Predictive Coding Inspired Noise Robust Network

Application in image classification

Rong Zeng, Mohammad Shakil

Electrical and Computer Engineering Department
University of Rochester

May 9, 2025

Table of Contents

- Predictive coding (PC) for a network of L layers can be viewed as a generative model that minimizes a global energy function comprised of local errors.

$$\mathcal{F} = \sum_{l=0}^{L-1} \frac{1}{2} \|\epsilon_l\|^2 = \frac{1}{2} \sum_{l=0}^{L-1} \|\mu_l - x_l\|_2^2$$

where x_l is the actual representation and μ_l is the prediction for layer l

$$\mu_l = W_l^T \, x_{l+1}$$

with x_L being the generate instruction or label and x_0 being the generated image.

- Predictive coding (PC) for a network of L layers can be viewed as a generative model that minimizes a global energy function comprised of local errors.

$$\mathcal{F} = \sum_{l=0}^{L-1} \frac{1}{2} \|\epsilon_l\|^2 = \frac{1}{2} \sum_{l=0}^{L-1} \|\mu_l - x_l\|_2^2$$

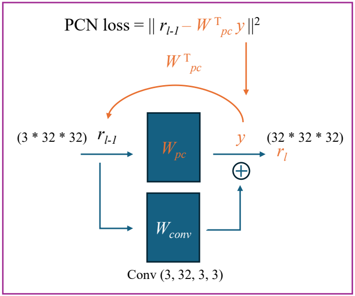
where x_l is the actual representation and μ_l is the prediction for layer l

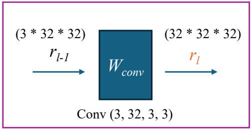
$$\mu_l = W_l^T \, x_{l+1}$$

with x_L being the generate instruction or label and x_0 being the generated model.

- The framework is flexible, we can also fix the x_0, minimizing the energy function above to get the prediction of the output label.

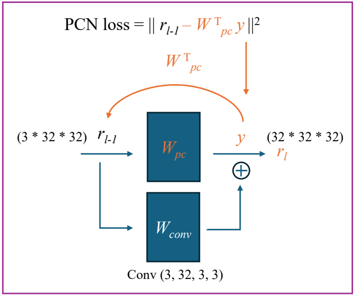
- Inspired by the PC algorithm, we develop a normal feed-forward convolutional neural network with local PC recurrent structure.

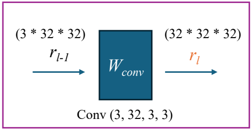




Predictive Coding Inspired Neural Network

- Inspired by the PC algorithm, we develop a normal feed-forward convolutional neural network with local PC recurrent structure.





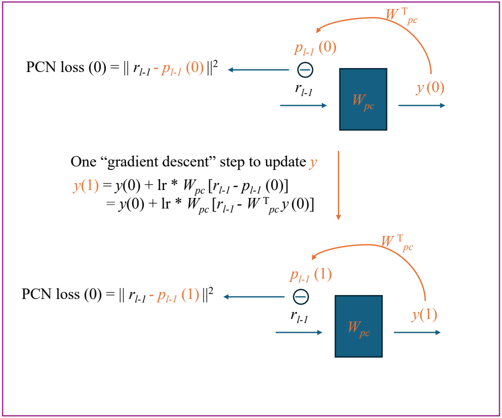
- With y being the output of the local PC recurrent, we try to minimize the local error ϵ_l with respect to y

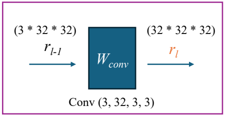
$$\nabla_y \frac{1}{2} \|\epsilon_{l-1}\|^2 = W_{pc}(r_{l-1} - W_{pc}^T\, y)$$

- The gradient of local error ϵ_l with respect to y

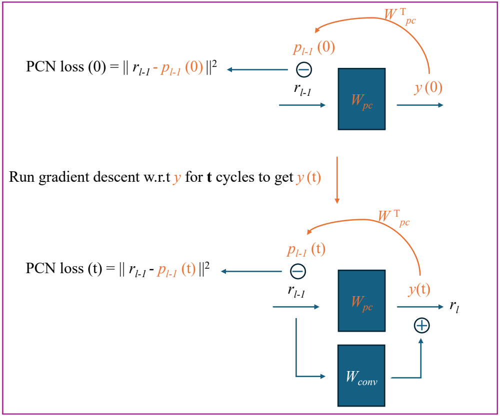
$$\nabla_y \frac{1}{2}\|\epsilon_{l-1}\|^2 = W_{pc}(r_{l-1} - W_{pc}^T\, y)$$

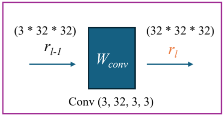
- Run gradient descent for t cycles with respect to y





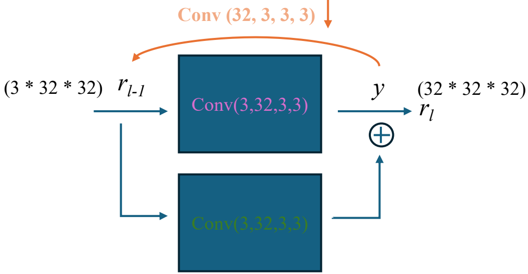
- Run gradient descent for t cycles with respect to y and add the result with the output of a normal convolution to get the final output of a layer





- Each gradient descent step can be viewed as one feedforward convolution layer plus a feedback transposed convolution layer.
- Unlike the naive way of optimizing the local PC loss, we use different sets of parameters for feedback and feedforward convolution.
- Feedforward (FF), Feedback (FB) and Bypass (BP) layers

$$\text{PCN loss} = \| r_{l-1} - W^{T}_{pc}\, y \|^2$$



UNIVERSITY of ROCHESTER

- Naive Implementation

$$y_{t+1} = y + lr * W_{FF}(r_{l-1} - W_{FF}^T\, y_t)$$

After T steps,

$$r_l = y_T + W_{BP}\, r_{l-1}$$

- More weights

$$y_{t+1} = y + lr * W_{FF}(r_{l-1} - W_{FB}\, y_t)$$

- Even adding non-linearity

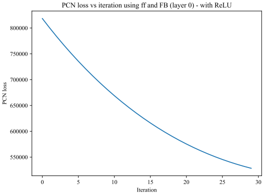
$$y_{t+1} = y + lr * W_{FF}[ReLU(r_{l-1} - W_{FB}\, y_t)]$$

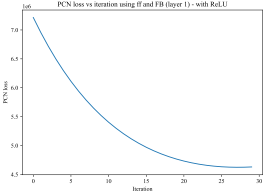
Experiment Result

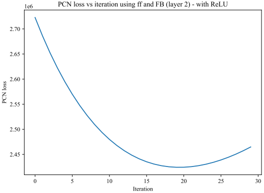
- Train and evaluate a 5-layer convolutional PC network on cifar-10.

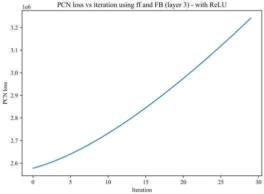
PCN ($\sim$0.59M)	**ResNet-18** ($\sim$11M)	**ResNet-34** ($\sim$21M)
91.11%	93.07%	93.34%

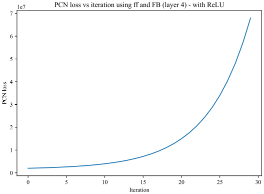
- Local PC loss versus the number of "gradient steps" for each layer.











- Different inference scheme for the trained model.
  - MVM = Matrix-vector Multiplication, running feedforward and feedback convolution for multiple cycles.
  - Direct = use gradient descent to directly minimizing the local PC loss.

Method	Conv layers	With ReLU	Acc
MVM	FB, FF, BP	No	50.08%
		Yes	90.02%
	FB, BP	Yes	13.64%
	FB, BP	No	14.10%
Direct	FB, BP	No	13.08%

- Naive Implementation

$$y_{t+1} = y + lr * W_{FF}(r_{l-1} - W_{FF}^{T}\, y_t)$$

After T steps,

$$r_l = y_T + W_{BP}\, r_{l-1}$$

- More weights

$$y_{t+1} = y + lr * W_{FF}(r_{l-1} - W_{FB}\, y_t)$$

- Even adding non-linearity

$$y_{t+1} = y + lr * W_{FF}[ReLU(r_{l-1} - W_{FB}\, y_t)]$$

- Tie BP to FF

$$r_l = y_T + W_{FF}\, r_{l-1}$$

Experiment Result

- Explore more ways of training the model
  - Tie FF and FB
  - Tie BP with FF
  - Remove ReLU between FF and FB

Tie FF/FB	Tie BP	With ReLU	Param (M)	Acc
UnTied	Untied	Yes	0.59	91.11%
		No	0.92	89.96%
			0.59	89.58%
	Tied	Yes	0.87	92.24%
		No	0.87	89.61%
Tied	Untied	Yes	0.86	91.21%
		No	0.86	88.31%
	Tied	Yes	0.77	91.62%
		No	0.77	88.48%

- The last two layers are not really optimizing the local PC loss. But if we tie the weights of FF/FB and remove ReLU in between, we can save the accuracy by adding more parameters (still < 1M).
- The ReLU connection in between does improve the accuracy, even when the weights of FF/FB are tied.
- By tying the weights of Bypass convolution, we can reduce the number of parameters and keep the accuracy.