Review briefly predictive encoding and create a predictive encoding model on CIFAR10 and attack it to verify its robustness.

This task said to create a predictive encoding model on CIFAR10, and attack it to verify its robustness.

So first, I have a look at [2].Then i know that, [3] is a deep convolutional recurrent neural network inspired by the principles of predictive coding.
But it is trained for next-frame video prediction not the image.
And then i find a paper named [1], they propose a predictive coding networks(PCN) for Object Recognition.
The PCN is used for object recogntion, so i have a implement with it on CIFAR10.

The network structure is like the next figure.

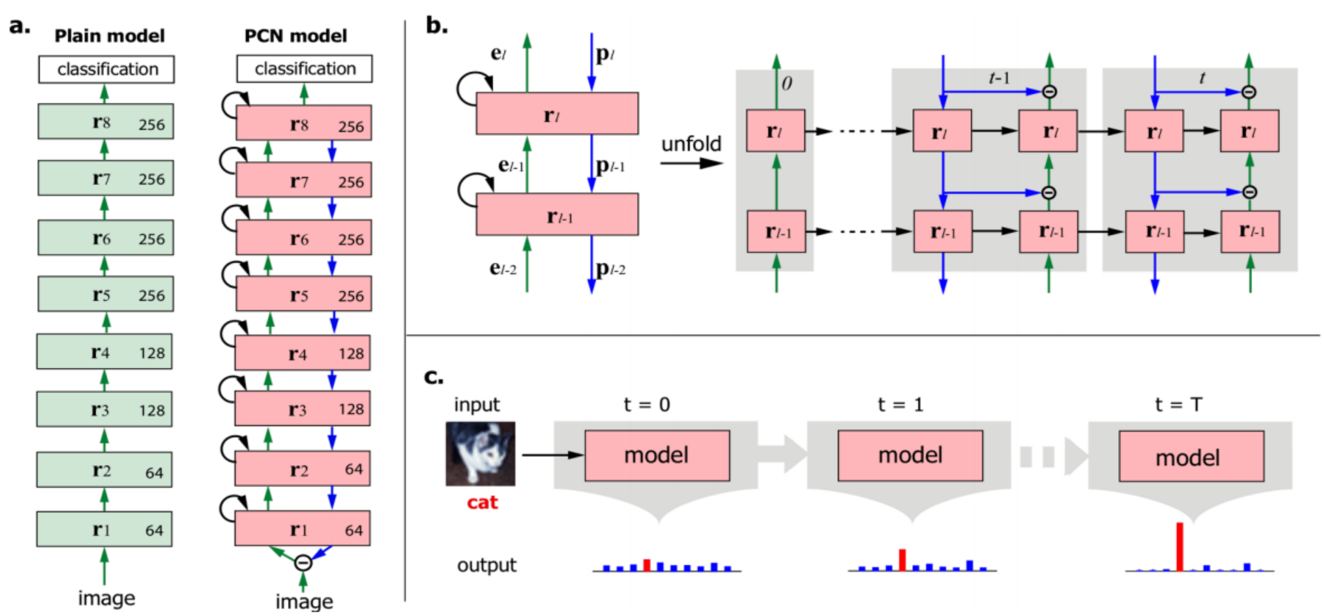

Figure 1. a) An example PCN with 9 layers and its CNN counterpart (or the plain model). b) Two-layer substructure of PCN. Feedback (blue), feedforward (green), and recurrent (black) connections convey the top-down prediction, the bottom-up prediction error, and the past information, respectively. c) The dynamic process in the PCN iteratively updates and refines the representation of visual input over time. PCN outputs the probability over candidate categories for object recognition. The bar height indicates the probability and the red indicates the ground truth.

So this network have 9 layers, like a.
b is Two-layer substructure of PCN.Feedback (blue), feedforward (green), and recurrent (black) connections convey the top-down prediction, the bottom-up prediction error, and the past information, respectively.

Model Training

The next is the Algorithm for the PCN network.

Algorithm 1 Deep Predictive Coding Network

Input static image: $\mathbf{x}$

$\mathbf{r}_0(t) \leftarrow \mathbf{x}$

% initialize representations

for $l = 0$ **to** L-1 **do**

$\quad \mathbf{r}_{l+1}(0) \leftarrow \mathrm{ReLU}\left(\mathrm{FFConv}(\mathbf{r}_l(0))\right)$

% recurrent computation with T cycles

for t = 1 to T **do**

$\quad$ *% nonlinear feedback process*

$\quad$ **for** $l = \mathrm{L}$ **to** 1 **do**

$\quad\quad \mathbf{p}_{l-1}(t\text{-}1) \leftarrow \mathrm{FBConv}\left(\mathbf{r}_l(t\text{-}1)\right)$

$\quad\quad$ **if** $l > 1$ **do**

$\quad\quad\quad \mathbf{r}_{l-1}(t\text{-}1) \leftarrow \mathrm{ReLU}\left((1\text{-}b)\mathbf{r}_{l\text{-}1}(t\text{-}1) + b\mathbf{p}_{l\text{-}1}(t\text{-}1)\right)$

$\quad$ *% nonlinear feedforward process*

$\quad$ **for** $l = 0$ **to** L-1 **do**

$\quad\quad \mathbf{e}_l(t) \leftarrow \mathbf{r}_l(t) - \mathbf{p}_l(t\text{-}1)$

$\quad\quad \mathbf{r}_{l+1}(t) \leftarrow \mathrm{ReLU}\left(\mathbf{r}_{l+1}(t\text{-}1) + a\,\mathrm{FFConv}(\mathbf{e}_l(t))\right)$

% classification

Output $\mathbf{r}_\mathrm{L}(\mathrm{T})$ for classification

> These figure from [1].

The traing log is below:

```
Training Setting: batchsize=128 | epoch=246 | lr=1.0e-05
Training: Epoch=246 | Loss: 0.000 |  Acc: 99.998% (49999/50000) | best acc: 93.750
Testing: Epoch=246 | Loss: 0.467 |  Acc: 93.580% (9358/10000) | best_acc: 93.750

Training Setting: batchsize=128 | epoch=247 | lr=1.0e-05
Training: Epoch=247 | Loss: 0.000 |  Acc: 100.000% (50000/50000) | best acc: 93.750
Testing: Epoch=247 | Loss: 0.467 |  Acc: 93.590% (9359/10000) | best_acc: 93.750

Training Setting: batchsize=128 | epoch=248 | lr=1.0e-05
Training: Epoch=248 | Loss: 0.000 |  Acc: 99.994% (49997/50000) | best acc: 93.750
Testing: Epoch=248 | Loss: 0.466 |  Acc: 93.600% (9360/10000) | best_acc: 93.750

Training Setting: batchsize=128 | epoch=249 | lr=1.0e-05
Training: Epoch=249 | Loss: 0.001 |  Acc: 99.988% (49994/50000) | best acc: 93.750
Testing: Epoch=249 | Loss: 0.466 |  Acc: 93.590% (9359/10000) | best_acc: 93.750
```

And then, like task B3, i use the adversarial attack to verify its robustness.
The training log is saved in ./log folder, and the adversarial attack result is saved in the same folder.
The result is below:

```
without attack  Test Accuracy = 9375 / 10000 = 93.75%
Epsilon: 0      Test Accuracy = 5348 / 10000 = 53.48%
Epsilon: 0.05   Test Accuracy = 4372 / 10000 = 43.72%
Epsilon: 0.1    Test Accuracy = 3749 / 10000 = 37.49%
Epsilon: 0.15   Test Accuracy = 3312 / 10000 = 33.12%
Epsilon: 0.2    Test Accuracy = 2989 / 10000 = 29.89%
Epsilon: 0.25   Test Accuracy = 2752 / 10000 = 27.52%
Epsilon: 0.3    Test Accuracy = 2547 / 10000 = 25.47%
Epsilon: 0.5    Test Accuracy = 2027 / 10000 = 20.27%
Epsilon: 1.0    Test Accuracy = 1483 / 10000 = 14.83%
```

I train the PCN in 250 epochs, and adversarial attack epsilons in [0, .05, .1, .15, .2, .25, .3, .5, 1.]

Reference

1. Wen H, Han K, Shi J, et al. Deep predictive coding network for object recognition[C]//International Conference on Machine Learning. PMLR, 2018: 5266-5275.
2. a-new-kind-of-deep-neural-networks
3. PredNet
4. Chalasani R, Principe J C. Deep predictive coding networks[J]. arXiv preprint arXiv:1301.3541, 2013.