

Generative programming of multilayer optical processor

Petr Šícho

supervisors

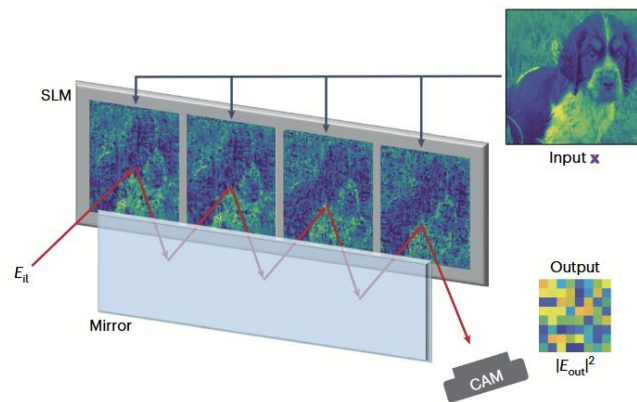
Niyazi Ulas Dinç

Mustafa Yildirim

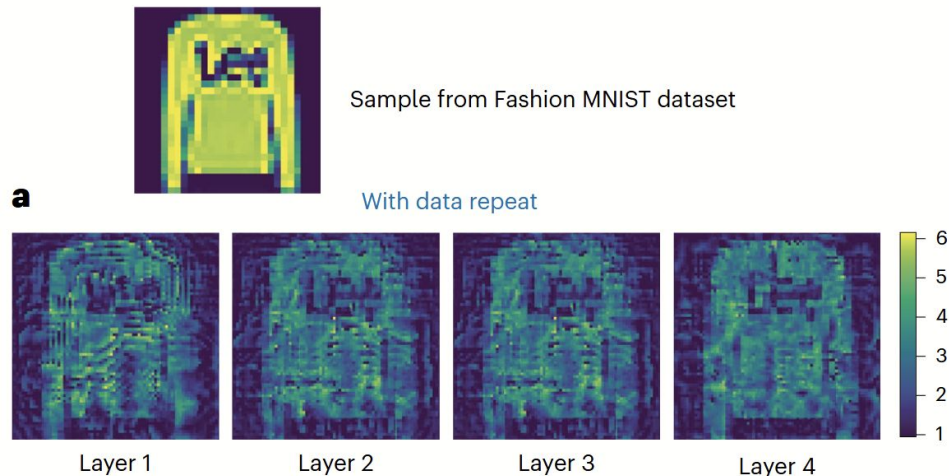
Final Presentation

Optical neural networks

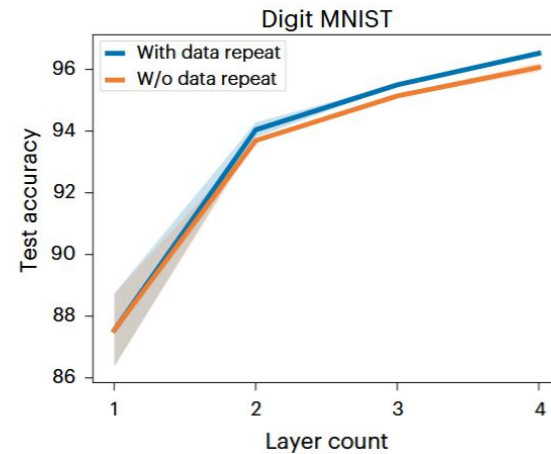
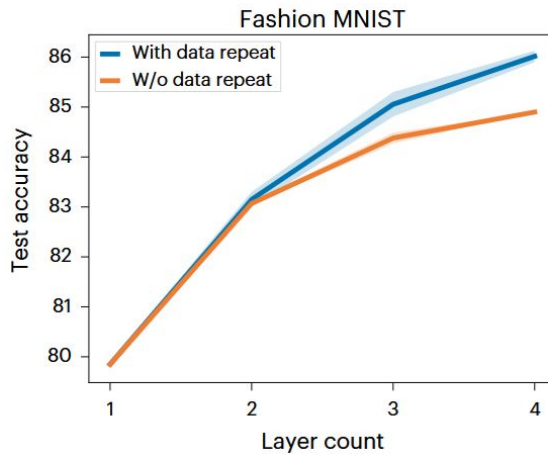
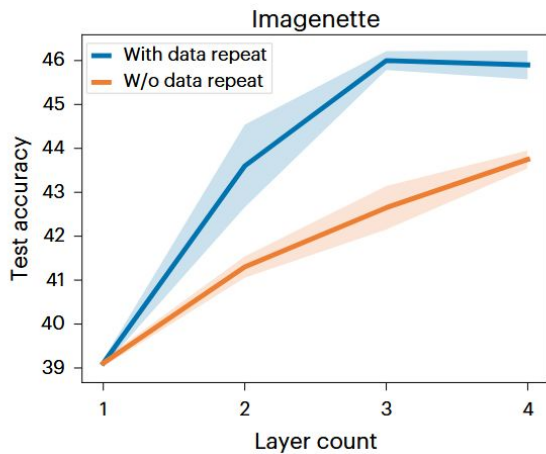
- Optical neural networks energy-efficient alternative
- Multiple SLM layers changing phase of light
- Transformed input image is on each layer



- Scale and bias in each layer
 - Backpropagation
- Average pool output intensity
- Use linear classifier -> label

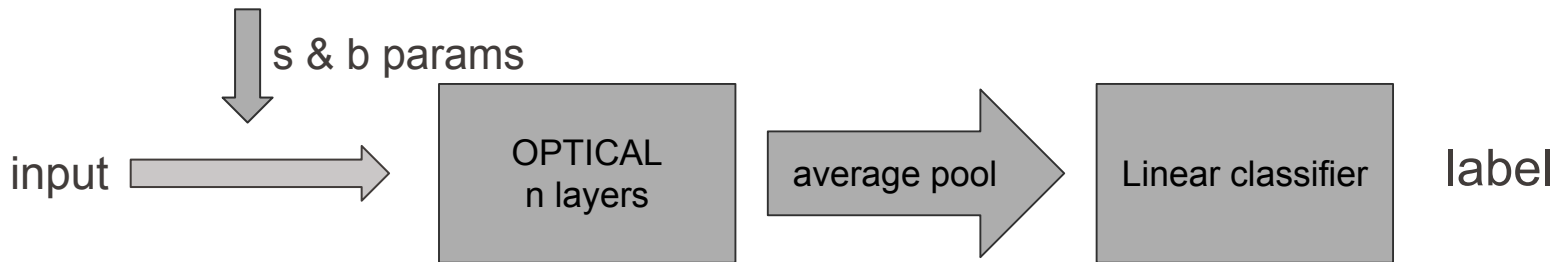


- Performance does not increase with growing layer count
- It is hard to train deeper networks
- Large parameter count -> overfitting



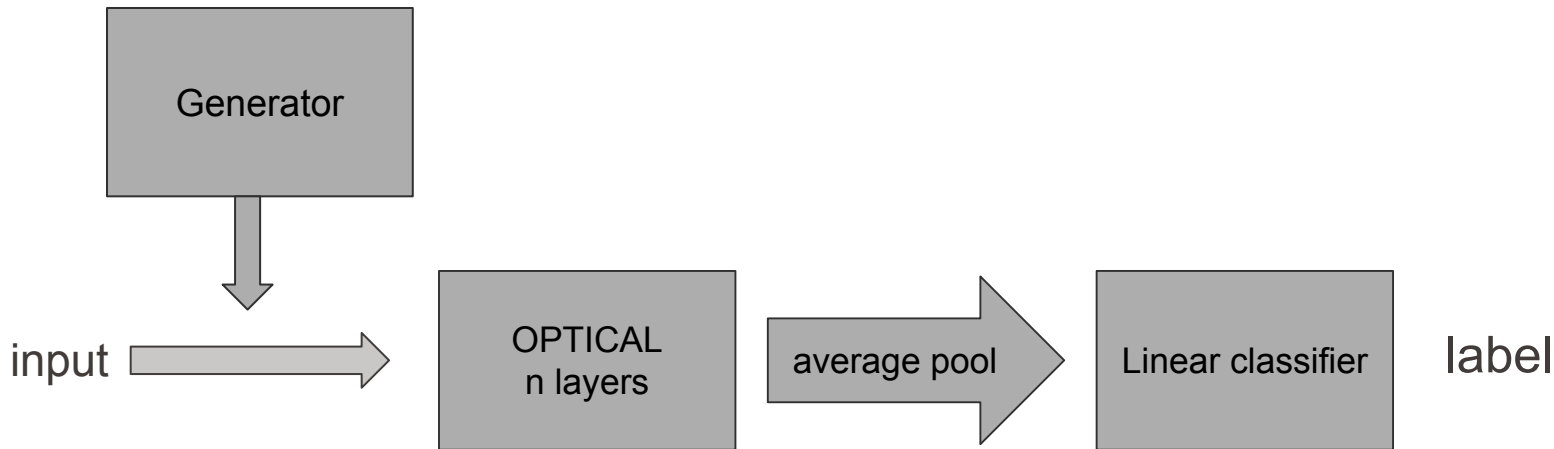
Improve learning performance with higher layer count

- Use implicit neural representation (Generators)
- This should make the gradient landscape smoother



Generate scale & bias using a neural network

- Choose a generator that could nicely generate smooth mask
- Lower parameter count

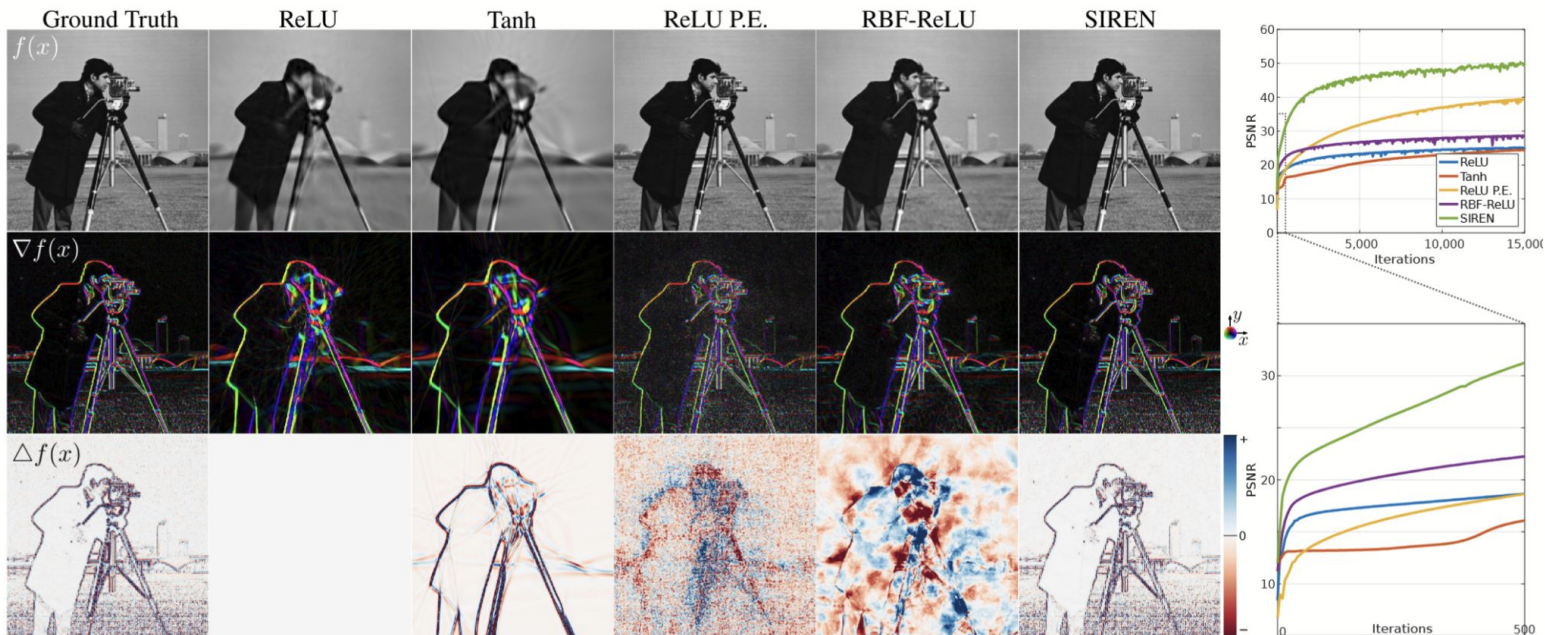


SIREN

Implicit Neural Representations with Periodic Activation Functions

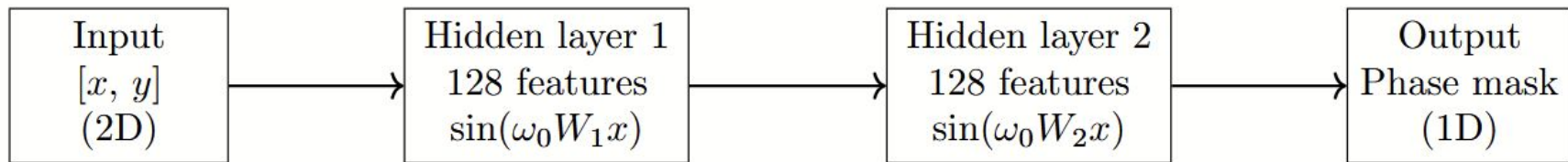
- Use periodic activation function $\sin(\omega_0 \cdot (ax + b))$
 - base frequency should match the data
- Useful for representing natural data - images, 3D models
- Works well with physics (Solving Helmholtz and Wave Equations)

Nicely represents zero, first and second order derivatives

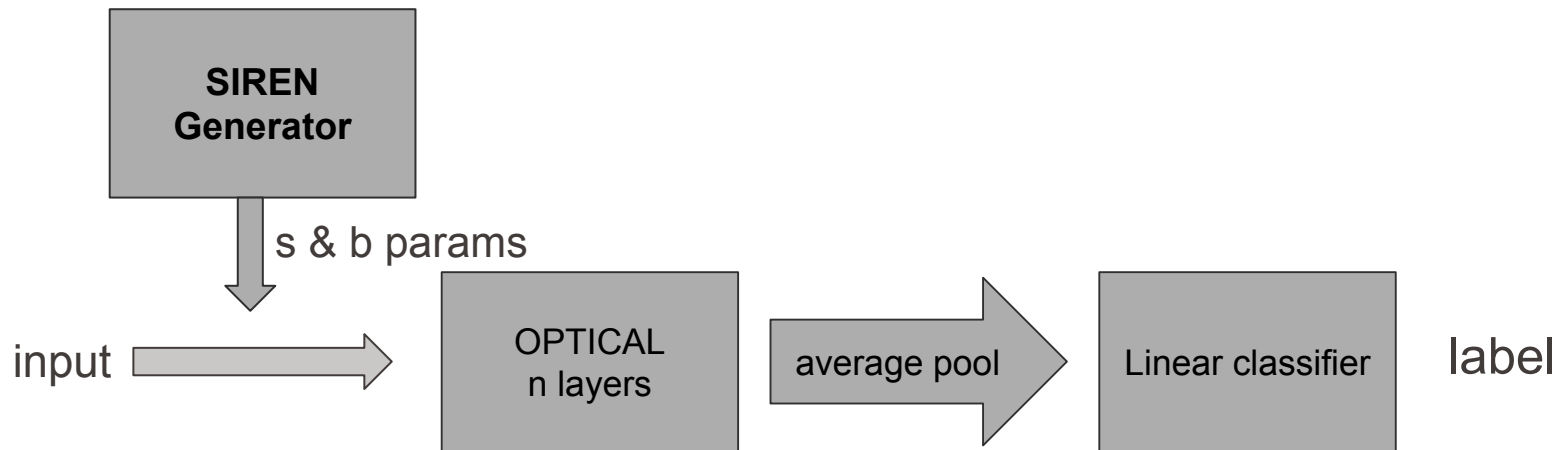


**SIREN
Generator**

- Use SIREN to generate scale and bias parameters
- Input are 2D coordinates $(-1 \text{ to } 1) \times (-1 \text{ to } 1)$
- Output is a phase mask



Use SIREN to generate scale and bias parameters



Process

SIREN sensitive to hyperparameters - sensitive to ω and learning rate

→ need to do parameter search

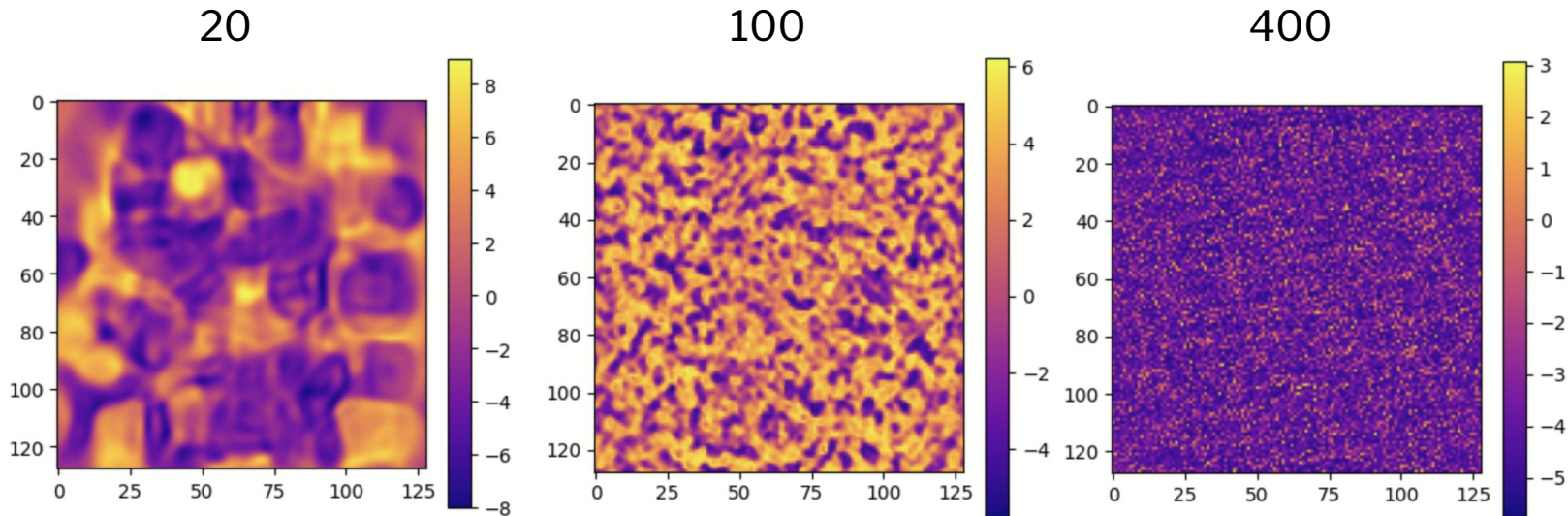
Training takes a very long time (> 1hr per epoch with 20 epochs)

- Optimize the code
- Use smaller optical layer size to find what works, then train full-size

Base frequency omega

$$\sin(\omega_0 \cdot (ax + b))$$

- In the original SIREN paper, they used 30 for all the applications
- In our testing, the range 200 - 400 worked the best



$$\sin(\omega_0 \cdot (ax + b))$$

- SIREN is very sensitive to learning rate
- For each setting, we did a search for the best initial learning rate
- Found out a relationship that helped future parameter search

$$\text{learning rate} \approx \frac{1}{\omega_0}$$

Introduced learning rate scheduler -> 10x decrease over 10 epochs

- This helped cut training time in half

Single SIREN outputting all the masks vs. one for each of the parameters

- Hard to tune for all layer counts
- Using the same parameters and multiplying the generators scales

Data augmentation

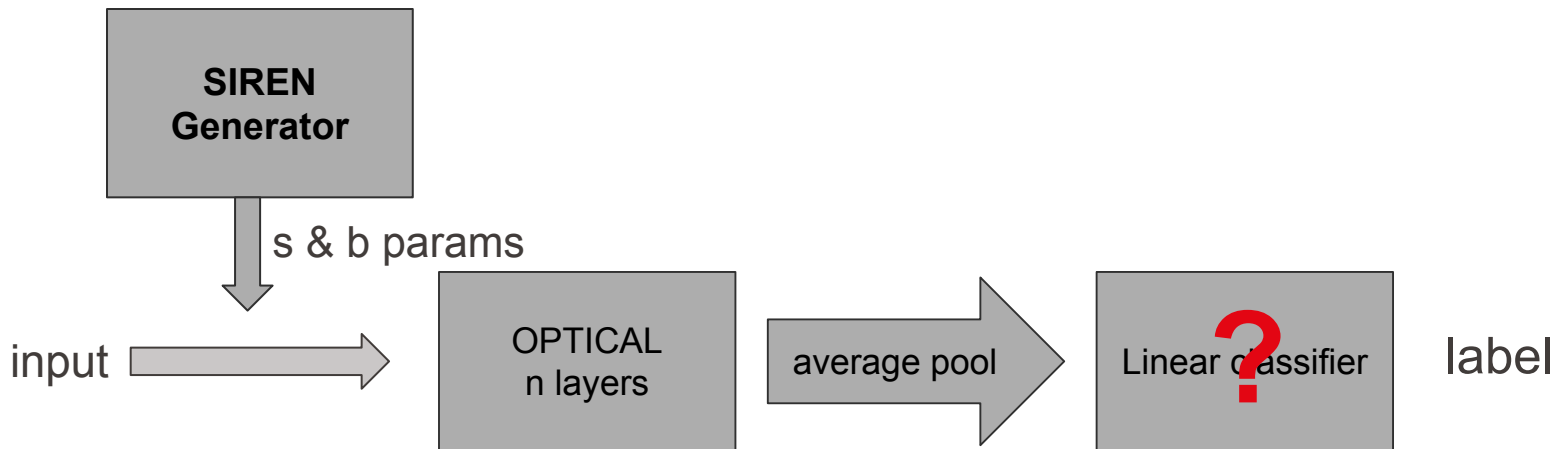
- With higher layer counts overfit a lot
 - Introduced data augmentation (flip, crop)

Normalization

- Crucial to good performance

At higher layer count, the classifier does not add too much performance

- Can lead to fully optical network



Results

Using CIFAR-10 dataset

- 60'000 images 32x32 pixels with 10 classes
- Normally RGB, but here we convert to grayscale

Classical results

- Fully Connected: 48 %
- LeNet (Convolutional Net): 64 %

Baseline results - using pure backpropagation (window size 2x of mask)

- Learnable parameters 128x128: **47 %**

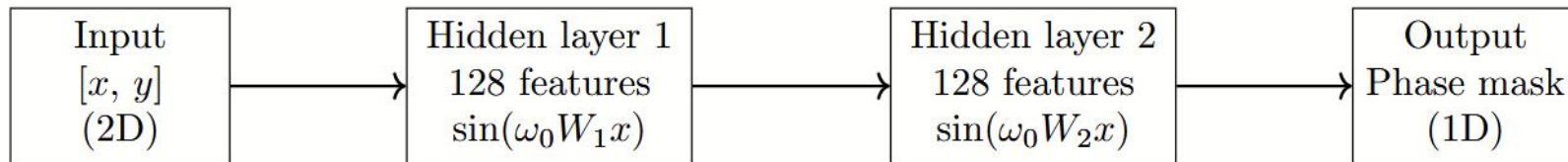
Layers	1	2	3	4	6
Accuracy	39.66 %	44.19 %	45.55 %	46.79 %	47.28 %

- Learnable parameters 512x512: **50 %**

Layers	1	2	3	4	6
Accuracy	40.92 %	46.51 %	47.58 %	48.63 %	49.54 %

SIREN results - 128x128 learnable parameters (2 hidden layers)

- Optical size 512x512, but 16x less parameters than backpropagation



SIREN results - 128x128 learnable parameters (2 inner layers)

- Optical size 128x128: **47 %** → **51 %**

Layers	1	2	3	4	6
Baseline	39.66 %	44.19 %	45.55 %	46.79 %	47.28 %
SIREN	40.94 %	46.61 %	45.74 %	48.90 %	50.91 %

- Optical size 512x512: **49 %** → **53 %**

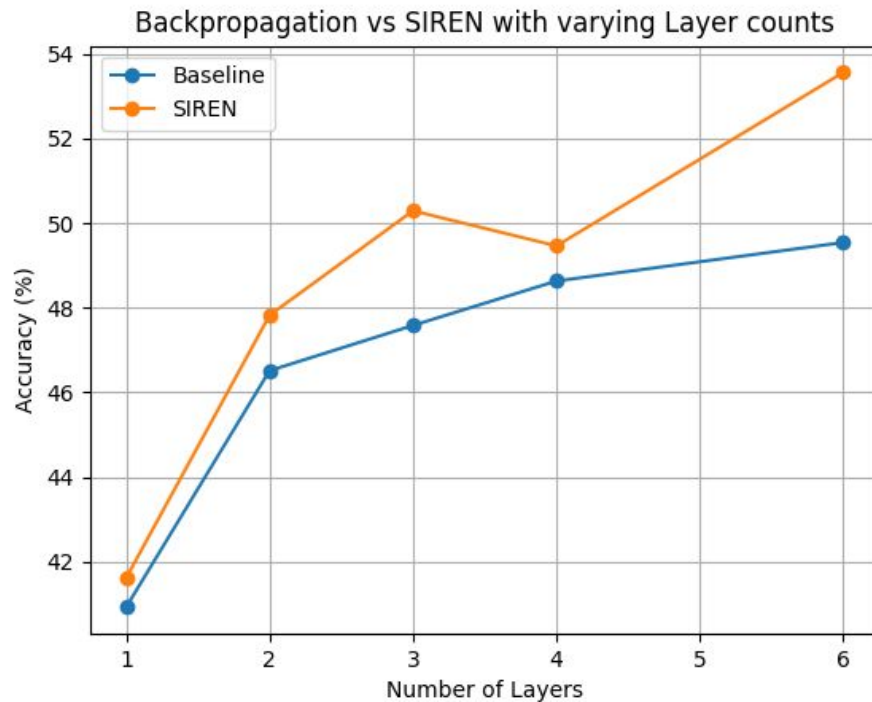
Layers	1	2	3	4	6
Baseline	40.92 %	46.51 %	47.58 %	48.63 %	49.54 %
SIREN	41.61 %	47.82 %	50.29 %	49.46 %	53.56 %

Performance growth

- Bigger increase with layers

Gain of about 4 %

at higher layer count



Conclusion

Generator (SIREN) can:

- reduce the parameter count while keeping high optical resolution
- Increase performance of the model (**49% -> 53%**)
- With increasing layer count, the difference grows

But

- SIRENs are sensitive to hyperparameters

Thank you