(4) define the LSTM model;

(a) input layer.

(b) LSTM layers(s)

(c) dense layers.

(5) compile the model

(a) Loss function : mean squared error

(b) optimiser adam.

(6) Train the model on the training data.

(7) predict on the test data

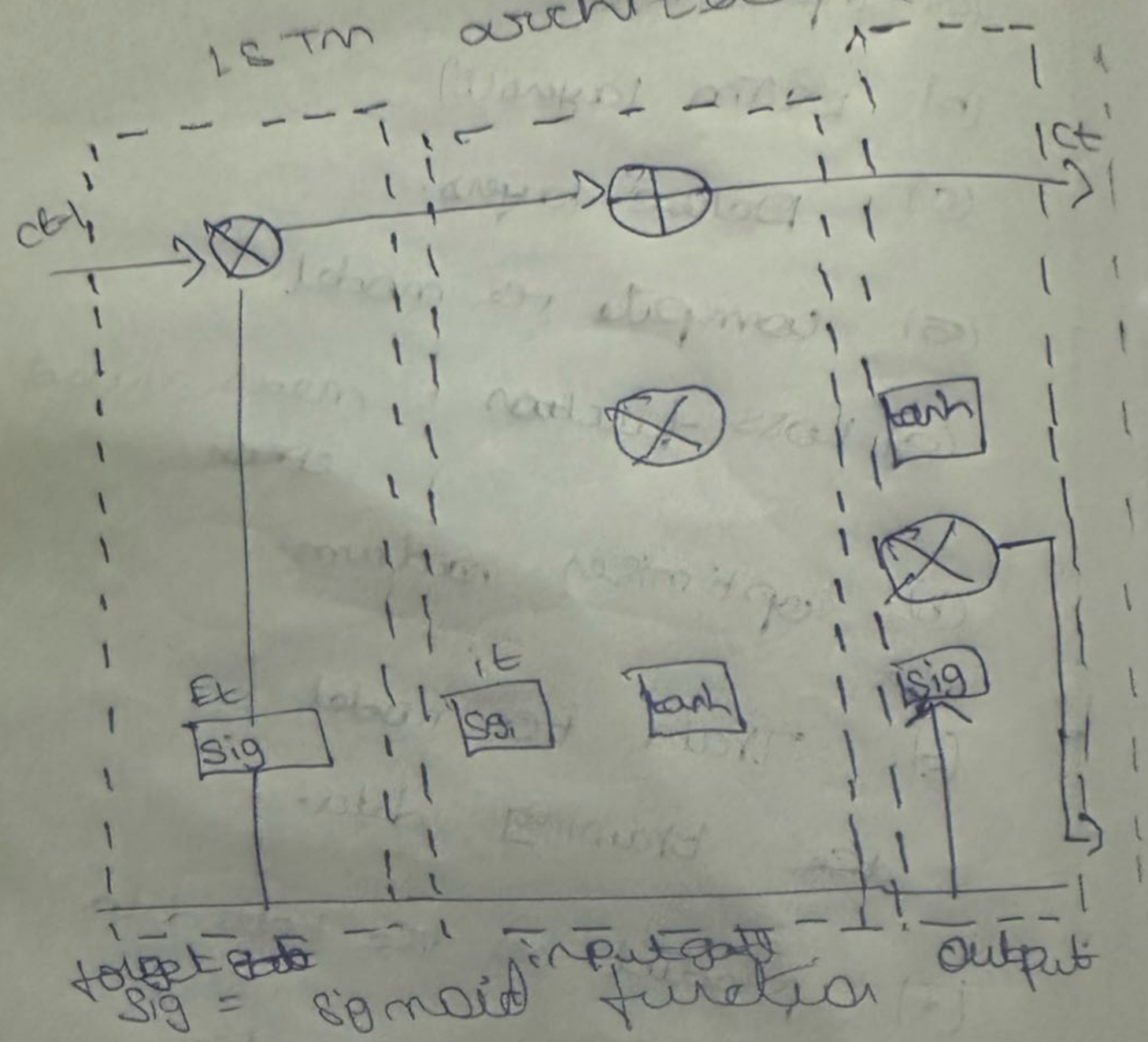
(8) Inverse transform the predicted values

(9) evaluate the models

(a) Plot actual vs predicted values.

(b) calculate RMSE or MAE.

LSTM architecture



Sig = sigmoid function

tanh = tanh function

⊗ = Point by point multiplication

⊕ = Point by point addition

→ = vector connection

Observation

(1) initial Performance

when the model was trained with a small no of epochs, the LSTM show basic learning capability but had relatively higher error values

(2) improved Accuracy with more training:

increasing the no of epochs to 100 and 200 significantly improved the model's performance.

(3) signs of overfitting

At 300 epochs, the improvement in performance on the test data slightly worsened.

(4) model stability

overall, the LSTM model was stable and efficitive for time series prediction.

Ex 9

network

Aim

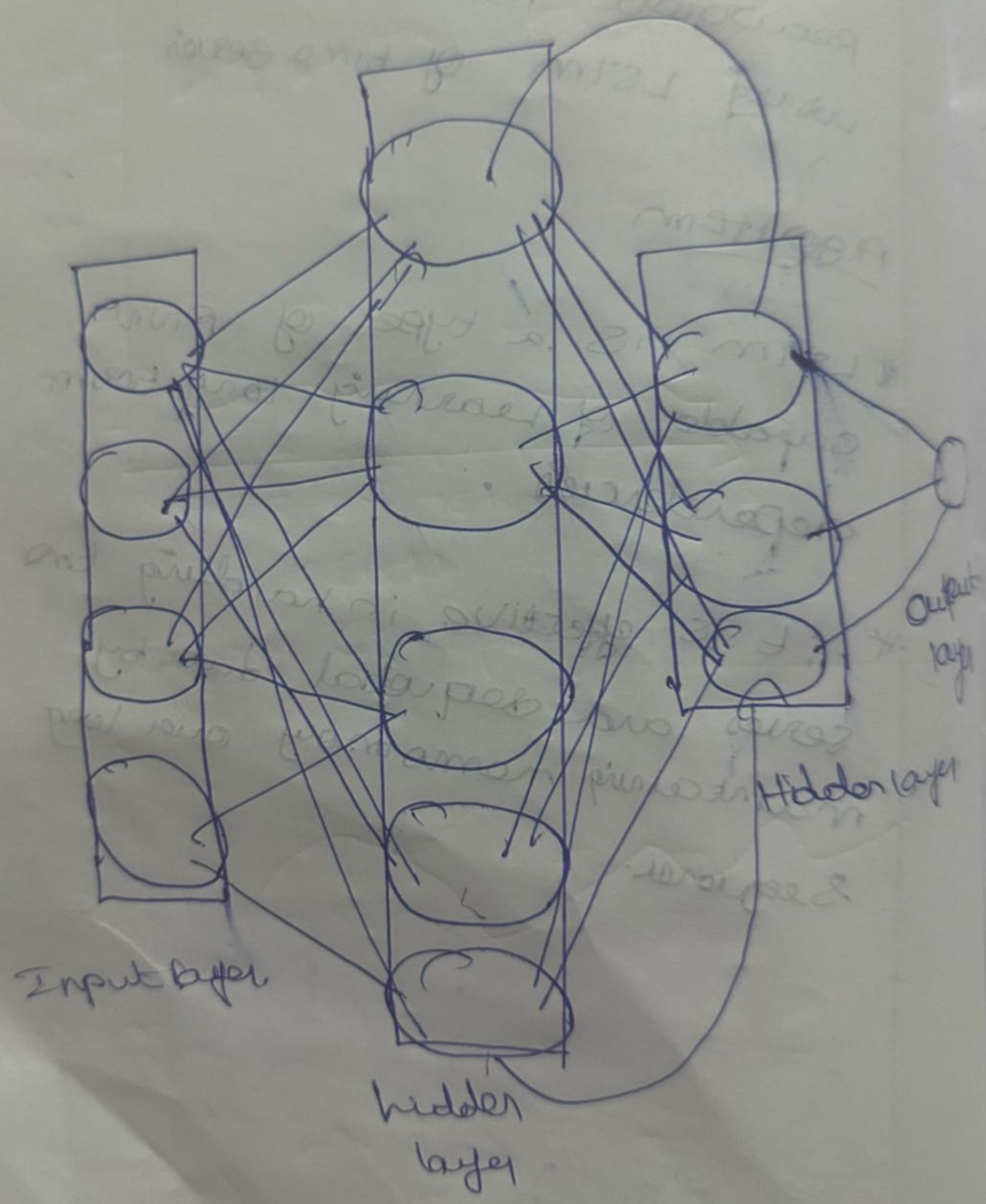
To build and implement a recurrent neural network using LSTM of time series

Algorithm

* LSTM is a type of RNN capable of learning long term dependencies.

* it is effective in handling time series and sequential data by maintaining memory over long sequence.

RNN architecture



pseudo code

import necessarily libraries (numpy, Pandas, keras)

load or generate time series data

(a) Normalize the data using minmaxscaler.

(b) convert data into input output sequence using sliding windows

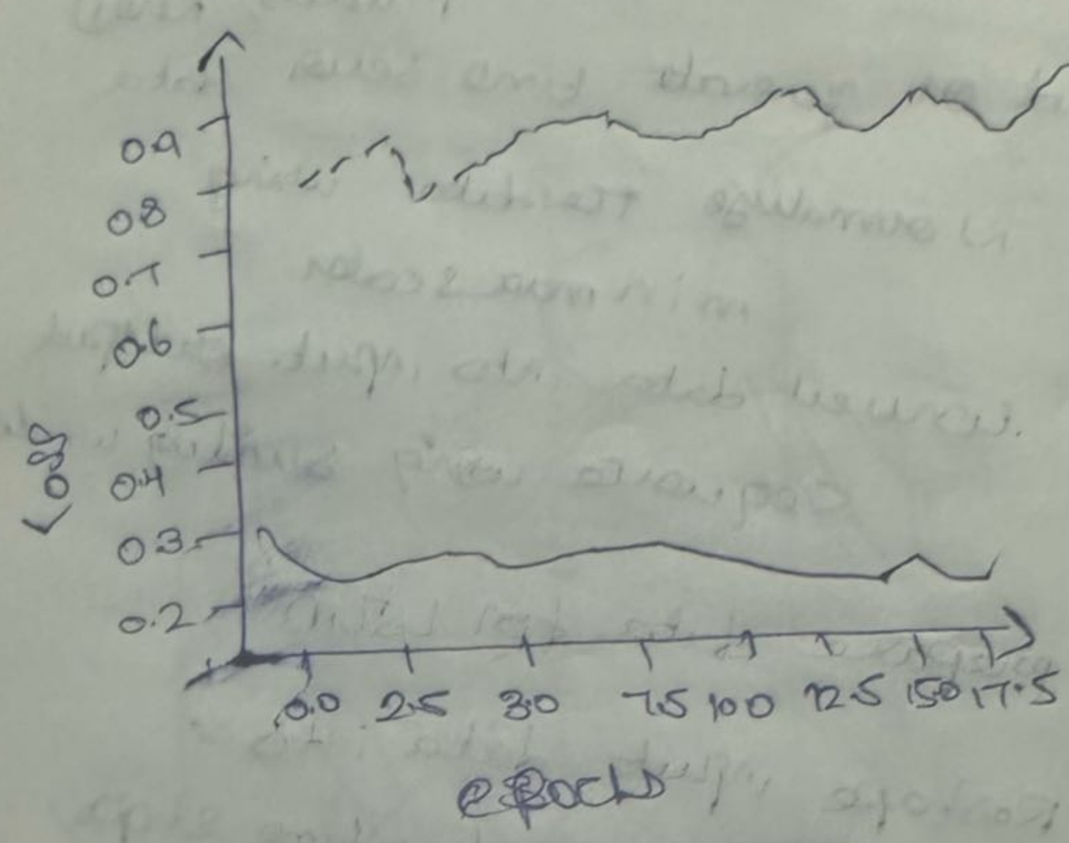
(3) prepare data for LSTM

(a) Reshape input data into [sample, time steps, features].

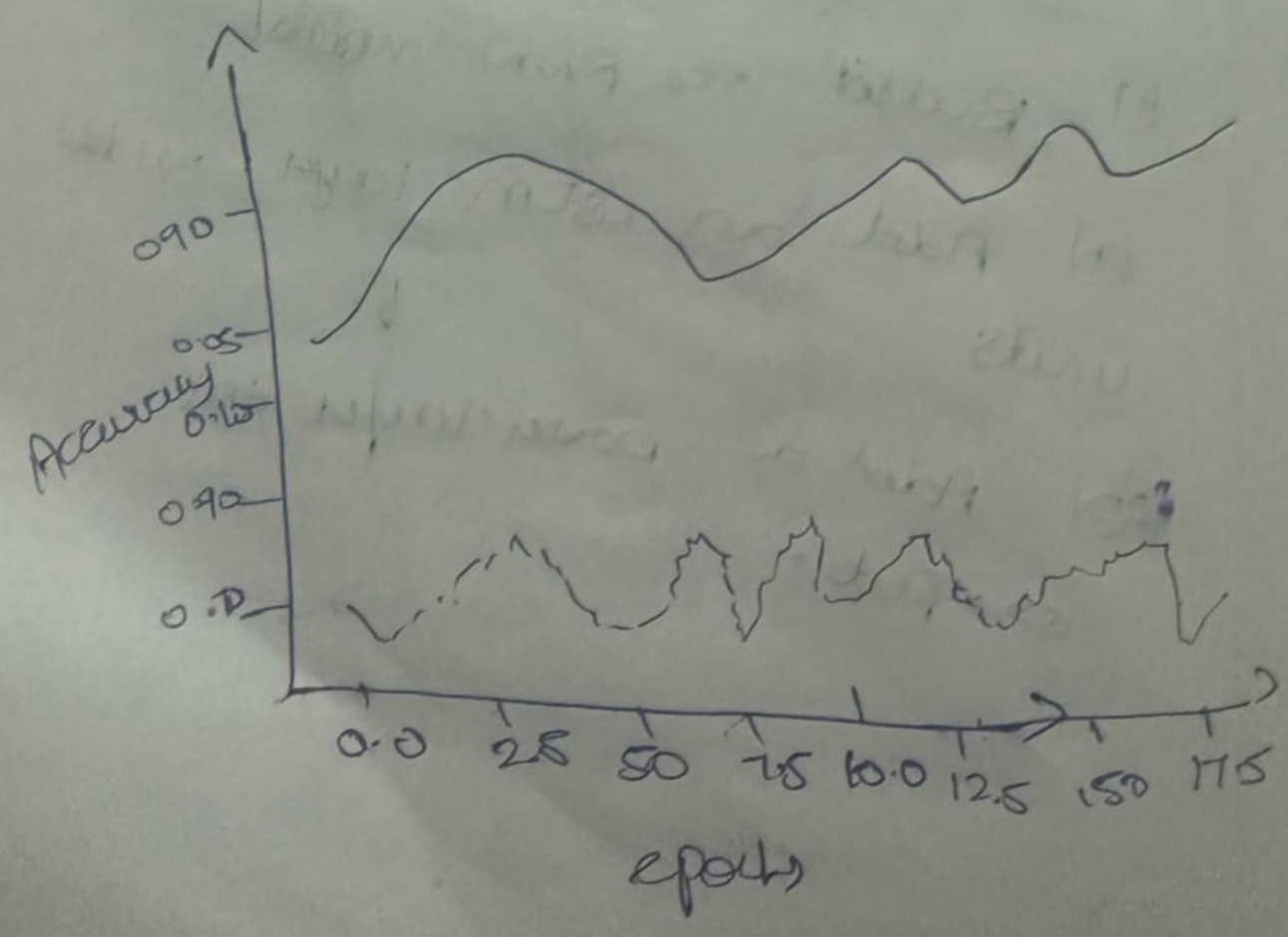
(4) Build the RNN model.

(a) Add an LSTM layer with units

(b) Add a Dense layer for output.



Accuracy vs epochs



(6) data for no of epochs.

(7) make prediction on the input data

(a) inverse transform prediction if normalized

(8) visualize actual vs predicted value

(9) evaluate model performance

observation

(1) The LSTM model learned the sequence pattern the time series data effectively after several epochs of training

(2) During initial epochs, the model showed higher error due to limited learning.

(3) The model was able to follow the trend of the actual values, indicating that the LSTM layer captured the temporal dependencies well.

(4) overfitting was not observed in this case due to a simple data set.

[5] Visualization of predicted vs actual values showed that the model was making accurate

Result

The RNN model using LSTM successfully

pseudo code:

(1) import libraries

(2) write model architecture layer wise

(3) pass a sample image through a model

(4) visualize the feature map

5) observe how filters detect low to high level features

Observation

→ The initial convolution layers detect edges and

pseudo code:

(1) import libraries

(2) print model architecture layer wise

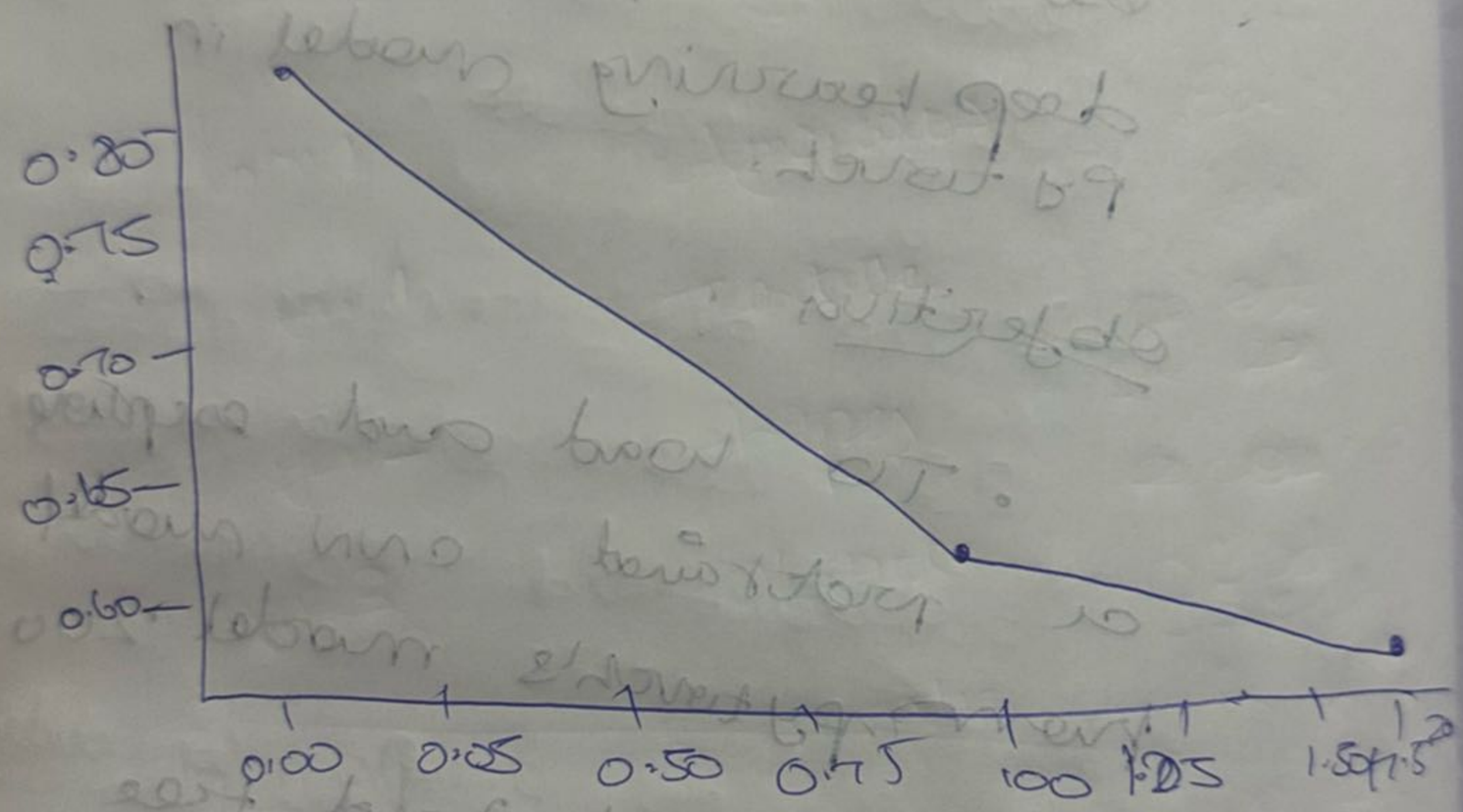
(3) pass a sample image through a model

(4) visualize the feature map.

5) observe how filters detect low to high level features

Observation

→ The initial convolution layers detect edges and color gradient.


0.80
0.75
0.70
0.65
0.60
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75

→ Deeper layers capture complex features like shapes and textures

→ The hierarchical structure help the network learn them, how leads to demonstrate

→ pre trained models like Resnet 18 are rich hierarchical feature extractor

Result

A pre trained Resnet 18 model was successfully used and used to

epoch

	feature type	Description
layer 1	edges, colours	Basic visual feas
layer 1	Texture	patterns stripes
layer 2	shapes	recognize
layer 3	object	Detect object
layer 4	semantic	high level context

automated

```python
# CELL 1: confirm GPU
import torch
print("torch:", torch.__version__)
print("cuda available:", torch.cuda.is_available())
if torch.cuda.is_available():
    print("device:", torch.cuda.get_device_name(0))
```

```
torch: 2.9.0+cu128
cuda available: True
device: Tesla T4
```

```python
# CELL 2: install dependencies
# You can skip if already satisfied, but safe to run
!pip install --quiet torch torchvision matplotlib

# Colab cell (Code)
!pip install --upgrade torch torchvision matplotlib gdown
```

```
Requirement already satisfied: torch in
/usr/local/lib/python3.12/dist-packages (2.9.0)
Requirement already satisfied: torchvision in
/usr/local/lib/python3.12/dist-packages (0.24.0)
Requirement already satisfied: matplotlib in
/usr/local/lib/python3.12/dist-packages (3.10.7)
Requirement already satisfied: gdown in
/usr/local/lib/python3.12/dist-packages (5.2.0)
Requirement already satisfied: filelock in
/usr/local/lib/python3.12/dist-packages (from torch) (3.20.0)
Requirement already satisfied: typing-extensions>=4.10.0 in
/usr/local/lib/python3.12/dist-packages (from torch) (4.15.0)
Requirement already satisfied: setuptools in
/usr/local/lib/python3.12/dist-packages (from torch) (75.2.0)
Requirement already satisfied: sympy>=1.13.3 in
/usr/local/lib/python3.12/dist-packages (from torch) (1.13.3)
Requirement already satisfied: networkx>=2.5.1 in
/usr/local/lib/python3.12/dist-packages (from torch) (3.5)
Requirement already satisfied: jinja2 in
/usr/local/lib/python3.12/dist-packages (from torch) (3.1.6)
Requirement already satisfied: fsspec>=0.8.5 in
/usr/local/lib/python3.12/dist-packages (from torch) (2025.3.0)
Requirement already satisfied: nvidia-cuda-nvrtc-cu12==12.8.93 in
/usr/local/lib/python3.12/dist-packages (from torch) (12.8.93)
Requirement already satisfied: nvidia-cuda-runtime-cu12==12.8.90 in
/usr/local/lib/python3.12/dist-packages (from torch) (12.8.90)
Requirement already satisfied: nvidia-cuda-cupti-cu12==12.8.90 in
/usr/local/lib/python3.12/dist-packages (from torch) (12.8.90)
Requirement already satisfied: nvidia-cudnn-cu12==9.10.2.21 in
/usr/local/lib/python3.12/dist-packages (from torch) (9.10.2.21)
Requirement already satisfied: nvidia-cublas-cu12==12.8.4.1 in
/usr/local/lib/python3.12/dist-packages (from torch) (12.8.4.1)
```

```
Requirement already satisfied: nvidia-cufft-cu12==11.3.3.83 in
/usr/local/lib/python3.12/dist-packages (from torch) (11.3.3.83)
Requirement already satisfied: nvidia-curand-cu12==10.3.9.90 in
/usr/local/lib/python3.12/dist-packages (from torch) (10.3.9.90)
Requirement already satisfied: nvidia-cusolver-cu12==11.7.3.90 in
/usr/local/lib/python3.12/dist-packages (from torch) (11.7.3.90)
Requirement already satisfied: nvidia-cusparse-cu12==12.5.8.93 in
/usr/local/lib/python3.12/dist-packages (from torch) (12.5.8.93)
Requirement already satisfied: nvidia-cusparselt-cu12==0.7.1 in
/usr/local/lib/python3.12/dist-packages (from torch) (0.7.1)
Requirement already satisfied: nvidia-nccl-cu12==2.27.5 in
/usr/local/lib/python3.12/dist-packages (from torch) (2.27.5)
Requirement already satisfied: nvidia-nvshmem-cu12==3.3.20 in
/usr/local/lib/python3.12/dist-packages (from torch) (3.3.20)
Requirement already satisfied: nvidia-nvtx-cu12==12.8.90 in
/usr/local/lib/python3.12/dist-packages (from torch) (12.8.90)
Requirement already satisfied: nvidia-nvjitlink-cu12==12.8.93 in
/usr/local/lib/python3.12/dist-packages (from torch) (12.8.93)
Requirement already satisfied: nvidia-cufile-cu12==1.13.1.3 in
/usr/local/lib/python3.12/dist-packages (from torch) (1.13.1.3)
Requirement already satisfied: triton==3.5.0 in
/usr/local/lib/python3.12/dist-packages (from torch) (3.5.0)
Requirement already satisfied: numpy in
/usr/local/lib/python3.12/dist-packages (from torchvision) (2.0.2)
Requirement already satisfied: pillow!=8.3.*,>=5.3.0 in
/usr/local/lib/python3.12/dist-packages (from torchvision) (11.3.0)
Requirement already satisfied: contourpy>=1.0.1 in
/usr/local/lib/python3.12/dist-packages (from matplotlib) (1.3.3)
Requirement already satisfied: cycler>=0.10 in
/usr/local/lib/python3.12/dist-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in
/usr/local/lib/python3.12/dist-packages (from matplotlib) (4.60.1)
Requirement already satisfied: kiwisolver>=1.3.1 in
/usr/local/lib/python3.12/dist-packages (from matplotlib) (1.4.9)
Requirement already satisfied: packaging>=20.0 in
/usr/local/lib/python3.12/dist-packages (from matplotlib) (25.0)
Requirement already satisfied: pyparsing>=3 in
/usr/local/lib/python3.12/dist-packages (from matplotlib) (3.2.5)
Requirement already satisfied: python-dateutil>=2.7 in
/usr/local/lib/python3.12/dist-packages (from matplotlib)
(2.9.0.post0)
Requirement already satisfied: beautifulsoup4 in
/usr/local/lib/python3.12/dist-packages (from gdown) (4.13.5)
Requirement already satisfied: requests[socks] in
/usr/local/lib/python3.12/dist-packages (from gdown) (2.32.4)
Requirement already satisfied: tqdm in /usr/local/lib/python3.12/dist-
packages (from gdown) (4.67.1)
Requirement already satisfied: six>=1.5 in
/usr/local/lib/python3.12/dist-packages (from python-dateutil>=2.7-
```

```
>matplotlib) (1.17.0)
Requirement already satisfied: mpmath<1.4,>=1.1.0 in
/usr/local/lib/python3.12/dist-packages (from sympy>=1.13.3->torch)
(1.3.0)
Requirement already satisfied: soupsieve>1.2 in
/usr/local/lib/python3.12/dist-packages (from beautifulsoup4->gdown)
(2.8)
Requirement already satisfied: MarkupSafe>=2.0 in
/usr/local/lib/python3.12/dist-packages (from jinja2->torch) (3.0.3)
Requirement already satisfied: charset_normalizer<4,>=2 in
/usr/local/lib/python3.12/dist-packages (from requests[socks]->gdown)
(3.4.4)
Requirement already satisfied: idna<4,>=2.5 in
/usr/local/lib/python3.12/dist-packages (from requests[socks]->gdown)
(3.11)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.12/dist-packages (from requests[socks]->gdown)
(2.5.0)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.12/dist-packages (from requests[socks]->gdown)
(2025.10.5)
Requirement already satisfied: PySocks!=1.5.7,>=1.5.6 in
/usr/local/lib/python3.12/dist-packages (from requests[socks]->gdown)
(1.7.1)
```

```python
import torch
import torch.nn as nn
import torch.optim as optim
import torchvision
import torchvision.datasets as dset
import torchvision.transforms as transforms
import torchvision.utils as vutils
import matplotlib.pyplot as plt
import numpy as np

# Root directory for dataset
dataroot = "./data"

# Number of workers for dataloader
workers = 2

# Batch size
batch_size = 128

# Image size (64x64 color images)
image_size = 64

# Number of channels (RGB = 3)
nc = 3
```

```python
# Size of latent vector (z)
nz = 100

# Size of feature maps in generator
ngf = 64

# Size of feature maps in discriminator
ndf = 64

# Number of training epochs
num_epochs = 25

# Learning rate
lr = 0.0001

# Beta1 hyperparam for Adam optimizer
beta1 = 0.5

# Device
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
```

```
!mkdir -p ./data
!wget -c
https://www.dropbox.com/s/8iq1zqg4x9jqk2v/img_align_celeba.zip?dl=1 -O
./data/img_align_celeba.zip
!unzip -qq ./data/img_align_celeba.zip -d ./data/celeba
```

```
--2025-10-26 15:53:59--
https://www.dropbox.com/s/8iq1zqg4x9jqk2v/img_align_celeba.zip?dl=1
Resolving www.dropbox.com (www.dropbox.com)... 162.125.5.18,
2620:100:601d:18::a27d:512
Connecting to www.dropbox.com (www.dropbox.com)|162.125.5.18|:443...
connected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [text/html]
Saving to: ‘./data/img_align_celeba.zip’

           ./data/im     [<=>                 ]       0  --.-KB/s
./data/img_align_ce     [ <=>                ]  86.44K  --.-KB/s    in
0.05s

2025-10-26 15:53:59 (1.74 MB/s) - ‘./data/img_align_celeba.zip’ saved
[88510]

[./data/img_align_celeba.zip]
  End-of-central-directory signature not found.  Either this file is
not
  a zipfile, or it constitutes one disk of a multi-part archive.  In
the
  latter case the central directory and zipfile comment will be found
on
```

10:08 5G

vertopal.com_LAB12.pdf Done

```
Length: unspecified [text/html]
Saving to: ‘./data/img_align_celeba.zip’

           ./data/im     [<=>                 ]       0  --.-KB/s
./data/img_align_ce     [ <=>                ]  86.44K  --.-KB/s    in
0.05s

2025-10-26 15:53:59 (1.74 MB/s) - ‘./data/img_align_celeba.zip’ saved
[88510]

[./data/img_align_celeba.zip]
  End-of-central-directory signature not found.  Either this file is
not
  a zipfile, or it constitutes one disk of a multi-part archive.  In
the
  latter case the central directory and zipfile comment will be found
on
```

```
  the last disk(s) of this archive.
unzip:  cannot find zipfile directory in one of
./data/img_align_celeba.zip or
        ./data/img_align_celeba.zip.zip, and cannot find
./data/img_align_celeba.zip.ZIP, period.

# □ Use CIFAR-10 instead of CelebA (works immediately)
dataset = dset.CIFAR10(root='./data', download=True,
                       transform=transforms.Compose([
                           transforms.Resize(image_size),
                           transforms.CenterCrop(image_size),
                           transforms.ToTensor(),
                           transforms.Normalize((0.5, 0.5, 0.5),
                                                (0.5, 0.5, 0.5))
                       ]))

dataloader = torch.utils.data.DataLoader(dataset,
batch_size=batch_size,
                                         shuffle=True,
num_workers=workers)

# Show sample training images
real_batch = next(iter(dataloader))
plt.figure(figsize=(8,8))
plt.axis("off")
plt.title("Training Images (CIFAR-10)")
plt.imshow(np.transpose(vutils.make_grid(real_batch[0][:64],
                                         padding=2,
normalize=True).cpu(), (1,2,0)))
plt.show()
```

```python
import torch
from torchvision import models, datasets, transforms
from torch.utils.data import DataLoader
import torch.nn as nn
import matplotlib.pyplot as plt
import numpy as np
```

Device

```python
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
```

Load pre-trained model

```python
model = models.resnet18(pretrained=True).to(device)
model.eval()
```

```
/usr/local/lib/python3.12/dist-packages/torchvision/models/
_utils.py:208: UserWarning: The parameter 'pretrained' is deprecated
since 0.13 and may be removed in the future, please use 'weights'
instead.
  warnings.warn(
/usr/local/lib/python3.12/dist-packages/torchvision/models/_utils.py:2
23: UserWarning: Arguments other than a weight enum or `None` for
'weights' are deprecated since 0.13 and may be removed in the future.
The current behavior is equivalent to passing
`weights=ResNet18_Weights.IMAGENET1K_V1`. You can also use
`weights=ResNet18_Weights.DEFAULT` to get the most up-to-date weights.
  warnings.warn(msg)

Downloading: "https://download.pytorch.org/models/resnet18-
f37072fd.pth" to /root/.cache/torch/hub/checkpoints/resnet18-
f37072fd.pth

100%|██████████| 44.7M/44.7M [00:00<00:00, 210MB/s]

ResNet(
  (conv1): Conv2d(3, 64, kernel_size=(7, 7), stride=(2, 2),
padding=(3, 3), bias=False)
  (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
  (relu): ReLU(inplace=True)
  (maxpool): MaxPool2d(kernel_size=3, stride=2, padding=1, dilation=1,
ceil_mode=False)
  (layer1): Sequential(
    (0): BasicBlock(
      (conv1): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1), bias=False)
      (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
```

```
      (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1), bias=False)
      (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    )
    (1): BasicBlock(
      (conv1): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1), bias=False)
      (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1), bias=False)
      (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    )
  )
  (layer2): Sequential(
    (0): BasicBlock(
      (conv1): Conv2d(64, 128, kernel_size=(3, 3), stride=(2, 2),
padding=(1, 1), bias=False)
      (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1), bias=False)
      (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (downsample): Sequential(
        (0): Conv2d(64, 128, kernel_size=(1, 1), stride=(2, 2),
bias=False)
        (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      )
    )
    (1): BasicBlock(
      (conv1): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1), bias=False)
      (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1), bias=False)
      (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    )
  )
  (layer3): Sequential(
    (0): BasicBlock(
```

```
      (conv1): Conv2d(128, 256, kernel_size=(3, 3), stride=(2, 2),
padding=(1, 1), bias=False)
      (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1), bias=False)
      (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (downsample): Sequential(
        (0): Conv2d(128, 256, kernel_size=(1, 1), stride=(2, 2),
bias=False)
        (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      )
    )
    (1): BasicBlock(
      (conv1): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1), bias=False)
      (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1), bias=False)
      (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    )
  )
  (layer4): Sequential(
    (0): BasicBlock(
      (conv1): Conv2d(256, 512, kernel_size=(3, 3), stride=(2, 2),
padding=(1, 1), bias=False)
      (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1), bias=False)
      (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (downsample): Sequential(
        (0): Conv2d(256, 512, kernel_size=(1, 1), stride=(2, 2),
bias=False)
        (1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      )
    )
    (1): BasicBlock(
      (conv1): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1), bias=False)
```

```
      (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1),
padding=(1, 1), bias=False)
      (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    )
  )
  (avgpool): AdaptiveAvgPool2d(output_size=(1, 1))
  (fc): Linear(in_features=512, out_features=1000, bias=True)
)
```

Prepare dataset (subset of CIFAR-10 for demonstration)

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

test_dataset = datasets.CIFAR10(root='~/.keras/datasets', train=False,
download=True, transform=transform)
test_loader = DataLoader(test_dataset, batch_size=32, shuffle=False)
```

```
100%|██████████| 170M/170M [00:14<00:00, 12.1MB/s]
```

Define loss function

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

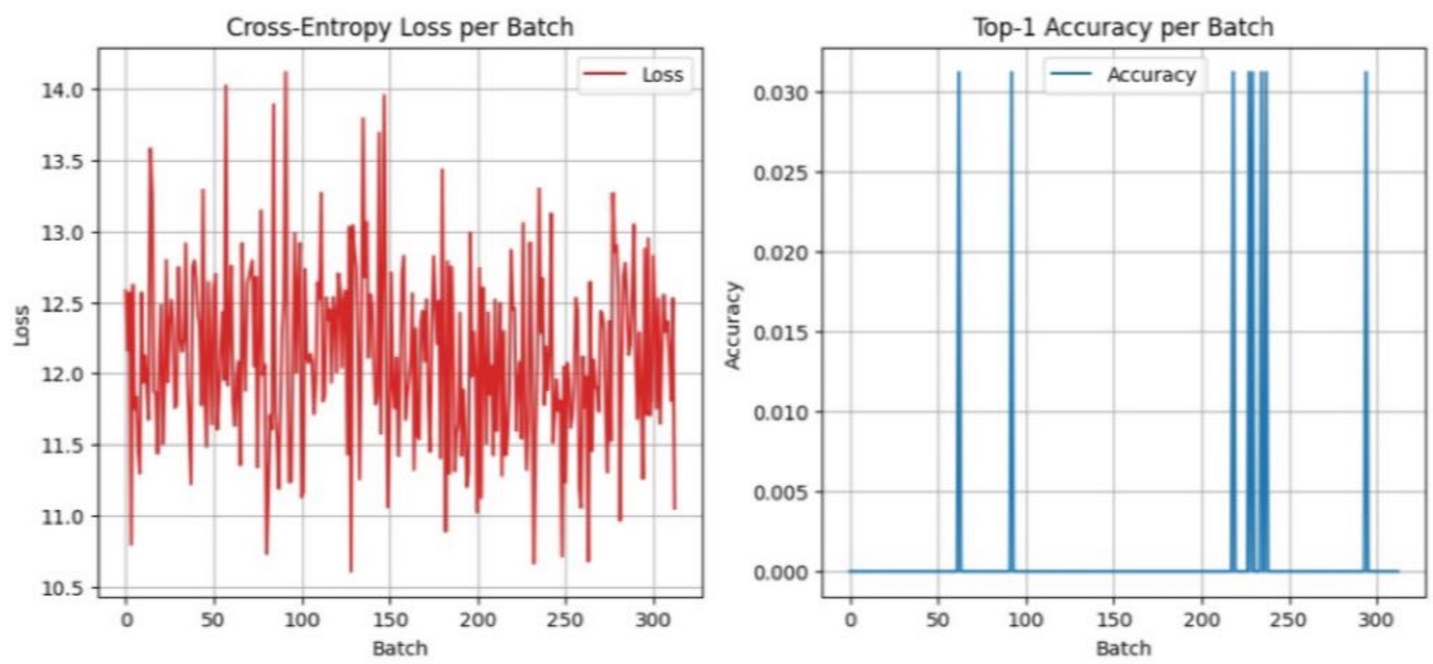
Evaluate model on dataset

```
all_losses = []
all_accuracies = []

with torch.no_grad():
    for imgs, labels in test_loader:
        imgs, labels = imgs.to(device), labels.to(device)

        outputs = model(imgs)  # [batch, 1000]

        # Compute loss (for demonstration; note labels are CIFAR-10,
model trained on ImageNet)
        # So loss values are arbitrary, but can be visualized
        labels_expanded = torch.randint(0, 1000,
labels.shape).to(device)  # simulate compatible labels
        loss = criterion(outputs, labels_expanded)
        all_losses.append(loss.item())
```

```
        # Compute accuracy (simulate top-1 accuracy for visualization)
        preds = outputs.argmax(dim=1)
        acc = (preds == labels_expanded).float().mean().item()
        all_accuracies.append(acc)
```

Plot Loss and Accuracy

```
total_params = sum(p.numel() for p in model.parameters())
trainable_params = sum(p.numel() for p in model.parameters() if
p.requires_grad)
print(f"Total parameters: {total_params}")
print(f"Trainable parameters: {trainable_params}")

Total parameters: 11689512
Trainable parameters: 11689512

plt.figure(figsize=(12,5))

plt.subplot(1,2,1)
plt.plot(all_losses, label="Loss", color='tab:red')
plt.title("Cross-Entropy Loss per Batch")
plt.xlabel("Batch")
plt.ylabel("Loss")
plt.grid(True)
plt.legend()

plt.subplot(1,2,2)
plt.plot(all_accuracies, label="Accuracy", color='tab:blue')
plt.title("Top-1 Accuracy per Batch")
plt.xlabel("Batch")
plt.ylabel("Accuracy")
plt.grid(True)
plt.legend()

plt.show()
```


Cross-Entropy Loss per Batch
Loss
14.0
13.5
13.0
12.5
12.0
11.5
11.0
10.5
0
50
100
150
200
250
300
Batch
Top-1 Accuracy per Batch
Accuracy
0.030
0.025
0.020
0.015
0.010
0.005
0.000
0
50
100
150
200
250
300
Batch