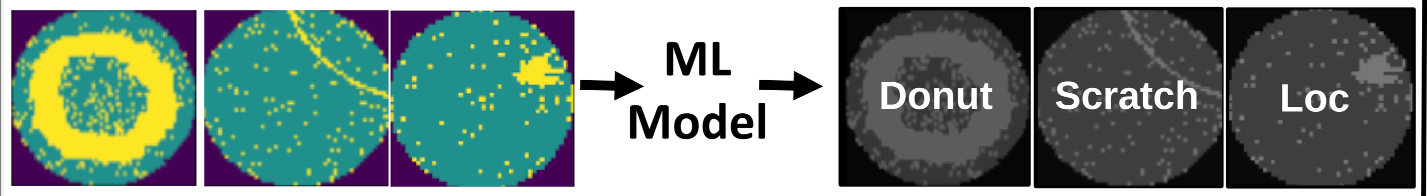
Energy-efficient Wafer Defect Detection using Spiking Neural Networks

Henry Alexander Lepp, Electrical Engineering B.S.E.
Mentor: Dr. Leslie Hwang, Assistant Professor
Electrical, Computer, and Energy Engineering (ECEE)

Impact Statement

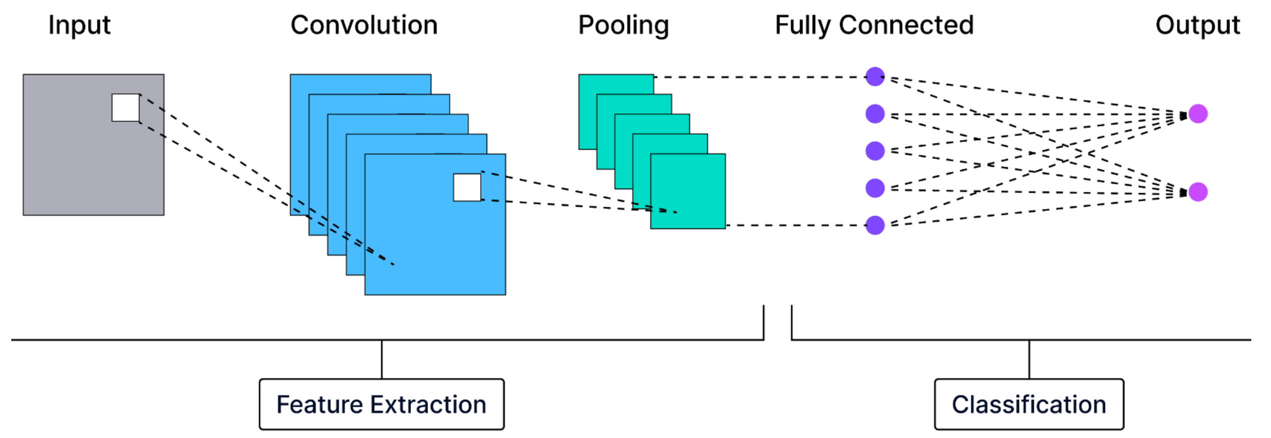
Developing an energy-efficient method for silicon defect detection would allow assessment of production techniques in a more sustainable way

Wafer Defect Classification



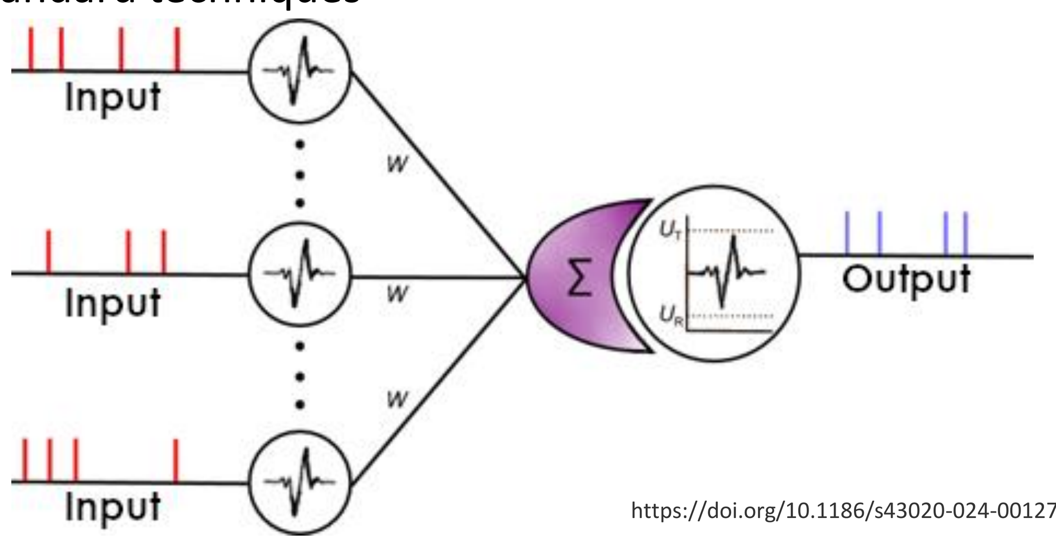
CNN vs. SNN

- Convolutional Neural Networks are a widely-used form of Deep Learning Model
- Feature Extraction
  - Convolution applies "filters" to the image, but increases size
  - Pooling reduces size by grouping several pixels into one
- Classification
  - Dense layers translate flattened image into output values
  - Result is a vector of probabilities



https://zilliz.com/glossary/convolutional-neural-network

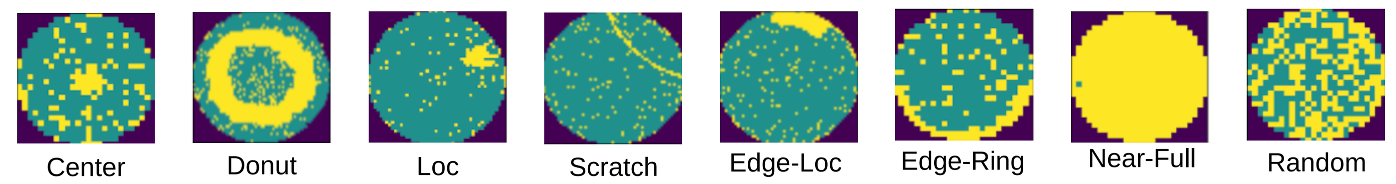
- Spiking Neural Networks mimic real brains
- Feature extraction is similar to CNNs but data is encoded as impulses
- Each impulse increases the neurons membrane potential until it activates at a certain threshold value
- Since neurons aren't always active, it generally uses less power
- However, it cannot be trained as effectively using standard techniques

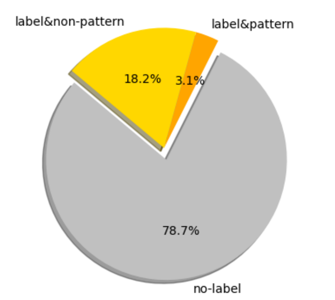


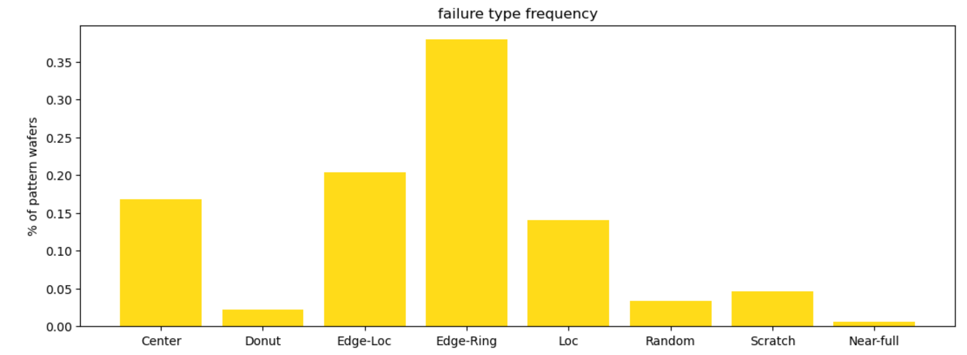
https://doi.org/10.1186/s43020-024-00127-8

Results

http://mirlab.org/dataSet/public/





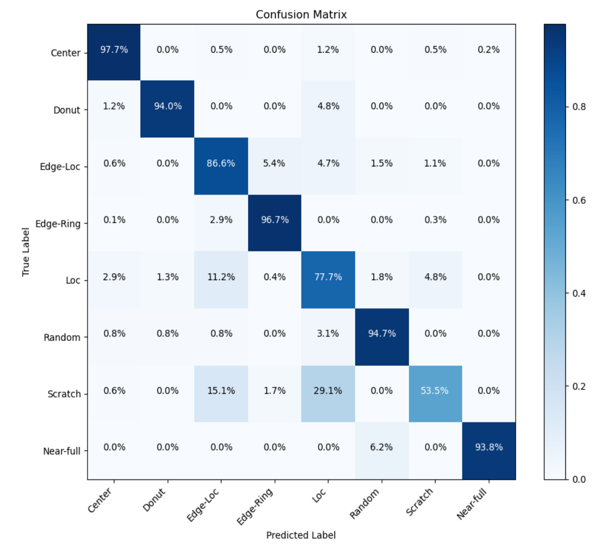


- The dataset used for the wafer images was WM811K
- Only patterned wafers could be used for training

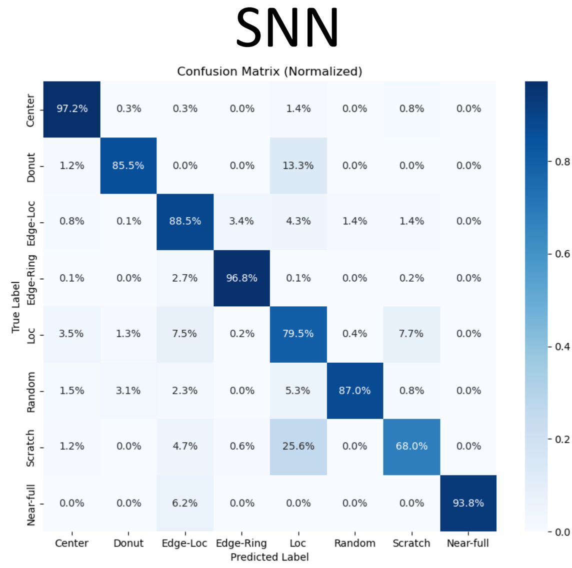
WM811K Data Distribution

Total Wafers	Labelled Wafers	Patterned Wafers	Training Set
811,457	172,950	25,519	17,863

CNN



SNN



Model Performance Comparison

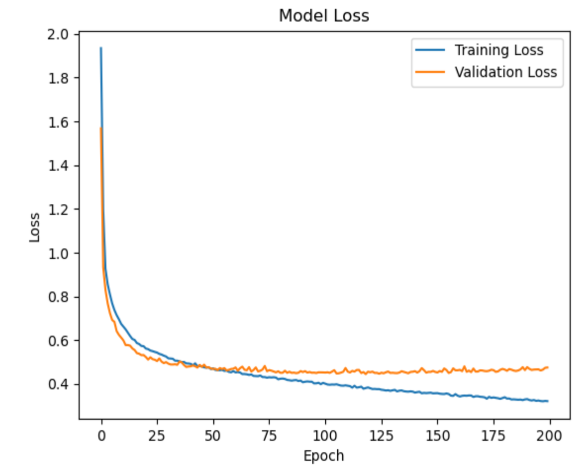
	Accuracy	GFLOPS	Energy	Inference Time
CNN	0.90	0.22	15.96 Wh	2.68 ms
SNN	0.91	0.20	1.579 Wh	717.35 ms

Model Hyperparameters

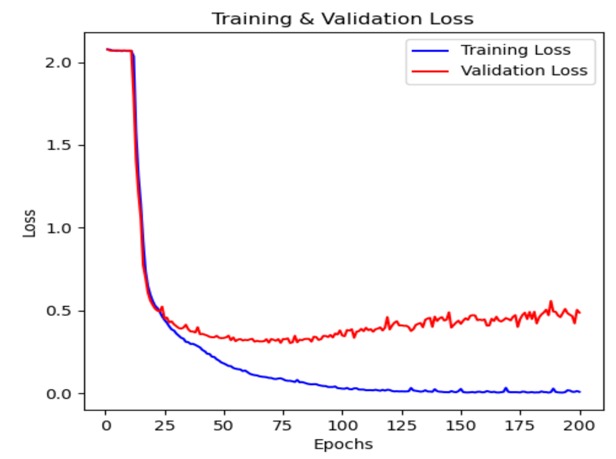
Hyperparameter	CNN	SNN
Epochs	200	200
Batch Size	32	32
Dropout Rate	0.50	0.50
Dense Layers	2	2
Neurons	256	128
Layer Sizes	64,128,256	32,64,128
Learning Rate	0.0001	0.001

Hyper-parameter	CNN	SNN
Activation Fn	ReLu	N/A
Regularizer	0.001	N/A
Spike Threshold	N/A	1
Decay Rate	N/A	0.9
Time Steps	N/A	5

CNN



SNN



Experimental Methods

List of Hyperparameters

- Learning Rate – How much the model changes each epoch
- Epochs – # of times a model is tested and adjusted (trained)
- Batch Size – # of samples analyzed before weights are adjusted
- Dropout Rate – Percent of inputs removed to prevent overfitting
- Activation Function (CNN) – Function to add non-linearity to output
- Layer Size – # of filters for feature extraction
- Surrogate Gradient (SNN) – Approximates spikes as differentiable functions
- Decay Rate (SNN) – Rate membrane potential decays w/ time
- Regularizer – Changes the model less as training progresses
- # Neurons – How many neurons are in the fully connected layers
- Spike Threshold (SNN) – min. membrane potential before neuron "fires"

Model Training Procedure

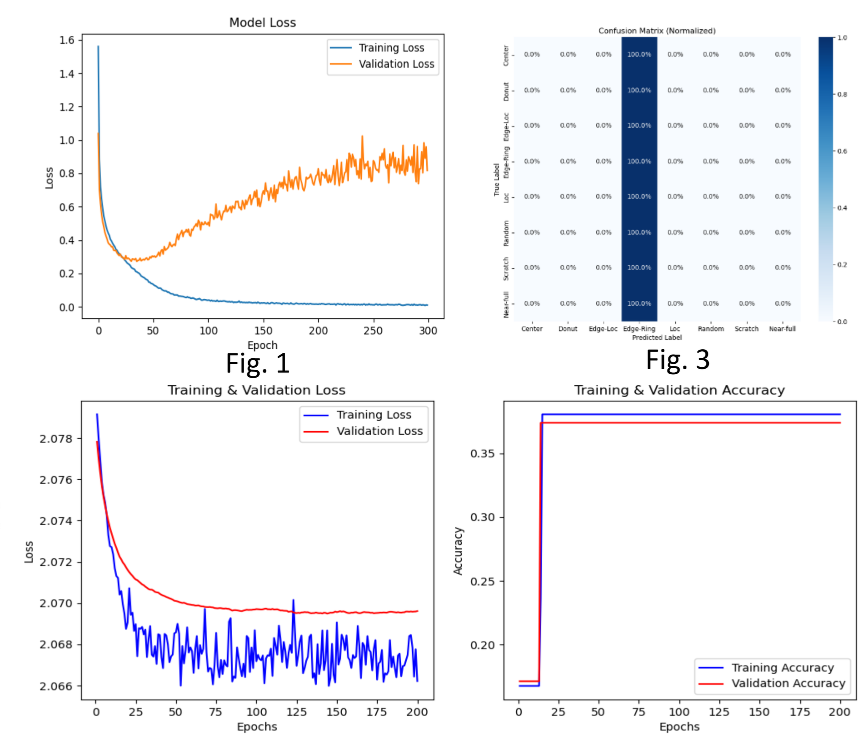
1) A hyperparameter was tuned and its new value recorded
2) The accuracy and loss of the new model was recorded along with the effect of the hyperparameter

$$Loss = -\frac{1}{N}\sum_{i=1}^{N} y_i * \ln(\hat{y}_i)$$

$N = \# \, Samples; y_i = True \, Label; \, \hat{y}_i = Probability \, of \, Prediction$

Challenges Faced

- Initial installation of Python was broken, crashing runs
- SNN initially couldn't detect GPUs; instead, trained using slower CPU
- Validation loss for CNN kept increasing overtime, rather than converging to a value (Fig. 1)
  - Resolved by adding a regularizer
- SNN accuracy was held constant for many epochs (Fig. 2); model predicted everything to be one class (Fig. 3)
  - Resolved by removing the surrogate gradient



FURI

Ira A. Fulton Schools of Engineering
Arizona State University