Commands + Code + Text ▷ Run all

[1] ✓ 6s
```
!pip install timm -q
```

[2] ✓ 15s
```python
import torch
import torch.nn as nn
import torch.optim as optim
from torchvision.datasets import CIFAR10
from torch.utils.data import DataLoader, random_split
import torchvision.transforms as transforms
import timm
import numpy as np
from tqdm import tqdm
import matplotlib.pyplot as plt
import random
import os
```

[3] ✓ 0s
```python
DEVICE = "cuda" if torch.cuda.is_available() else "cpu"
torch.manual_seed(42)
np.random.seed(42)
random.seed(42)
```

Release notes

Please follow our blog to see more information about new features, tips and tricks, and featured notebooks such as Analyzing a Bank Failure with Colab.

2025-11-13

- **VS Code Extension**: Connect to Colab runtimes from VS Code (extension, repository).
- **"Notebook Info" Moved**: The "Notebook Info" command has been moved from the View dropdown menu to the File dropdown menu, aligning with other Workspace products.
- **Improved Error Handling**: Error messages when failing to create a Colab session have been improved.
- **Improved Accessibility**: Tooltips in the table of contents has been updated.
- **Add prompt cells**: It is now possible to add prompt cells within Colab (Blog).
- **OAuth flow**: Improved support and error handling during OAuth flow.
- **Open in Colab**: Released version 1.2.1

{} Variables ☐ Terminal

✓ 10:26 T4 (Python 3)

```python
train_tf = transforms.Compose([
    transforms.Resize(224),
    transforms.RandomHorizontalFlip(),
    transforms.RandomCrop(224, padding=16),
    transforms.ToTensor(),
    transforms.Normalize(
        [0.485, 0.456, 0.406],
        [0.229, 0.224, 0.225]
    )
])

test_tf = transforms.Compose([
    transforms.Resize(224),
    transforms.ToTensor(),
    transforms.Normalize(
        [0.485, 0.456, 0.406],
        [0.229, 0.224, 0.225]
    )
])
```

```python
dataset = CIFAR10("./data", train=True, download=True, transf
test_set = CIFAR10("./data", train=False, download=True, trar
```

```python
dataset  = CIFAR10("./data", train=True, download=True, transf
test_set = CIFAR10("./data", train=False, download=True, trar

train_set, val_set = random_split(
    dataset, [45000, 5000],
    generator=torch.Generator().manual_seed(42)
)


test_set = torch.utils.data.Subset(test_set, range(5000))
```

```
100%|██████████| 170M/170M [00:10<00:00, 15.6MB/s]
```

```python
train_loader = DataLoader(train_set, batch_size=128, shuffle=
val_loader   = DataLoader(val_set, batch_size=128)
test_loader  = DataLoader(test_set, batch_size=128)
```

```python
class SEBlock(nn.Module):
    def __init__(self, channels, reduction=16):
        super().__init__()
        self.fc = nn.Sequential(
            nn.Linear(channels, channels // reduction),
```

Release notes

Please follow our blog to see more information about new features, tips and tricks, and featured notebooks such as Analyzing a Bank Failure with Colab.

2025-11-13

- **VS Code Extension**: Connect to Colab runtimes from VS Code (extension, repository).
- **"Notebook Info" Moved:** The "Notebook Info" command has been moved from the View dropdown menu to the File dropdown menu, aligning with other Workspace products.
- **Improved Error Handling**: Error messages when failing to create a Colab session have been improved.
- **Improved Accessibility**: Tooltips in the table of contents has been updated.
- **Add prompt cells**: It is now possible to add prompt cells within Colab (Blog).
- **OAuth flow**: Improved support and error handling during OAuth flow.
- **Open in Colab**: Released version 1.2.1

```python
class SEBlock(nn.Module):
    def __init__(self, channels, reduction=16):
        super().__init__()
        self.fc = nn.Sequential(
            nn.Linear(channels, channels // reduction),
            nn.ReLU(),
            nn.Linear(channels // reduction, channels),
            nn.Sigmoid()
        )

    def forward(self, x):
        b, c, _, _ = x.size()
        y = x.mean(dim=[2, 3])
        y = self.fc(y).view(b, c, 1, 1)
        return x * y
```

```python
class ConvNeXtAttention(nn.Module):
    def __init__(self, num_classes=10):
        super().__init__()
        self.backbone = timm.create_model(
            "convnext_tiny",
            pretrained=True,
            features_only=True
        )
        self.attn = SEBlock(768)
```

Release notes

Please follow our blog to see more information about new features, tips and tricks, and featured notebooks such as Analyzing a Bank Failure with Colab.

2025-11-13

- **VS Code Extension**: Connect to Colab runtimes from VS Code (extension, repository).
- **"Notebook Info" Moved**: The "Notebook Info" command has been moved from the View dropdown menu to the File dropdown menu, aligning with other Workspace products.
- **Improved Error Handling**: Error messages when failing to create a Colab session have been improved.
- **Improved Accessibility**: Tooltips in the table of contents has been updated.
- **Add prompt cells**: It is now possible to add prompt cells within Colab (Blog).
- **OAuth flow**: Improved support and error handling during OAuth flow.
- **Open in Colab**: Released version 1.2.1

```
                            self.head = nn.Linear(768, num_classes)

    def forward(self, x):
        feat = self.backbone(x)[-1]
        feat = self.attn(feat)
        feat = self.pool(feat).flatten(1)
        return self.head(feat)
```

```
model = ConvNeXtAttention().to(DEVICE)
print("Parameters:", sum(p.numel() for p in model.parameters(
```

```
/usr/local/lib/python3.12/dist-packages/huggingface_hub/utils/
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in y
You will be able to reuse this secret in all of your notebooks
Please note that authentication is recommended but still optio
  warnings.warn(
model.safetensors: 100%          114M/114M [00:01<00:00, 130MB/s]

Parameters: 27 M
```

```
criterion = nn.CrossEntropyLoss(label_smoothing=0.05)
```

```
                    model.parameters(),
                    lr=3e-4,
                    weight_decay=0.05
)

scheduler = optim.lr_scheduler.CosineAnnealingLR(optimizer, T
```

```python
def run_epoch(model, loader, train=True):
    model.train() if train else model.eval()
    correct, total, loss_sum = 0, 0, 0

    with torch.set_grad_enabled(train):
        for x, y in tqdm(loader, leave=False):
            x, y = x.to(DEVICE), y.to(DEVICE)

            if train:
                optimizer.zero_grad()

            out = model(x)
            loss = criterion(out, y)

            if train:
                loss.backward()
                optimizer.step()
```

```
        loss_sum += loss.item()
        correct += (out.argmax(1) == y).sum().item()
        total += y.size(0)


    return loss_sum / len(loader), correct / total
```

```
EPOCHS = 15
best_val = 0
history = {"train": [], "val": []}

for ep in range(EPOCHS):
    _, tr_acc = run_epoch(model, train_loader, True)
    _, va_acc = run_epoch(model, val_loader, False)
    scheduler.step()

    history["train"].append(tr_acc)
    history["val"].append(va_acc)

    print(f"Epoch {ep+1}: Train={tr_acc:.4f}, Val={va_acc:.4f

    if va_acc > best_val:
        best_val = va_acc
        torch.save(model.state_dict(), "level3_best.pth")
```

```
993, Val=0.9744
996, Val=0.9742
999, Val=0.9746
998, Val=0.9730
998, Val=0.9740
998, Val=0.9742
964, Val=0.9622
                              Epoch 15: Train=0.9881, Val=0.9620
```

```python
model.load_state_dict(torch.load("level3_best.pth"))
_, test_acc = run_epoch(model, test_loader, False)

print("\nLEVEL-3 TEST ACCURACY:", round(test_acc * 100, 2), '
```

```
LEVEL-3 TEST ACCURACY: 97.1 %
```

```python
classes = ['airplane','automobile','bird','cat','deer',
           'dog','frog','horse','ship','truck']

class_correct = [0]*10
class_total = [0]*10

model.eval()
```

```
        class_total[y[i]] += 1
        class_correct[y[i]] += (preds[i]==y[i]).item()


for i in range(10):
    print(f"{classes[i]:12s}: {100*class_correct[i]/class_tot
```

```
airplane    : 97.54%
automobile  : 98.42%
bird        : 97.46%
cat         : 93.76%
deer        : 97.44%
dog         : 94.06%
frog        : 98.57%
horse       : 98.79%
ship        : 98.02%
truck       : 96.88%
```

```python
plt.plot([a*100 for a in history["train"]], label="Train")
plt.plot([a*100 for a in history["val"]], label="Val")
plt.axhline(y=91, linestyle="--", color="green", label="Targe
plt.xlabel("Epoch")
plt.ylabel("Accuracy (%)")
plt.title("Level-3 Training Curve")
plt.legend()
plt.grid(alpha=0.3)
```

```
[17]   plt.title( Level-3 Training Curve )
✓ 0s   plt.legend()
       plt.grid(alpha=0.3)
       plt.show()
```


Level-3 Training Curve

Legend: Train, Val, Target

Release notes   ⋮   ✕

Please follow our blog to see more information about new features, tips and tricks, and featured notebooks such as Analyzing a Bank Failure with Colab.
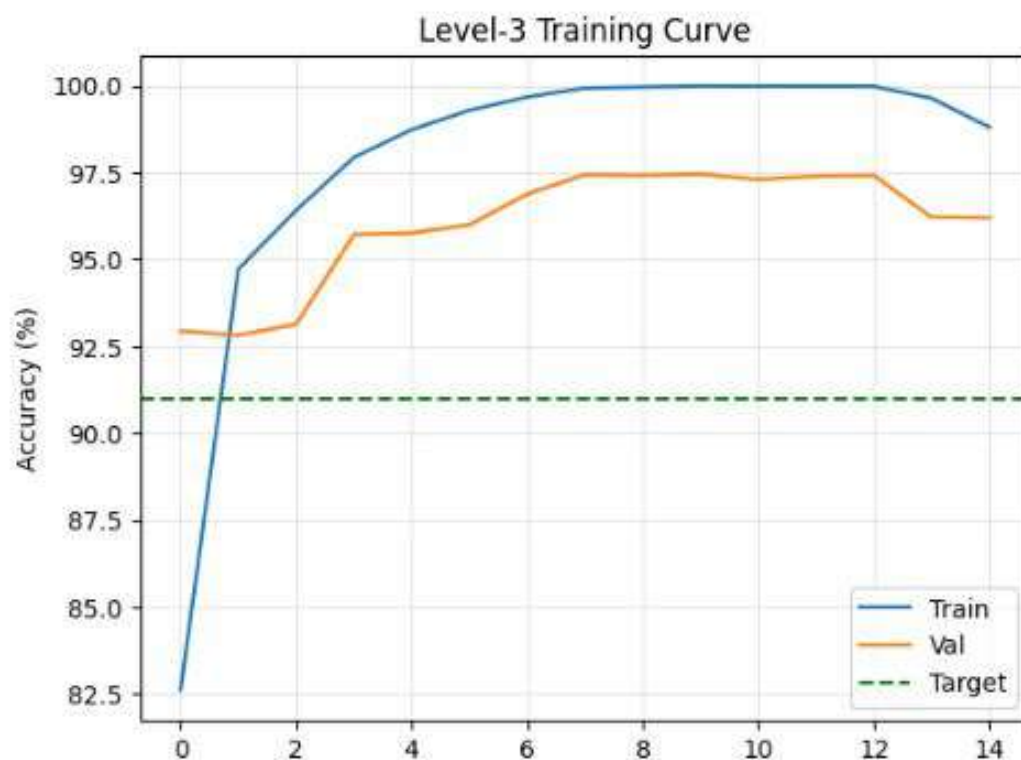
2025-11-13

- **VS Code Extension**: Connect to Colab runtimes from VS Code (extension, repository).
- **"Notebook Info" Moved:** The "Notebook Info" command has been moved from the View dropdown menu to the File dropdown menu, aligning with other Workspace products.
- **Improved Error Handling**: Error messages when failing to create a Colab session have been improved.
- **Improved Accessibility**: Tooltips in the table of contents has been updated.
- **Add prompt cells**: It is now possible to add prompt cells within Colab (Blog).
- **OAuth flow**: Improved support and error handling during OAuth flow

Level-3 Training Curve