PyTorch Hook

Underrated Tool for Debugging and Modifying Deep Learning Model Blindly

Rian Adam Rajagede – PyCon ID 2023

Code available:

Rian Adam Rajagede

- **Ph.D. Student** at CECS, **University of Central Florida** working on Machine Learning Reliability
- **Lecturer** at Dept. informatics, **Universitas Islam Indonesia** **researcher** at **Center of Data Science, UII**

Website: https://structilmy.com

Konteks

Mau apa? Sharing pengalaman pakai PyTorch hook setahun terakhir

Konteks

Mau apa? Sharing pengalaman pakai PyTorch hook setahun terakhir

Itu apa? TLDR: Satu fitur/fungsi dari framework deep learning PyTorch

Konteks

Mau apa? Sharing pengalaman pakai PyTorch hook setahun terakhir

Itu apa? TLDR: Satu fitur/fungsi dari framework deep learning PyTorch

Kenapa perlu tahu? PyTorch hook dokumentasinya singkat dan tutorialnya masih sedikit. Saya merasakan manfaatnya ketika perlu "melapisi" model DL tanpa mau paham arsitekturnya, siapa tahu ada yang punya masalah sama

Konteks

Mau apa? Sharing pengalaman pakai PyTorch hook setahun terakhir

Itu apa? TLDR: Satu fitur/fungsi dari framework deep learning PyTorch

Kenapa perlu tahu? PyTorch hook dokumentasinya singkat dan tutorialnya masih sedikit. Saya merasakan manfaatnya ketika perlu "melapisi" model DL tanpa mau paham arsitekturnya, siapa tahu ada yang punya masalah sama

Saya *enggak* pakai PyTorch, *tuh*? Saya bakal kenalin sedikit tentang PyTorch dan coba jelasin semudah mungkin.

Semoga tetap ada hikmah yang bisa dipetik ya :)

Konteks

Mau apa? Sharing pengalaman pakai PyTorch hook setahun terakhir

Itu apa? TLDR: Satu fitur/fungsi dari framework deep learning PyTorch

Kenapa perlu tahu? PyTorch hook dokumentasinya singkat dan tutorialnya masih sedikit. Saya merasakan manfaatnya ketika perlu "melapisi" model DL tanpa mau paham arsitekturnya, siapa tahu ada yang punya masalah sama

Saya *enggak* pakai PyTorch, *tuh*? Saya bakal kenalin sedikit tentang PyTorch dan coba jelasin semudah mungkin.

Semoga tetap ada hikmah yang bisa dipetik ya :)

Agenda

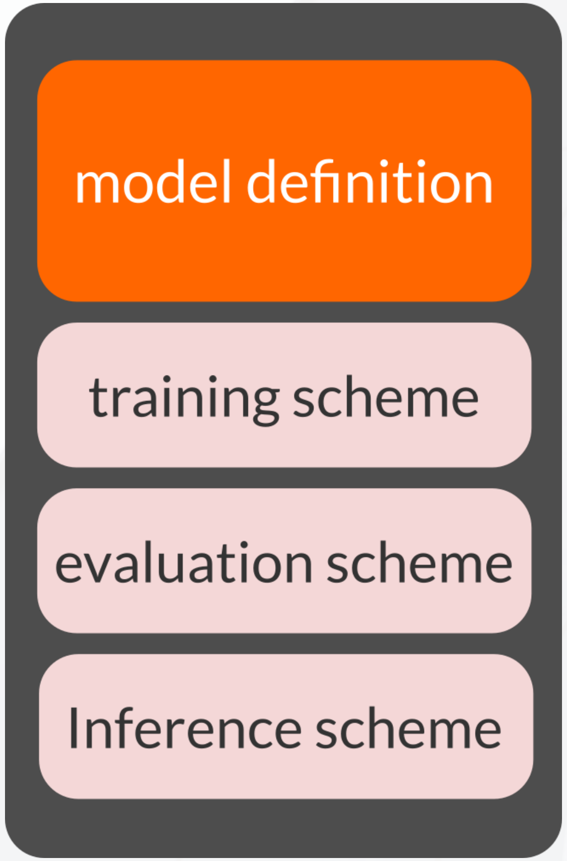
- **Sekilas PyTorch**
- **Mengulik model "blindly"**
- **PyTorch hook & studi kasus**

A Glimpse of PyTorch

Deep learning framework dari Meta.ai
Sekarang bagian dari Linux Foundation

Struktur

Secara umum, struktur kode PyTorch sama seperti Tensorflow/Keras*:



model definition

training scheme

evaluation scheme

Inference scheme

*tapi lebih banyak yang harus ditulis

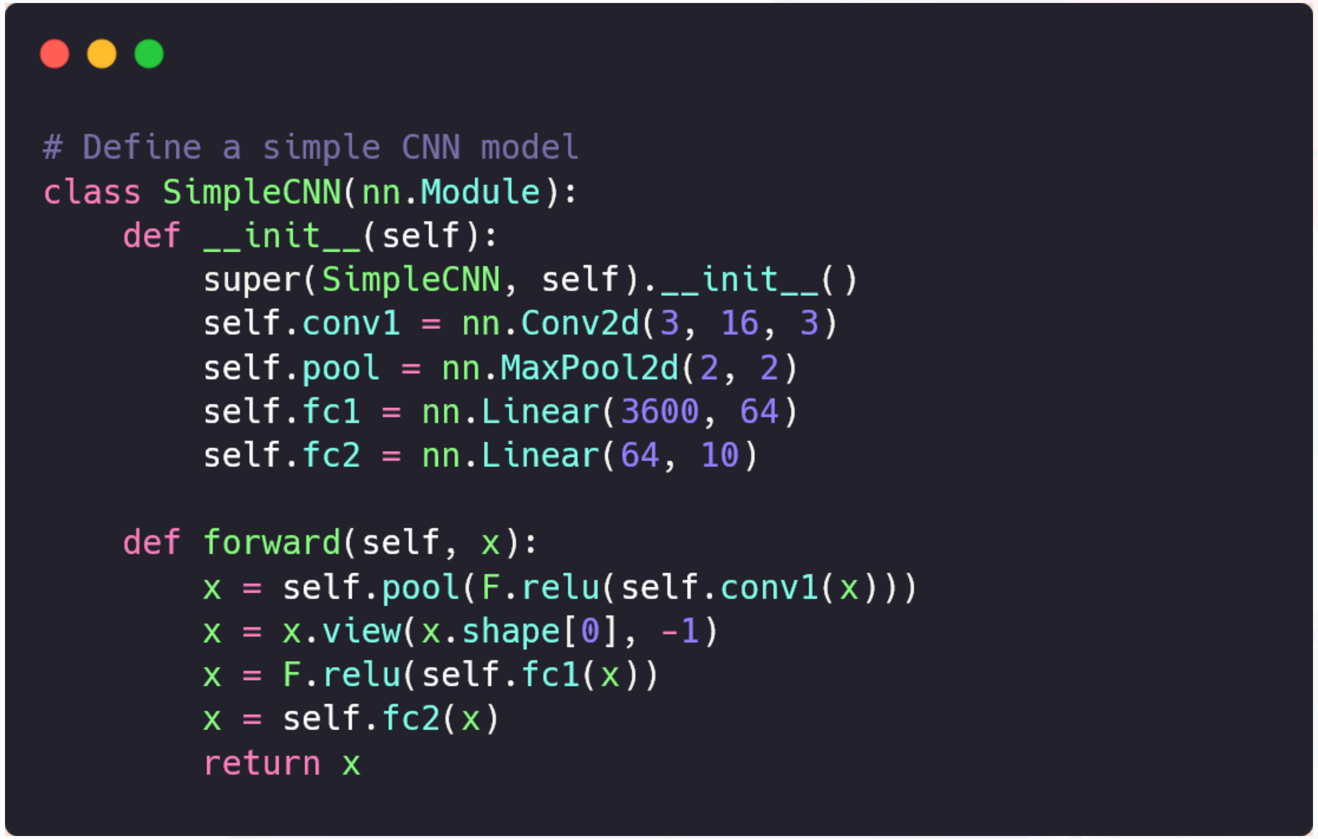
Mendefinisikan Model

- model definition
- training scheme
- evaluation scheme
- Inference scheme

Buat class model

Definisikan layer-layer

Definisikan forward propagation



```
# Define a simple CNN model
class SimpleCNN(nn.Module):
    def __init__(self):
        super(SimpleCNN, self).__init__()
        self.conv1 = nn.Conv2d(3, 16, 3)
        self.pool = nn.MaxPool2d(2, 2)
        self.fc1 = nn.Linear(3600, 64)
        self.fc2 = nn.Linear(64, 10)

    def forward(self, x):
        x = self.pool(F.relu(self.conv1(x)))
        x = x.view(x.shape[0], -1)
        x = F.relu(self.fc1(x))
        x = self.fc2(x)
        return x
```

Training Scheme

model definition

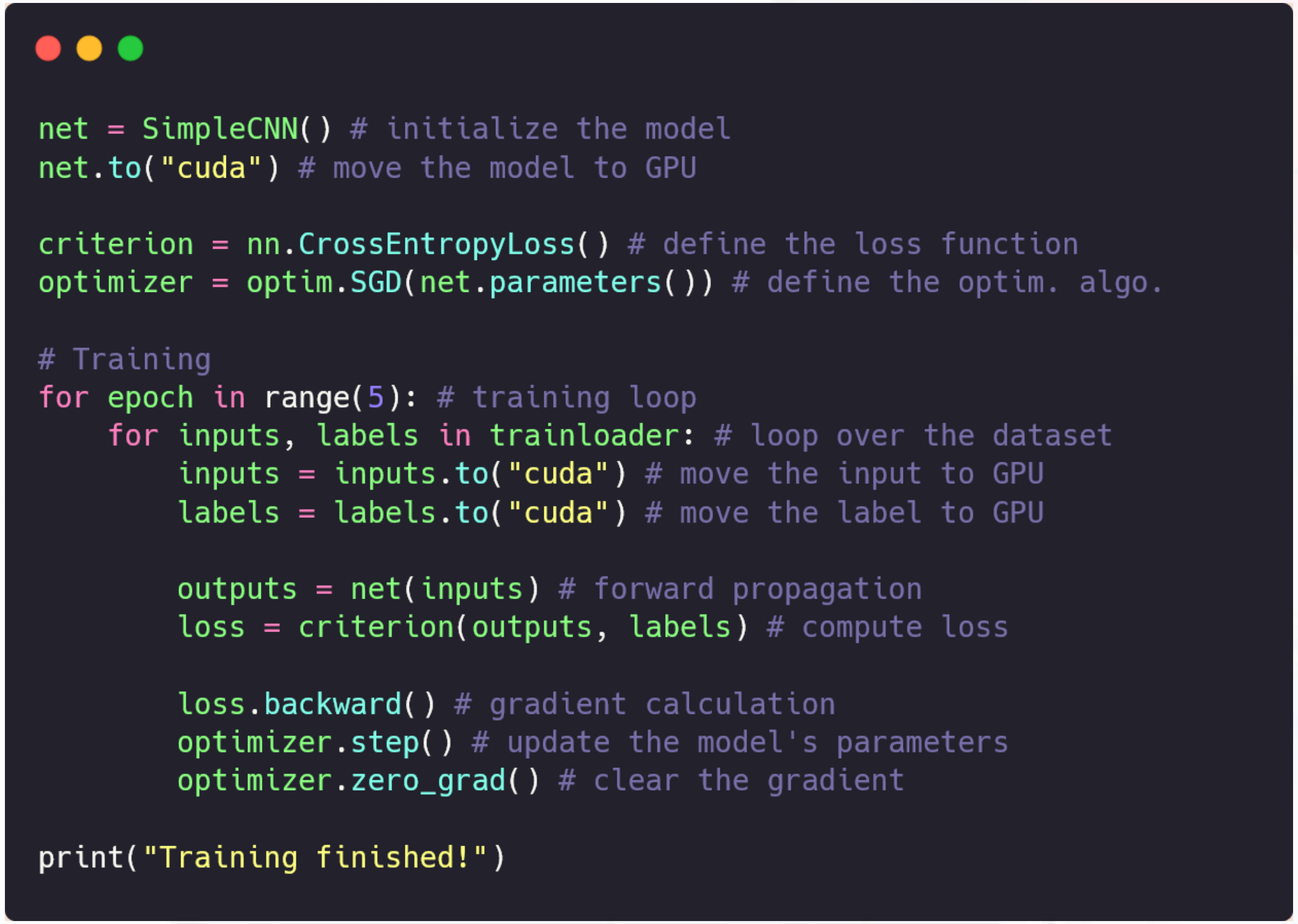
training scheme

evaluation scheme

Inference scheme

Initialize model,
optimizer,
loss function

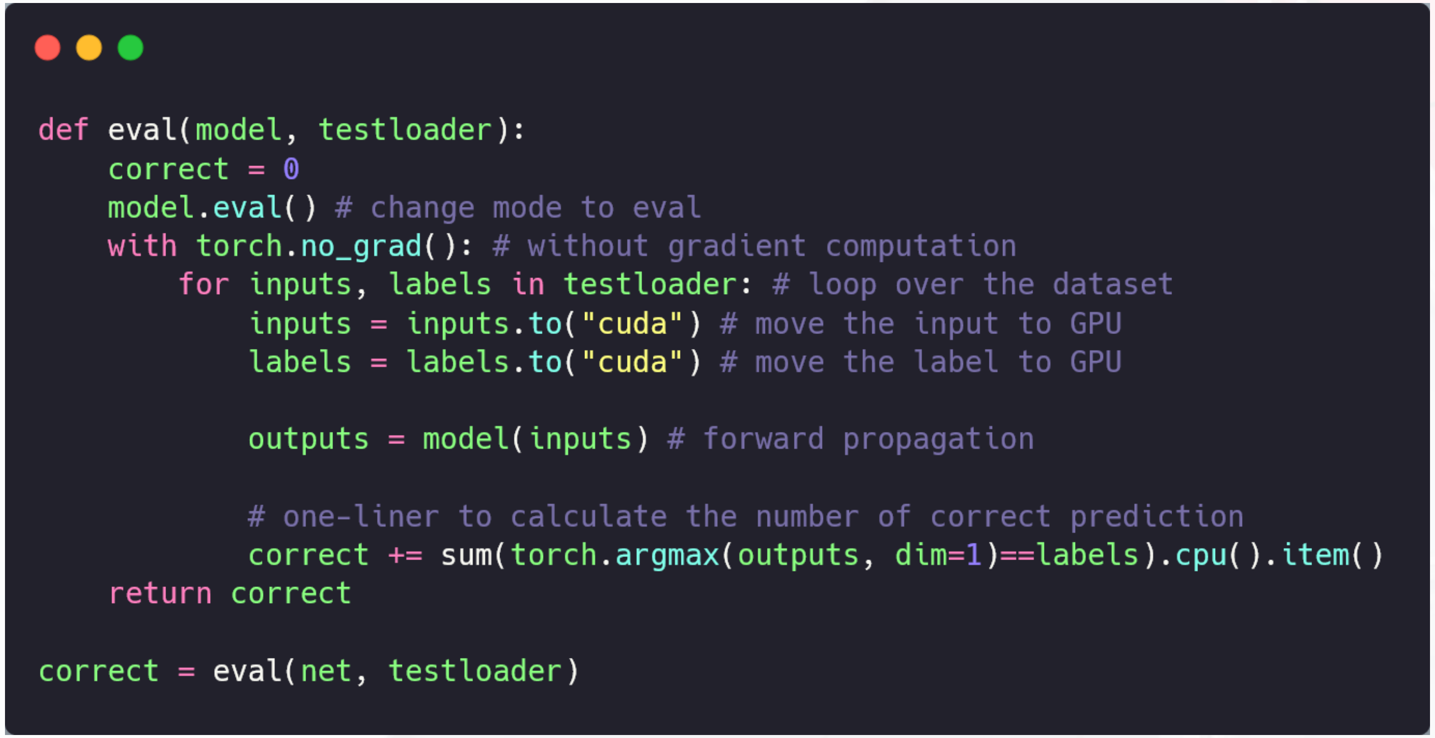
Training loop



```
net = SimpleCNN() # initialize the model
net.to("cuda") # move the model to GPU

criterion = nn.CrossEntropyLoss() # define the loss function
optimizer = optim.SGD(net.parameters()) # define the optim. algo.

# Training
for epoch in range(5): # training loop
    for inputs, labels in trainloader: # loop over the dataset
        inputs = inputs.to("cuda") # move the input to GPU
        labels = labels.to("cuda") # move the label to GPU

        outputs = net(inputs) # forward propagation
        loss = criterion(outputs, labels) # compute loss

        loss.backward() # gradient calculation
        optimizer.step() # update the model's parameters
        optimizer.zero_grad() # clear the gradient

print("Training finished!")
```

Evaluation Scheme

Sama seperti training, hanya saja tanpa sintaks-sintaks yang berhubungan dengan backward propagation / optimization



```python
def eval(model, testloader):
    correct = 0
    model.eval() # change mode to eval
    with torch.no_grad(): # without gradient computation
        for inputs, labels in testloader: # loop over the dataset
            inputs = inputs.to("cuda") # move the input to GPU
            labels = labels.to("cuda") # move the label to GPU

            outputs = model(inputs) # forward propagation

            # one-liner to calculate the number of correct prediction
            correct += sum(torch.argmax(outputs, dim=1)==labels).cpu().item()
    return correct

correct = eval(net, testloader)
```

Biasanya dibuat dalam bentuk fungsi biar enak kalau perlu panggil lagi

model definition

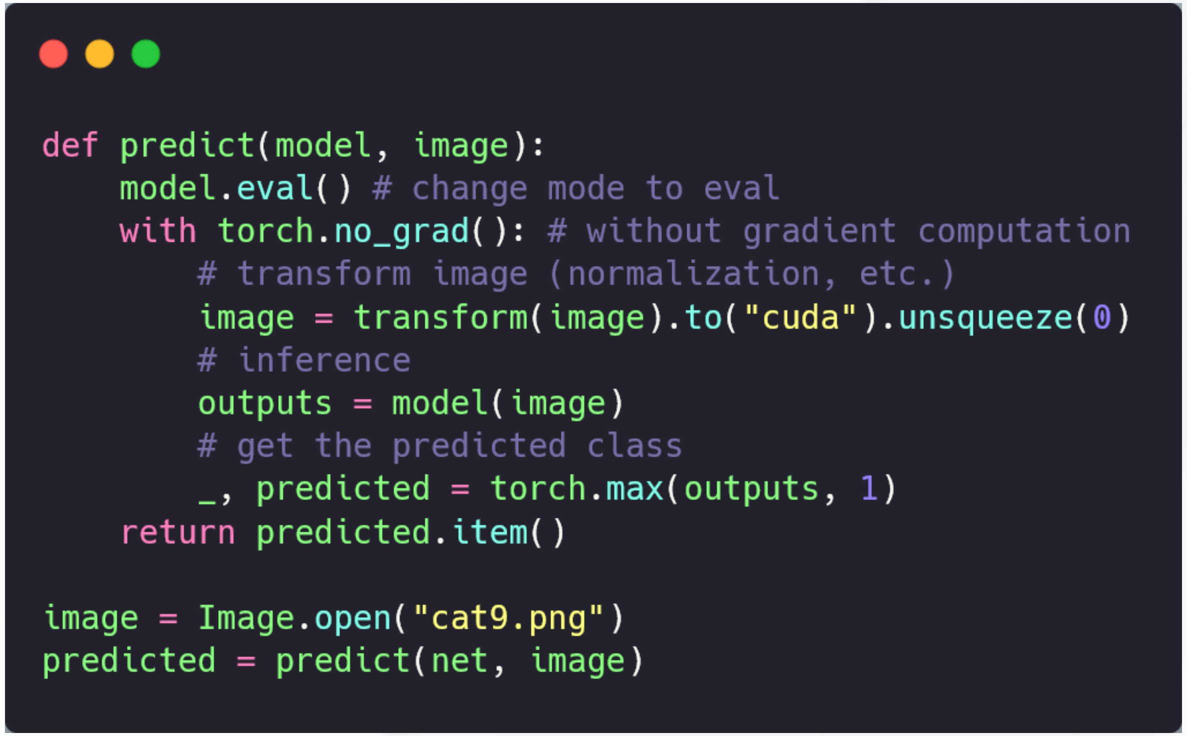
training scheme

evaluation scheme

Inference scheme

Inference Scheme

Sama seperti evaluasi, tapi hanya dari satu gambar raw



```python
def predict(model, image):
    model.eval() # change mode to eval
    with torch.no_grad(): # without gradient computation
        # transform image (normalization, etc.)
        image = transform(image).to("cuda").unsqueeze(0)
        # inference
        outputs = model(image)
        # get the predicted class
        _, predicted = torch.max(outputs, 1)
    return predicted.item()

image = Image.open("cat9.png")
predicted = predict(net, image)
```

model definition

training scheme

evaluation scheme

Inference scheme

Mengulik Model “Blindly”

Karena model semakin kompleks

Mendefinisikan Model

Karena model semakin kompleks, bagian mendefinisikan model dan juga proses training jadi semakin kompleks

model definition

training scheme

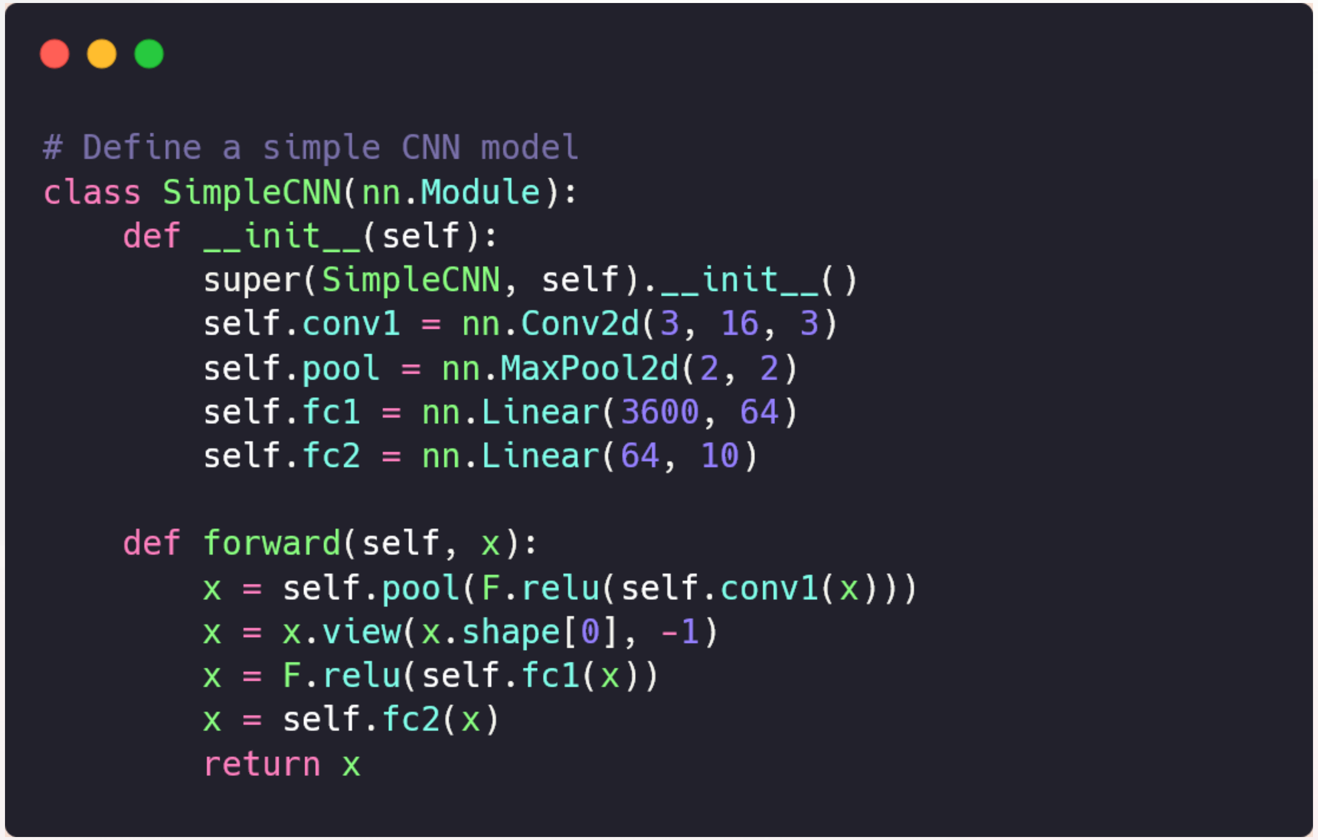
evaluation scheme

Inference scheme

Buat class model

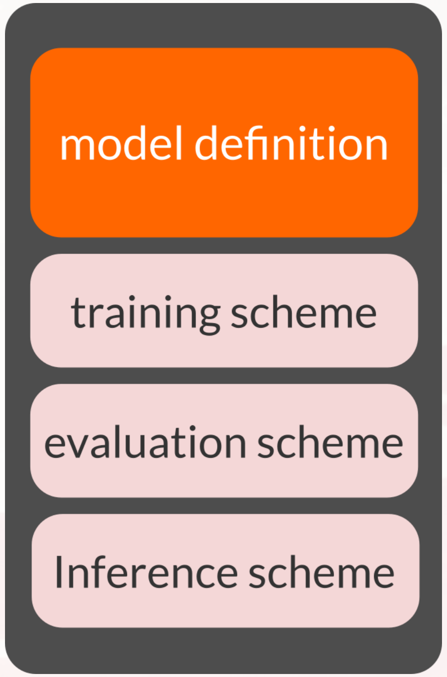
Definisikan layer-layer

Definisikan forward propagation



```
# Define a simple CNN model
class SimpleCNN(nn.Module):
    def __init__(self):
        super(SimpleCNN, self).__init__()
        self.conv1 = nn.Conv2d(3, 16, 3)
        self.pool = nn.MaxPool2d(2, 2)
        self.fc1 = nn.Linear(3600, 64)
        self.fc2 = nn.Linear(64, 10)

    def forward(self, x):
        x = self.pool(F.relu(self.conv1(x)))
        x = x.view(x.shape[0], -1)
        x = F.relu(self.fc1(x))
        x = self.fc2(x)
        return x
```

Mendefinisikan Model



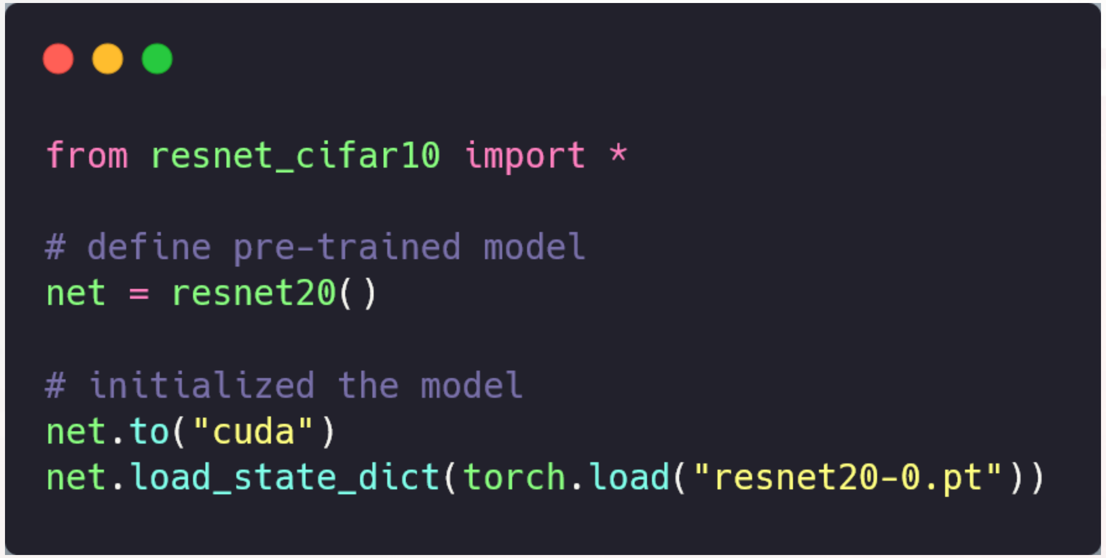
model definition

training scheme

evaluation scheme

Inference scheme

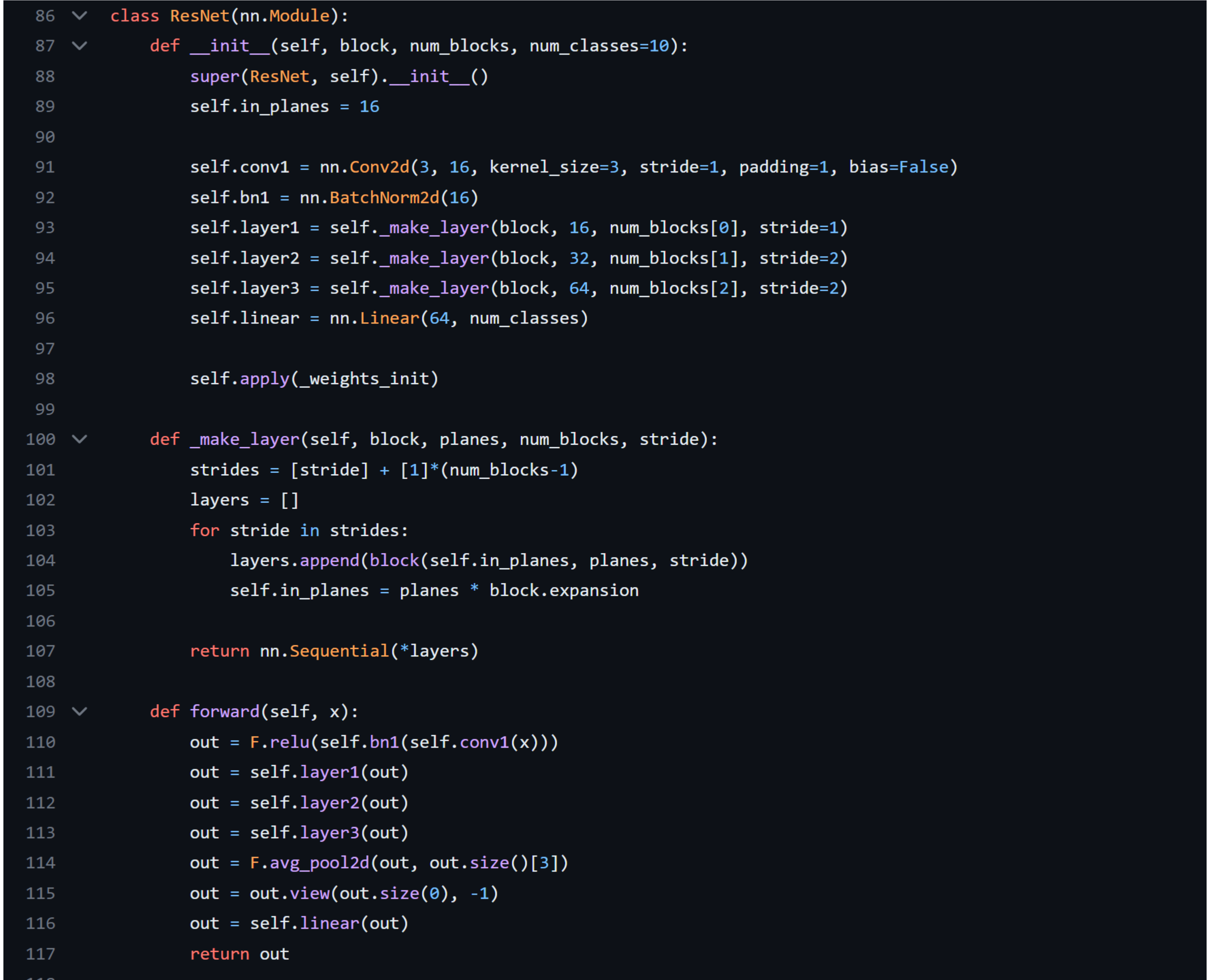
Solusi: Download model yang sudah jadi (pre-trained model)
- simple
- kualitas sudah bagus



```
from resnet_cifar10 import *

# define pre-trained model
net = resnet20()

# initialized the model
net.to("cuda")
net.load_state_dict(torch.load("resnet20-0.pt"))
```

Model ResNet20 bisa didownload di
https://github.com/akamaster/pytorch_resnet_cifar10

Sekilas dibalik `resnet20()`



```python
class ResNet(nn.Module):
    def __init__(self, block, num_blocks, num_classes=10):
        super(ResNet, self).__init__()
        self.in_planes = 16

        self.conv1 = nn.Conv2d(3, 16, kernel_size=3, stride=1, padding=1, bias=False)
        self.bn1 = nn.BatchNorm2d(16)
        self.layer1 = self._make_layer(block, 16, num_blocks[0], stride=1)
        self.layer2 = self._make_layer(block, 32, num_blocks[1], stride=2)
        self.layer3 = self._make_layer(block, 64, num_blocks[2], stride=2)
        self.linear = nn.Linear(64, num_classes)

        self.apply(_weights_init)

    def _make_layer(self, block, planes, num_blocks, stride):
        strides = [stride] + [1]*(num_blocks-1)
        layers = []
        for stride in strides:
            layers.append(block(self.in_planes, planes, stride))
            self.in_planes = planes * block.expansion

        return nn.Sequential(*layers)

    def forward(self, x):
        out = F.relu(self.bn1(self.conv1(x)))
        out = self.layer1(out)
        out = self.layer2(out)
        out = self.layer3(out)
        out = F.avg_pool2d(out, out.size()[3])
        out = out.view(out.size(0), -1)
        out = self.linear(out)
        return out
```

Gimana kalau kita perlu memodifikasi pre-trained model?

https://imgflip.co

Buat model sendiri tapi panjang ngodingnya
Pakai model orang tapi susah pahamnya
JAKE-CLARK.TUMBLR
imgflip.com

Gimana kalau kita perlu memodifikasi* pre-trained model?

Buat model sendiri tapi panjang ngodingnya
Pakai model orang tapi susah pahamnya
JAKE-CLARK.TUMBLR
imgflip.com

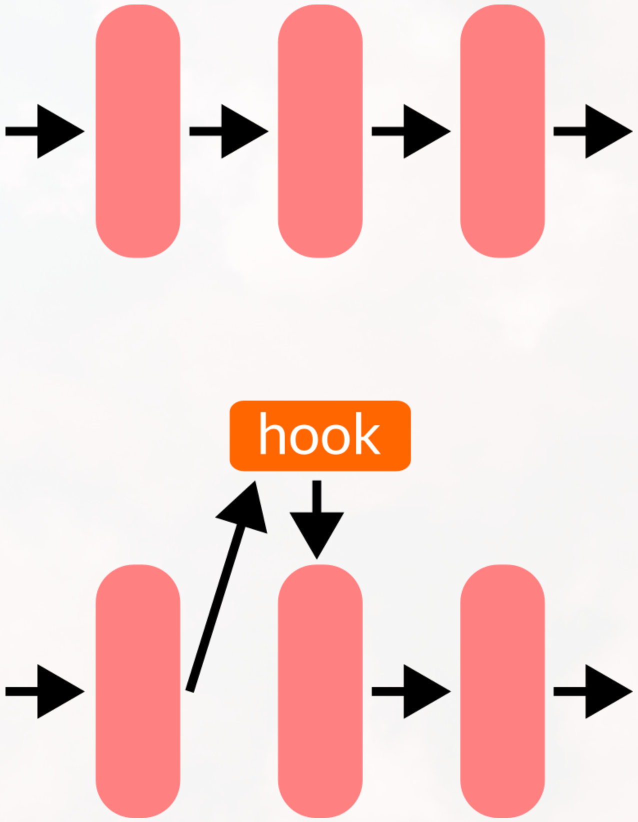
***Disclaimer:**
Memodifikasi di sini bisa jadi beda-beda tergantung kebutuhan. Solusi yang kita bahas bisa jadi tidak cocok

Introducing PyTorch Hook

Karena model semakin kompleks

PyTorch Hook

PyTorch Hook memungkinkan kita **untuk menaruh fungsi** di tengah-tengah model (di suatu modul).



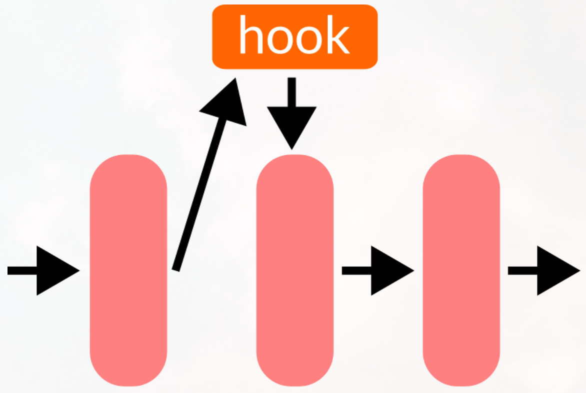
PyTorch Hook

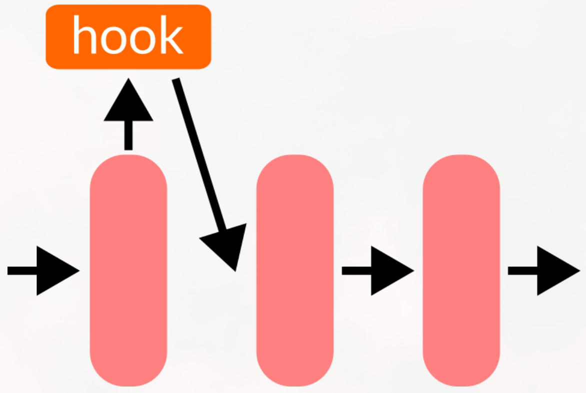
Saat ini ada 3 skema yang tersedia:

- **forward hook** : fungsi dijalankan setelah modul running

- **forward pre hook** : fungsi dijalankan sebelum modul running

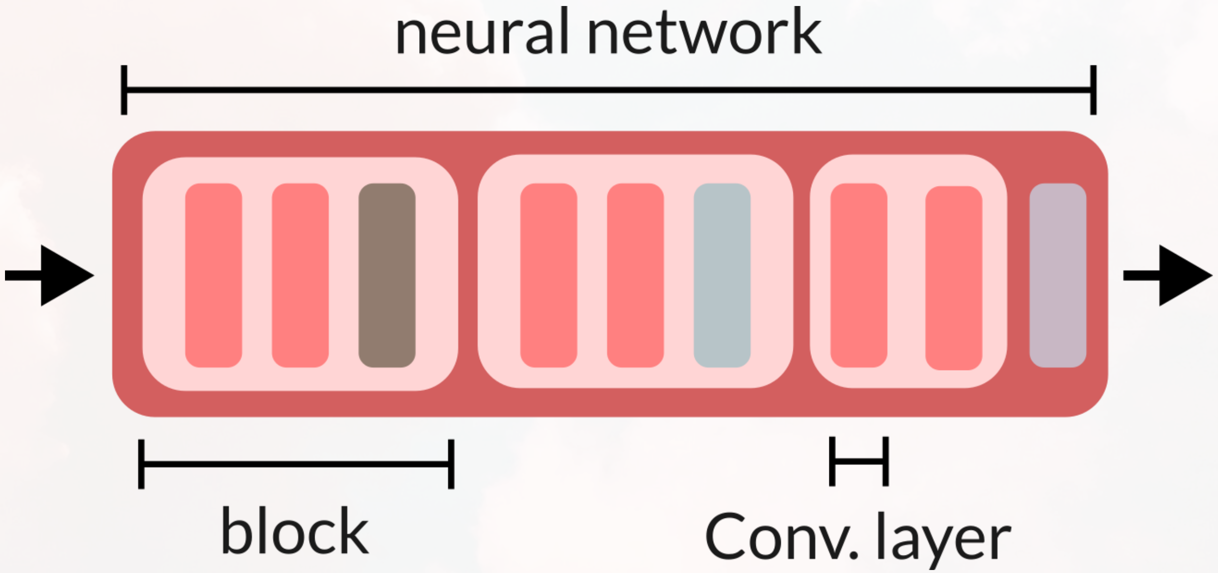
- **backward hook** : fungsi dijalankan saat proses backward propagation





“Modul” apa, *toh*?

Model Neural Network di PyTorch dibangun dari sekumpulan modul



PyTorch hook bisa dikaitkan di modul manapun

TKP

https:/ d-2023

https://github.com/rianrajagede/pyconid-2023

Thank You!

PyTorch Hook

Underrated Tool for Debugging and Modifying Deep Learning Model Blindly

Rian Adam Rajagede – PyCon ID 2023

Email: rian.adam@uii.ac.id

Code available: