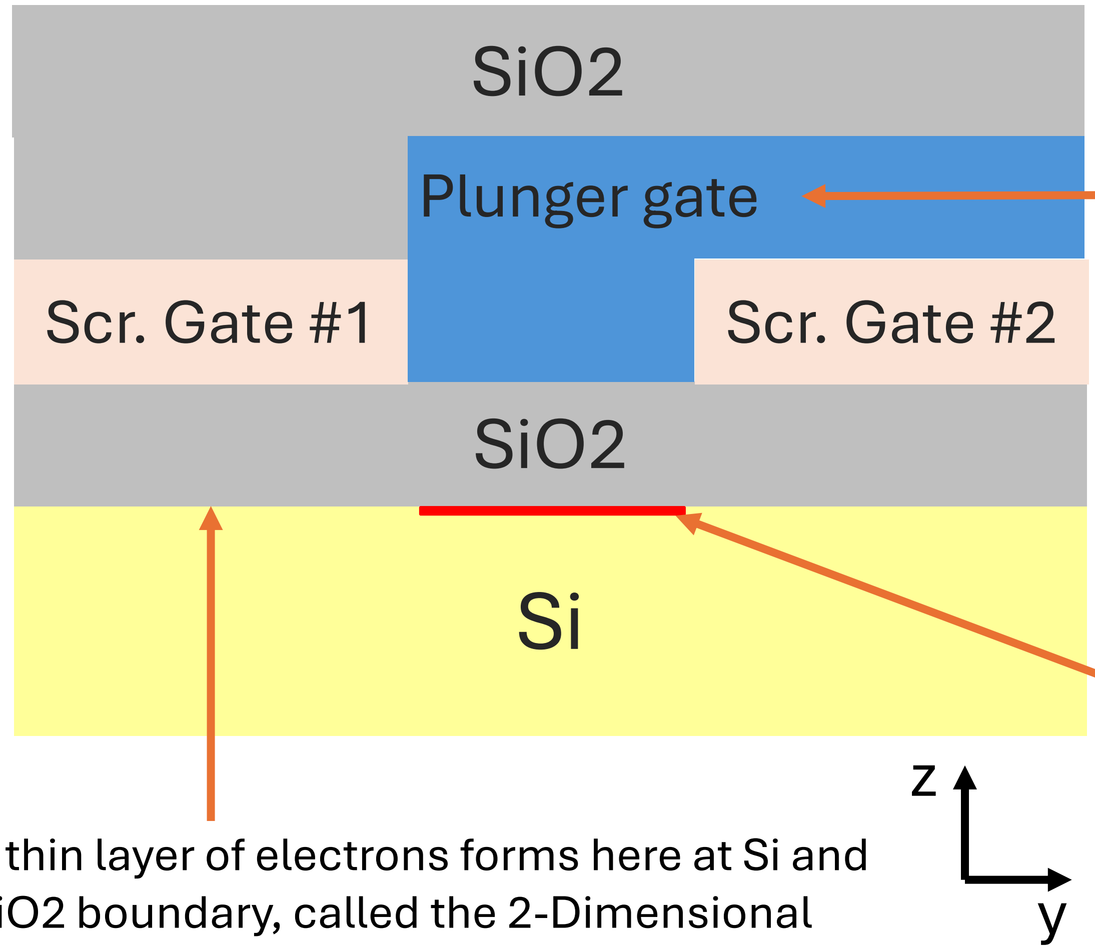


3) This could also be done for 2 electrons, and that is then called a **double dot.**

Plots taken from Burkard et al.: Semiconductor spin qubits

What our Device looks like

Cross-sectional view

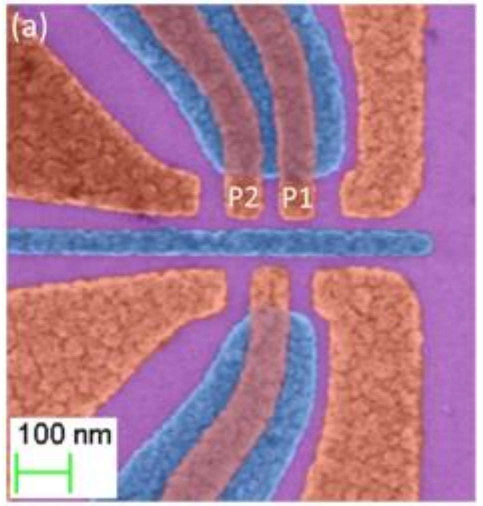


Plunger gates are electrodes that apply voltages and cause potential wells to be formed, trapping single electrons

A thin layer of electrons forms here at Si and SiO2 boundary, called the 2-Dimensional Electron Gas (2DEG)

quantum dots are formed right underneath plunger gate, in the 2DEG

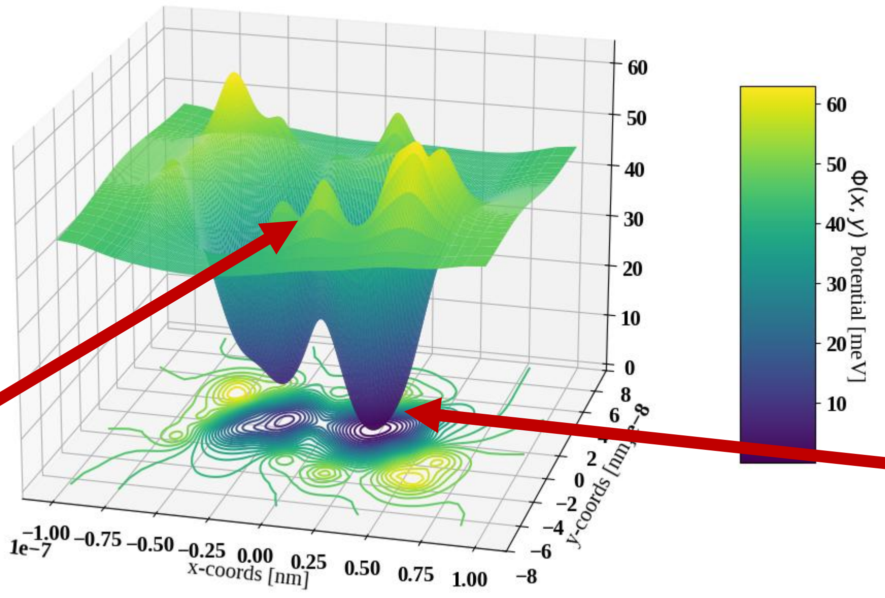
Top view of different gates



Electrostatic disorder in device

- Due to “charge defects” in the device, the electrostatic potential landscape of the quantum dot device could be affected.
- If we know defects’ position, we can

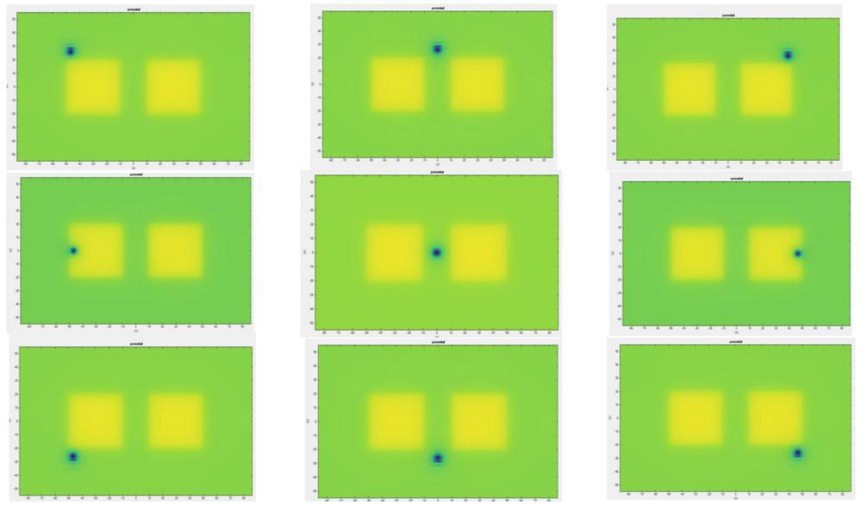
3D potential landscape of the double-dot device



Disorder in the potential landscape due to charge defects

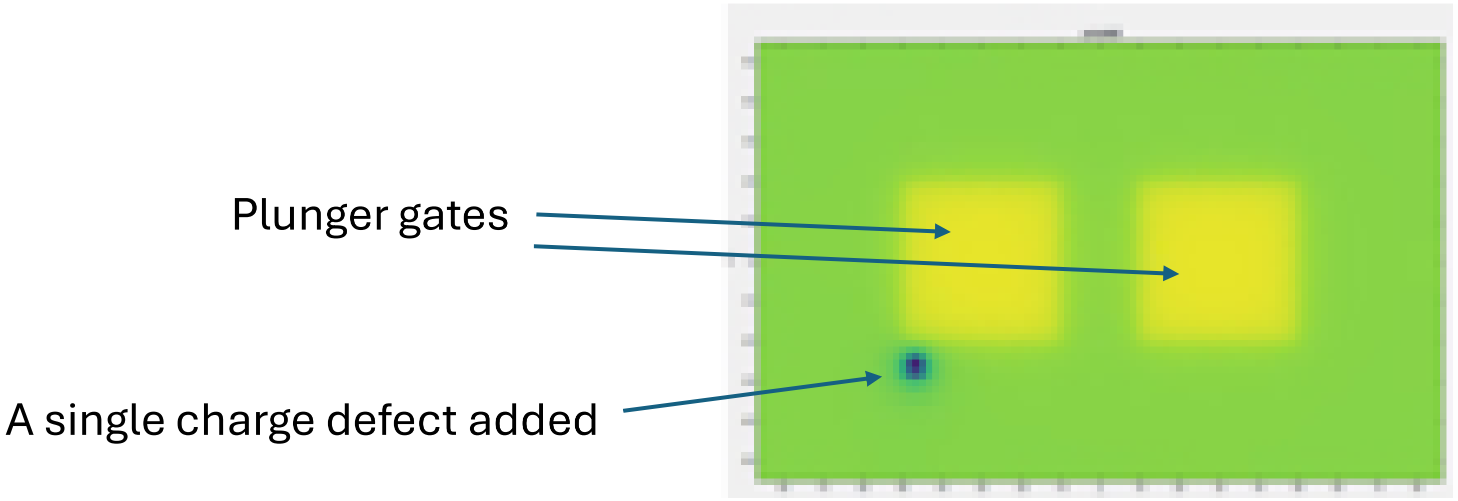
Potential wells in which the two electrons are trapped, forming the two quantum dots

Simulating disorder

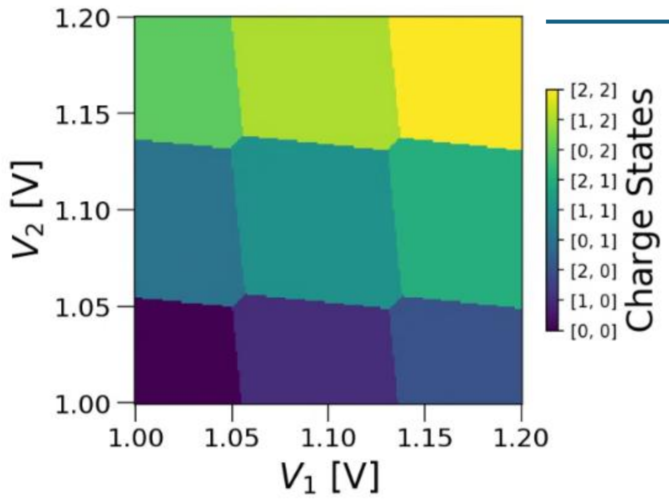


Adding disorder manually:

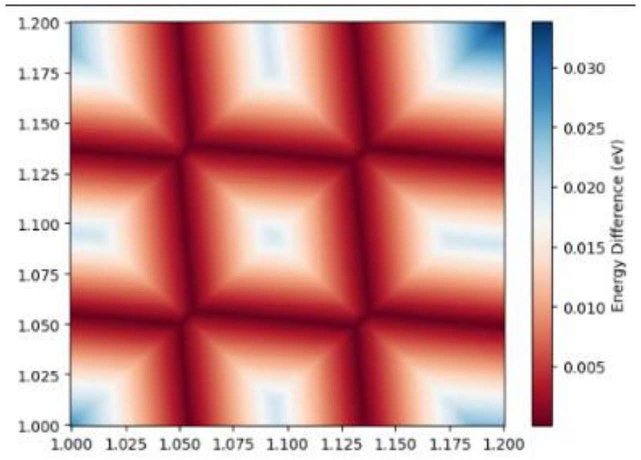
- Charge: 1e per sphere
- Radius 2nm
- Fixed in a plane 4nm above the quantum dot layer

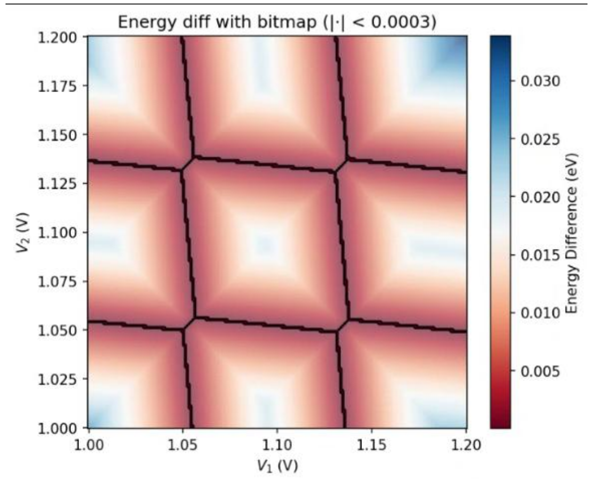


CNN Input

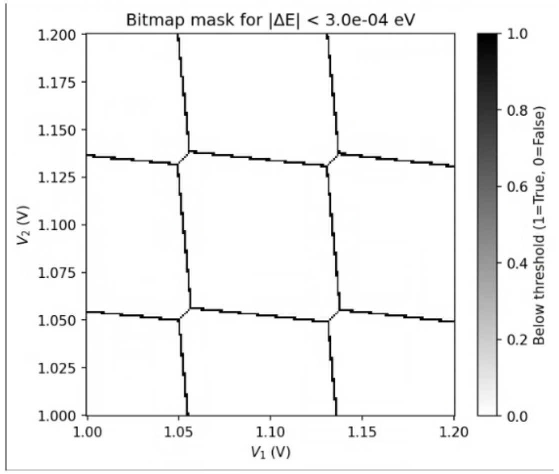


Charge Stability Diagram (CSD)



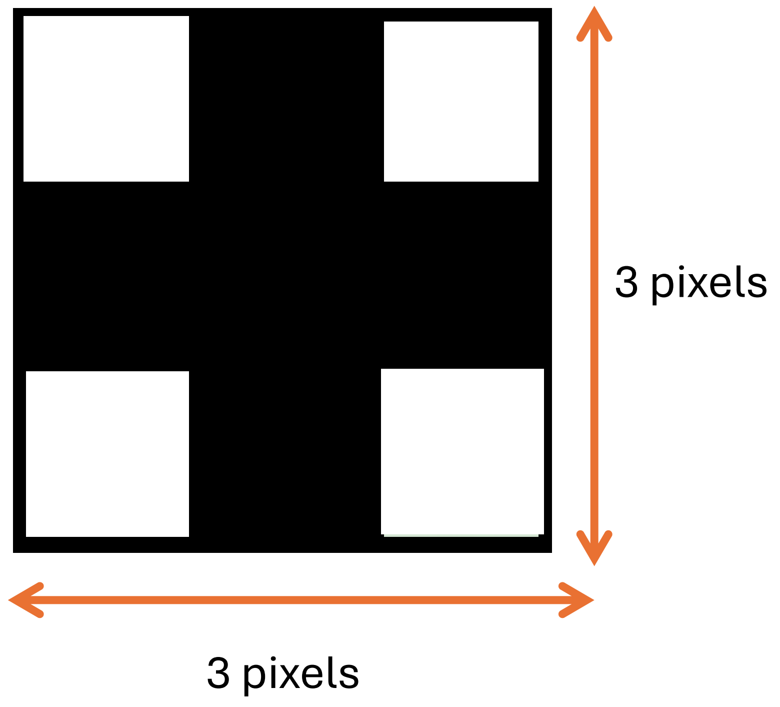


Input

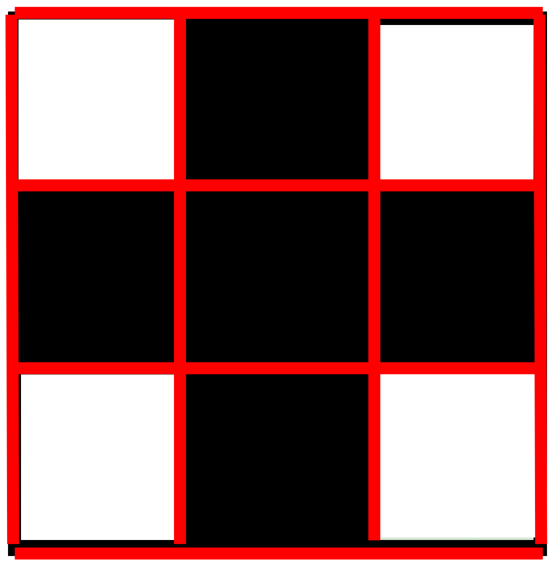


Implementing a CNN with CSDs

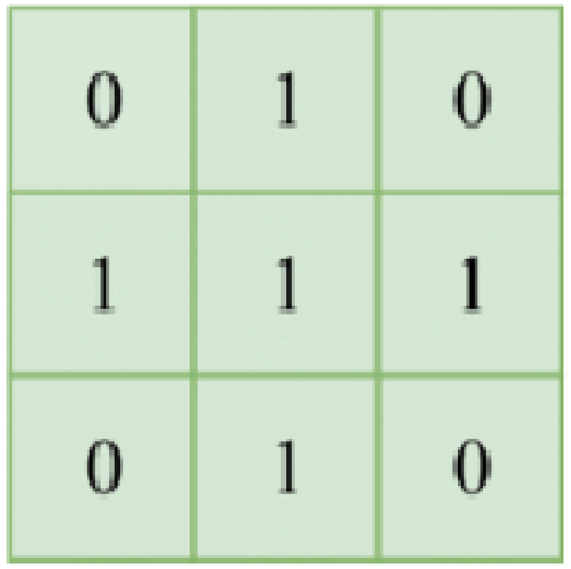
Instructional diagrams taken from the book “The Hundred Page Machine Learning Book”



Say we have a black and white image (of area 3x3 pixels)



We can impose a 3x3 grid on it, letting each box encompass 1 pixel

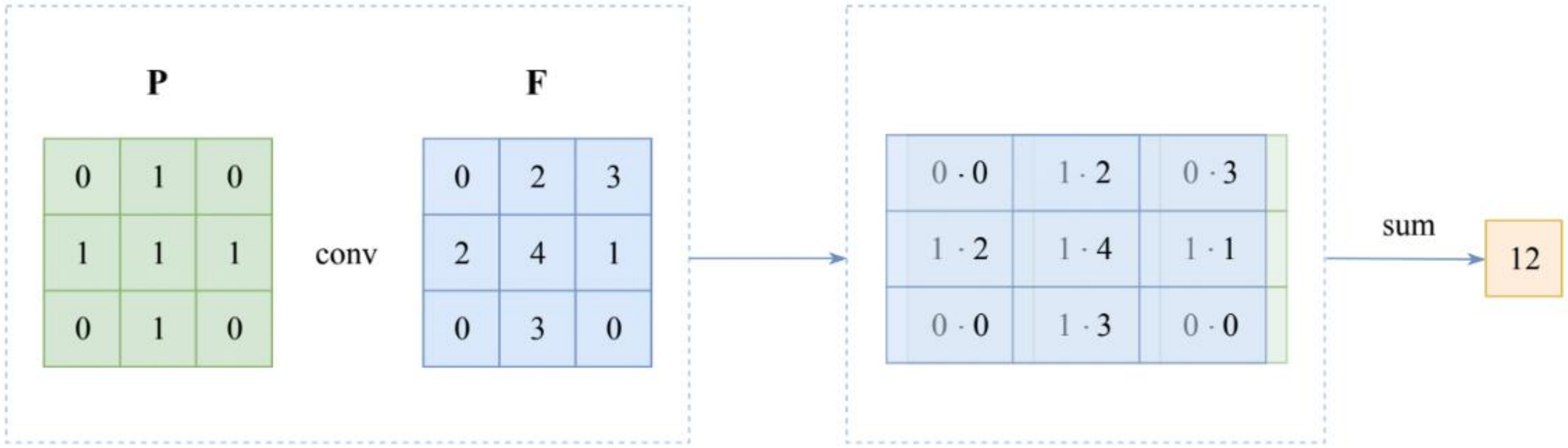


0	1	0
1	1	1
0	1	0

Then convert the image into a 3x3 matrix by assigning 1 to black box and 0 to any white box

Convolution operation:

- maps matrices to real numbers



$$P = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{pmatrix} \quad \text{conv} \quad F = \begin{pmatrix} 0 & 2 & 3 \\ 2 & 4 & 1 \\ 0 & 3 & 0 \end{pmatrix} \rightarrow \begin{pmatrix} 0 \cdot 0 & 1 \cdot 2 & 0 \cdot 3 \\ 1 \cdot 2 & 1 \cdot 4 & 1 \cdot 1 \\ 0 \cdot 0 & 1 \cdot 3 & 0 \cdot 0 \end{pmatrix} \xrightarrow{\text{sum}} \boxed{12}$$

matrix from the previous slide

a “filter” matrix of our choice

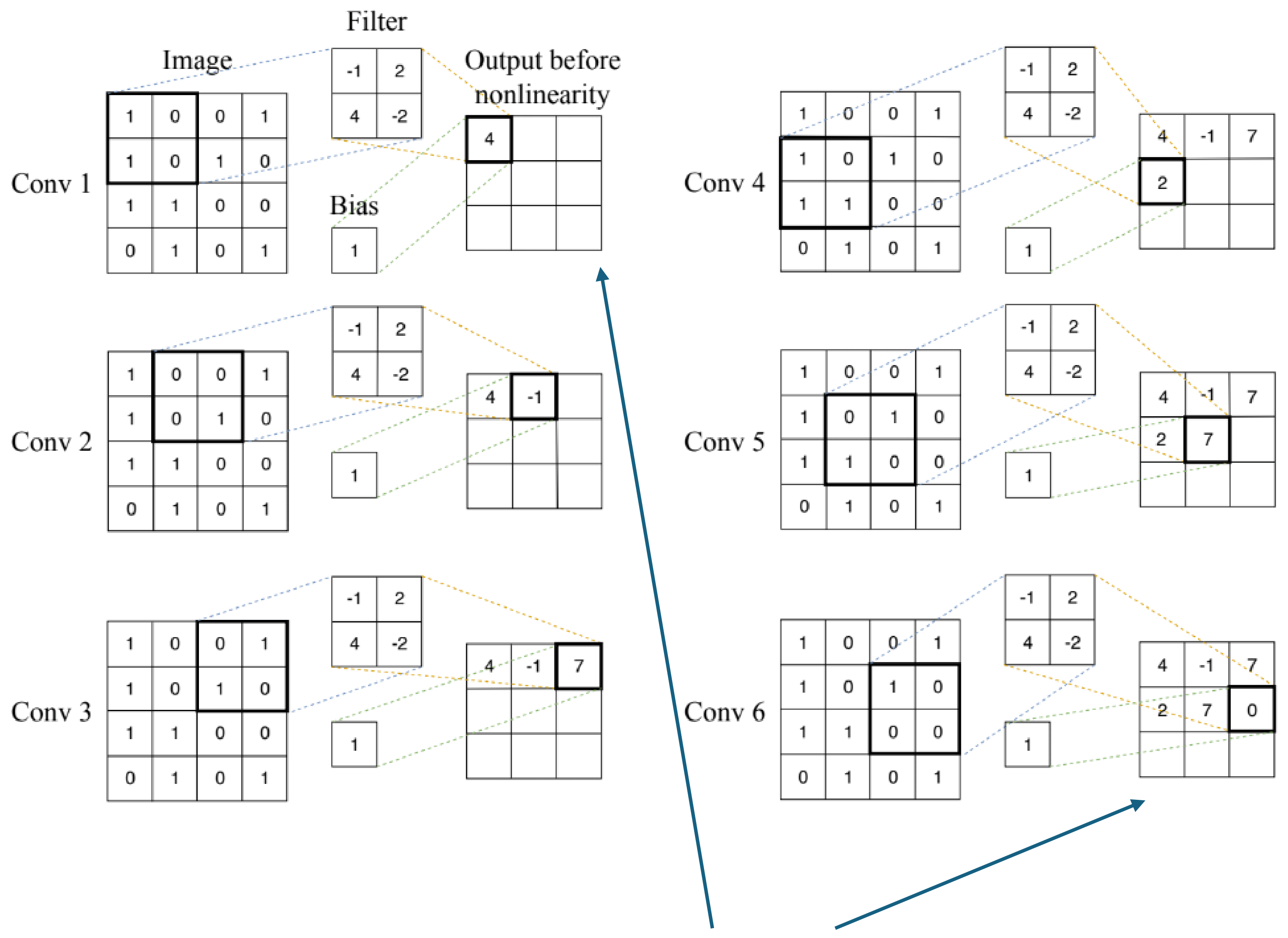
Applying the same filter to a different matrix gives a different real number

$$\mathbf{P} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} \text{ conv } \overset{\mathbf{F}}{\begin{array}{|c|c|c|} \hline 0 & 2 & 3 \\ \hline 2 & 4 & 1 \\ \hline 0 & 3 & 0 \\ \hline \end{array}} \xrightarrow{\text{overlay}} \begin{bmatrix} 1*0 & 0*2 & 0*3 \\ 1*2 & 0*4 & 0*0 \\ 1*0 & 1*3 & 1*0 \end{bmatrix} \xrightarrow{\text{sum}} 5$$

Application

Big images have many pixels so their matrix's dimensions would be very large.

Using a "moving window" approach we could apply filter to "windows" of the matrix and try to end up with a smaller matrix, while trying to maintain information about its "features".



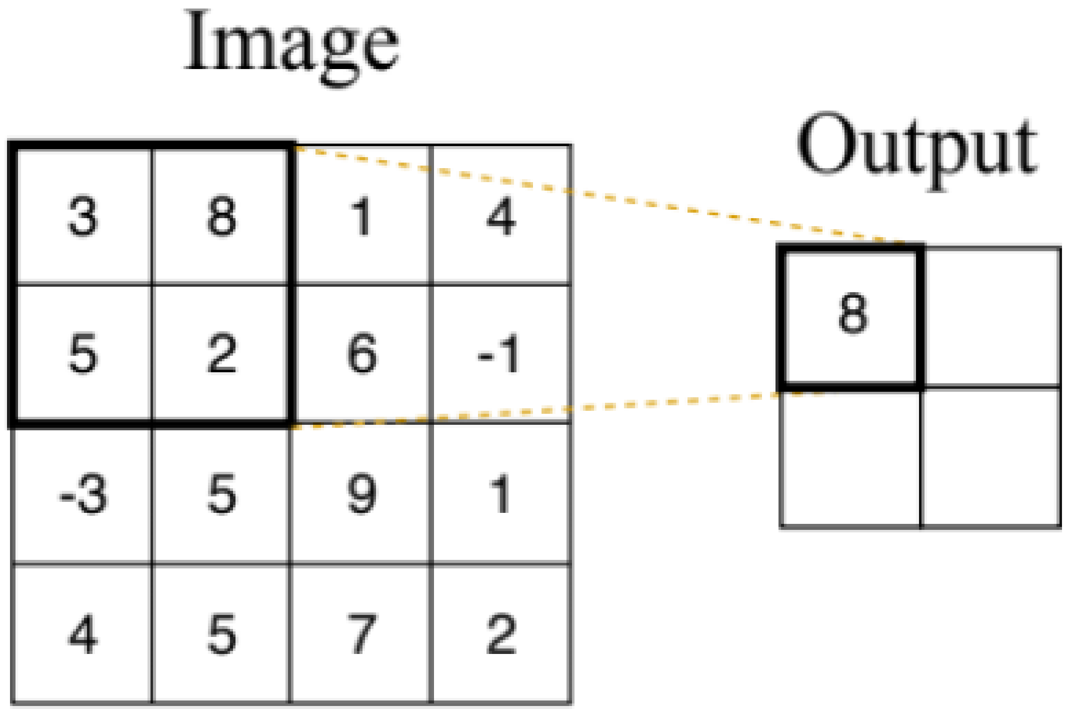
New smaller matrix being built using "moving window" approach

Pooling:

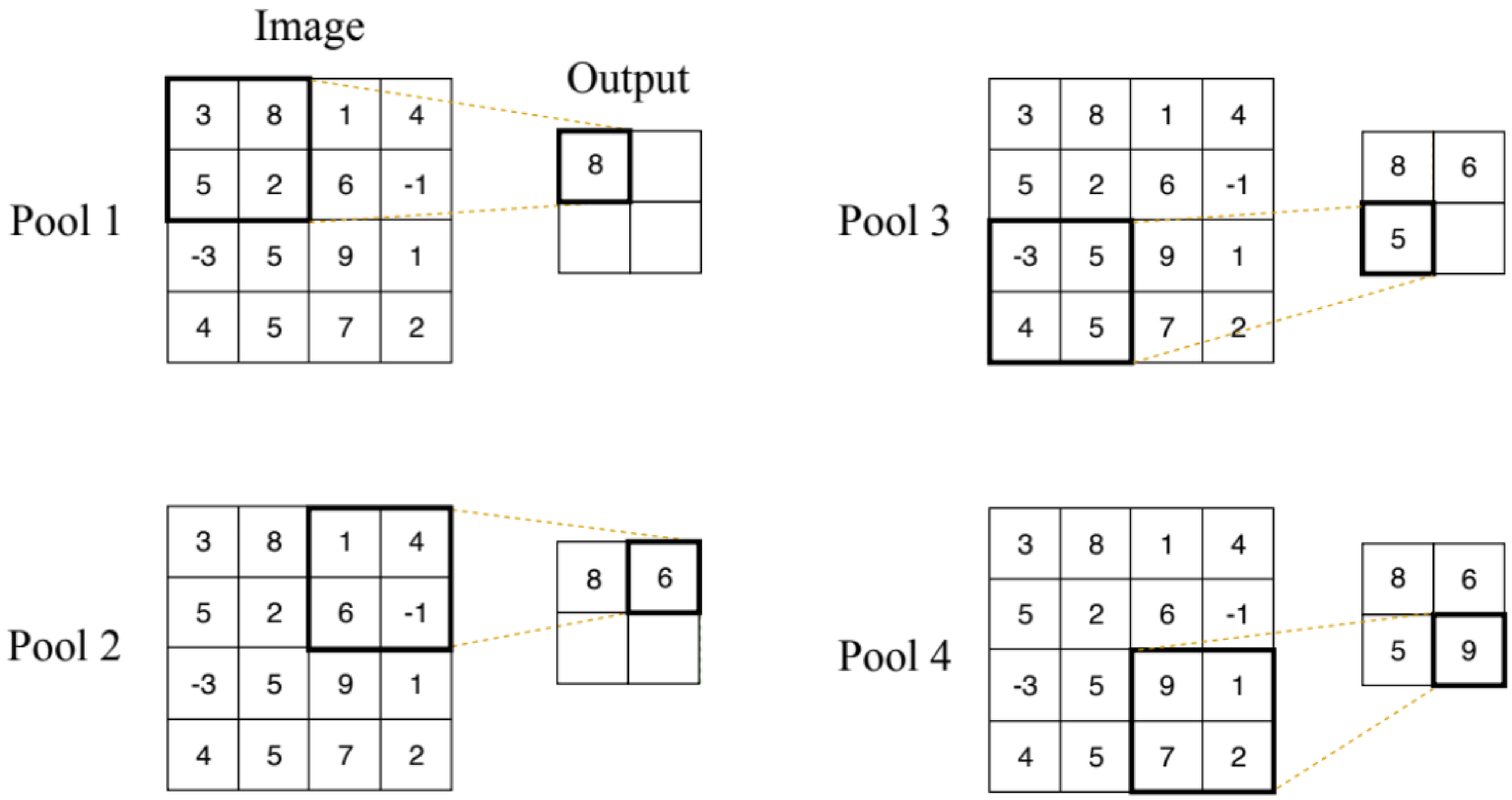
A similar operation to convolution, can involve summing up all entries in a "window".

Another attempt at reducing the bigger matrix down into a smaller one without losing info about features.

Is usually done following a convolution layer

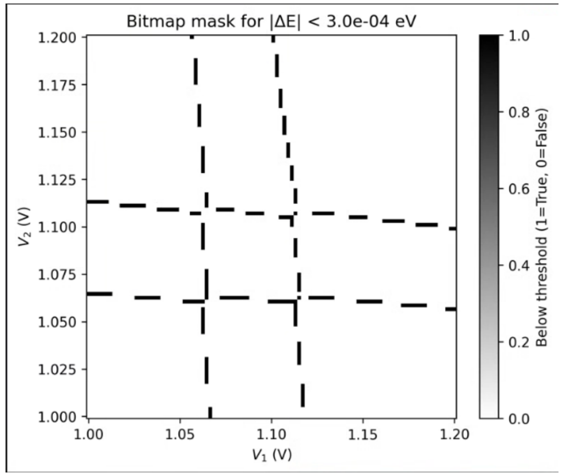


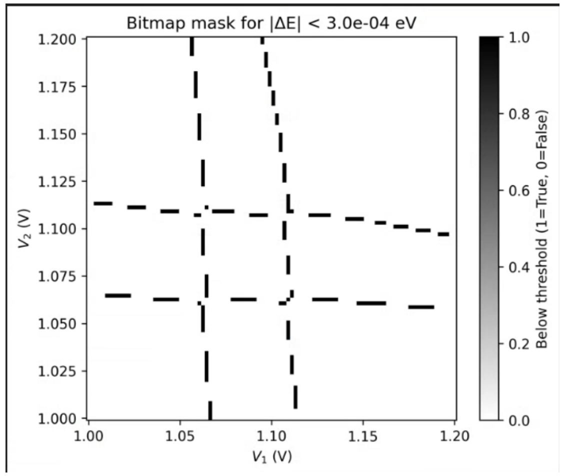
This is also done using a “moving window” approach

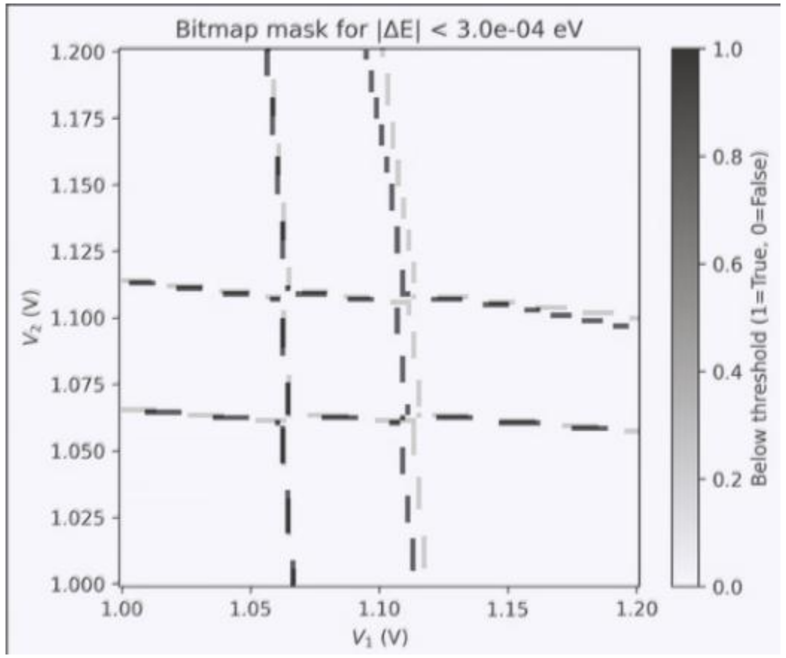


Sample application

This is what the overlap between two of our disordered CSDs looks like. This is for disorder in position (1,1) vs (1,2).



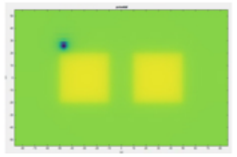


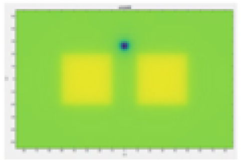


Defect in position (1,1)

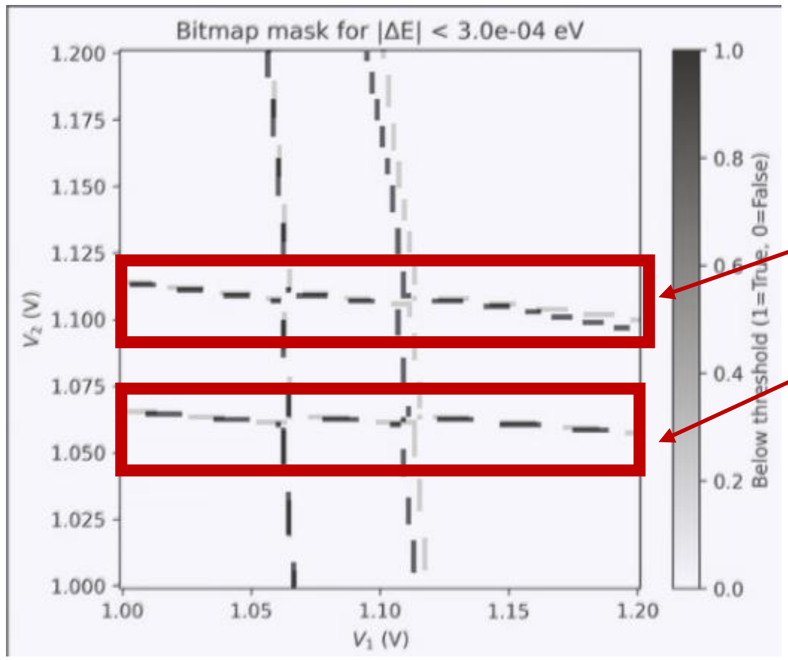
Defect in position (1,2)

Overlapping the two images

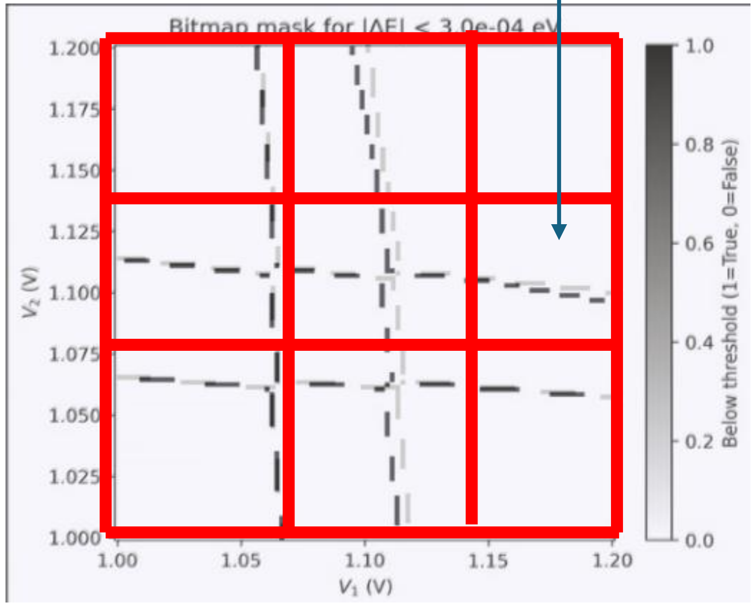


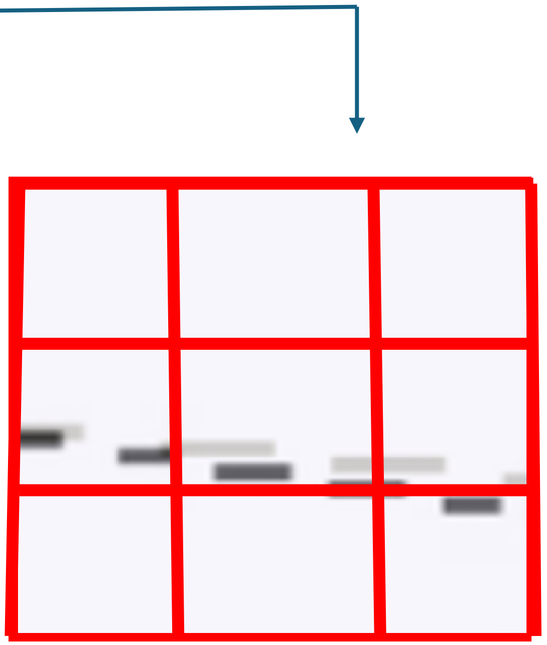


Sample application

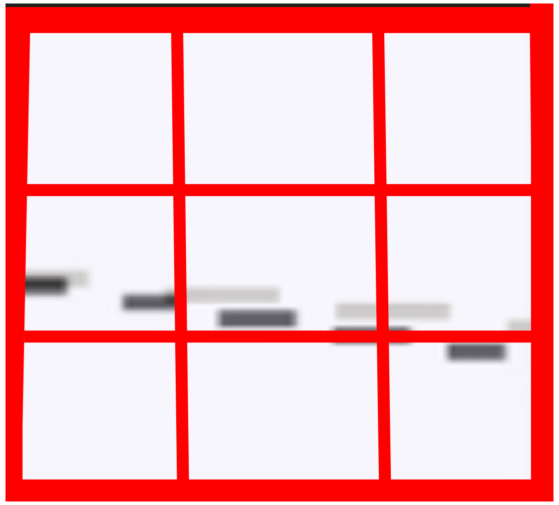


These areas seem important for trying to differentiate between the two CSDs





Say we take this region of the image and convert it into a matrix (2 matrices in this case, since the image shows two overlapping CSDs, so a matrix for each CSD)



Assume value of each pixel of these colors is:

□ = 0

■ =1 (for CSD 1)

░ =1 (for CSD 2)

Matrix entries for each grid box can then take values for % occupied

Matrix for CSD 1 (defect in position (1,1)):

$$\begin{bmatrix} 0 & 0 & 0 \\ 20 & 12 & 3 \\ 0 & 0 & 7 \end{bmatrix}$$

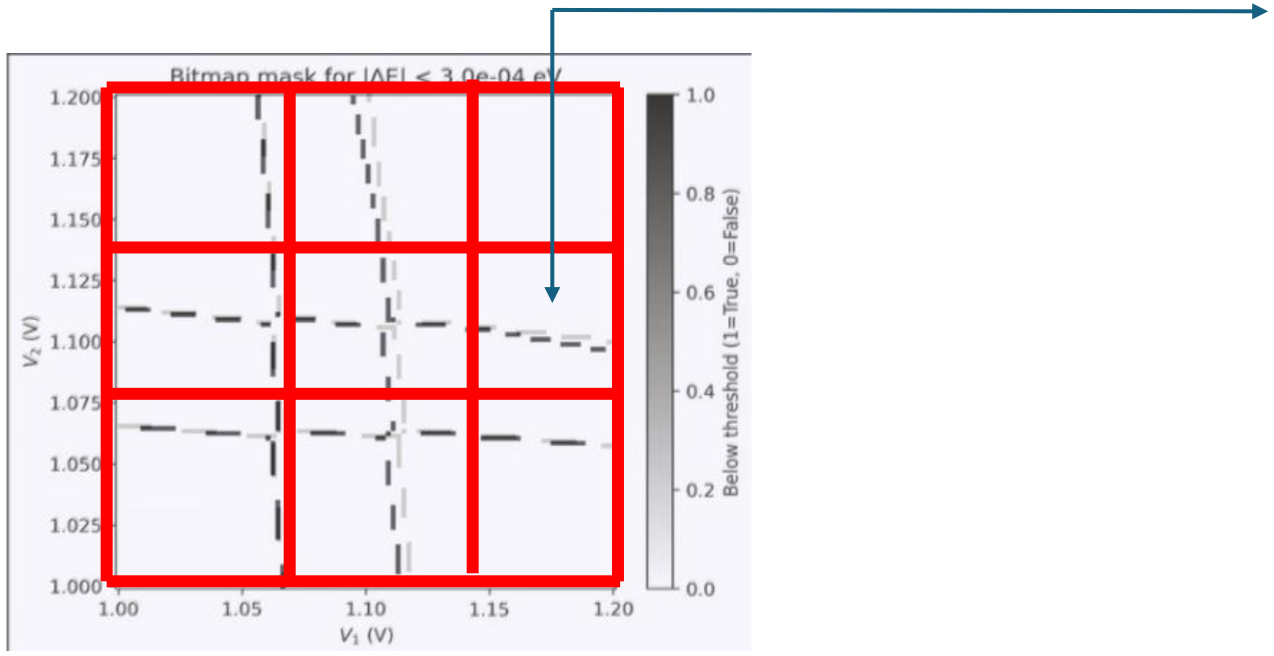
Matrix for CSD 2 (defect in position (1,2)):

$$\begin{bmatrix} 0 & 0 & 0 \\ 10 & 26 & 15 \\ 0 & 0 & 0 \end{bmatrix}$$

Sample convolution filter that assigns higher weight to difference between cells (2,1), (2,2) and (2,3) to extract/ emphasize horizontal shift:

$$\begin{bmatrix} 0 & 0 & 0 \\ 5 & 5 & 5 \\ 0 & 0 & 0 \end{bmatrix}$$

Value after convolution for CSD 1: 20*5+12*5+3*5= **175**
Value after convolution for CSD 2: 10*5+26*5+15*5= **255**



So just in this grid cell, defect in (1,1) has value 175 while defect in position (1,2) has value 255.

So an ML model can draw a “decision boundary” to classify a defect in position (1,1) vs (1,2) such as:

if grid cell value <215 , output = (1,1)
if grid cell value >215 , output = (1,2)

In practice, an ML model uses optimization techniques to come up good filter matrices in order to create decision boundaries that most frequently predict the correct classes.

Specifications of our CNN

Architecture:
→ 3 Convolution blocks, no pooling, so local features are preserved
→ 1×1 channel mixing
→ delayed pooling
→ global average pooling
→ fully-connected classifier

Training images: 5 per class, Testing images: 5 per class
Highest accuracy achieved: 87% ,
Lowest validation loss achieved: 0.85 , Lowest training loss achieved: 0.46
Steadily decreasing validation loss

Is the model physically accurate?

- Increasing accuracy of simulations adds cost – the most accurate simulations of the electrostatic potential of the device would entail evaluating the Schoedinger equation as well as the Poisson equation, which involves first finding correct “turn-on voltages” for each unique defect case. Furthermore Schroedinger-Poisson simulations for our device (using nextnano) take up a lot of time: ~255 minutes (4.25 hours) for each **unique** defect configuration to be simulated! (we had at the **minimum 90 unique defect cases**!)
- Alternatively, what if we train the model on close estimations of the correct electrostatic potential and evaluate (validate) it on the more physically accurate simulations?

Thank you!