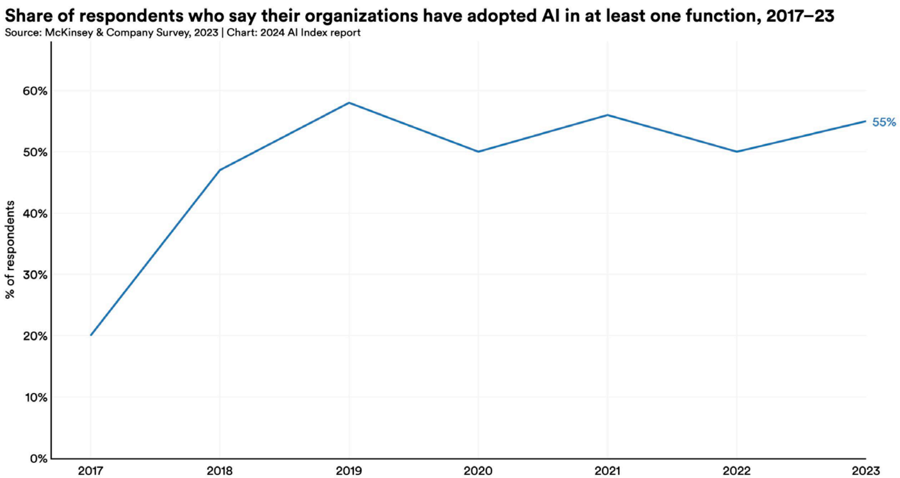
THE OHIO STATE UNIVERSITY

Data-Aware Tuning of Deep Learning Models

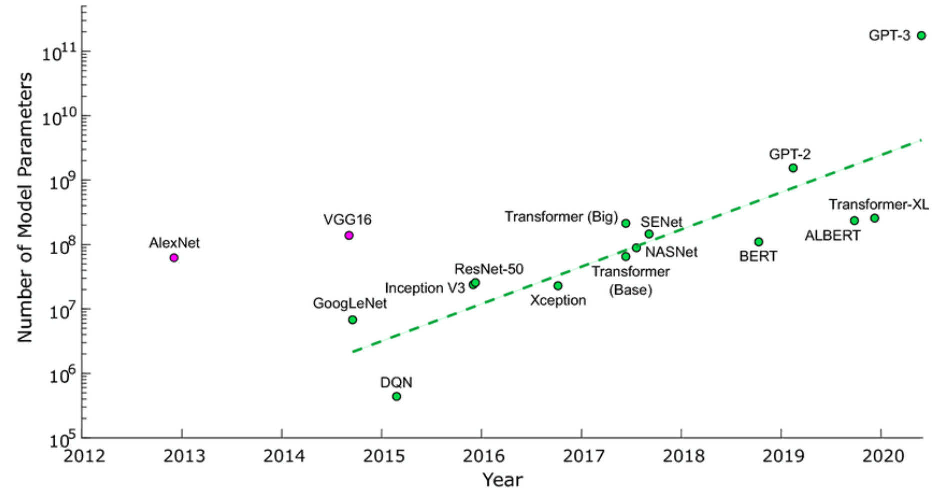
Anthony Baietto

Committee: Dr. Chris Stewart, Dr. Trevor Bihl (AFRL)
Dr. Radu Teodorescu, Dr. Mi Zhang

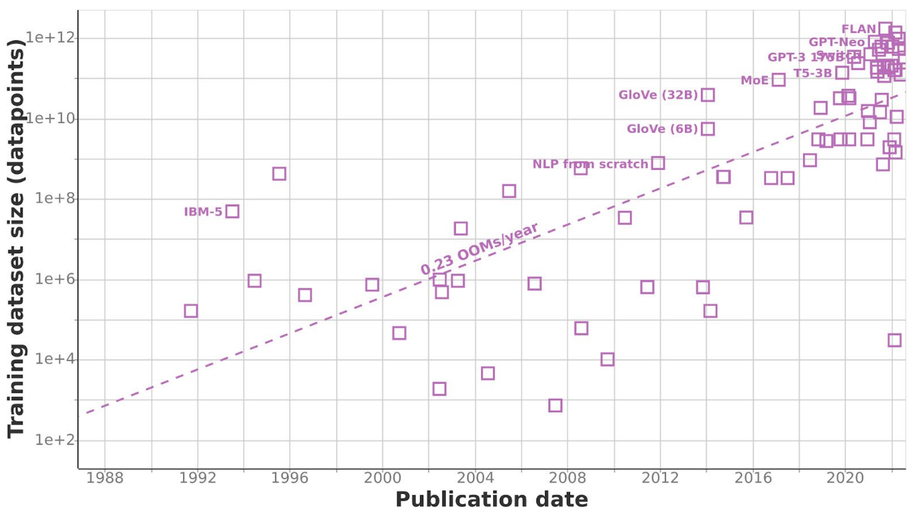
- **Chapter 1 – Introduction**
  - Motivation
  - Thesis Statement
  - Contributions and Outline
- **Neural Network Architecture**
  - Chapter 2 – "Lean Neural Networks for Autonomous Radar Waveform Design" (§1, §2,§4)
  - Chapter 2.5 – "Lean Neural Networks for Real-time Embedded Spectral Notching Waveform Design" (§2,§5)
- **Neural Network Training Data**
  - Chapter 3 – "Dataset Augmentation for Robust Spiking Neural Networks" (§2,§3,§4)
  - Chapter 4 – "Generative Data for Neuromorphic Computing" (§2,§5)
  - Chapter 5 – "Dataset Assembly for Training Spiking Neural Networks" (§2,§3,§4)
- **Neural Network Initialization**
  - Chapter 6 – "Generative Samples for Smooth Weight Transitioning to Spiking Neural Networks" (§3,§5)
- **Conclusions & Future Work Opportunities**


Share of respondents who say their organizations have adopted AI in at least one function, 2017–23
Source: McKinsey & Company Survey, 2023 | Chart: 2024 AI Index report
% of respondents
0%
10%
20%
30%
40%
50%
60%
55%
2017
2018
2019
2020
2021
2022
2023

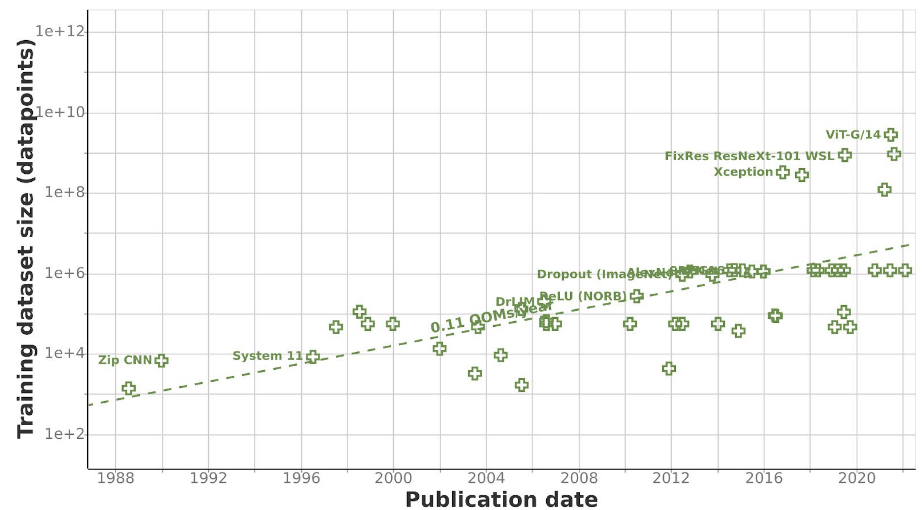
[1]


Number of Model Parameters
Year
2012
2013
2014
2015
2016
2017
2018
2019
2020
AlexNet
VGG16
GoogLeNet
DQN
Inception V3
ResNet-50
Xception
Transformer (Big)
Transformer (Base)
SENet
NASNet
BERT
GPT-2
ALBERT
Transformer-XL
GPT-3

[2]


Training dataset size (datapoints)
Publication date
1e+2
1e+4
1e+6
1e+8
1e+10
1e+12
1988
1992
1996
2000
2004
2008
2012
2016
2020
0.23 OOMs/year
IBM-5
NLP from scratch
GloVe (6B)
GloVe (32B)
MoE
T5-3B
GPT-Neo
FLAN

[3]


Training dataset size (datapoints)
Publication date
1e+2
1e+4
1e+6
1e+8
1e+10
1e+12
1988
1992
1996
2000
2004
2008
2012
2016
2020
0.11 OOMs/year
Zip CNN
System 11
ReLU (NORB)
ViT-G/14
FixRes ResNeXt-101 WSL
Xception

[3]

Successful neural network solutions require leveraging existing knowledge found within extant solutions to provide hardware flexibility, unerring resiliency, and superior performance with minimal regression. Specifically, attentive design, implementation, and execution must be a part of neural network development, including:

§1. Incorporation of existing problem-specific information into neural network architecture choices.

§2. Emphasis on maintaining low SWaP (size, weight, and power) solutions without sacrificing performance.

§3. Supplying self-correcting abilities via augmentative training data.

§4. Providing equivalent robustness to tried and true existing solutions.

§5. Insuring flexibility for deployment to the latest hardware including neuromorphic processors.

Chapters 3, 4, 5:
Improving neural network training data through generative augmentation

Chapter 2:
Improving neural network building using domain-specific information

Chapter 6:
Improving neural network initialization for changes in hardware

Algorithm Neural Network Development

```
/* Use sensors to collect dataset based on some objective */
procedure COLLECTDATA(sensors, criterion)
    dataset ← []                                    // Begin with empty dataset
    while !criterion do                             // Continue until some stopping criterion
        dataset += CAPTURE(sensors)                 // Append newly collected sample from sensors to dataset
    end while
    return dataset
end procedure

/* Given a list of options for each hyperparemter (# of layers, width of each layer, activation function per layer, etc.), select an option and apply it to NN */
procedure BUILDNEURALNETWORK(hyperparameterOptions)
    NN ← {}                                         // Create empty neural network
    for hyperparameterOptions in hyperparametersOptions do   // Repeat for all available hyperparameters
        choose hyperparameter ∈ hyperparameterOptions        // Select hyperparameter from options based on some criterion
        APPLY(NN, hyperparameter)
    end for
    return NN
end procedure

/* Given a strucutre of a neural network, initialize each layer's weights and overall optimizer */
procedure INITIALIZENEURALNETWORK(NN)
    choose optimizer ∈ [SGD, RMSprop, Adagrad, Adam, ...]   // Also select learning rate and optional momentum
    for layer in NN do
        INIT(layer)                                 // INIT ∈ [zeros, ones, uniform, normal, Xavier, Kaiming, ...]
    end for
    return NN
end procedure

/* Given an initialized neural network and a dataset, train the neural network */
procedure TRAINNEURALNETWORK(NN, dataset)
    ...
end procedure

procedure MAIN
    dataset ← COLLECTDATA(...)
    NN ← BUILDNEURALNETWORK(...)
    INITIALIZENEURALNETWORK(NN)
    TRAINNEURALNETWORK(NN, dataset)
end procedure
```

Previous Related Activity

- **Neural Network Architecture**
  - **Low-SWaP** [7, 8, 9, 10, 11]
  - **End-to-End Pipeline** [4, 5, 6]
  - **Hardware Portability** [6, 7, 8, 9, 10, 11]
  - **Intelligent Design** [4, 5, 6, 7]
- **Neural Network Training Data**
  - **Generative Spiking** [14, 15, 16]
  - **SNN Robustness** [16, 17, 18, 19]
  - **Spike Viewpoint Encoding** [16, 19]
  - **Adversarial Attacks** [17, 18]
- **Neural Network Initialization**
  - **Activation Substitution** [20, 22, 23, 24, 25]
  - **Post Correction** [23, 24]
  - **Arbitrary Architecture** [20, 22, 25]
- **Signal Processing**
  - **Iterative process: ERA** [4]
  - **Convex Optimization: RUWO** [5]

This Research

- **Neural Network Architecture**
  - **Low-SWaP** **[AB1] [AB2]**
  - **End-to-End Pipeline** **[AB1] [AB2]**
  - **Hardware Portability** **[AB1] [AB2]**
  - **Intelligent Design** **[AB1] [AB2]**
- **Neural Network Training Data**
  - **Generative Spiking** **[AB3] [AB4] [AB5]**
  - **SNN Robustness** **[AB3] [AB5]**
  - **Spike Viewpoint Encoding** **[AB3] [AB5]**
- **Neural Network Initialization**
  - **Activation Substitution** **[AB6]**
  - **Post Correction** **[AB6]**
  - **Arbitrary Architecture** **[AB6]**
- **Signal Processing**
  - **Lean ANNs** **[AB1] [AB2] [AB7] [AB8]**

[AB1] "Lean Neural Networks for Autonomous Radar Waveform Design" Sensors 2022

[AB2] "Lean Neural Networks for Real-time Embedded Spectral Nothing Waveform Design" IEEE ISIE 2022

[AB3] "Dataset Augmentation for Robust Spiking Neural Networks" IEEE ACSOS 2023

[AB4] "Generative Data for Neuromorphic Computing" HICSS 2025

[AB5] "Dataset Assembly for Training Spiking Neural Networks" Neurocomputing. *In review*

[AB6] "Generative Samples for Smooth Weight Transitioning to Spiking" *In preparation*

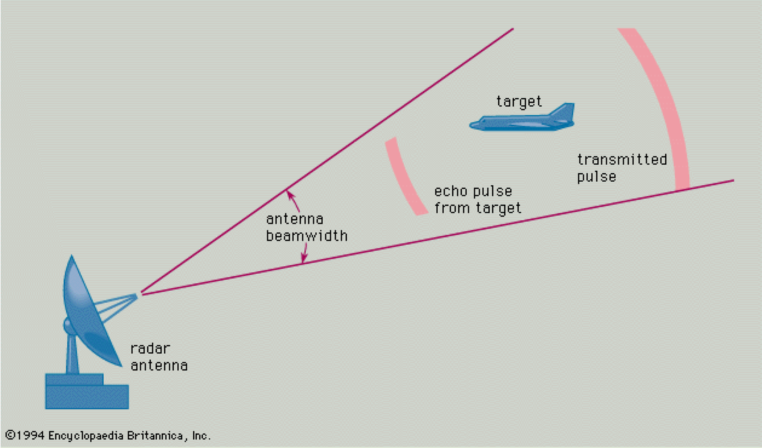
[AB7] "Method of Analyzing and Correcting a Dynamic Waveform Using Multivariate Error Loss Functions" IP 18/418,576

[AB8] "Method of Analyzing and Correcting a Dynamic Waveform by Real and Imaginary Partitioning and Recombination" IP 18/418,585

- **Chapter 1 – Introduction**
  - Motivation
  - Thesis Statement
  - Contributions and Outline
- **Neural Network Architecture**
  - Chapter 2 – "Lean Neural Networks for Autonomous Radar Waveform Design" (§1, §2,§4)
  - Chapter 2.5 – "Lean Neural Networks for Real-time Embedded Spectral Notching Waveform Design" (§2,§5)
- **Neural Network Training Data**
  - Chapter 3 – "Dataset Augmentation for Robust Spiking Neural Networks" (§2,§3,§4)
  - Chapter 4 – "Generative Data for Neuromorphic Computing" (§2,§5)
  - Chapter 5 – "Dataset Assembly for Training Spiking Neural Networks" (§2,§3,§4)
- **Neural Network Initialization**
  - Chapter 6 – "Generative Samples for Smooth Weight Transitioning to Spiking Neural Networks" (§3,§5)
- **Conclusions & Future Work Opportunities**

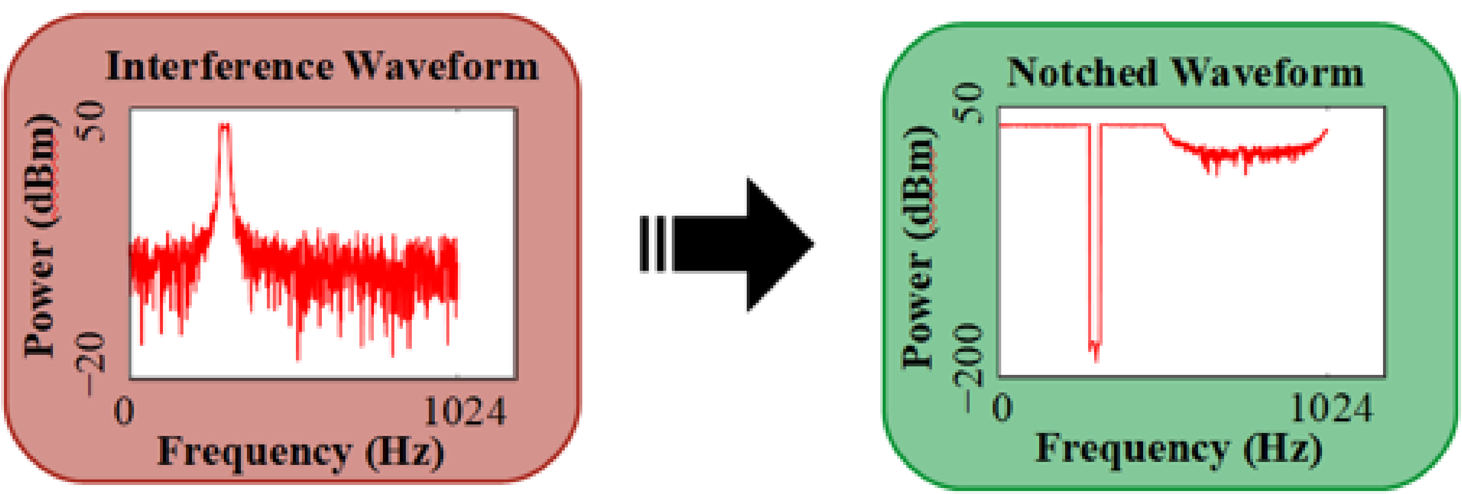
Increased Wireless Spectrum Interference

- 4G/5G telecommunication networks
- Mobile sensors
- IoT devices

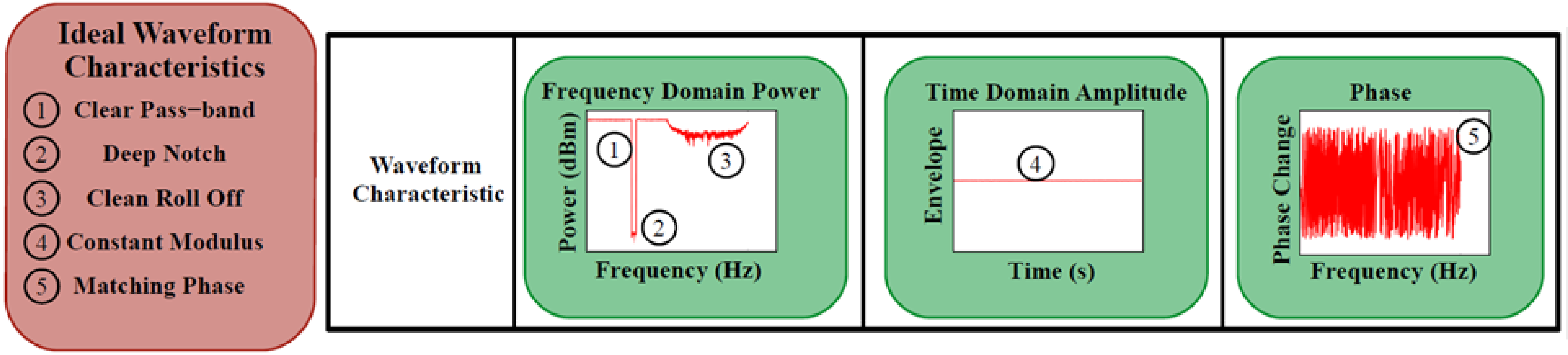


Interference Mitigation with Spectral Notching

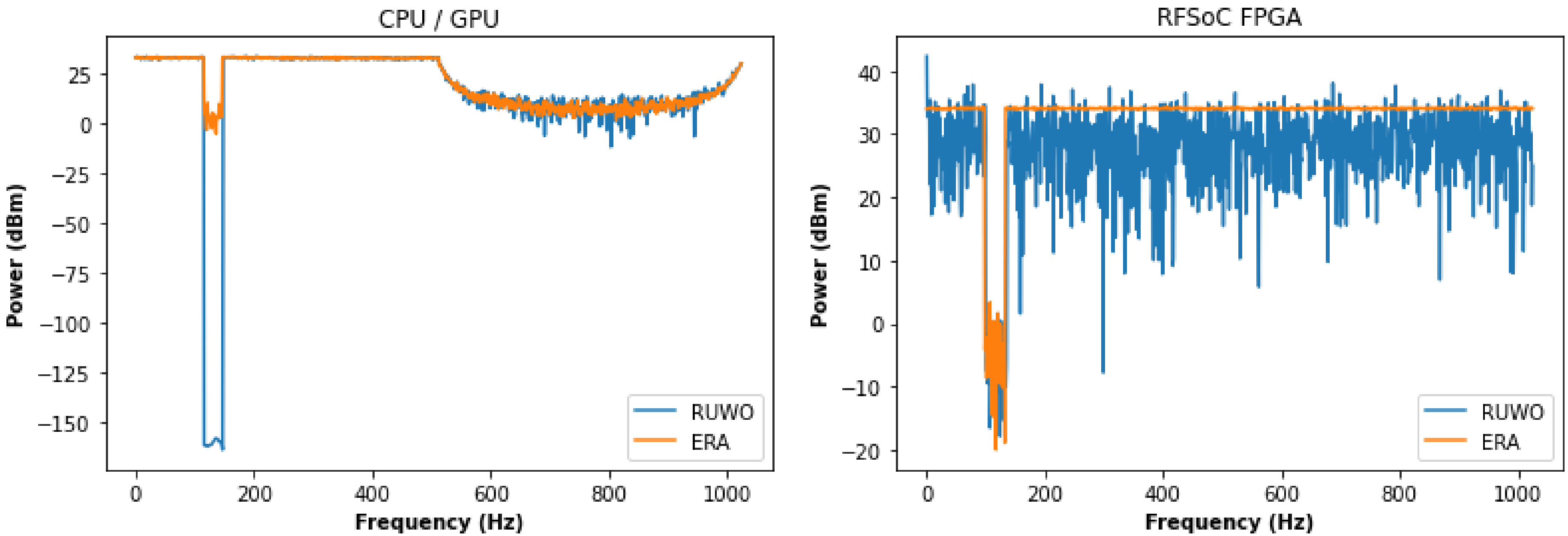
- Sample RF environment
- Determine interfered stopband
- Modify transmit waveform to avoid stopband



- Difficult task with multiple constraints that must be met for radar functionality and power efficiency
- Trade-off between runtime/power and precision
  - Want near real-time without sacrificing performance



- Solutions must also be portable to different hardware
  - RFSoC FPGA (Radio Frequency System on Chip Field-Programmable Gate Array) has fixed-point representation limit for example



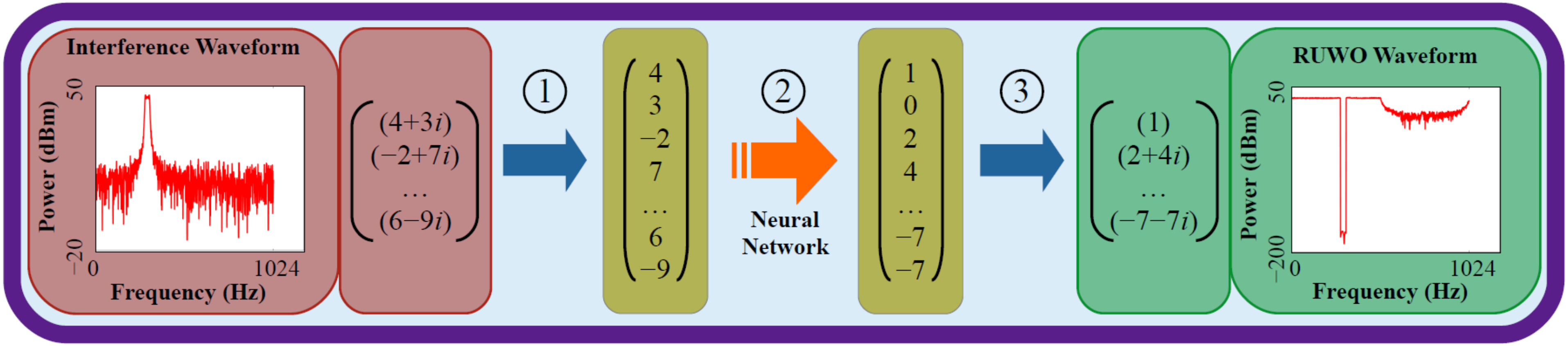
Work	Low SWaP	End-to-End	Hardware Portability	Intelligent Design
Error Reduction Algorithm (ERA) [4]		✓		✓
Re-Iterative Uniform Weight Optimization Algorithm (RUWO) [5]		✓		✓
MIMO GPU [6]		✓	✓	✓
TCNRWR [7]	✓		✓	✓
RVTDCNN [8]	✓		✓	
Autowave pre-computed [9, 10, 11]	✓		✓	
My work [Chapter 2, Chapter 2.5]	✓	✓	✓	✓

- AutoWave → Artificial Intelligence (AI) implementation of an adaptive radar system which uses neural networks to adjust transmitted waveforms to avoid sources of interference
  - Treat RUWO as absolute, train NN to learn RUWO
- Naive assumption: increasing neural network size will result in better performance

Algorithm	GPU Latency (μs)	CPU Latency (μs)	Cosine Similarity	Null Depth (dBm)
NN MSE 1 Layer	747.71 ± 5.23	786.44 ± 5.01	$0.9901 \pm 7.69 \times 10^{-5}$	28.54 ± 0.16
NN MSE 2 Layers	749.40 ± 5.61	797.92 ± 5.99	$0.9900 \pm 7.89 \times 10^{-5}$	29.17 ± 0.22
NN MSE 3 Layers	797.72 ± 10.03	855.34 ± 6.38	$0.9898 \pm 9.87 \times 10^{-5}$	26.57 ± 0.23

Moving away from Mean Squared Error (MSE)

- Numerical comparisons between coefficient vectors prone to errors

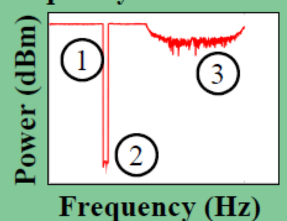
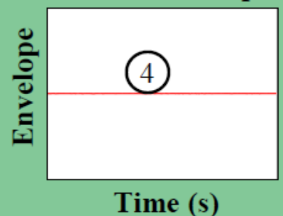
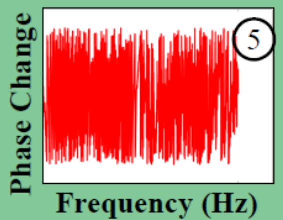


Tailor loss function to radar waveform design

1. Provide quicker learning to <u>valid</u> solutions compared to MSE
2. Discourage “close enough” waveforms which are similar but not <u>valid</u>
3. Encourage neural network to always produce <u>valid</u> waveforms (even if not identical to RUWO)

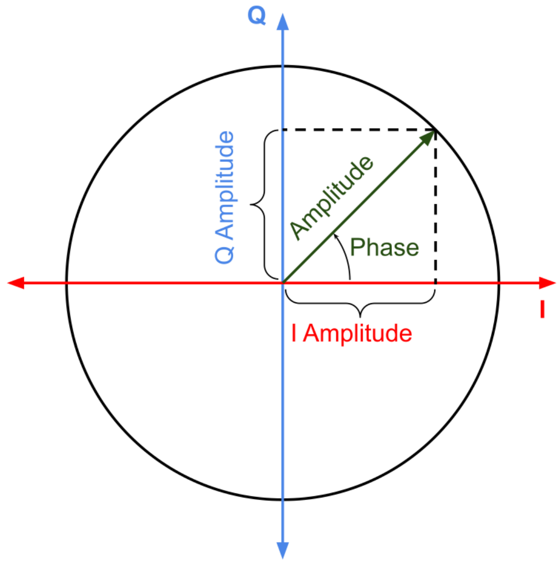
Ideal Waveform Characteristics

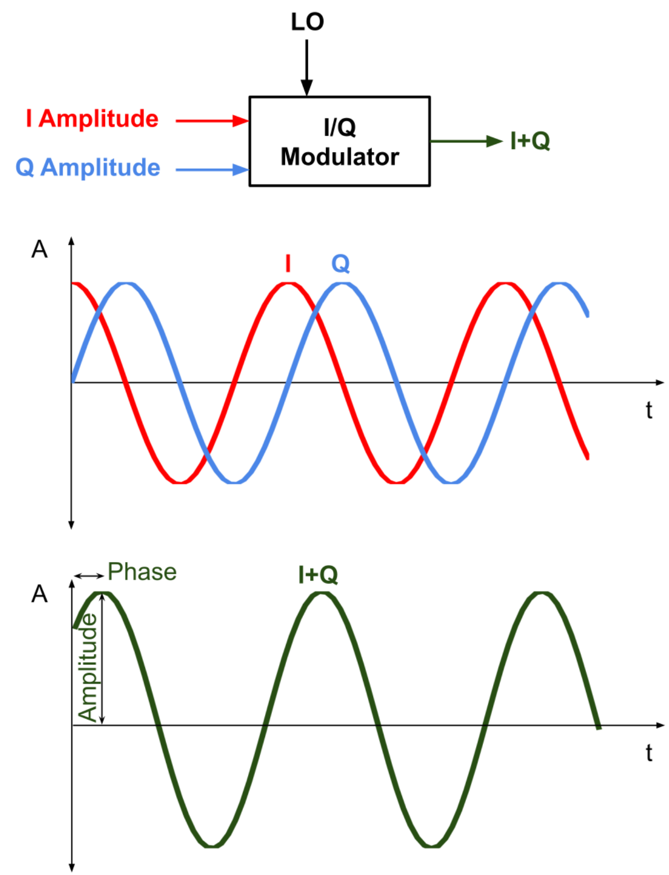
1. Clear Pass−band
2. Deep Notch
3. Clean Roll Off
4. Constant Modulus
5. Matching Phase

Waveform Characteristic	**Frequency Domain Power** 	**Time Domain Amplitude** 	**Phase** 
Reflection in Loss Function	$\frac{1}{n}\sum_{i=1}^{n}(20*\log_{10}(FFT(\lvert z_i\rvert)) - 20*\log_{10}(FFT(\lvert\hat{z}_i\rvert)))^2$	$\frac{1}{n}\sum_{i=1}^{n}(IFFT(\lvert z_i\rvert) - IFFT(\lvert\hat{z}_i\rvert))^2$	$\frac{1}{n}\sum_{i=1}^{n}(\angle FFT(\lvert z_i\rvert) - \angle FFT(\lvert\hat{z}_i\rvert))^2$

Split processing into 2 parallel neural networks

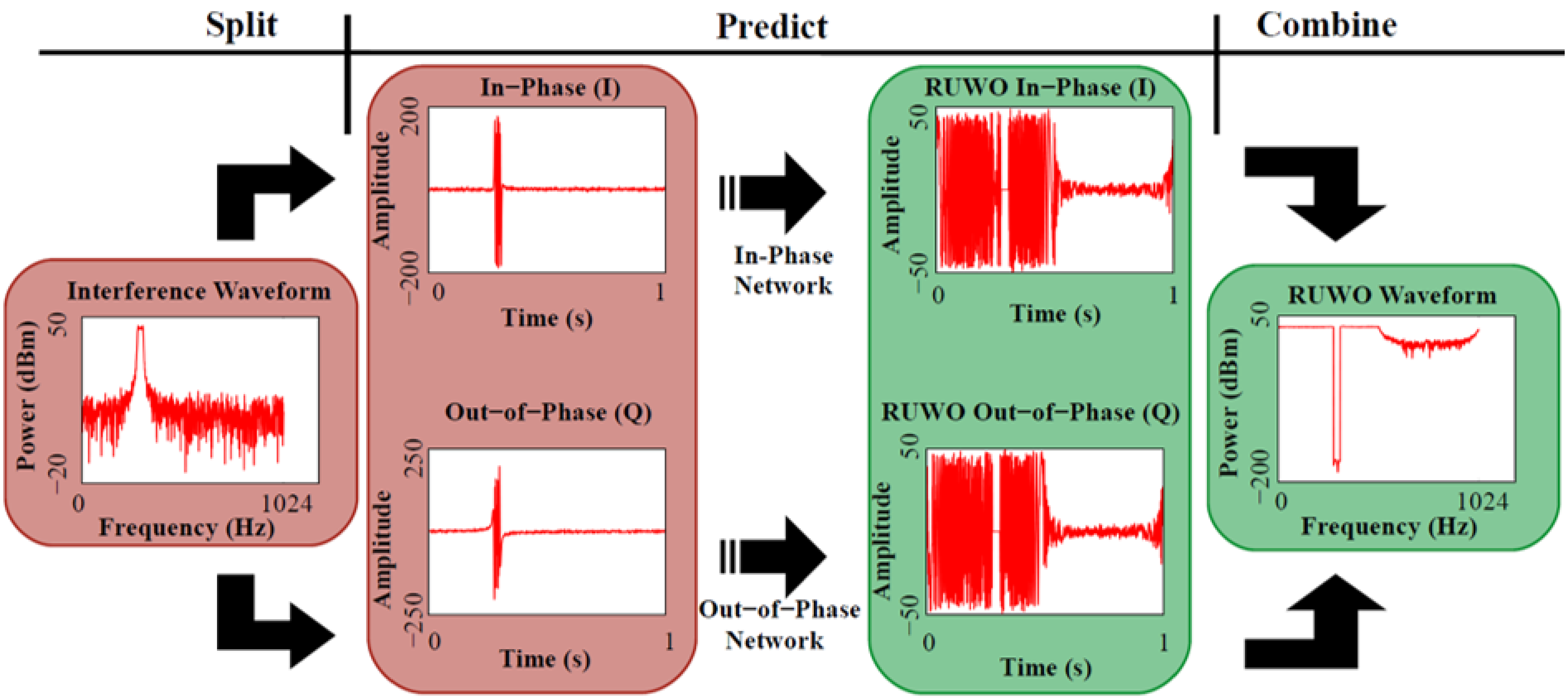
- Quadrature radar waveforms are <u>separate</u>





I Amplitude (V)	Q Amplitude (V)	I+Q	
		Amplitude (V)	Phase (°)
1	0	1	0
0	1	1	90
-1	0	1	180
0	-1	1	270

Split processing into 2 parallel neural networks



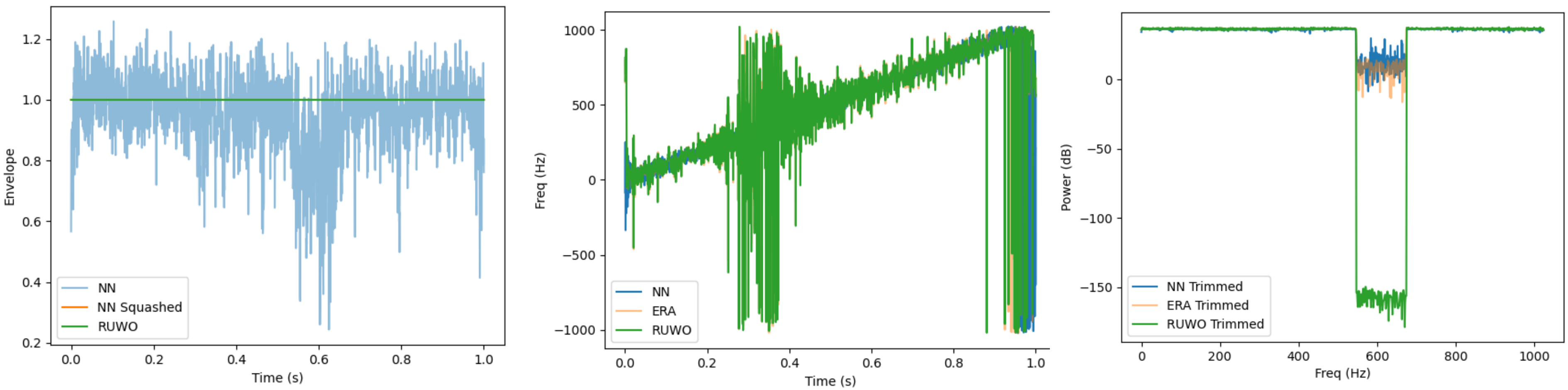
CPU / GPU Simulation

Algorithm	Cosine Similarity	Null Depth (dBm)
RUWO	1.0 ± 0.0	202.23 ± 0.0
ERA	0.9982 ± 0.0	31.89 ± 0.0
NN MSE	$0.9901 \pm 7.69 \times 10^{-5}$	28.54 ± 0.16
NN Custom Loss	$0.9789 \pm 9.53 \times 10^{-5}$	22.32 ± 0.13
NN Split	$0.9900 \pm 1.08 \times 10^{-4}$	29.75 ± 0.12

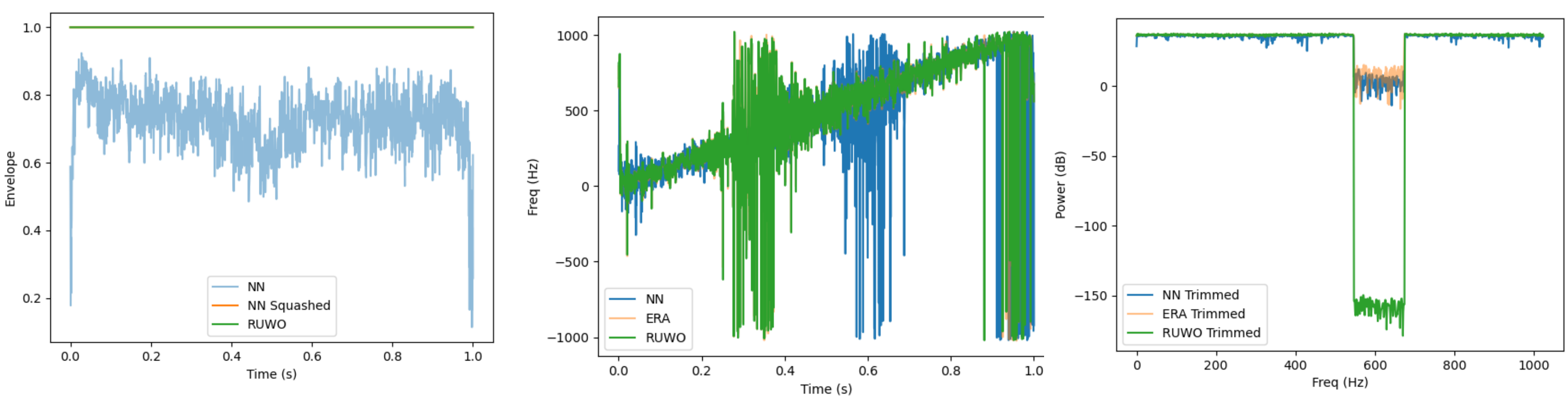
RFSoC FPGA Open-Air Trials

Algorithm	Null Depth (dBm)
RUWO	33.62 ± 0.041
ERA	37.13 ± 0.057
NN MSE	28.17 ± 0.213
NN Custom Loss	21.22 ± 0.138
NN Split	28.93 ± 0.285

Custom Loss



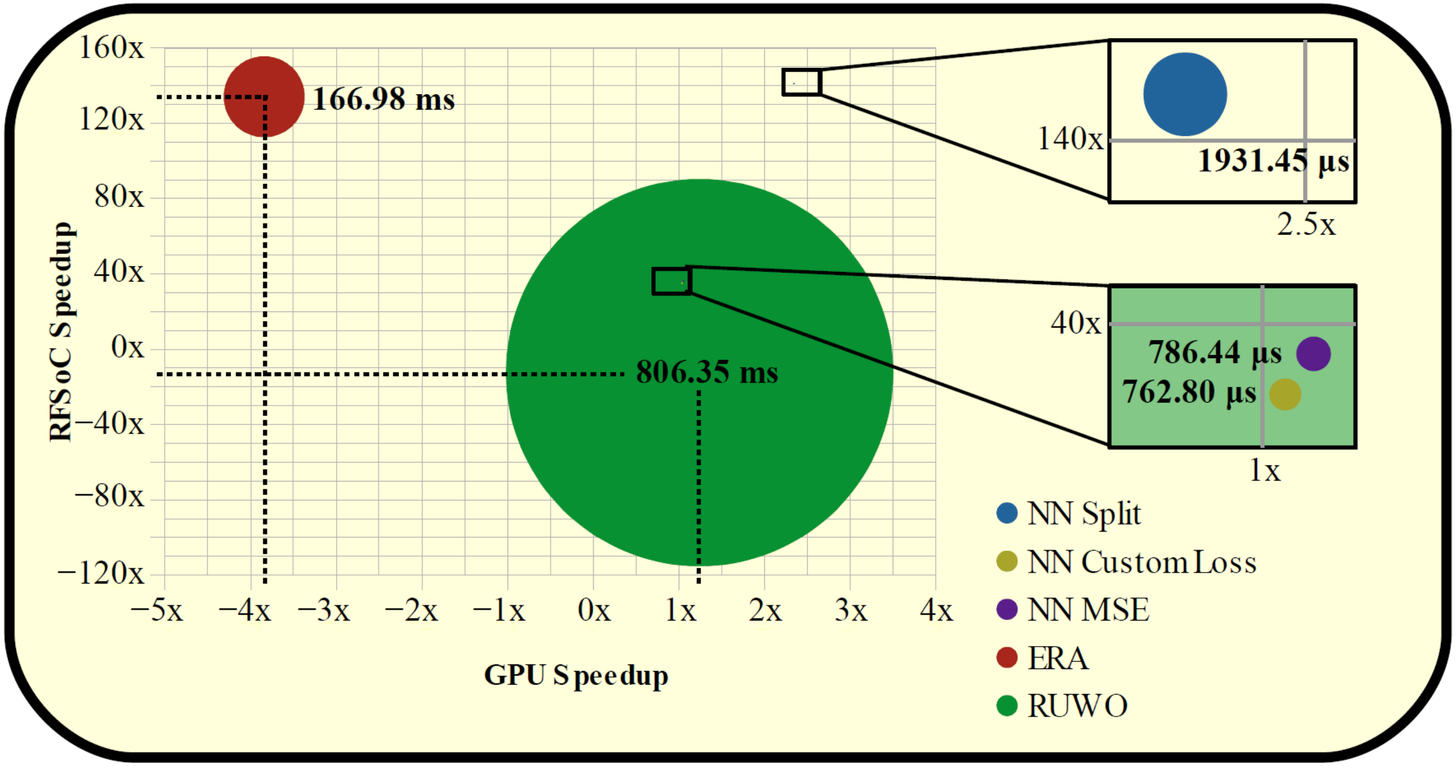
MSE



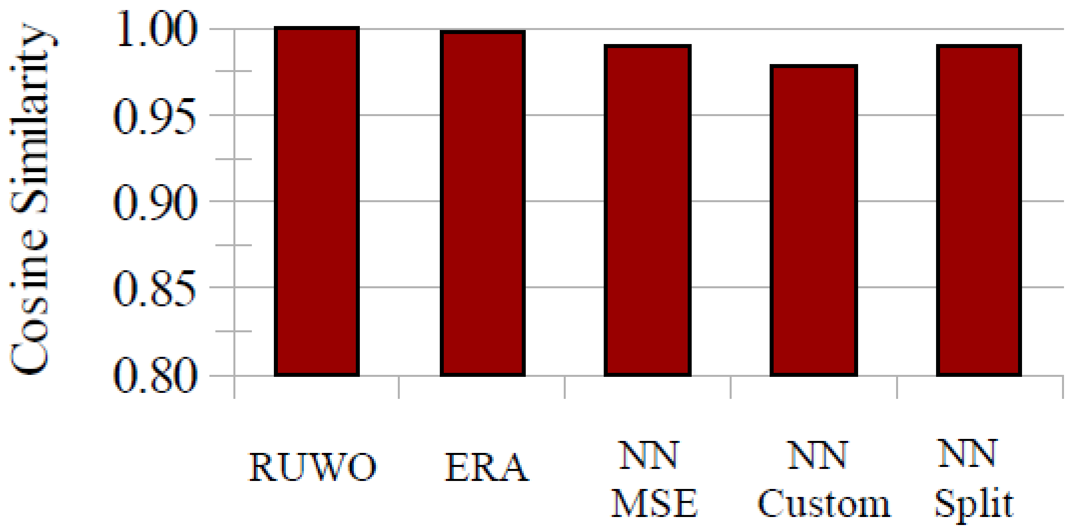
Neural Network Architecture

Chapter 2 – "Lean Neural Networks for Autonomous Radar Waveform Design" (§1,§2,§4)

Algorithm	CPU Latency (µs)	GPU Latency (µs)	RFSoC Latency (µs)
RUWO	806,347.0 ± 11,860.82	649,581.0 ± 33,168.78	10,060,000.0 ± 999.0
ERA	166,982.0 ± 3465.06	641,441.0 ± 20,921.13	1246.0 ± 8.8
NN MSE	786.4 ± 5.01	747.71 ± 5.23	21.7 ± 0.0
NN Custom Loss	762.8 ± 5.04	735.68 ± 7.99	21.7 ± 0.0
NN Split	1931.5 ± 9.75	823.63 ± 6.76	13.7 ± 0.0



- Specifically target low power embedded devices (Raspberry Pi 3B)



Null Depth (dBm)	202.2	31.9	28.5	22.3	29.8

Algorithm	Dell r720 2x Intel E5-2670, NVIDIA GT 1030, 144GB RAM		Raspberry Pi 3B Broadcom BCM2837, 1GB RAM	
	Latency (ms)	Energy (J)	Latency (ms)	Energy (J)
RUWO	1064.98 ± 10.94	261.3 ± 6.5	453,965.43 ± 4131.61	1510.5 ± 14.8
ERA	185.47 ± 3.87	45.5 ± 1.4	1982.04 ± 29.27	6.5 ± 0.1
NN MSE	23.19 ± 1.86	**3.7 ± 0.3**	**230.98 ± 2.74**	**0.6 ± 0.01**
NN Tailored Loss Function	**20.72 ± 0.44**	**3.7 ± 0.1**	233.92 ± 3.16	**0.6 ± 0.01**
NN Tailored Network Architecture	23.35 ± 0.29	4.1 ± 0.6	250.90 ± 0.63	0.7 ± 0.01

- **Chapter 1 – Introduction**
  - Motivation
  - Thesis Statement
  - Contributions and Outline
- **Neural Network Architecture**
  - Chapter 2 – "Lean Neural Networks for Autonomous Radar Waveform Design" (§1, §2,§4)
  - Chapter 2.5 – "Lean Neural Networks for Real-time Embedded Spectral Notching Waveform Design" (§2,§5)
- **Neural Network Training Data**
  - Chapter 3 – "Dataset Augmentation for Robust Spiking Neural Networks" (§2,§3,§4)
  - Chapter 4 – "Generative Data for Neuromorphic Computing" (§2,§5)
  - Chapter 5 – "Dataset Assembly for Training Spiking Neural Networks" (§2,§3,§4)
- **Neural Network Initialization**
  - Chapter 6 – "Generative Samples for Smooth Weight Transitioning to Spiking Neural Networks" (§3,§5)
- **Conclusions & Future Work Opportunities**

Spiking Neural Networks (SNNs)

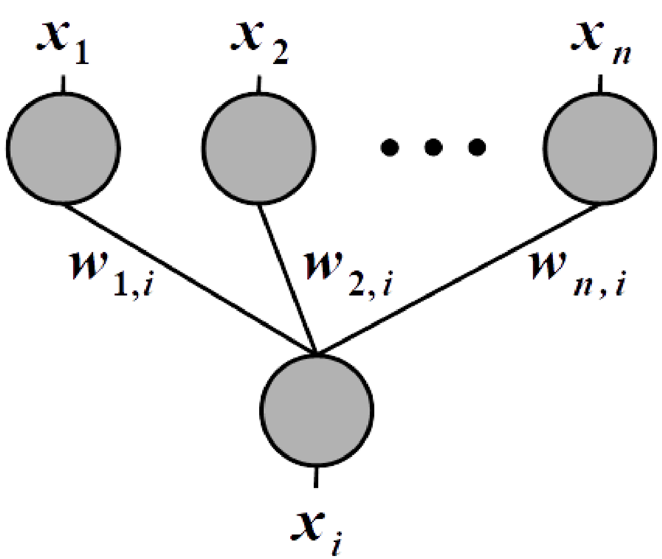
- Biologically inspired 3rd generation neural networks
- Neurons communicate via discrete pulses over time
- Great for time-series data
- SNN processing consumes less power when realized on neuromorphic hardware such as Intel Loihi [12]

https://www.intel.com/content/www/us/en/newsroom/news/intel-unveils-neuromorphic-loihi-2-lava-software.html#gs.4ve63w

ANNs vs SNNs

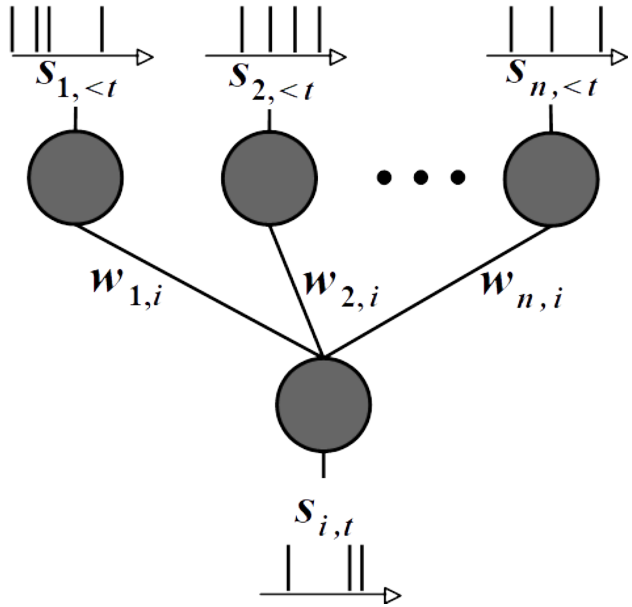
ANNs

- Operate on continuous values $x_1, x_2, \ldots, x_n$
- Information propagates instantaneously



SNNs

- Operate on discrete spike trains $S_1, S_2, \ldots, S_n$
- Must be run over a period of time



ANNs vs SNNs

ANNs

$$E = L(y, \hat{y})$$

$$net_j = \sum_i w_{ij} x_i + b$$

$$o_j = \varphi(net_j)$$

$$\delta_j = \begin{cases} \frac{\partial L(y, o_j)}{\partial o_j} \frac{d\varphi(net_j)}{dnet_j} & j \text{ output} \\ \left(\sum_k w_{jk}\delta_k\right) \frac{d\varphi(net_j)}{dnet_j} & j \text{ hidden} \end{cases}$$

$$\Delta w_{ij} = -\eta o_j \delta_j$$

SNNs

Input $s_i(t) = \sum_f \delta\left(t - t_i^{(f)}\right)$

$$a_i(t) = (\epsilon * s_i)(t) \quad \epsilon(t) = \frac{t}{\tau_s} exp\left(1 - \frac{t}{\tau_s}\right)\Theta(t)$$

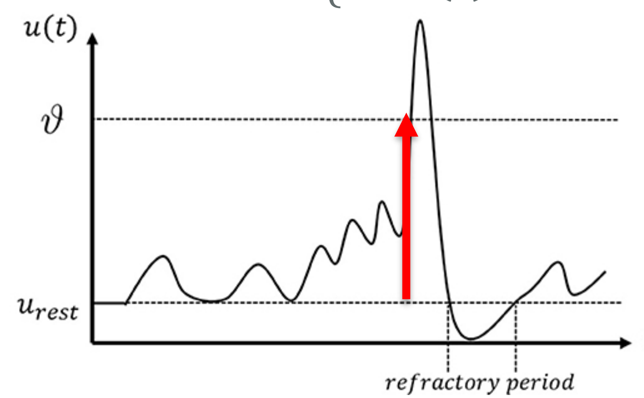
$$v_i(t) = (v * s)(t) \quad \mathrm{v}(t) = -2\vartheta exp\left(1 - \frac{t}{\tau_r}\right)\Theta(t)$$

$$u(t) = \sum_i w_i a_i(t) + v_i(t)$$

$$f_s(u) : u \to s$$

$$s(t) \coloneqq s(t) + \delta\left(t - t^{(f+1)}\right)$$

$$t^{(f+1)} = min\{t : u(t) = \vartheta, t > t^{(f)}\}$$



SNN Data

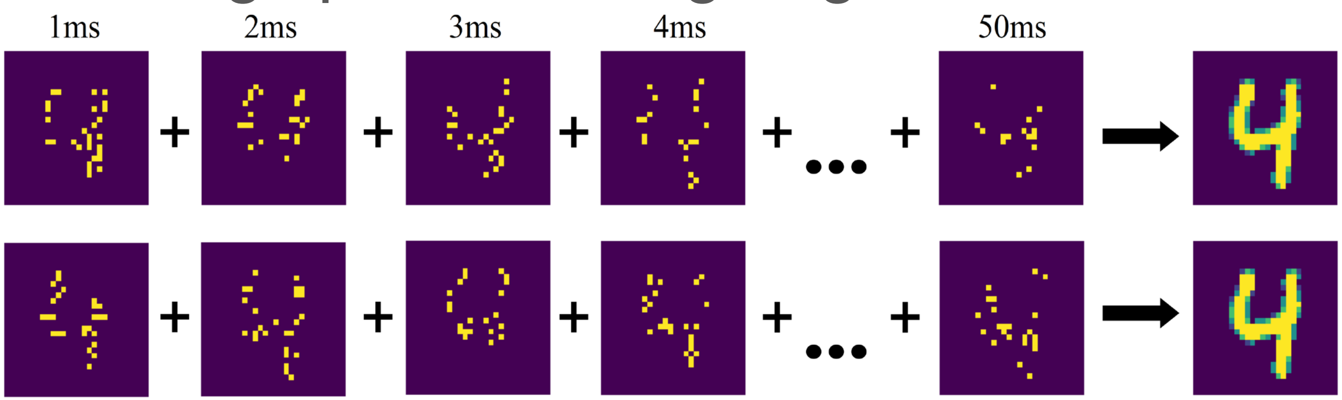
- SNNs operate on discrete spike trains
- Can be either generated from static data using integrate-and-fire (IF) neurons or captured directly using a Dynamic Vision Sensor (DVS) camera which produces event data:

https://inilabs.com/products/

$$[\boldsymbol{x}\ \text{coordinate}, \boldsymbol{y}\ \text{coordinate}, \boldsymbol{t}\ \text{timestep}, \boldsymbol{p}\ \text{polarity of light} - \text{intensity change}]$$

Spike Distribution Dependencies

- For a given static image, there are a copious number of valid spike trains which can be created/captured depending on IF neuron parameters, DVS camera settings, or lighting properties of the subject
- Surrogate gradient SNN training can fixate on the intervals of training spikes leading to generalization issues



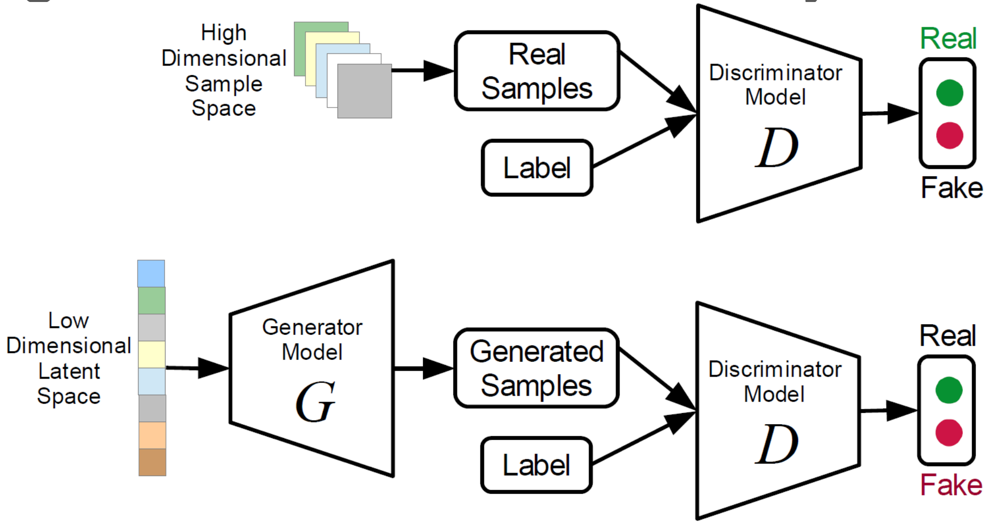
Work	Generative Spiking	SNN Robustness	Spike Viewpoint	Adversarial Attacks
Spiking-GAN [14]	✓			
SpikeGAN [15]	✓			
Deep CovDenseSNN [16]	✓	✓	✓ (encoding)	
Ozdenizci et al. [17]		✓		✓
SNN-RAT [18]		✓		✓
StepReLU [19]		✓	✓ (encoding)	
My work [Chapter 3, Chapter 4, Chapter 5]	✓	✓	✓	

My approach

- Using a spiking GAN, generate **valid** samples of **varying** spike distributions
- Augmented dataset provides additional robustness against samples different from the original training set
- Generated samples enrichen dataset without additional manual collection of data and without dataset growth for each possible spike distribution

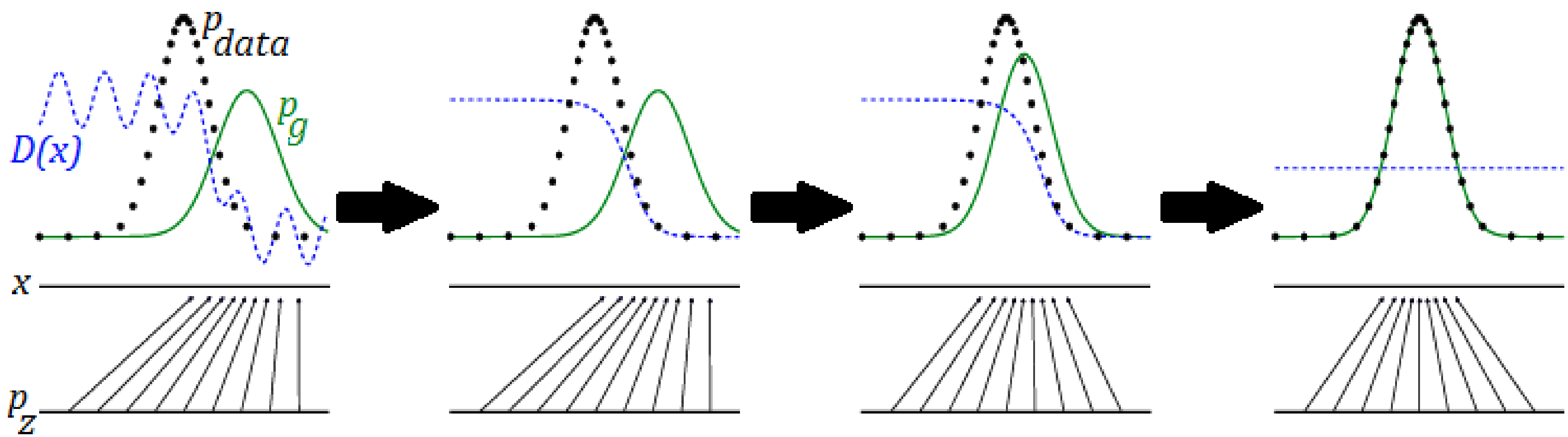
Generative Adversarial Networks (GANs)

- Adversarial learning paradigm in which a generator model G synthesizes artificial samples, and a discriminator model D classifies samples as either real or fake
- G and D "compete" against each other i.e., they are playing a minimax game to each better themselves



GAN Training

- G and D are trained simultaneously in which D identifies areas in which it can more easily identify fake samples
- These areas then become the focus for where G updates its weights
- Given sufficient capacity, G and D converge to where $p_g \approx p_{data}$ and $D(x) = 0.5$ for all input
- $\min_G \max_D V(D, G) = \mathbb{E}_{x \sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}\left[\log\left(1 - D\left(G(z)\right)\right)\right]$

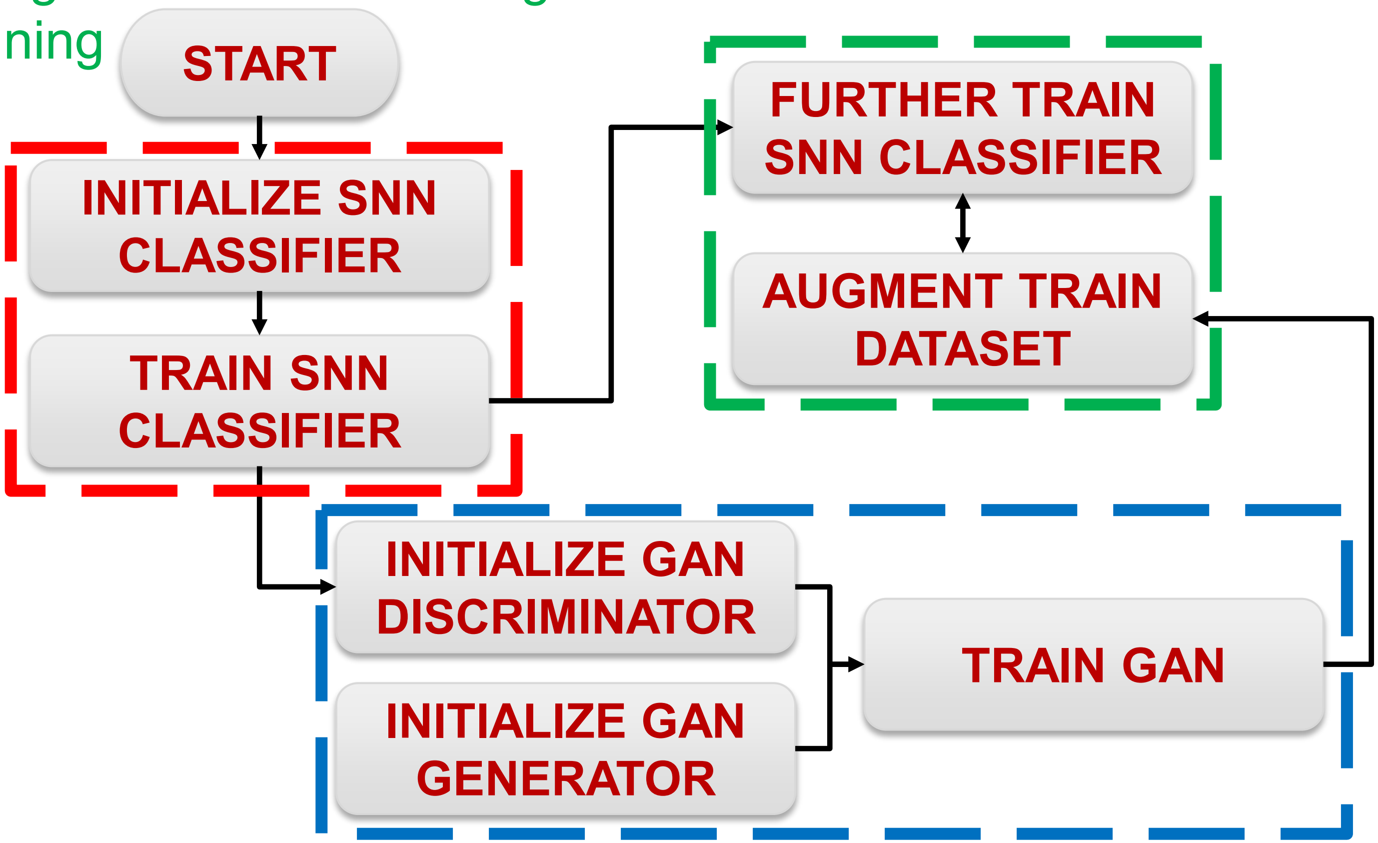


[13]

(1) SNN classifier trained to convergence

(2) GAN trained using classifier weights to seed discriminator

(3) Trained GAN generator used to augment train dataset for further classifier training



My approach (cont'd)

- During augmentation, samples are generated on an as-needed basis determined by the relative class performances
- Difficulty of correct classification is not uniform across all classes of data → disproportionate number of samples can achieve same overall accuracy

My approach (cont'd)

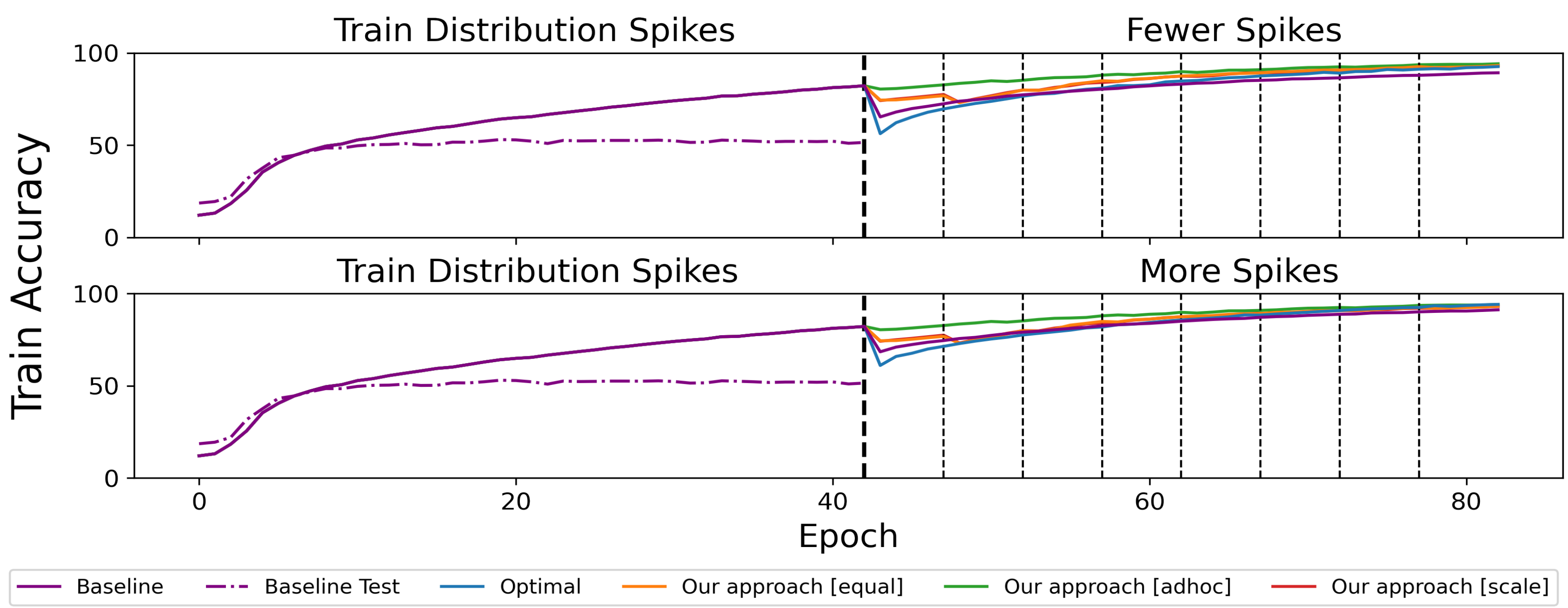
- Three schemes used to determine the number of additional samples needed for the next iteration:
  1) **equal**: same number of samples across all classes
  2) **adhoc**: only samples from the 3 worst performing classes added
  3) **scale**: number of samples added correlated to relative performance of each class

Setup

- CIFAR-10 training spike trains generated from $X \sim U(100, 200)$ firing rate distribution using LIF (leaky integrate-and-fire neuron) in Nengo [20] simulator
- Models evaluated on **fewer** spikes and **more** spikes distributions → half ($X \sim U(50, 100)$) and double ($X \sim U(200, 400)$) the number of spikes compared to training distribution

Training

- My models quickly responded to the changing spike distribution



Testing

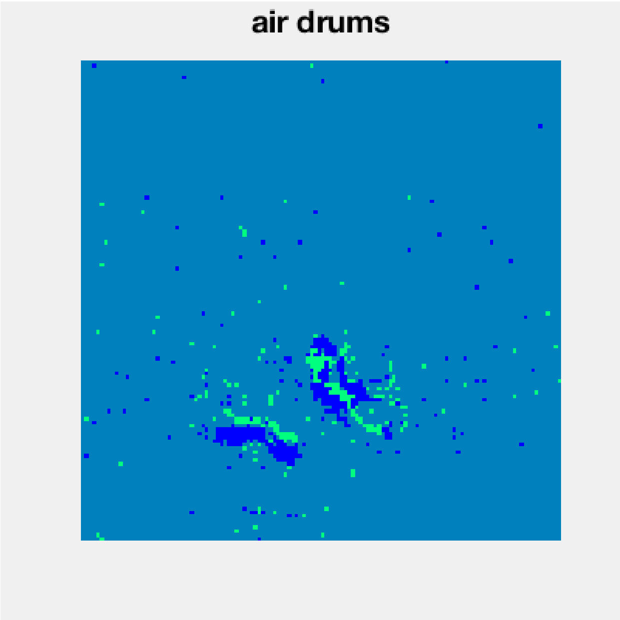
- All models perform worse as samples drifted further from the training distribution
- My models outperformed baseline classifier by an average **1.80%** and had an average **1.02%** lesser reduction in accuracy

Model	**Testing Spike Distribution** Fewer Spikes	Train Dist. Spikes	More Spikes
Baseline	37.76 ± 0.34	52.67 ± 0.31	42.73 ± 0.52
Our approach [equal]	39.09 ± 0.25	$\mathbf{54.57 \pm 0.27}$	44.07 ± 0.52
Our approach [adhoc]	$\mathbf{39.33 \pm 0.28}$	54.25 ± 0.30	$\mathbf{44.67 \pm 0.52}$
Our approach [scale]	38.52 ± 0.39	54.05 ± 0.32	43.51 ± 0.29

- **Introduction**
  - Motivation
  - Thesis Statement
  - Contributions and Outline
- **Neural Network Architecture**
  - Chapter 2 – “Lean Neural Networks for Autonomous Radar Waveform Design” (§1, §2,§4)
  - Chapter 2.5 – “Lean Neural Networks for Real-time Embedded Spectral Notching Waveform Design” (§2,§5)
- **Neural Network Training Data**
  - Chapter 3 – “Dataset Augmentation for Robust Spiking Neural Networks” (§2,§3,§4)
  - Chapter 4 – “Generative Data for Neuromorphic Computing” (§2,§5)
  - Chapter 5 – “Dataset Assembly for Training Spiking Neural Networks” (§2,§3,§4)
- **Neural Network Initialization**
  - Chapter 6 – “Generative Samples for Smooth Weight Transitioning to Spiking Neural Networks” (§3,§5)
- **Conclusions & Future Work Opportunities**

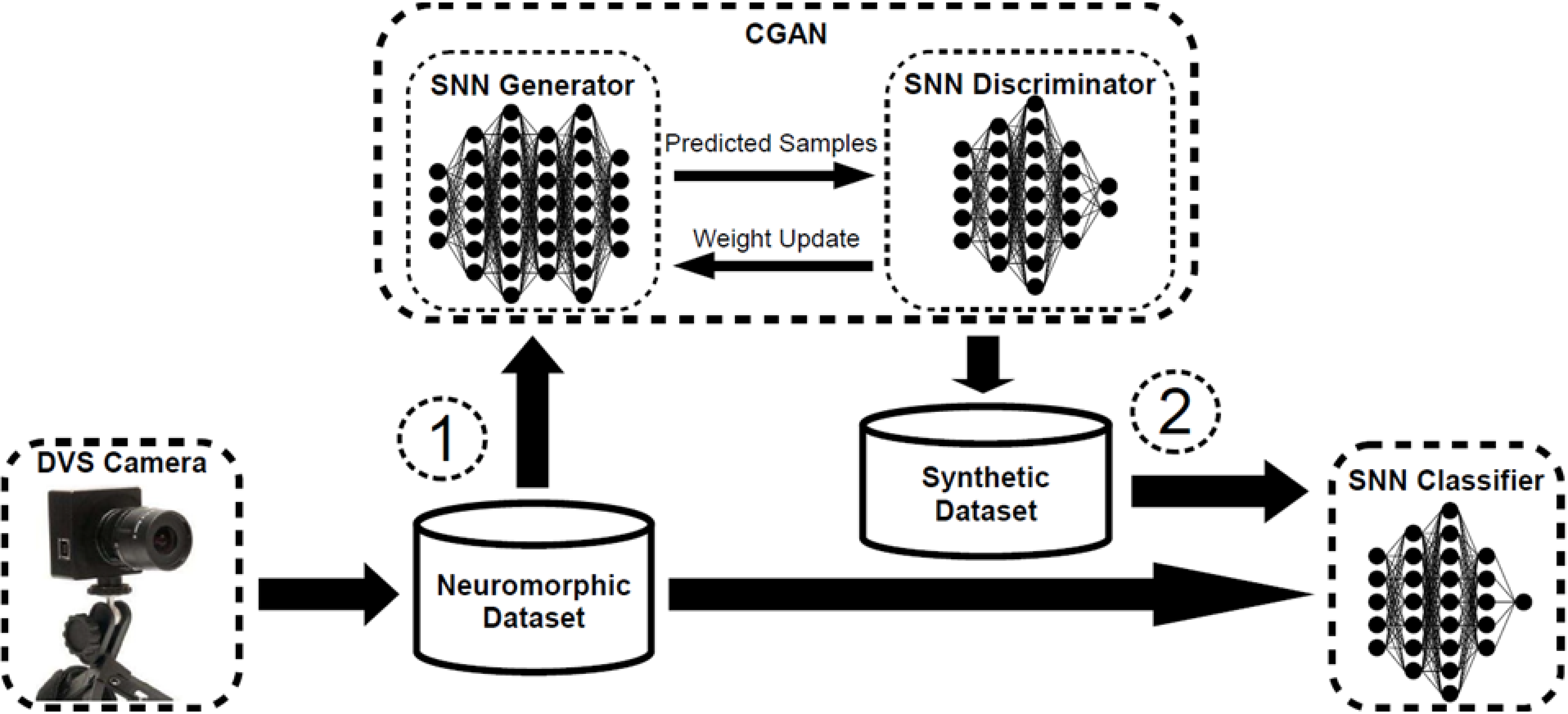
Objective

- Address shortage of neuromorphic datasets
- Apply generative augmentation to natively spiking dataset: **IBM DVSGesture**
- Unlock new SNN developments with greater access to quality spiking datasets



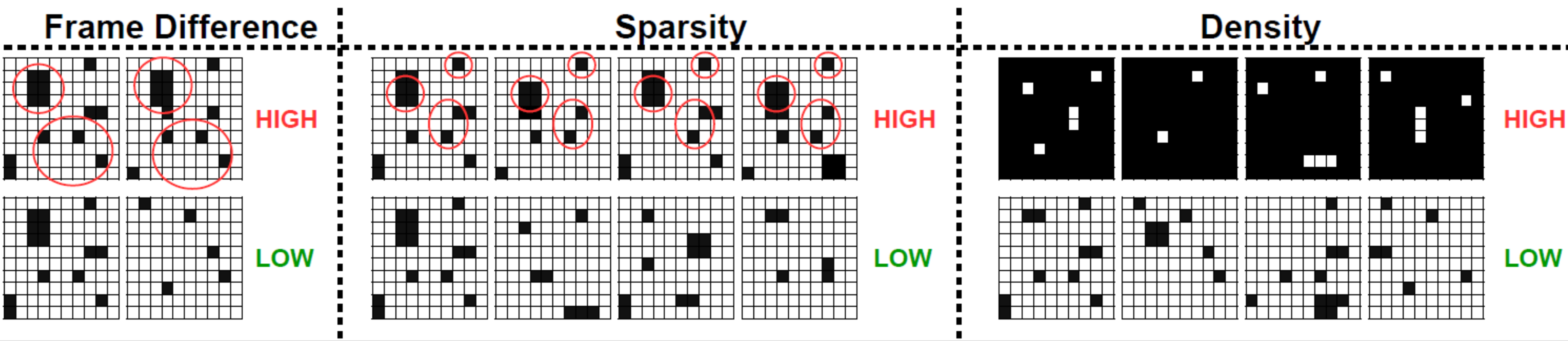
Neural Network Training Data

Chapter 4 – “Generative Data for Neuromorphic Computing” (§2,§5)



Neuromorphic Quality Metrics

- Frame Difference – cosine similarity between frames
- Sparsity – avg. number of events per pixel
- Density – avg. number of pixels firing per timestep



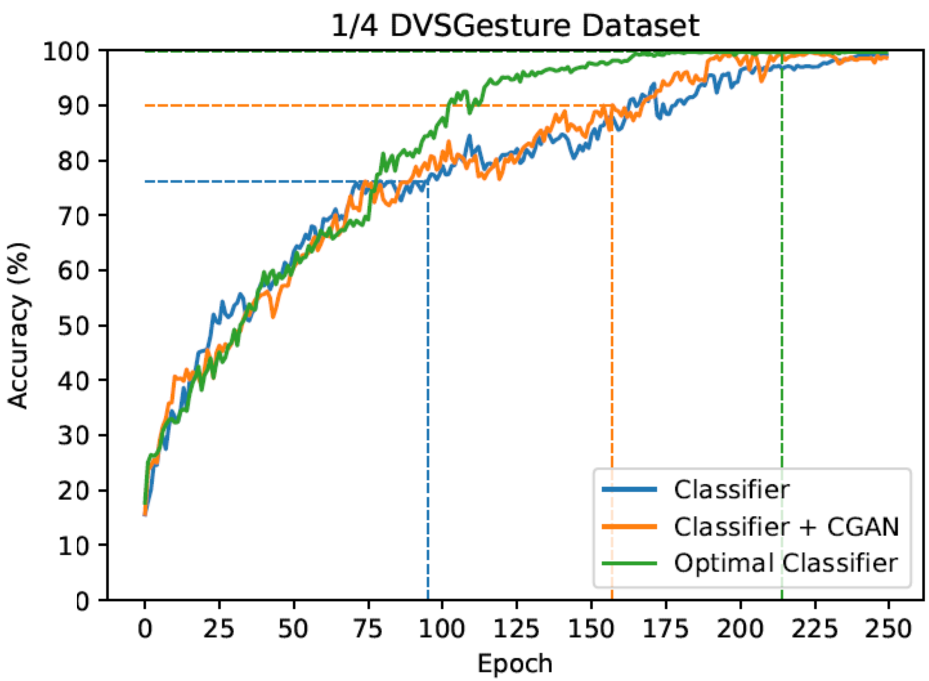
Sample Quality

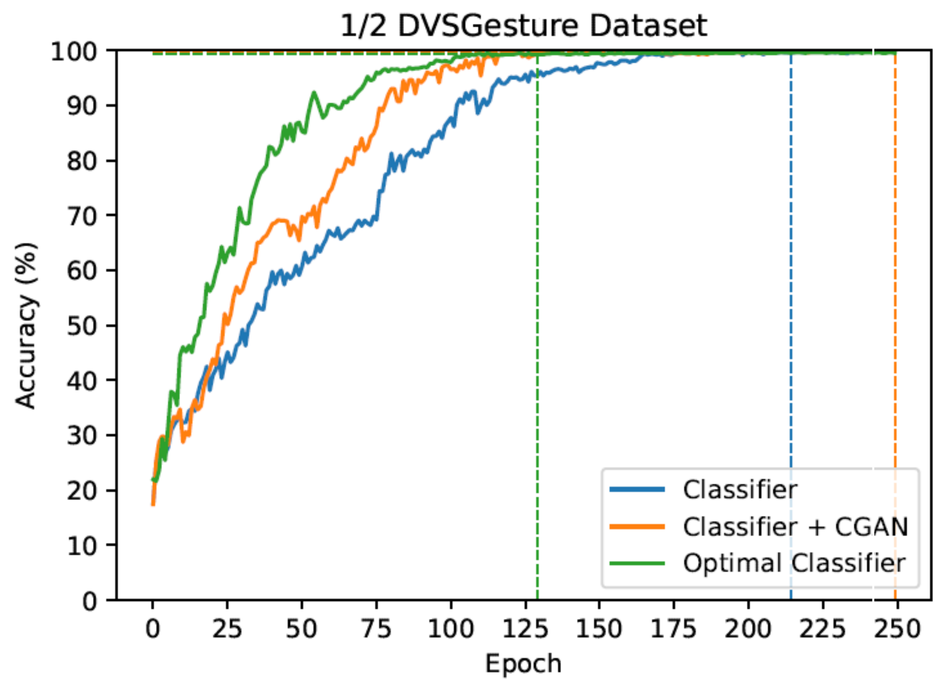
- Generative samples behave roughly similar to their real counterparts

Dataset	Frame Difference	Sparsity	Density
DVSGesture	0.07105	0.00152	0.00152
CGAN 1/4 DVSGesture	0.04033	0.00424	0.00424
CGAN 1/2 DVSGesture	0.00020	0.00030	0.00030
CGAN Entire DVSGesture	0.00010	0.00030	0.00030

Sample Behavior in Training

- Generative samples improved training performance





- **Chapter 1 – Introduction**
  - Motivation
  - Thesis Statement
  - Contributions and Outline
- **Neural Network Architecture**
  - Chapter 2 – "Lean Neural Networks for Autonomous Radar Waveform Design" (§1, §2,§4)
  - Chapter 2.5 – "Lean Neural Networks for Real-time Embedded Spectral Notching Waveform Design" (§2,§5)
- **Neural Network Training Data**
  - Chapter 3 – "Dataset Augmentation for Robust Spiking Neural Networks" (§2,§3,§4)
  - Chapter 4 – "Generative Data for Neuromorphic Computing" (§2,§5)
  - Chapter 5 – "Dataset Assembly for Training Spiking Neural Networks" (§2,§3,§4)
- **Neural Network Initialization**
  - Chapter 6 – "Generative Samples for Smooth Weight Transitioning to Spiking Neural Networks" (§3,§5)
- **Conclusions & Future Work Opportunities**

Continuation of chapter 3

- Spike Viewpoint Dependencies in natively spiking dataset: **IBM DVSGesture**
- DVSGesture contains lighting conditions:
  - Fluorescent
  - Fluorescent LED
  - LAB
  - LED
  - Natural

Lighting Condition Effects

START	TARGET FLUORESCENT	TARGET FLUORESCENT LED	TARGET LAB	TARGET LED	TARGET NATURAL
FLUORESCENT	**79.80**	73.33	72.73	78.79	72.23
FLUORESCENT LED	80.30	83.03	78.79	77.78	**86.36**
LAB	55.56	64.24	**66.67**	55.05	63.64
LED	69.70	**76.97**	74.75	76.26	76.52
NATURAL	74.24	**79.39**	72.73	78.28	78.79

Lighting	Frame Difference	Sparsity	Density
FLUORESCENT	0.069	0.0015	0.0015
FLUORESCENT LED	0.0662	0.0014	0.0014
LAB	0.1008	0.0021	0.0021
LED	0.0592	0.0015	0.0015
NATURAL	0.06	0.0016	0.0016

Optimal Transport Dataset Distance (OTDD)[21]

	FLUORESCENT	FLUORESCENT LED	LAB	LED	NATURAL
FLUORESCENT	0.03466	137,741.798	166,953.851	139,224.688	143,268.443
FLUORESCENT LED	137,741.798	0.03466	164,983.994	137,254.965	141,306.234
LAB	166,953.851	164,983.994	0.03466	166,473.620	170,445.217
LED	139,224.688	137,254.965	166,473.620	0.03466	142,756.693
NATURAL	143,268.443	141,306.234	170,445.217	142,756.693	0.03466

	FLUORESCENT	FLUORESCENT LED	LAB	LED	NATURAL
FLUORESCENT	X	1.109	1.000	0.171	1.165
FLUORESCENT LED	0.381	X	0.494	0.736	-0.453
LAB	0.976	0.216	X	1.024	0.261
LED	1.079	-0.118	0.208	X	-0.041
NATURAL	0.812	-0.109	0.909	0.091	X

Accuracy & OTDD correlation. Positive values indicate a decrease in accuracy from the starting lighting condition with an increase in OTDD while negative values indicate the opposite. A larger magnitude indicates a larger change in accuracy corresponding to a larger OTDD difference.

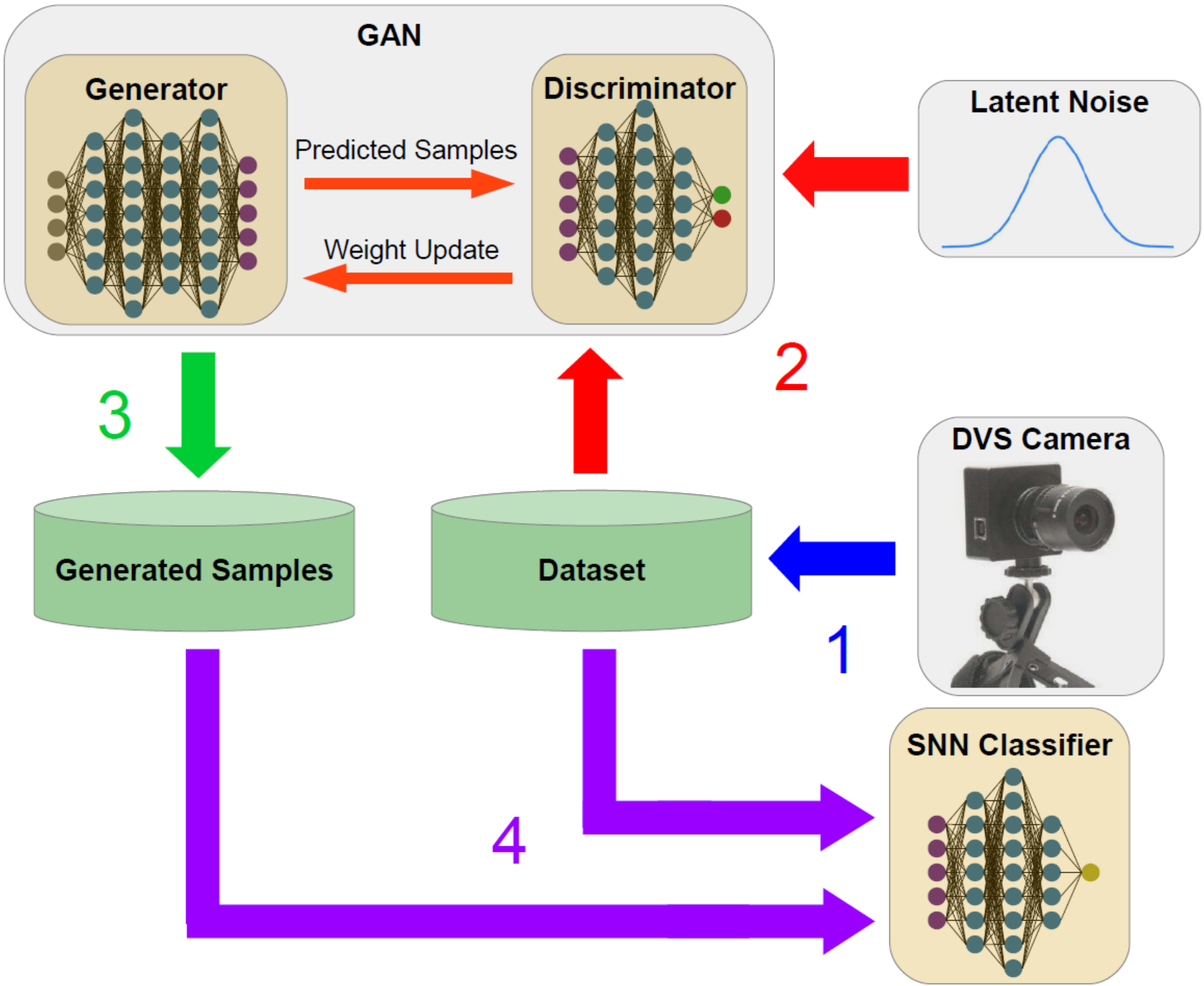
Helper Lighting (based on OTDD)

START		HELPER LIGHTING	TARGET: FLUORESCENT	FLUORESCENT LED	LAB	LED	NATURAL
FLUORESCENT			0.00	9.09	14.14	3.03	15.90
FLUORESCENT LED			4.04	0.00	5.05	6.57	2.27
LAB	+	TARGET	27.77	20.61	0.00	26.26	28.79
LED			16.16	4.24	9.09	0.00	12.88
NATURAL			14.14	6.06	10.10	5.05	0.00
FLUORESCENT			4.54	9.09	10.10	3.03	15.91
FLUORESCENT LED			6.06	-1.82	5.05	4.04	-0.76
LAB	+	CLOSEST	27.77	20.00	17.17	27.78	20.46
LED			16.66	5.46	9.09	8.08	9.09
NATURAL			12.12	3.64	7.07	4.55	9.85
FLUORESCENT			3.54	10.91	8.08	3.53	9.85
FLUORESCENT LED			3.03	1.81	1.01	5.05	-2.27
LAB	+	FURTHEST	33.33	27.28	16.16	32.32	18.94
LED			11.11	7.27	9.09	5.05	11.36
NATURAL			14.65	12.12	8.08	9.09	13.63

Helper Lighting	Accuracy Improvement (%)
START + TARGET	9.65
START + CLOSEST	10.16
START + FURTHEST	10.96

Neural Network Training Data

Chapter 5 – "Dataset Assembly for Training Spiking Neural Networks" (§2,§3,§4)

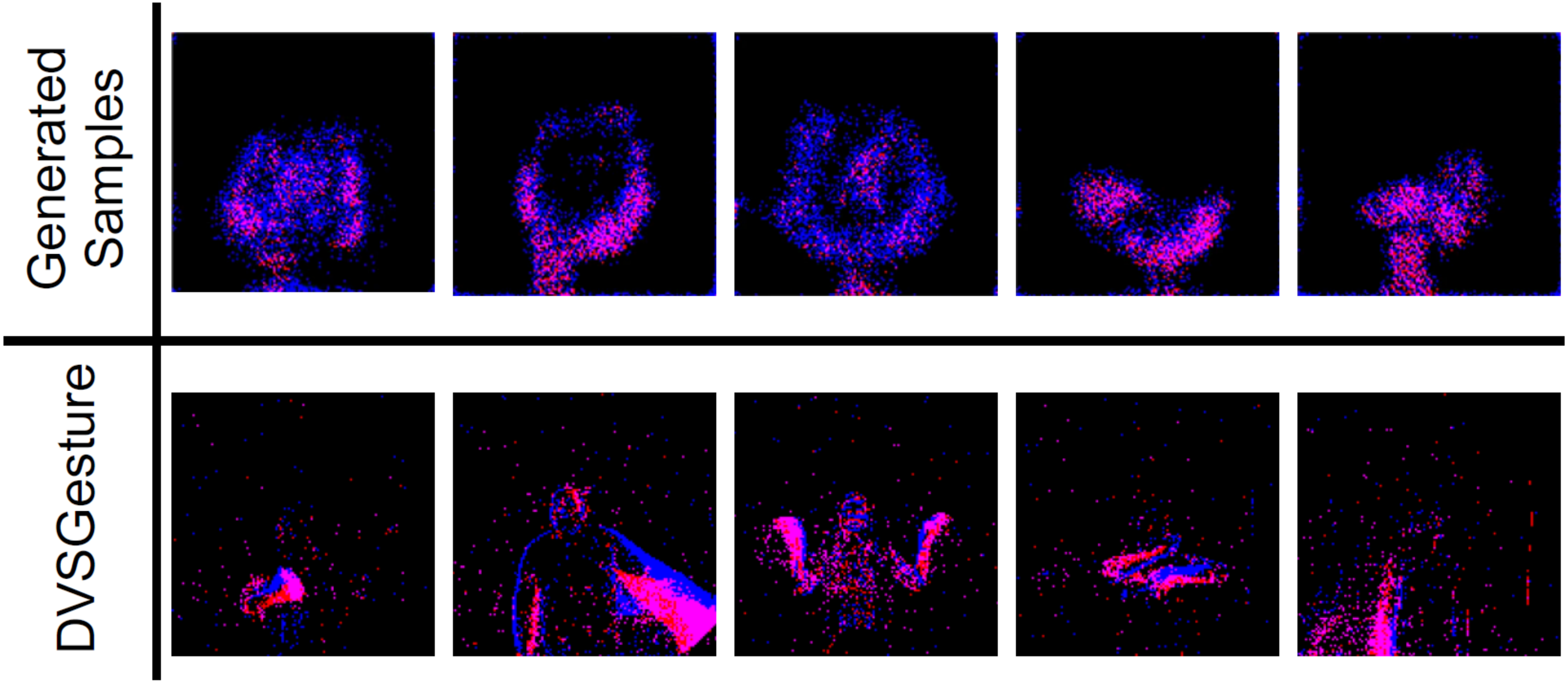


Generative Augmentation

START		HELPER LIGHTING	TARGET FLUORESCENT	TARGET FLUORESCENT LED	TARGET LAB	TARGET LED	TARGET NATURAL
FLUORESCENT	+	CGAN	1.52	6.67	5.05	0.00	5.30
FLUORESCENT LED			0.50	1.21	2.02	3.54	-4.54
LAB			6.59	10.30	9.09	8.08	6.06
LED			3.53	0.00	0.00	-2.02	6.06
NATURAL			4.55	3.64	6.06	-1.01	6.06
FLUORESCENT	+	CGAN & FURTHEST	5.55	12.12	14.14	2.02	12.12
FLUORESCENT LED			4.55	6.06	8.08	6.06	0.76
LAB			29.29	27.88	15.15	29.80	28.03
LED			5.55	5.45	7.07	0.50	12.12
NATURAL			10.61	9.09	12.12	4.04	12.88

Neural Network Training Data

Chapter 5 – "Dataset Assembly for Training Spiking Neural Networks" (§2,§3,§4)



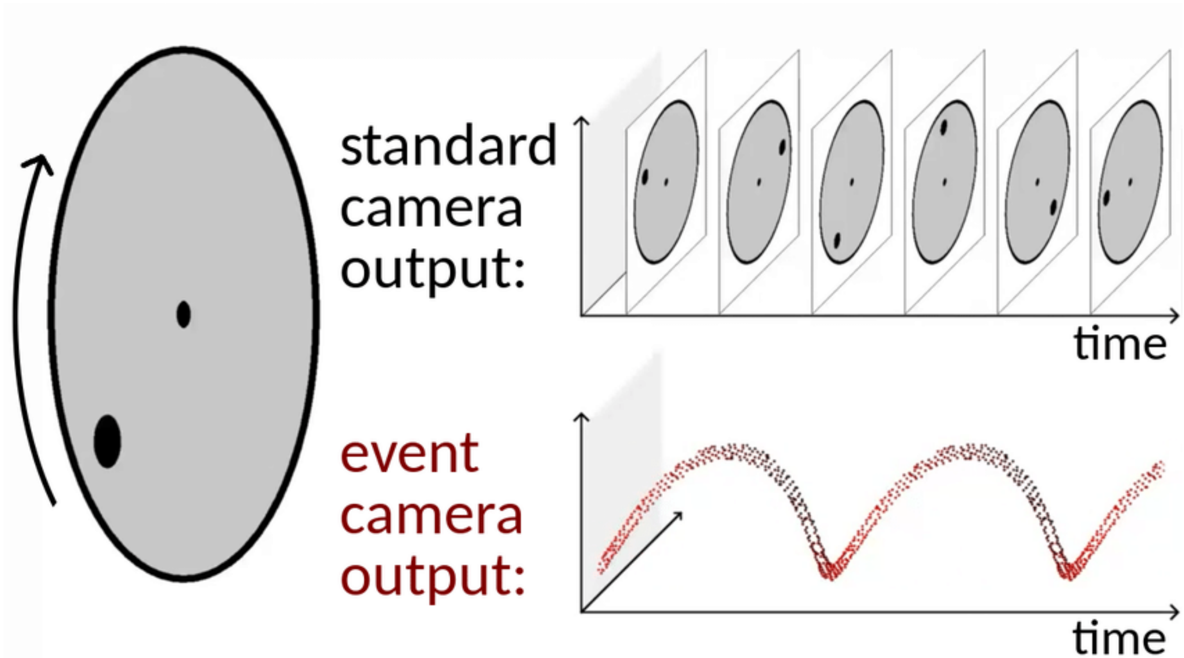
$$\text{accuracy robustness}\left(accuracy, START_{accuracy}\right) = \frac{accuracy - START_{accuracy}}{100 - accuracy}$$

Approach	Accuracy Robustness
START + TARGET	0.4451
START + CLOSEST	0.4598
START + FURTHEST	0.5108
START + CGAN	0.1613
START + CGAN + FURTHEST	0.5165
NOTHING	-0.1848

- **Chapter 1 – Introduction**
  - Motivation
  - Thesis Statement
  - Contributions and Outline
- **Neural Network Architecture**
  - Chapter 2 – "Lean Neural Networks for Autonomous Radar Waveform Design" (§**1**, §**2**,§**4**)
  - Chapter 2.5 – "Lean Neural Networks for Real-time Embedded Spectral Notching Waveform Design" (§**2**,§**5**)
- **Neural Network Training Data**
  - Chapter 3 – "Dataset Augmentation for Robust Spiking Neural Networks" (§**2**,§**3**,§**4**)
  - Chapter 4 – "Generative Data for Neuromorphic Computing" (§**2**,§**5**)
  - Chapter 5 – "Dataset Assembly for Training Spiking Neural Networks" (§**2**,§**3**,§**4**)
- **Neural Network Initialization**
  - Chapter 6 – "Generative Samples for Smooth Weight Transitioning to Spiking Neural Networks" (§**3**,§**5**)
- **Conclusions & Future Work Opportunities**

Big Idea

- Want to run ANN on neuromorphic hardware without needing to start from scratch SNN
- Need method for transitioning weights to higher dimensional sample space



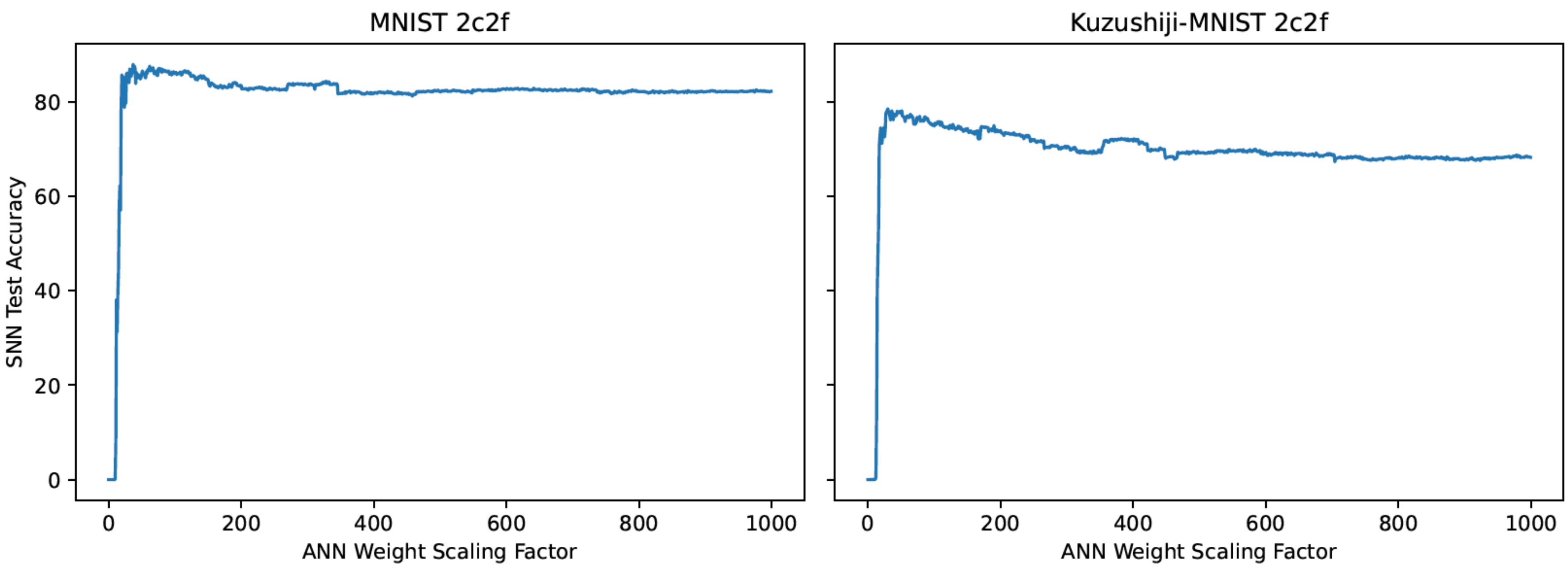
Work	Activation Substitution	Arbitrary Architecture	Post Correction	ANN Preservation
Nengo [20]	✓	✓		
Spikingjelly [22]	✓	✓		
Bu et al. [23]	✓ (quantized)		✓	
Hao et al. [24]	✓ (quantized)		✓	
SpikeZIP-TF [25]	✓ (quantized)	✓		
My work [Chapter 6]	✓	✓	✓	✓

Observation

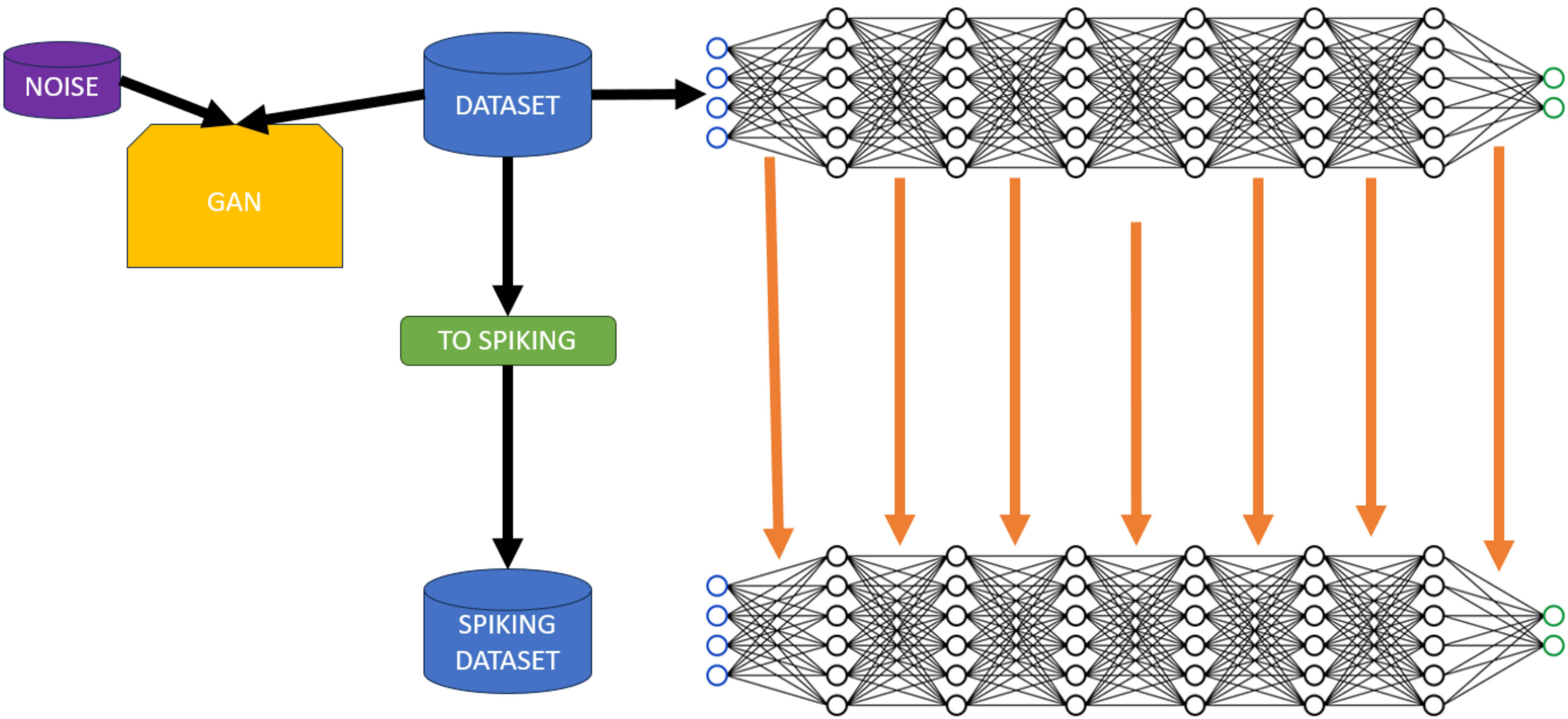
- Raw weight copying failed, weight scale needed
- Accuracy degradation moving from ANN to SNN

Dataset	Architecture	ANN	Weight Copy	Weight Copy + Scale
MNIST	2f	96.72	0.00	96.32
	2c2f	98.49	0.00	87.13
	3c3f	96.95	0.00	82.69
Kuzushiji MNIST	2f	91.46	0.00	90.65
	2c2f	96.98	0.00	84.97
	3c3f	91.45	0.00	78.15

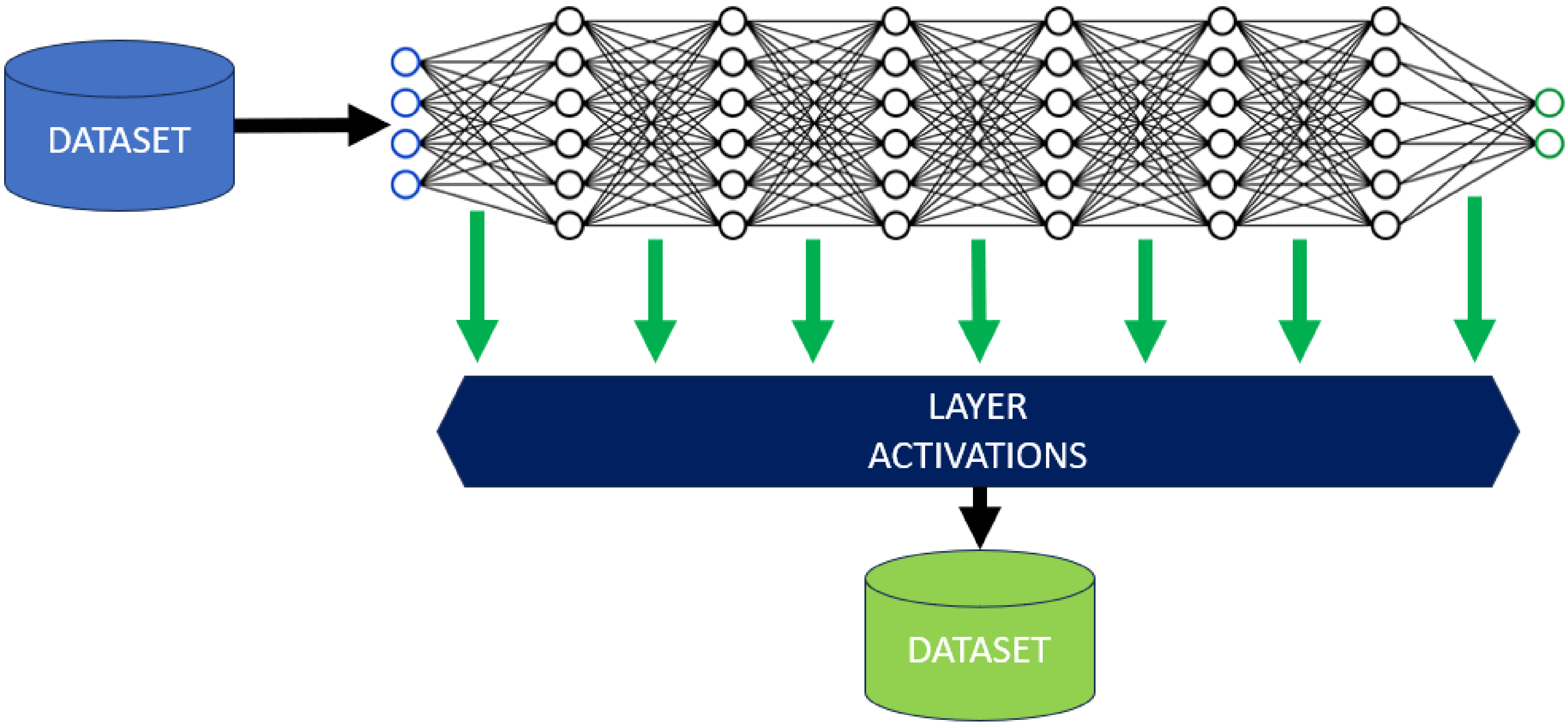
Dataset	Architecture	Best Scaling Factor
MNIST	2f	14.00 ± 1.309
	2c2f	28.33 ± 1.382
	3c3f	39.00 ± 2.760
Kuzushiji MNIST	2f	17.71 ± 0.837
	2c2f	35.43 ± 2.181
	3c3f	47.43 ± 1.757



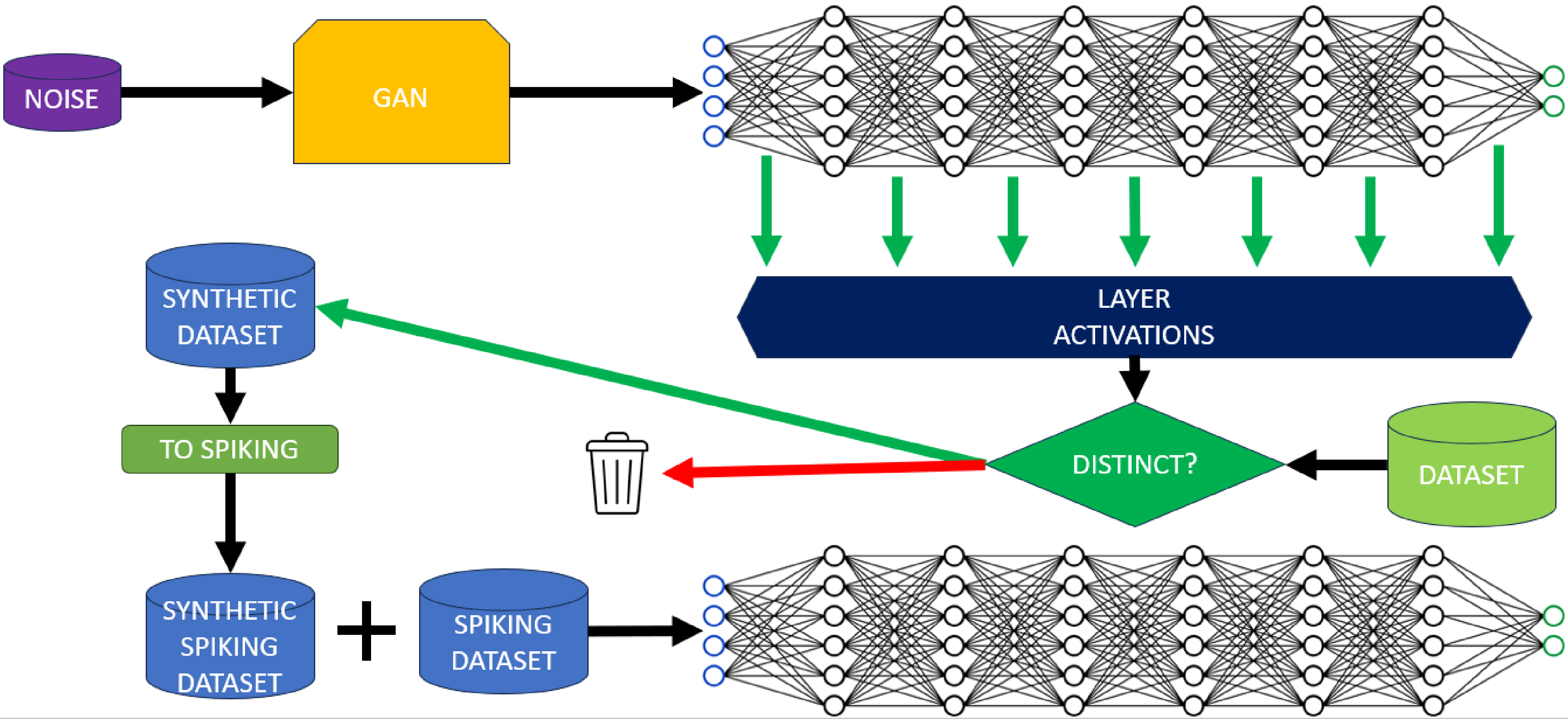
Step 1 – Copy weights & train GAN



Step 2 – Store layer activations



Step 3 – Generate unique samples



Observation

- Addition of GAN samples improves performance during re-training across all architectures/datasets

Dataset	Architecture	ANN	Weight Copy	Weight Copy + Scale	Weight Copy + Scale + re-Train	Weight Copy + Scale + re-Train + GAN	New SNN
MNIST	2f	96.72	0.00	96.32	89.08	90.78	90.94
	2c2f	98.49	0.00	87.13	87.16	91.38	87.75
	3c3f	96.95	0.00	82.69	78.69	80.86	13.17
Kuzushiji MNIST	2f	91.46	0.00	90.65	74.92	75.38	81.67
	2c2f	96.98	0.00	84.97	30.31	42.06	78.25
	3c3f	91.45	0.00	78.15	63.47	68.47	10.87

- **Chapter 1 – Introduction**
  - Motivation
  - Thesis Statement
  - Contributions and Outline
- **Neural Network Architecture**
  - Chapter 2 – "Lean Neural Networks for Autonomous Radar Waveform Design" (§1, §2,§4)
  - Chapter 2.5 – "Lean Neural Networks for Real-time Embedded Spectral Notching Waveform Design" (§2,§5)
- **Neural Network Training Data**
  - Chapter 3 – "Dataset Augmentation for Robust Spiking Neural Networks" (§2,§3,§4)
  - Chapter 4 – "Generative Data for Neuromorphic Computing" (§2,§5)
  - Chapter 5 – "Dataset Assembly for Training Spiking Neural Networks" (§2,§3,§4)
- **Neural Network Initialization**
  - Chapter 6 – "Generative Samples for Smooth Weight Transitioning to Spiking Neural Networks" (§3,§5)
- **Conclusions & Future Work Opportunities**

§1. Incorporation of existing problem-specific information into neural network architecture choices.
- Chapter 2

§2. Emphasis on maintaining low SWaP (size, weight, and power) solutions without sacrificing performance.
- Chapters 2 – 5

§3. Supplying self-correcting abilities via augmentative training data.
- Chapters 3, 5, 6

§4. Providing equivalent robustness to tried and true existing solutions.
- Chapters 2, 3, 5

§5. Insuring flexibility for deployment to the latest hardware including neuromorphic processors.
- Chapters 2, 4, 6

Future Work Opportunities

1. Applying generative augmentation to spiking transformers (spiking LLMs)
    - My work dealt with SNN simulations
    - Explore generative augmentation benefits on neuromorphic hardware
    - Incorporate latest transformer architectures on SNN platforms

2. Condensing my work for converting algorithms to SNNs in a single step
    - My work converts traditional algorithms to intelligently designed ANNs
    - My work also addressed converting ANNs to SNNs
    - As demand for SNNs increases, a single step conversion could prove to be important

3. Incorporating code/text analysis for automatic domain knowledge extraction from existing software solutions
    - My work “manually” extracted and domain knowledge for inclusion in ANN design (conversations, literature reading, etc.)
    - Code analysis (existing software solutions)
    - Text analysis via LLMs (existing literature)

THE OHIO STATE UNIVERSITY

Questions?

References

[Chapter 2] Baietto, A.; Boubin, J.; Farr, P.; Bihl, T.J.; Jones, A.M.; Stewart, C. Lean Neural Networks for Autonomous Radar Waveform Design. Sensors 2022, 22, 1317. https://doi.org/10.3390/s22041317

[Chapter 2.5] A. Baietto, J. Boubin, P. Farr and T. J. Bihl, "Lean Neural Networks for Real-time Embedded Spectral Notching Waveform Design," 2022 IEEE 31st International Symposium on Industrial Electronics (ISIE), Anchorage, AK, USA, 2022, pp. 1121-1126, doi: 10.1109/ISIE51582.2022.9831772.

[Chapter 3] Anthony Baietto, Christopher Stewart, Trevor J. Bihl. “Dataset Augmentation for Robust Spiking Neural Networks”. IEEE International Conference on Autonomic Computing and Self-Organizing Systems, Toronto, Canada, 2023

[Chapter 4] Anthony Baietto, Trevor J. Bihl. “Generative Data for Neuromorphic Computing”. Hawaii International Conference on Systems Sciences, Waikoloa Village, Big Island, HI, USA, 2025

[Chapter 5] “Dataset Assembly for Training Spiking Neural Networks”, Neurocomputing. Under review.

[Chapter 6] “Generative Samples for Smooth Weight Transitioning to Spiking Neural Networks”. In preparation.

References

[1] N. Maslej, L. Fattorini, R. Perrault, V. Parli, A. Reuel, E. Brynjolfsson,J. Etchemendy, K. Ligett, T. Lyons, J. Manyika, J. C. Niebles, Y. Shoham,R. Wald, and J. Clark, “Artificial intelligence index report 2024,” 2024.

[2] L. Bernstein, A. Sludds, R. Hamerly, V. Sze, J. Emer, and D. Englund,“Freely scalable and reconfigurable optical hardware for deep learning,” Sci-entific Reports, vol. 11, 02 2021.

[3] P. Villalobos and A. Ho, “Trends in training dataset sizes,” 2022. Accessed:2023-10-3.

[4] Gerchberg, R.W. (1972) A Practical Algorithm for the Determination of Phase from Image and Diffraction Plane Pictures. Optik, 35, 237-246.

[5] T. Higgins, T. Webster, and A. K. Shackelford, “Mitigating interference via spatial and spectra nulling,” IET Radar, Sonar & Navigation, vol. 8, no. 2, pp. 84–93, 2014. [Online]. Available: https://ietresearch.onlinelibrary.wiley.com/doi/abs/10.1049/iet-rsn.2013.0194

[6] Liu, G., et al.: A GPU-based real-time processing system for frequency division multiple-input-multiple-output radar. IET Radar Sonar Navig. 17(10), 1524–1537 (2023). https://doi.org/10.1049/rsn2.12439

[7] Xia, Y., Ma, Z., Huang, Z.: Radar waveform recognition based on a two-stream convolutional network and software defined radio. IET Radar Sonar Navig. 16(5), 837–851 (2022). https://doi.org/10.1049/rsn2.12224

[8] R. Michev, Y. Shu, D. Werbunat, J. Hasch and C. Waldschmidt, "Adaptive Compensation of Hardware Impairments in Digitally Modulated Radars Using ML-Based Behavioral Models," in IEEE Transactions on Microwave Theory and Techniques, doi: 10.1109/TMTT.2023.3285438.

References

[9] J. Boubin, A. M. Jones, and T. Bihl, “Neurowav: Toward real-time waveform design for vanets using neural networks,” in 2019 IEEE Vehicular Networking Conference (VNC), 2019, pp. 1–4.

[10] P. John-Baptiste, G. E. Smith, A. M. Jones, and T. Bihl, “Rapid waveform design through machine learning,” in 2019 IEEE 8th International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP), 2019, pp. 659–663.

[11] P. Farr, A. M. Jones, T. Bihl, J. Boubin, and A. DeMange, “Waveform design implemented on neuromorphic hardware,” in 2020 IEEE International Radar Conference (RADAR), 2020, pp. 934–939.

[12] M. Davies, N. Srinivasa, T.-H. Lin, G. Chinya, Y. Cao, S. H. Choday, G. Dimou, P. Joshi, N. Imam, S. Jain, Y. Liao, C.-K. Lin, A. Lines, R. Liu, D. Mathaikutty, S. McCoy, A. Paul, J. Tse, G. Venkataramanan, Y.-H. Weng, A. Wild, Y. Yang, and H. Wang, “Loihi: A neuromorphic manycore processor with on-chip learning,” IEEE Micro, vol. 38, no. 1, pp. 82–99, 2018.

[13] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, & Yoshua Bengio. (2014). Generative Adversarial Networks.

[14] V. Kotariya and U. Ganguly, “Spiking-gan: A spiking generative adversarialnetwork using time-to-first-spike coding,” 2021.

[15] B. Rosenfeld, O. Simeone, and B. Rajendran, “Spiking generative adversar-ial networks with a neural network discriminator: Local training, bayesianmodels, and continual meta-learning,” 2021.

[16] Q. Xu, J. Peng, J. Shen, H. Tang, and G. Pan, “Deep covdensesnn: Ahierarchical event-driven dynamic framework with spiking neurons in noisyenvironment,” Neural Networks, vol. 121, pp. 512–519, 2020.

References

[17] O. Ozdenizci and R. Legenstein, “Adversarially robust spiking neural net-works through conversion,” 2024.

[18] J. Ding, T. Bu, Z. yu, T. Huang, and J. Liu, “Snn-rat: Robustness-enhancedspiking neural network through regularized adversarial training,” 10 2022.

[19] B. Wang, J. Cao, J. Chen, S. Feng, and Y. Wang, “A new ann-snn conver-sion method with high accuracy, low latency and good robustness,” in Pro-ceedings of the Thirty-Second International Joint Conference on ArtificialIntelligence, IJCAI-23 (E. Elkind, ed.), pp. 3067–3075, International JointConferences on Artificial Intelligence Organization, 8 2023. Main Track.

[20] Bekolay, T., Bergstra, J., Hunsberger, E., DeWolf, T., Stewart, T., Rasmussen, D., Choo, X., Voelker, A., & Eliasmith, C. (2014). Nengo: a Python tool for building large-scale functional brain models. Frontiers in Neuroinformatics, 7(48), 1–13.

[21] D. Alvarez-Melis and N. Fusi, “Geometric dataset distances via optimaltransport,” 2020.

[22] W. Fang, Y. Chen, J. Ding, Z. Yu, T. Masquelier, D. Chen, L. Huang,H. Zhou, G. Li, and Y. Tian, “Spikingjelly: An open-source machine learn-ing infrastructure platform for spike-based intelligence,” Science Advances,vol. 9, no. 40, p. eadi1480, 2023.

[23] T. Bu, W. Fang, J. Ding, P. Dai, Z. Yu, and T. Huang, “Optimal ann-snnconversion for high-accuracy and ultra-low-latency spiking neural networks,”2023.

[24] Hao, Z., Bu, T., Ding, J., Huang, T., & Yu, Z. (2023). Reducing ANN-SNN Conversion Error through Residual Membrane Potential. Proceedings of the AAAI Conference on Artificial Intelligence, 37(1), 11-21. https://doi.org/10.1609/aaai.v37i1.25071

References

[25] K. You, Z. Xu, C. Nie, Z. Deng, Q. Guo, X. Wang, and Z. He, “Spikezip-tf:Conversion is all you need for transformer-based snn,” 2024.