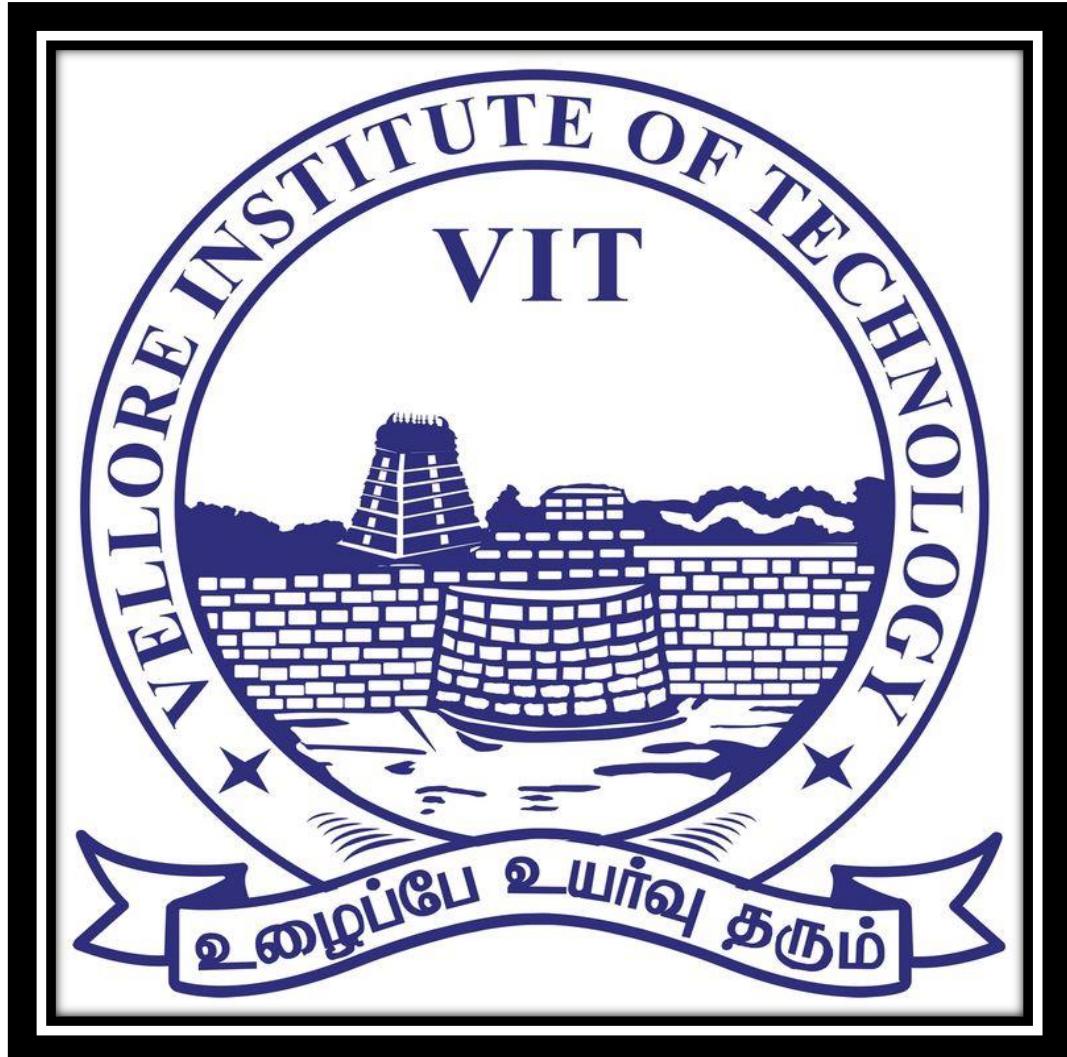




VIT[®]

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI



VELLORE INSTITUTE OF TECHNOLOGY

VIT – CHENNAI CAMPUS

DEEP LEARNING LAB

LAB ASSIGNMENT – 5

FACULTY : DR. HARINI. S



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

CHENNAI

BCSE332P - DL Lab

Lab 5 - DL Lab Assignment

INTRODUCTION TO PYTORCH & GPU

1. Executing the code.

CODE:

```
import torch
import numpy as np
data = [[1, 2], [3, 4]]
x_data = torch.tensor(data)
np_array = np.array(data)
x_np = torch.from_numpy(np_array)
x_ones = torch.ones_like(x_data) # retains the properties of x_data
print(f"Ones Tensor: \n {x_ones} \n")

x_rand = torch.rand_like(x_data, dtype=torch.float) # overrides the
datatype of x_data
print(f"Random Tensor: \n {x_rand} \n")
shape = (2, 3,)
rand_tensor = torch.rand(shape)
ones_tensor = torch.ones(shape)
zeros_tensor = torch.zeros(shape)

print(f"Random Tensor: \n {rand_tensor} \n")
print(f"Ones Tensor: \n {ones_tensor} \n")
print(f"Zeros Tensor: \n {zeros_tensor} ")
tensor = torch.rand(3, 4)

print(f"Shape of tensor: {tensor.shape}")
print(f"Datatype of tensor: {tensor.dtype}")
print(f"Device tensor is stored on: {tensor.device}")
if torch.cuda.is_available():
    tensor = tensor.to('cuda')
```



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

```
print(f"Device tensor is stored on: {tensor.device}")

tensor = torch.ones(4, 4)
tensor[:,1] = 0
print(tensor)

t1 = torch.cat([tensor, tensor, tensor], dim=1)
print(t1)

# This computes the element-wise product
print(tensor.mul(tensor) \n {tensor.mul(tensor)} \n)

# Alternative syntax:
print(tensor * tensor \n {tensor * tensor})

params = list(net.parameters())
print(len(params))

print(params[0].size()) # conv1's .weight
input = torch.randn(1, 1, 32, 32)
out = net(input)
print(out)

import torch
import torchvision
import torchvision.transforms as transforms
transform = transforms.Compose(
    [transforms.ToTensor(),
     transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))])

batch_size = 4

trainset = torchvision.datasets.CIFAR10(root='./data', train=True,
                                         download=True,
                                         transform=transform)
trainloader = torch.utils.data.DataLoader(trainset,
                                         batch_size=batch_size,
                                         shuffle=True, num_workers=2)

testset = torchvision.datasets.CIFAR10(root='./data', train=False,
                                         download=True,
                                         transform=transform)
testloader = torch.utils.data.DataLoader(testset,
                                         batch_size=batch_size,
                                         shuffle=False, num_workers=2)

classes = ('plane', 'car', 'bird', 'cat',
           'deer', 'dog', 'frog', 'horse', 'ship', 'truck')

import matplotlib.pyplot as plt
```



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

```
import numpy as np

# functions to show an image

def imshow(img):
    img = img / 2 + 0.5      # unnormalize
    npimg = img.numpy()
    plt.imshow(np.transpose(npimg, (1, 2, 0)))
    plt.show()

# get some random training images
dataiter = iter(trainloader)
images, labels = next(dataiter)

# show images
imshow(torchvision.utils.make_grid(images))
# print labels
print(' '.join(f'{classes[labels[j]]}:5s' for j in range(batch_size)))
# Alternative syntax:
print(f"tensor @ tensor.T \n {tensor @ tensor.T}")
print(tensor, "\n")
tensor.add_(5)
print(tensor)
t = torch.ones(5)
print(f"t: {t}")
n = t.numpy()
print(f"n: {n}")
t.add_(1)
print(f"t: {t}")
print(f"n: {n}")
n = np.ones(5)
t = torch.from_numpy(n)
np.add(n, 1, out=n)
print(f"t: {t}")
print(f"n: {n}")
import torch
from torchvision.models import resnet18, ResNet18_Weights
model = resnet18(weights=ResNet18_Weights.DEFAULT)
data = torch.rand(1, 3, 64, 64)
labels = torch.rand(1, 1000)
prediction = model(data) # forward pass
```



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

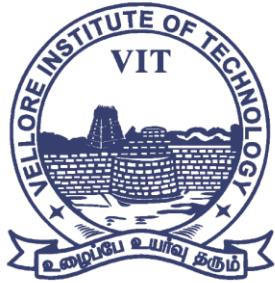
```
loss = (prediction - labels).sum()
loss.backward() # backward pass
optim = torch.optim.SGD(model.parameters(), lr=1e-2, momentum=0.9)
optim.step() #gradient descent
import torch
import torch.nn as nn
import torch.nn.functional as F

class Net(nn.Module):

    def __init__(self):
        super(Net, self).__init__()
        # 1 input image channel, 6 output channels, 5x5 square
        convolution
        # kernel
        self.conv1 = nn.Conv2d(1, 6, 5)
        self.conv2 = nn.Conv2d(6, 16, 5)
        # an affine operation: y = Wx + b
        self.fc1 = nn.Linear(16 * 5 * 5, 120) # 5*5 from image
        dimension
        self.fc2 = nn.Linear(120, 84)
        self.fc3 = nn.Linear(84, 10)

    def forward(self, x):
        # Max pooling over a (2, 2) window
        x = F.max_pool2d(F.relu(self.conv1(x)), (2, 2))
        # If the size is a square, you can specify with a single number
        x = F.max_pool2d(F.relu(self.conv2(x)), 2)
        x = torch.flatten(x, 1) # flatten all dimensions except the
        batch dimension
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = self.fc3(x)
        return x

net = Net()
print(net)
```



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

SCREENSHOTS OF THE CODE SNIPPET:

The screenshot shows a Google Colab notebook titled "21BAI1364 BCSE332P LAB ASSIGN.5.ipynb". The code cell [2] contains:[2]: import torch
import numpy as np

Under the "Tensor Initialization" section, there are two sub-sections: "From a NumPy array" and "From Another Tensor".

Code cell [3] contains:[3]: np_array = np.array(data)
x_np = torch.from_numpy(np_array)

Code cell [4] contains:[4]: x_ones = torch.ones_like(x_data) # retains the properties of x_data
print(f"Ones Tensor: \n {x_ones} \n")

x_rand = torch.rand_like(x_data, dtype=torch.float) # overrides the datatype of x_data
print(f"Random Tensor: \n {x_rand} \n")

A tooltip from Google Account shows the user's name: "vittavishnu.datta2021@vitstudent.ac.in".

The screenshot shows the same Google Colab notebook. The code cell [5] contains:[5]: shape = (2, 3)
rand_tensor = torch.rand(shape)
ones_tensor = torch.ones(shape)
zeros_tensor = torch.zeros(shape)

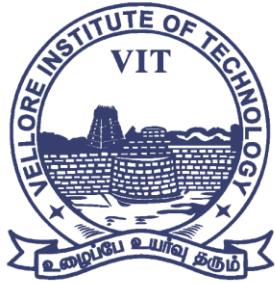
print(f"Random Tensor: \n {rand_tensor} \n")
print(f"Ones Tensor: \n {ones_tensor} \n")
print(f"Zeros Tensor: \n {zeros_tensor} \n")

Code cell [6] contains:[6]: Random Tensor:
tensor([0.4983, 0.4045, 0.9663],
[0.3377, 0.3466, 0.3870])

Ones Tensor:
tensor([[1., 1., 1.],
[1., 1., 1.]])

Zeros Tensor:
tensor([[0., 0., 0.],
[0., 0., 0.]])

A tooltip from Google Account shows the user's name: "vittavishnu.datta2021@vitstudent.ac.in".



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

A screenshot of a Jupyter Notebook in Google Colab. The notebook has two code cells:

```
[5] Ones Tensor:  
tensor([[1., 1., 1.],  
       [1., 1., 1.]])  
  
[6] Zeros Tensor:  
tensor([[0., 0., 0.],  
       [0., 0., 0.]])  
  
[7] tensor = torch.rand(3, 4)  
print(f"Shape of tensor: {tensor.shape}")  
print(f"Datatype of tensor: {tensor.dtype}")  
print(f"Device tensor is stored on: {tensor.device}")  
  
[8] if torch.cuda.is_available():  
    tensor = tensor.to('cuda')  
    print(f"Device tensor is stored on: {tensor.device}")
```

The notebook interface includes a sidebar with sections like 'Tensor Attributes' and 'Tensor Operations'. A floating Google account sidebar shows a profile picture and the message 'Hi, Vitta Vishnu Datta!'. The system tray at the bottom shows the date as 04-03-2024.

A screenshot of a Jupyter Notebook in Google Colab. The notebook has three code cells:

```
[8] tensor = torch.ones(4, 4)  
tensor[:, :] = 0  
print(tensor)  
  
[9] t1 = torch.cat([tensor, tensor], dim=1)  
print(t1)  
  
[10] # This computes the element-wise product  
print(f"tensor.mul(tensor) \n{tensor.mul(tensor)} \n")  
# Alternative syntax:  
print(f"tensor * tensor \n{tensor * tensor}")
```

The notebook interface includes a sidebar with sections like 'Tensor Attributes' and 'Tensor Operations'. A floating Google account sidebar shows a profile picture and the message 'Hi, Vitta Vishnu Datta!'. The system tray at the bottom shows the date as 04-03-2024.



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

21BAI1364 BCSE332P LAB ASSIGN.5.ipynb

```
[10] tensor * tensor
      tensor([[1., 0., 1., 1.],
             [1., 0., 1., 1.],
             [1., 0., 1., 1.],
             [1., 0., 1., 1.]])  

[11] print(f"tensor.matmul(tensor.T) \n {tensor.matmul(tensor.T)} \n")
      # Alternative syntax:
      print(f"tensor @ tensor.T \n {tensor @ tensor.T}")  

      tensor.matmul(tensor.T)
      tensor([[3., 3., 3., 3.],
             [3., 3., 3., 3.],
             [3., 3., 3., 3.],
             [3., 3., 3., 3.]])  

      tensor @ tensor.T
      tensor([[3., 3., 3., 3.],
             [3., 3., 3., 3.],
             [3., 3., 3., 3.],
             [3., 3., 3., 3.]])  

[12] print(tensor, "\n")
      tensor.add_(5)
      print(tensor)  

      tensor([[1., 0., 1., 1.],
             [1., 0., 1., 1.],
             [1., 0., 1., 1.],
             [1., 0., 1., 1.]])  

      tensor([[6., 5., 6., 6.],
```

vittavishnu.data202@vitstudent.ac.in
Managed by vitstudent.ac.in

Hi, Vitta Vishnu Datta!

Manage your Google Account

Add account Sign out

Privacy Policy Terms of Service

21BAI1364 BCSE332P LAB ASSIGN.5.ipynb

```
[12] tensor([[6., 5., 6., 6.],
      [6., 5., 6., 6.],
      [6., 5., 6., 6.],
      [6., 5., 6., 6.]])  

With NumPY  

[13] t = torch.ones(5)
      print(f"t: {t}")
      n = t.numpy()
      print(f"n: {n}")
      t: tensor([1., 1., 1., 1., 1.])
      n: [1. 1. 1. 1. 1.]  

[14] t.add_(1)
      print(f"t: {t}")
      print(f"n: {n}")
      t: tensor([2., 2., 2., 2., 2.])
      n: [2. 2. 2. 2. 2.]  

[15] n = np.ones(5)
      t = torch.from_numpy(n)
      np.add(n, 1, out=n)
      print(f"t: {t}")
      t: tensor([2., 2., 2., 2., 2.])
```

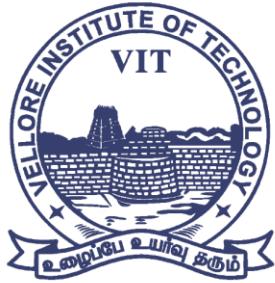
vittavishnu.data202@vitstudent.ac.in
Managed by vitstudent.ac.in

Hi, Vitta Vishnu Datta!

Manage your Google Account

Add account Sign out

Privacy Policy Terms of Service



VIT[®]

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

```
21BAI1364 BCSE332P LAB ASSIGN.5.ipynb
```

```
[17]: import torch
from torchvision.models import resnet18, ResNet18_Weights
model = resnet18(weights=ResNet18_Weights.DEFAULT)
data = torch.randn(1, 3, 64, 64)
labels = torch.randint(0, 1000, (1,))

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, 74.0MB/s]
```

```
[18]: prediction = model(data) # forward pass
```

```
[19]: loss = (prediction - labels).sum()
loss.backward() # backward pass
```

```
[20]: optim = torch.optim.SGD(model.parameters(), lr=1e-2, momentum=0.9)
```

```
[21]: optim.step() #gradient descent
```

27°C Partly cloudy

ENG IN 23:08 04-03-2024

```
21BAI1364 BCSE332P LAB ASSIGN.5.ipynb
```

```
[18]: prediction = model(data) # forward pass
```

```
[19]: loss = (prediction - labels).sum()
loss.backward() # backward pass
```

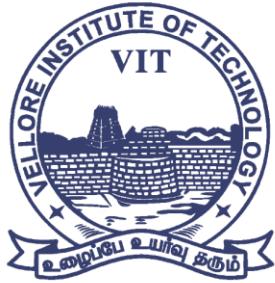
```
[20]: optim = torch.optim.SGD(model.parameters(), lr=1e-2, momentum=0.9)
```

```
[21]: optim.step() #gradient descent
```

```
[22]: class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        # 1 input image channel, 6 output channels, 5x5 square convolution
        # kernel
        self.conv1 = nn.Conv2d(1, 6, 5)
        self.conv2 = nn.Conv2d(6, 16, 5)
        # an affine operation: y = mx + b
        self.fc1 = nn.Linear(16 * 5 * 5, 120) # 5*5 from image dimension
        self.fc2 = nn.Linear(120, 10)
```

27°C Partly cloudy

ENG IN 23:09 04-03-2024



VIT[®]

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

21BAI1364 BCSE332P LAB ASSIGN.5.ipynb

```
import torch
import torch.nn as nn
import torch.nn.functional as F

class Net(nn.Module):

    def __init__(self):
        super(Net, self).__init__()
        # 1 input image channel, 6 output channels, 5x5 square convolution
        # kernel
        self.conv1 = nn.Conv2d(1, 6, 5)
        self.conv2 = nn.Conv2d(6, 16, 5)
        # an affine operation: y = Wx + b
        self.fc1 = nn.Linear(16 * 5 * 5, 120) # 5*5 from image dimension
        self.fc2 = nn.Linear(120, 84)
        self.fc3 = nn.Linear(84, 10)

    def forward(self, x):
        # Max pooling over a 2, 2 window
        x = F.max_pool2d(F.relu(self.conv1(x)), (2, 2))
        # If the size is a square, you can specify with a single number
        x = F.max_pool2d(F.relu(self.conv2(x)), 2)
        x = torch.flatten(x, 1) # flatten all dimensions except the batch dimension
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = self.fc3(x)
        return x

net = Net()
print(net)
```

vittavishnu.data2021@vitstudent.ac.in
Managed by vitstudent.ac.in

Hi, Vitta Vishnu Datta!

Add account Sign out

Privacy Policy Terms of Service

27°C Partly cloudy Search ENG IN 23:09 04-03-2024

21BAI1364 BCSE332P LAB ASSIGN.5.ipynb

```
Net(
    (conv1): Conv2d(1, 6, kernel_size=(5, 5), stride=(1, 1))
    (conv2): Conv2d(6, 16, kernel_size=(5, 5), stride=(1, 1))
    (fc1): Linear(in_features=480, out_features=120, bias=True)
    (fc2): Linear(in_features=120, out_features=84, bias=True)
    (fc3): Linear(in_features=84, out_features=10, bias=True)
)

[23] params = list(net.parameters())
print(len(params))
print(params[0].size()) # conv1's .weight

10
torch.Size([6, 1, 5, 5])

[24] input = torch.randn(1, 1, 32, 32)
out = net(input)
print(out)

tensor([[[-0.0033, -0.0990, -0.0097,  0.1367,  0.1192, -0.0109, -0.1653, -0.1234,
         0.0046, -0.0339]], grad_fn=<AddmmBackward0>])

Simple Classification using PyTorch

[25] import torch
import torchvision
import torchvision.transforms as transforms

[26] transform = transforms.Compose(
```

vittavishnu.data2021@vitstudent.ac.in
Managed by vitstudent.ac.in

Hi, Vitta Vishnu Datta!

Add account Sign out

Privacy Policy Terms of Service

27°C Partly cloudy Search ENG IN 23:09 04-03-2024



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

21BAI1364 BCSE332P LAB ASSIG... +

File Edit View Insert Runtime Tools Help All changes saved

```
[26]: transform = transforms.Compose([
    transforms.ToTensor(),
    transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))])

batch_size = 4

trainset = torchvision.datasets.CIFAR10(root='./data', train=True,
                                         download=True, transform=transform)
trainloader = torch.utils.data.DataLoader(trainset, batch_size=batch_size,
                                          shuffle=True, num_workers=2)

testset = torchvision.datasets.CIFAR10(root='./data', train=False,
                                         download=True, transform=transform)
testloader = torch.utils.data.DataLoader(testset, batch_size=batch_size,
                                         shuffle=False, num_workers=2)

classes = ('plane', 'car', 'bird', 'cat',
           'deer', 'dog', 'frog', 'horse', 'ship', 'truck')

Downloading https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz to ./data/cifar-10-python.tar.gz
100% [██████████] 170498071 /170498071 [00:03:00:00, 48272881.44it/s]
Extracting ./data/cifar-10-python.tar.gz to ./data
Files already downloaded and verified
```

```
import matplotlib.pyplot as plt
import numpy as np

# functions to show an image
```

0s completed at 11:00 PM

vittavishnu.data202@vitstudent.ac.in
Managed by vitstudent.ac.in

Hi, Vitta Vishnu Datta!
Manage your Google Account

+ Add account Sign out

Privacy Policy Terms of Service

27°C Partly cloudy 23:09 04-03-2024 ENG IN

21BAI1364 BCSE332P LAB ASSIG... +

File Edit View Insert Runtime Tools Help All changes saved

```
# functions to show an image

def imshow(img):
    img = img / 2 + 0.5      # unnormalize
    npimg = img.numpy()
    plt.imshow(np.transpose(npimg, (1, 2, 0)))
    plt.show()

# get some random training images
dataiter = iter(trainloader)
images, labels = next(dataiter)

# show images
imshow(torchvision.utils.make_grid(images))
# print labels
print(' '.join(f'{classes[labels[j]]}' for j in range(batch_size)))
```

0s completed at 11:00 PM

vittavishnu.data202@vitstudent.ac.in
Managed by vitstudent.ac.in

Hi, Vitta Vishnu Datta!
Manage your Google Account

+ Add account Sign out

Privacy Policy Terms of Service

27°C Partly cloudy 23:09 04-03-2024 ENG IN



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

2. Executing the PYTORCH & GPU

CODE:

```
import torch
import torchvision
import torchvision.transforms as transforms
device = torch.device('cuda:0' if torch.cuda.is_available() else 'cpu')

# Assuming that we are on a CUDA machine, this should print a CUDA
device:

print(device)
transform = transforms.Compose(
    [transforms.ToTensor(),
     transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))])

batch_size = 4

trainset = torchvision.datasets.CIFAR10(root='./data', train=True,
                                         download=True,
                                         transform=transform)
trainloader = torch.utils.data.DataLoader(trainset,
                                         batch_size=batch_size,
                                         shuffle=True, num_workers=2)

testset = torchvision.datasets.CIFAR10(root='./data', train=False,
                                         download=True,
                                         transform=transform)
testloader = torch.utils.data.DataLoader(testset,
                                         batch_size=batch_size,
                                         shuffle=False, num_workers=2)

classes = ('plane', 'car', 'bird', 'cat',
           'deer', 'dog', 'frog', 'horse', 'ship', 'truck')
import matplotlib.pyplot as plt
import numpy as np

# functions to show an image
```



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

```
def imshow(img):
    img = img / 2 + 0.5      # unnormalize
    npimg = img.numpy()
    plt.imshow(np.transpose(npimg, (1, 2, 0)))
    plt.show()

# get some random training images
dataiter = iter(trainloader)
images, labels = next(dataiter)

# show images
imshow(torchvision.utils.make_grid(images))
# print labels
print(' '.join(f'{classes[labels[j]]}:5s' for j in range(batch_size)))
import torch.nn as nn
import torch.nn.functional as F

class Net(nn.Module):
    def __init__(self):
        super().__init__()
        self.conv1 = nn.Conv2d(3, 6, 5)
        self.pool = nn.MaxPool2d(2, 2)
        self.conv2 = nn.Conv2d(6, 16, 5)
        self.fc1 = nn.Linear(16 * 5 * 5, 120)
        self.fc2 = nn.Linear(120, 84)
        self.fc3 = nn.Linear(84, 10)

    def forward(self, x):
        x = self.pool(F.relu(self.conv1(x)))
        x = self.pool(F.relu(self.conv2(x)))
        x = torch.flatten(x, 1) # flatten all dimensions except batch
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = self.fc3(x)
        return x

net = Net()
net.to(device)
import torch.optim as optim

criterion = nn.CrossEntropyLoss()
```



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

```
optimizer = optim.SGD(net.parameters(), lr=0.001, momentum=0.9)
for epoch in range(2): # loop over the dataset multiple times
    running_loss = 0.0
    for i, data in enumerate(trainloader, 0):
        # Assuming data is a list containing inputs and labels
        inputs, labels = data[0], data[1] # Access elements using
        integer indices

        # Move data to device (if applicable)
        if device is not None:
            inputs = inputs.to(device)
            labels = labels.to(device)

        # zero the parameter gradients
        optimizer.zero_grad()

        # forward + backward + optimize
        outputs = net(inputs)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()

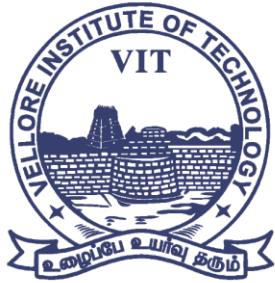
        # print statistics
        running_loss += loss.item()
        if i % 2000 == 1999: # print every 2000 mini-batches
            print(f'{epoch + 1}, {i + 1:5d}] loss: {running_loss / 2000:.3f}')
            running_loss = 0.0

print('Finished Training')

PATH = './cifar_net.pth'
torch.save(net.state_dict(), PATH)
dataiter = iter(testloader)
images, labels = next(dataiter)

# print images
imshow(torchvision.utils.make_grid(images))
print('GroundTruth: ', ' '.join(f'{classes[labels[j]]}:5s' for j in
range(4)))
```

LINK TO THE GOOGLE COLAB NOTEBOOK



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

https://colab.research.google.com/drive/14n5SwP1q5zy50pKOwn-VKLIWJCJLcSdE?usp=chrome_ntp

SCREENSHOTS OF THE CODE SNIPPET:

The screenshot shows a Google Colab notebook titled "21BAI1364 BCSE332P LAB ASSIGN.5.ipynb". The code cell [3] contains the following Python code:[3]: device = torch.device('cuda:0' if torch.cuda.is_available() else 'cpu')
Assuming that we are on a CUDA machine, this should print a CUDA device:
print(device)

The output of the code is "cuda:0". Below the code, there is a section titled "Steps:" with the following steps listed:

- Load and normalize the CIFAR10 training and test datasets using torchvision
- Define a Convolutional Neural Network
- Define a loss function
- Train the network on the training data
- Test the network on the test data

A small window on the right shows a Google account profile for "vittavishnu.datta2021@vitstudent.ac.in".

The screenshot shows a Google Colab notebook titled "21BAI1364 BCSE332P LAB ASSIGN.5.ipynb". The code cell [4] contains the following Python code:[4]: transform = transforms.Compose([transforms.ToTensor(),
 transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))])

batch_size = 4

trainset = torchvision.datasets.CIFAR10(root='./data', train=True,
 download=True, transform=transform)
trainloader = torch.utils.data.DataLoader(trainset, batch_size=batch_size,
 shuffle=True, num_workers=2)

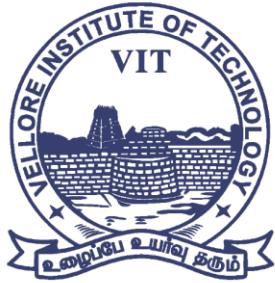
testset = torchvision.datasets.CIFAR10(root='./data', train=False,
 download=True, transform=transform)
testloader = torch.utils.data.DataLoader(testset, batch_size=batch_size,
 shuffle=False, num_workers=2)

classes = ('plane', 'car', 'bird', 'cat',
 'deer', 'dog', 'frog', 'horse', 'ship', 'truck')

Below the code, it shows the download of the CIFAR-10 dataset from <https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz> to the local directory. The code cell [5] contains:[5]: import matplotlib.pyplot as plt
import numpy as np

functions to show an image

A small window on the right shows a Google account profile for "vittavishnu.datta2021@vitstudent.ac.in".



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

21BAI1364 BCSE332P LAB ASSIGN.5.ipynb

```
[4] classes = ('plane', 'car', 'bird', 'cat',
              'deer', 'dog', 'frog', 'horse', 'ship', 'truck')

[5] import matplotlib.pyplot as plt
    import numpy as np

    # functions to show an image

    def imshow(img):
        img = img / 2 + 0.5      # unnormalize
        npimg = img.numpy()
        plt.imshow(np.transpose(npimg, (1, 2, 0)))
        plt.show()

    # get some random training images
    dataiter = iter(trainloader)
    images, labels = next(dataiter)

    # show images
    imshow(torchvision.utils.make_grid(images))
    # print labels
    print(' '.join(f'{classes[labels[j]]}:s' for j in range(batch_size)))
```

vittavishnu.datta2021@vitstudent.ac.in
Managed by vitstudent.ac.in

Hi, Vitta Vishnu Datta!

Manage your Google Account

Add account Sign out

Privacy Policy Terms of Service

0s completed at 11:21 PM

27°C Partly cloudy

Search

ENG IN 23:26 04-03-2024

21BAI1364 BCSE332P LAB ASSIGN.5.ipynb

```
[5] def imshow(img):
    img = img / 2 + 0.5      # unnormalize
    npimg = img.numpy()
    plt.imshow(np.transpose(npimg, (1, 2, 0)))
    plt.show()

    # get some random training images
    dataiter = iter(trainloader)
    images, labels = next(dataiter)

    # show images
    imshow(torchvision.utils.make_grid(images))
    # print labels
    print(' '.join(f'{classes[labels[j]]}:s' for j in range(batch_size)))
```

```
[7] import torch.nn as nn
    import torch.nn.functional as F
```

vittavishnu.datta2021@vitstudent.ac.in
Managed by vitstudent.ac.in

Hi, Vitta Vishnu Datta!

Manage your Google Account

Add account Sign out

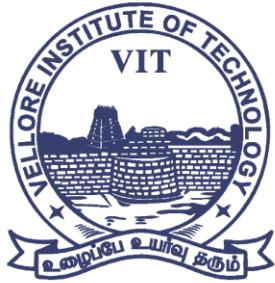
Privacy Policy Terms of Service

0s completed at 11:21 PM

27°C Partly cloudy

Search

ENG IN 23:27 04-03-2024



VIT[®]

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

21BAI1364 BCSE332P LAB ASSIGN.5.ipynb

```
import torch.nn as nn
import torch.nn.functional as F

class Net(nn.Module):
    def __init__(self):
        super().__init__()
        self.conv1 = nn.Conv2d(3, 6, 5)
        self.pool = nn.MaxPool2d(2, 2)
        self.conv2 = nn.Conv2d(6, 16, 5)
        self.fc1 = nn.Linear(16 * 5 * 5, 120)
        self.fc2 = nn.Linear(120, 84)
        self.fc3 = nn.Linear(84, 10)

    def forward(self, x):
        x = self.pool(F.relu(self.conv1(x)))
        x = self.pool(F.relu(self.conv2(x)))
        x = torch.flatten(x, 1) # flatten all dimensions except batch
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = self.fc3(x)
        return x

net = Net()

[8] net.to(device)
```

vittavishnu.datta2021@vitstudent.ac.in
Managed by vitstudent.ac.in

Hi, Vitta Vishnu Datta!

+ Add account | Sign out

Privacy Policy · Terms of Service

27°C Partly cloudy

21BAI1364 BCSE332P LAB ASSIGN.5.ipynb

```
[8] net.to(device)
```

Net(
 (conv1): Conv2d(3, 6, kernel_size=(5, 5), stride=(1, 1))
 (pool): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
 (conv2): Conv2d(6, 16, kernel_size=(5, 5), stride=(1, 1))
 (fc1): Linear(in_features=400, out_features=120, bias=True)
 (fc2): Linear(in_features=120, out_features=84, bias=True)
 (fc3): Linear(in_features=84, out_features=10, bias=True)
)

```
[9] import torch.optim as optim

criterion = nn.CrossEntropyLoss()
optimizer = optim.SGD(net.parameters(), lr=0.001, momentum=0.9)

for epoch in range(2): # loop over the dataset multiple times
    running_loss = 0.0
    for i, data in enumerate(trainloader, 0):
        # Assuming data is a list containing inputs and labels
        inputs, labels = data[0], data[1] # Access elements using integer indices

        # Move data to device (if applicable)
        if device is not None:
            inputs = inputs.to(device)
            labels = labels.to(device)

        # zero the parameter gradients
        optimizer.zero_grad()
```

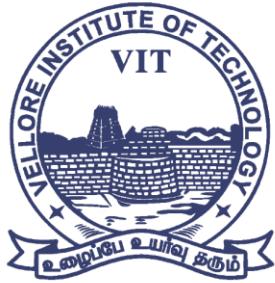
vittavishnu.datta2021@vitstudent.ac.in
Managed by vitstudent.ac.in

Hi, Vitta Vishnu Datta!

+ Add account | Sign out

Privacy Policy · Terms of Service

27°C Partly cloudy



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

21BAI1364 BCSE332P LAB ASSIGN.5.ipynb

```
[14] for epoch in range(2): # loop over the dataset multiple times
    running_loss = 0.0
    for i, data in enumerate(trainloader, 0):
        # Assuming data is a list containing inputs and labels
        inputs, labels = data[0], data[1] # Access elements using integer indices

        # Move data to device (if applicable)
        if device is not None:
            inputs = inputs.to(device)
            labels = labels.to(device)

        # zero the parameter gradients
        optimizer.zero_grad()

        # forward + backward + optimize
        outputs = net(inputs)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()

        # print statistics
        running_loss += loss.item()
        if i % 2000 == 1999: # print every 2000 mini-batches
            print(f'[{epoch + 1}, {i + 1:5d}] loss: {running_loss / 2000:.3f}')
            running_loss = 0.0

    print('Finished Training')

[1, 2000] loss: 2.209
[1, 4000] loss: 1.938
```

vittavishnu.datta2021@vitstudent.ac.in
Managed by vitstudent.ac.in

Hi, Vitta Vishnu Datta!

Add account Sign out

Privacy Policy Terms of Service

21BAI1364 BCSE332P LAB ASSIGN.5.ipynb

```
[14]     if i % 2000 == 1999: # print every 2000 mini-batches
        print(f'[{epoch + 1}, {i + 1:5d}] loss: {running_loss / 2000:.3f}')
        running_loss = 0.0

    print('Finished Training')

[1, 2000] loss: 2.209
[1, 4000] loss: 1.938
[1, 6000] loss: 1.715
[1, 8000] loss: 1.508
[1, 10000] loss: 1.515
[1, 12000] loss: 1.465
[2, 2000] loss: 1.429
[2, 4000] loss: 1.387
[2, 6000] loss: 1.366
[2, 8000] loss: 1.350
[2, 10000] loss: 1.315
[2, 12000] loss: 1.296
Finished Training

[15] PATH = './cifar_net.pth'
    torch.save(net.state_dict(), PATH)
```

vittavishnu.datta2021@vitstudent.ac.in
Managed by vitstudent.ac.in

Hi, Vitta Vishnu Datta!

Add account Sign out

Privacy Policy Terms of Service



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

The screenshot shows a Jupyter Notebook interface in Google Colab. The code cell [15] contains Python code for saving a PyTorch model and displaying a grid of four CIFAR-10 images. The output shows the images and their ground truth labels: cat, ship, ship, plane. A Google account sidebar is visible on the right.

3. Visualize the filters and features for GoogLeNet model using pytorch.

CODE FOR PRE-TRAINED GoogLeNet MODEL:

```
import torch
model = torch.hub.load('pytorch/vision:v0.10.0', 'googlenet',
pretrained=True)
model.eval()
# Download an example image from the pytorch website
import urllib
url, filename =
("https://github.com/pytorch/hub/raw/master/images/dog.jpg", "dog.jpg")
try: urllib.URLopener().retrieve(url, filename)
except: urllib.request.urlretrieve(url, filename)
# sample execution (requires torchvision)
from PIL import Image
from torchvision import transforms
input_image = Image.open(filename)
preprocess = transforms.Compose([
```



VIT[®]

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

```
transforms.Resize(256),  
transforms.CenterCrop(224),  
transforms.ToTensor(),  
transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224,  
0.225]),  
])  
input_tensor = preprocess(input_image)  
input_batch = input_tensor.unsqueeze(0) # create a mini-batch as  
expected by the model  
  
# move the input and model to GPU for speed if available  
if torch.cuda.is_available():  
    input_batch = input_batch.to('cuda')  
    model.to('cuda')  
  
with torch.no_grad():  
    output = model(input_batch)  
# Tensor of shape 1000, with confidence scores over ImageNet's 1000  
classes  
print(output[0])  
# The output has unnormalized scores. To get probabilities, you can run  
a softmax on it.  
probabilities = torch.nn.functional.softmax(output[0], dim=0)  
print(probabilities)  
# Download ImageNet labels  
!wget  
https://raw.githubusercontent.com/pytorch/hub/master/imagenet_classes.t  
xt  
# Read the categories  
with open("imagenet_classes.txt", "r") as f:  
    categories = [s.strip() for s in f.readlines()]  
# Show top categories per image  
top5_prob, top5_catid = torch.topk(probabilities, 5)  
for i in range(top5_prob.size(0)):  
    print(categories[top5_catid[i]], top5_prob[i].item())
```

SCREENSHOTS OF THE GoogLeNet MODEL:



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

```
import torch
model = torch.hub.load('pytorch/vision:v0.10.0', 'googlenet', pretrained=True)
model.eval()

  (bn): BatchNorm2d(32, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
)
  (inception3b): Inception(
    (branch1): BasicConv2d(
      (conv): Conv2d(256, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn): BatchNorm2d(128, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
    )
    (branch2): Sequential(
      (0): BasicConv2d(
        (conv): Conv2d(256, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (bn): BatchNorm2d(128, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
      )
      (1): BasicConv2d(
        (conv): Conv2d(128, 192, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
        (bn): BatchNorm2d(192, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
      )
    )
    (branch3): Sequential(
      (0): BasicConv2d(
        (conv): Conv2d(256, 32, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (bn): BatchNorm2d(32, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
      )
    )
)
  (inception3c): Inception(
    (branch1): BasicConv2d(
      (conv): Conv2d(480, 64, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn): BatchNorm2d(64, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
    )
)
  (inception4a): Inception(
    (branch1): BasicConv2d(
      (conv): Conv2d(512, 160, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn): BatchNorm2d(160, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
    )
    (branch2): Sequential(
      (0): BasicConv2d(
        (conv): Conv2d(512, 112, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (bn): BatchNorm2d(112, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
      )
      (1): BasicConv2d(
        (conv): Conv2d(112, 224, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
        (bn): BatchNorm2d(224, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
      )
    )
    (branch3): Sequential(
      (0): BasicConv2d(
        (conv): Conv2d(512, 24, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (bn): BatchNorm2d(24, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
      )
      (1): BasicConv2d(
        # Download an example image from the pytorch website
        import urllib
        url, filename = ("https://github.com/pytorch/hub/raw/master/images/dog.jpg", "dog.jpg")
        try: urllib.URLopener().retrieve(url, filename)
        except: urllib.request.urlretrieve(url, filename)
      )
    )
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
)
```

```
# Download an example image from the pytorch website
import urllib
url, filename = ("https://github.com/pytorch/hub/raw/master/images/dog.jpg", "dog.jpg")
try: urllib.URLopener().retrieve(url, filename)
except: urllib.request.urlretrieve(url, filename)
```



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

21BAI1364 BCSE332P LAB ASSIG... +

File Edit View Insert Runtime Tools Help All changes saved

```
[18] # Download an example image from the pytorch website
import urllib
url, filename = ("https://github.com/pytorch/hub/raw/master/images/dog.jpg", "dog.jpg")
try: urllib.URLopener().retrieve(url, filename)
except: urllib.request.urlretrieve(url, filename)

[19] # sample execution (requires torchvision)
from PIL import Image
from torchvision import transforms
input_image = Image.open(filename)
preprocess = transforms.Compose([
    transforms.Resize(256),
    transforms.CenterCrop(224),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]),
])
input_tensor = preprocess(input_image)
input_batch = input_tensor.unsqueeze(0) # create a mini-batch as expected by the model

# move the input and model to GPU for speed if available
if torch.cuda.is_available():
    input_batch = input_batch.to('cuda')
    model.to('cuda')

with torch.no_grad():
    output = model(input_batch)
# Tensor of shape 1000, with confidence scores over ImageNet's 1000 classes
print(output[0])
# The output has unnormalized scores. To get probabilities, you can run a softmax on it.
probabilities = torch.nn.functional.softmax(output[0], dim=0)
print(probabilities)
```

0s completed at 11:32 PM

vittavishnu.data202@vitstudent.ac.in
Managed by vitstudent.ac.in

Hi, Vitta Vishnu Datta!

Manage your Google Account

+ Add account Sign out

Privacy Policy Terms of Service

27°C Partly cloudy Search ENG IN 23:36 04-03-2024

21BAI1364 BCSE332P LAB ASSIG... +

File Edit View Insert Runtime Tools Help All changes saved

```
[19] probabilities = torch.nn.functional.softmax(output[0], dim=0)
print(probabilities)

9.6700e-06, 1.7964e-05, 1.4644e-05, 1.4619e-05, 3.0407e-05, 2.1671e-05,
2.8080e-05, 1.8943e-05, 4.1040e-06, 1.6279e-05, 1.7598e-05, 2.1738e-05,
1.5043e-05, 8.7795e-06, 7.7322e-06, 1.8247e-05, 3.3558e-05, 1.4045e-05,
1.5260e-05, 1.0509e-05, 2.4395e-05, 7.2626e-06, 2.3024e-05, 3.6498e-05,
6.3685e-06, 8.3994e-06, 1.1489e-05, 1.6684e-05, 8.2122e-06, 7.3346e-06,
1.7020e-05, 1.3606e-05, 1.0223e-05, 9.2187e-06, 8.8084e-06, 1.0283e-05,
4.5752e-06, 3.8138e-06, 4.0880e-05, 8.2313e-06, 1.6678e-05, 2.1501e-05,
3.1988e-05, 1.5459e-05, 3.5853e-05, 1.1873e-05, 2.7772e-05, 1.3308e-05,
6.6405e-06, 7.4611e-05, 1.0962e-05, 7.6554e-06, 4.1684e-05, 7.8274e-06,
1.7613e-05, 7.5711e-06, 2.1133e-05, 4.1397e-05, 1.7090e-05, 1.3629e-05,
1.8244e-05, 4.4653e-06, 7.7112e-05, 8.0590e-06, 1.0524e-05, 4.0827e-05,
2.0853e-05, 2.0898e-05, 3.5328e-05, 7.2529e-05, 3.3029e-05, 1.6842e-05,
3.4391e-06, 2.2082e-05, 5.8895e-05, 8.9084e-06, 2.7891e-05, 2.1940e-05,
1.2625e-05, 1.9812e-05, 1.7730e-05, 1.3376e-05, 8.9296e-06, 1.7879e-05,
2.5897e-05, 4.8600e-06, 7.4846e-06, 1.8515e-05, 2.4884e-05, 2.8516e-05,
3.3757e-06, 1.1566e-05, 8.4250e-06, 2.4185e-05, 3.5165e-05,
3.1746e-05, 1.2895e-05, 1.7137e-05, 5.4822e-05, 6.4239e-06, 3.5337e-05,
3.9793e-05, 1.0538e-05, 2.1348e-05, 3.1569e-05, 3.7572e-05, 2.5513e-05,
1.7718e-05, 2.4876e-05, 1.6158e-05, 2.4638e-05, 1.9813e-05, 1.6788e-05,
2.9668e-05, 1.7183e-05, 2.1136e-05, 1.7363e-05, 2.5277e-05, 1.6654e-05,
3.6772e-05, 1.2167e-05, 1.2761e-05, 1.6654e-05, 1.4937e-05, 1.4741e-05,
2.1206e-05, 1.2418e-05, 2.1546e-05, 4.0832e-05, 1.9972e-05, 4.2867e-06,
2.0955e-05, 1.0037e-05, 6.7363e-05, 3.8995e-05, 8.4601e-06, 1.2769e-05,
2.8060e-05, 9.3079e-06, 1.4956e-05, 6.5231e-06, 5.5716e-06, 1.1619e-05,
7.5368e-06, 4.3650e-05, 2.0084e-05, 5.7119e-06, 6.9347e-06, 5.0213e-06,
2.9980e-05, 6.8005e-05, 3.6459e-05, 7.2730e-05, 6.4800e-06, 4.6253e-06,
2.3459e-05, 4.5730e-05, 3.0053e-06, 2.3565e-05, 1.8075e-05, 8.5421e-05,
1.31110e-05, 1.1910e-05, 2.3439e-05, 6.6901e-06, 2.1100e-05, 5.5513e-05,
1.4768e-05, 2.2875e-05, 1.9278e-05, 4.1115e-05, 3.3092e-05, 1.9332e-05,
2.7779e-05, 1.3807e-05, 4.1089e-06, 2.2783e-05, 1.1784e-05, 8.7700e-05,
1.0019e-05, 1.5616e-05, 1.6477e-05, 1.6362e-05, 6.1439e-05, 1.0712e-05,
6.4766e-06, 3.7354e-05, 1.3070e-05, 3.0018e-05, 1.4566e-05, 3.7210e-05
```

0s completed at 11:32 PM

vittavishnu.data202@vitstudent.ac.in
Managed by vitstudent.ac.in

Hi, Vitta Vishnu Datta!

Manage your Google Account

+ Add account Sign out

Privacy Policy Terms of Service

27°C Partly cloudy Search ENG IN 23:36 04-03-2024



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

The screenshot shows a Jupyter Notebook interface with several code cells. Cell [19] contains a snippet of PyTorch code. Cell [20] shows the download of the 'imagenet_classes.txt' file from GitHub. Cell [21] reads the categories from the downloaded file and prints the top 5 results for a Samoyed dog image.

```
[19]:  
1.0939e-04, 2.3742e-05, 4.4047e-05, 6.339e-06, 1.3457e-05, 2.0588e-05,  
1.1749e-05, 1.1546e-05, 2.4948e-05, 2.5005e-05], device='cuda:0')  
  
[20]: # Download ImageNet labels  
!wget https://raw.githubusercontent.com/pytorch/hub/master/imagenet_classes.txt  
  
2024-03-04 18:02:23 -> https://raw.githubusercontent.com/pytorch/hub/master/imagenet_classes.txt  
Receiving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.109.133, 185.199.110.133, 185.199.111.133, ...  
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)[185.199.109.133]:443... connected.  
HTTP request sent, awaiting response... 200 OK  
Length: 10472 (10K) [text/plain]  
Saving to: 'imagenet_classes.txt'  
  
 imagenet_classes.txt 100%[=====] 10.23K --.-KB/s   in 0s  
2024-03-04 18:02:24 (107 MB/s) - "imagenet_classes.txt" saved [10472/10472]  
  
[21]: # Read the categories  
with open("imagenet_classes.txt", "r") as f:  
    categories = [s.strip() for s in f.readlines()]  
# Show top 5 categories per image  
top5_prob, top5_catid = torch.topk(probabilities, 5)  
for i in range(top5_prob.size(0)):  
    print(categories[top5_catid[i]], top5_prob[i].item())  
  
Samoyed 0.9378382563591003  
Pomeranian 0.008283416740596294  
Great Pyrenees 0.0056563071302175522  
Arctic fox 0.005527289704501629  
white wolf 0.004741060547530651
```

Google Account Sidebar:

vittavishnu.data202@vitstudent.ac.in
Managed by vitstudent.ac.in

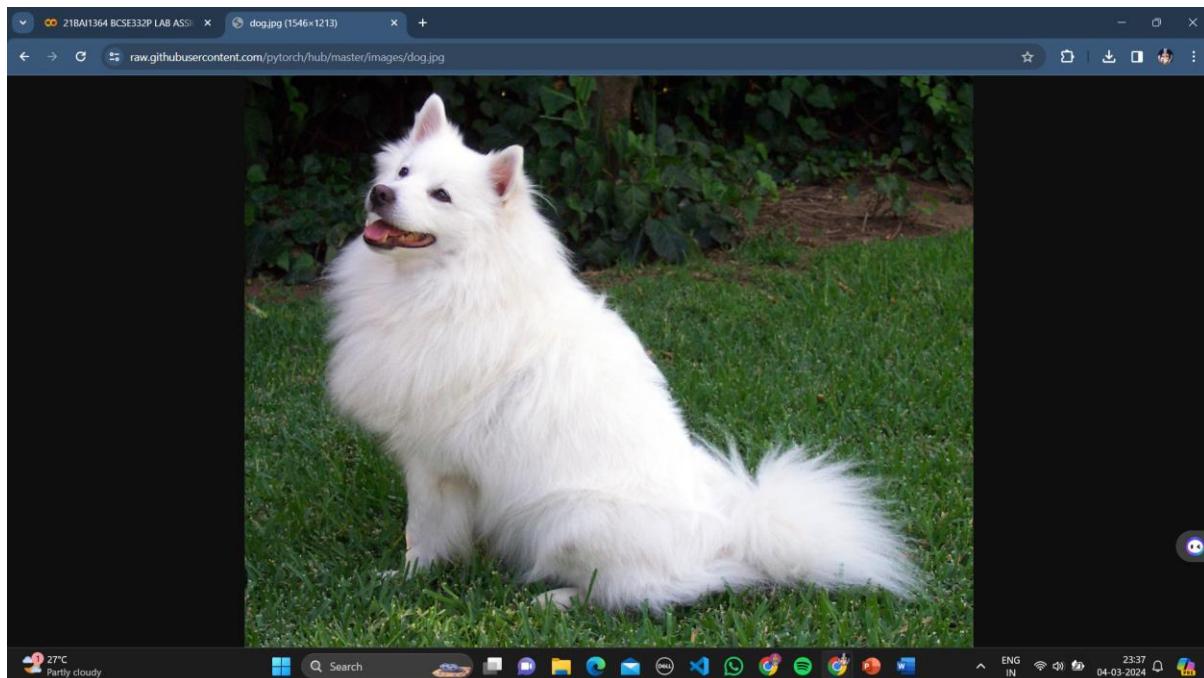
Hi, Vitta Vishnu Datta!

Manage your Google Account

+ Add account Sign out

Privacy Policy Terms of Service

DOG IMAGE:



CODE FOR PRE-TRAINED VGG 16 MODEL:



VIT[®]

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

```
import torch
import torch.nn as nn
import torchvision.models as models

# Define device (CPU or GPU)
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

class BasicBlock(nn.Module): # Basic building block for ResNet
    def __init__(self, in_channels, out_channels, stride=1):
        super(BasicBlock, self).__init__()

        # Define layers
        self.conv1 = nn.Conv2d(in_channels, out_channels,
kernel_size=3, stride=stride, padding=1, bias=False)
        self.bn1 = nn.BatchNorm2d(out_channels)
        self.relu = nn.ReLU(inplace=True)
        self.conv2 = nn.Conv2d(out_channels, out_channels,
kernel_size=3, padding=1, bias=False)
        self.bn2 = nn.BatchNorm2d(out_channels)

        # Shortcut connection for identity mapping (if dimensions
match)
        self.shortcut = nn.Sequential()
        if stride != 1 or in_channels != out_channels:
            self.shortcut = nn.Sequential(
                nn.Conv2d(in_channels, out_channels, kernel_size=1,
stride=stride, bias=False),
                nn.BatchNorm2d(out_channels)
            )

    def forward(self, x):
        # Forward pass through the block
        out = self.relu(self.bn1(self.conv1(x)))
        out = self.bn2(self.conv2(out))
        out += self.shortcut(x) # Add shortcut connection
        out = self.relu(out)
        return out

class ResNet(nn.Module): # Main ResNet architecture
    def __init__(self, block, num_blocks, num_classes=10):
        super(ResNet, self).__init__()
        self.in_channels = 64
```



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

```
# First convolutional layer
    self.conv1 = nn.Conv2d(3, 64, kernel_size=7, stride=2,
padding=3, bias=False)
    self.bn1 = nn.BatchNorm2d(64)
    self.relu = nn.ReLU(inplace=True)
    self.maxpool = nn.MaxPool2d(kernel_size=3, stride=2, padding=1)

    # Define residual layers
    self.layer1 = self._make_layer(block, 64, num_blocks[0])
    self.layer2 = self._make_layer(block, 128, num_blocks[1],
stride=2)
    self.layer3 = self._make_layer(block, 256, num_blocks[2],
stride=2)
    self.layer4 = self._make_layer(block, 512, num_blocks[3],
stride=2)

    # Final layers
    self.avgpool = nn.AdaptiveAvgPool2d((1, 1))
    self.fc = nn.Linear(512 * 1 * 1, num_classes) # Output layer
for desired number of classes

def _make_layer(self, block, out_channels, num_blocks, stride=1):
    # Helper function to create multiple residual blocks
    strides = [stride] + [1] * (num_blocks - 1) # Adjust strides
for the first and subsequent blocks
    layers = []
    for i in range(num_blocks):
        layers.append(block(self.in_channels, out_channels,
strides[i]))
        self.in_channels = out_channels # Update in_channels for
subsequent blocks
    return nn.Sequential(*layers)

def forward(self, x):
    # Forward pass through the network
    out = self.relu(self.bn1(self.conv1(x)))
    out = self.maxpool(out)
    out = self.layer1(out)
    out = self.layer2(out)
    out = self.layer3(out)
    out = self.layer4(out)
```



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

```
        out = self.avgpool(out)
        out = out.view(out.size(0), -1)  # Flatten for fully connected
layer
        out = self.fc(out)
        out

import tensorflow as tf
from tensorflow.keras import layers, models

def VGG16(input_shape=(224, 224, 3), num_classes=1000):
    model = models.Sequential()

    # Block 1
    model.add(layers.Conv2D(64, (3, 3), activation='relu',
padding='same', input_shape=input_shape))
    model.add(layers.Conv2D(64, (3, 3), activation='relu',
padding='same'))
    model.add(layers.MaxPooling2D((2, 2), strides=(2, 2)))

    # Block 2
    model.add(layers.Conv2D(128, (3, 3), activation='relu',
padding='same'))
    model.add(layers.Conv2D(128, (3, 3), activation='relu',
padding='same'))
    model.add(layers.MaxPooling2D((2, 2), strides=(2, 2)))

    # Block 3
    model.add(layers.Conv2D(256, (3, 3), activation='relu',
padding='same'))
    model.add(layers.Conv2D(256, (3, 3), activation='relu',
padding='same'))
    model.add(layers.Conv2D(256, (3, 3), activation='relu',
padding='same'))
    model.add(layers.MaxPooling2D((2, 2), strides=(2, 2)))

    # Block 4
    model.add(layers.Conv2D(512, (3, 3), activation='relu',
padding='same'))
    model.add(layers.Conv2D(512, (3, 3), activation='relu',
padding='same'))
    model.add(layers.Conv2D(512, (3, 3), activation='relu',
padding='same'))
    model.add(layers.MaxPooling2D((2, 2), strides=(2, 2)))
```



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

```
# Block 5
model.add(layers.Conv2D(512, (3, 3), activation='relu',
padding='same'))
model.add(layers.Conv2D(512, (3, 3), activation='relu',
padding='same'))
model.add(layers.Conv2D(512, (3, 3), activation='relu',
padding='same'))
model.add(layers.MaxPooling2D((2, 2), strides=(2, 2)))

# Flatten and fully connected layers
model.add(layers.Flatten())
model.add(layers.Dense(4096, activation='relu'))
model.add(layers.Dense(4096, activation='relu'))
model.add(layers.Dense(num_classes, activation='softmax'))

return model

# Example usage
input_shape = (224, 224, 3)
num_classes = 1000 # Change this to match your dataset's number of
classes
vgg16_model = VGG16(input_shape, num_classes)

# Print model summary
vgg16_model.summary()

#IMPLEMENT VGG16 FOR IMAGE RECOGNITION
import tensorflow as tf
from tensorflow.keras.applications import VGG16
from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications.vgg16 import
preprocess_input,decode_predictions
import numpy as np
import matplotlib.pyplot as plt
# Load the VGG16 model with pre-trained weights
model = VGG16(weights='imagenet')
# Load and preprocess the input image
img_path = '/content/paint.jpg'
img = image.load_img(img_path, target_size=(224, 224))
x = image.img_to_array(img)
x = np.expand_dims(x, axis=0)
```



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

```
x = preprocess_input(x)
# Perform inference
preds = model.predict(x)
# Decode and print the top-3 predicted labels
print('Predicted:', decode_predictions(preds, top=3)[0])
# Step 4: Visualize the features
plt.imshow(img)
plt.title('Input Image')
plt.axis('off')
plt.show()
```

SCREENSHOTS OF THE VGG16 MODEL:

The screenshot shows a Google Colab notebook titled "21BAI1364 BCSE332P LAB ASSIGN.5.ipynb". The code cell contains Python code for modifying a VGG16 neural network. A Google sign-in dialog box is overlaid on the right side of the screen, displaying the user's profile picture and name "vittavishnu.datta2021@vitstudent.ac.in". The dialog also includes options to "Manage your Google Account", "Add account", and "Sign out". The desktop taskbar at the bottom shows various application icons and system status.

```
import torch
import torchvision.models as models

# Define device (CPU or GPU)
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

# Define VGG16 model
model = models.vgg16(pretrained=True) # Optional: Load pre-trained weights

# Freeze convolutional layers to avoid training them (optional)
for param in model.features.parameters():
    param.requires_grad = False

# Modify the final layers for your desired number of output classes
num_classes = 10 # Replace with the actual number of classes
model.classifier[6] = torch.nn.Linear(in_features=4096, out_features=num_classes)

# Move model to chosen device
model.to(device)

# Define loss function and optimizer (adapt based on your task)
criterion = torch.nn.CrossEntropyLoss()
optimizer = torch.optim.SGD(model.classifier.parameters(), lr=0.001, momentum=0.9)

# Training and prediction steps (replace with your data handling and training logic)
# ...
```



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

```
# Training and prediction steps (replace with your data handling and training logic)
# ...

# Save the model (optional)
torch.save(model.state_dict(), "vgg16_model.pth")

# Define device (CPU or GPU)
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

class BasicBlock(nn.Module): # Basic building block for ResNet
    def __init__(self, in_channels, out_channels, stride=1):
        super(BasicBlock, self).__init__()

        # Define layers
        self.conv1 = nn.Conv2d(in_channels, out_channels, kernel_size=3, stride=stride, padding=1, bias=False)
        self.bn1 = nn.BatchNorm2d(out_channels)
        self.relu = nn.ReLU(inplace=True)
        self.conv2 = nn.Conv2d(out_channels, out_channels, kernel_size=3, padding=1, bias=False)
        self.bn2 = nn.BatchNorm2d(out_channels)

    def forward(self, x):
        # Forward pass through the block
        out = self.relu(self.bn1(self.conv1(x)))
        out = self.bn2(self.conv2(out))
        out += self.shortcut(x) # Add shortcut connection
        out = self.relu(out)
        return out

class ResNet(nn.Module): # Main ResNet architecture
    def __init__(self, block, num_blocks, num_classes=10):
        super(ResNet, self).__init__()
        self.in_channels = 64

        # First convolutional layer
        self.conv1 = nn.Conv2d(3, 64, kernel_size=7, stride=2, padding=3, bias=False)
        self.bn1 = nn.BatchNorm2d(64)
        self.relu = nn.ReLU(inplace=True)
        self.maxpool = nn.MaxPool2d(kernel_size=3, stride=2, padding=1)

        # Define residual layers
        self.layer1 = self._make_layer(block, 64, num_blocks[0])
        self.layer2 = self._make_layer(block, 128, num_blocks[1], stride=2)
        self.layer3 = self._make_layer(block, 256, num_blocks[2], stride=2)
        self.layer4 = self._make_layer(block, 512, num_blocks[3], stride=2)

        # Final layers
        self.avgpool = nn.AdaptiveAvgPool2d((1, 1))
        self.fc = nn.Linear(512 * 1 * 1, num_classes) # Output layer for desired number of classes

    def _make_layer(self, block, out_channels, num_blocks, stride=1):
        # Helper function to create multi-layer residual blocks
```

```
def forward(self, x):
    # Forward pass through the block
    out = self.relu(self.bn1(self.conv1(x)))
    out = self.bn2(self.conv2(out))
    out += self.shortcut(x) # Add shortcut connection
    out = self.relu(out)
    return out

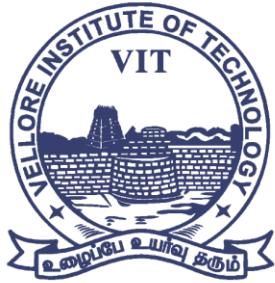
class ResNet(nn.Module): # Main ResNet architecture
    def __init__(self, block, num_blocks, num_classes=10):
        super(ResNet, self).__init__()
        self.in_channels = 64

        # First convolutional layer
        self.conv1 = nn.Conv2d(3, 64, kernel_size=7, stride=2, padding=3, bias=False)
        self.bn1 = nn.BatchNorm2d(64)
        self.relu = nn.ReLU(inplace=True)
        self.maxpool = nn.MaxPool2d(kernel_size=3, stride=2, padding=1)

        # Define residual layers
        self.layer1 = self._make_layer(block, 64, num_blocks[0])
        self.layer2 = self._make_layer(block, 128, num_blocks[1], stride=2)
        self.layer3 = self._make_layer(block, 256, num_blocks[2], stride=2)
        self.layer4 = self._make_layer(block, 512, num_blocks[3], stride=2)

        # Final layers
        self.avgpool = nn.AdaptiveAvgPool2d((1, 1))
        self.fc = nn.Linear(512 * 1 * 1, num_classes) # Output layer for desired number of classes

    def _make_layer(self, block, out_channels, num_blocks, stride=1):
        # Helper function to create multi-layer residual blocks
```



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

The screenshot shows a Google Colab notebook titled "21BAI1364 BCSE332P LAB ASSIGN.5.ipynb". The code implements a residual block using PyTorch. It defines a class `BasicBlock` that takes `in_channels`, `out_channels`, and `stride` as parameters. The block consists of two convolutional layers with batch normalization and ReLU activation, followed by a residual connection that adds the input to the output of the second layer. The forward pass method performs the forward pass through the block.

```
[ ] # Define device (CPU or GPU)
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

class BasicBlock(nn.Module): # Basic building block for ResNet
    def __init__(self, in_channels, out_channels, stride=1):
        super(BasicBlock, self).__init__()

        # Define layers
        self.conv1 = nn.Conv2d(in_channels, out_channels, kernel_size=3, stride=stride, padding=1, bias=False)
        self.bn1 = nn.BatchNorm2d(out_channels)
        self.relu = nn.ReLU(inplace=True)
        self.conv2 = nn.Conv2d(out_channels, out_channels, kernel_size=3, padding=1, bias=False)
        self.bn2 = nn.BatchNorm2d(out_channels)

        # Shortcut connection for identity mapping (if dimensions match)
        self.shortcut = nn.Sequential()
        if stride != 1 or in_channels != out_channels:
            self.shortcut = nn.Sequential(
                nn.Conv2d(in_channels, out_channels, kernel_size=1, stride=stride, bias=False),
                nn.BatchNorm2d(out_channels)
            )

    def forward(self, x):
        # Forward pass through the block
        out = self.relu(self.bn1(self.conv1(x)))
        out = self.bn2(self.conv2(out))
        out += self.shortcut(x) # Add shortcut connection
        out = self.relu(out)
        return out
```

The screenshot shows a Google Colab notebook titled "21BAI1364 BCSE332P LAB ASSIGN.5.ipynb". The code implements a residual network using PyTorch. It defines a class `ResNet` that takes `block`, `num_blocks`, and `num_classes` as parameters. The network consists of multiple residual blocks, each containing four layers: a convolutional layer with batch normalization and ReLU activation, a residual connection that adds the input to the output of the second layer, another convolutional layer with batch normalization and ReLU activation, and a final convolutional layer with batch normalization and ReLU activation. The forward pass method performs the forward pass through the network.

```
self.layer1 = self._make_layer(block, 64, num_blocks[0])
self.layer2 = self._make_layer(block, 128, num_blocks[1], stride=2)
self.layer3 = self._make_layer(block, 256, num_blocks[2], stride=2)
self.layer4 = self._make_layer(block, 512, num_blocks[3], stride=2)

# Final layers
self.avgpool = nn.AdaptiveAvgPool2d((1, 1))
self.fc = nn.Linear(512 * 1 * 1, num_classes) # Output layer for desired number of classes

def _make_layer(self, block, out_channels, num_blocks, stride=1):
    # Helper function to create multiple residual blocks
    strides = [stride] + [1] * (num_blocks - 1) # Adjust strides for the first and subsequent blocks
    layers = []
    for i in range(num_blocks):
        layers.append(block(self.in_channels, out_channels, strides[i]))
        self.in_channels = out_channels # Update in_channels for subsequent blocks
    return nn.Sequential(*layers)

def forward(self, x):
    # Forward pass through the network
    out = self.relu(self.bn1(self.conv1(x)))
    out = self.maxpool(out)
    out = self.layer1(out)
    out = self.layer2(out)
    out = self.layer3(out)
    out = self.layer4(out)
    out = self.avgpool(out)
    out = out.view(out.size(0), -1) # Flatten for fully connected layer
    out = self.fc(out)
```



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

21BAI1364 BCSE332P LAB ASSIGN.5.ipynb

```
import tensorflow as tf
from tensorflow.keras import layers, models

def VGG16(input_shape=(224, 224, 3), num_classes=1000):
    model = models.Sequential()

    # Block 1
    model.add(layers.Conv2D(64, (3, 3), activation='relu', padding='same', input_shape=input_shape))
    model.add(layers.Conv2D(64, (3, 3), activation='relu', padding='same'))
    model.add(layers.MaxPooling2D((2, 2), strides=(2, 2)))

    # Block 2
    model.add(layers.Conv2D(128, (3, 3), activation='relu', padding='same'))
    model.add(layers.Conv2D(128, (3, 3), activation='relu', padding='same'))
    model.add(layers.MaxPooling2D((2, 2), strides=(2, 2)))

    # Block 3
    model.add(layers.Conv2D(256, (3, 3), activation='relu', padding='same'))
    model.add(layers.Conv2D(256, (3, 3), activation='relu', padding='same'))
    model.add(layers.Conv2D(256, (3, 3), activation='relu', padding='same'))
    model.add(layers.MaxPooling2D((2, 2), strides=(2, 2)))

    # Block 4
    model.add(layers.Conv2D(512, (3, 3), activation='relu', padding='same'))
    model.add(layers.Conv2D(512, (3, 3), activation='relu', padding='same'))
    model.add(layers.Conv2D(512, (3, 3), activation='relu', padding='same'))
    model.add(layers.MaxPooling2D((2, 2), strides=(2, 2)))

    # Block 5
    model.add(layers.Conv2D(512, (3, 3), activation='relu', padding='same'))
    model.add(layers.Conv2D(512, (3, 3), activation='relu', padding='same'))
    model.add(layers.MaxPooling2D((2, 2), strides=(2, 2)))
```

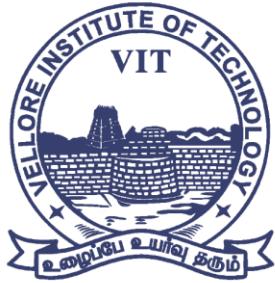
21BAI1364 BCSE332P LAB ASSIGN.5.ipynb

```
# Block 4
model.add(layers.Conv2D(512, (3, 3), activation='relu', padding='same'))
model.add(layers.Conv2D(512, (3, 3), activation='relu', padding='same'))
model.add(layers.Conv2D(512, (3, 3), activation='relu', padding='same'))
model.add(layers.MaxPooling2D((2, 2), strides=(2, 2)))

# Block 5
model.add(layers.Conv2D(512, (3, 3), activation='relu', padding='same'))
model.add(layers.Conv2D(512, (3, 3), activation='relu', padding='same'))
model.add(layers.Conv2D(512, (3, 3), activation='relu', padding='same'))
model.add(layers.MaxPooling2D((2, 2), strides=(2, 2)))

# Flatten and fully connected layers
model.add(layers.Flatten())
model.add(layers.Dense(4096, activation='relu'))
model.add(layers.Dense(4096, activation='relu'))
model.add(layers.Dense(num_classes, activation='softmax'))
```

Model: "sequential_1"



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

21BAI1364 BCSE332P LAB ASSIGN.5.ipynb

Model: "sequential_1"

Layer (type)	Output Shape	Param #
conv2d_13 (Conv2D)	(None, 224, 224, 64)	1792
conv2d_14 (Conv2D)	(None, 224, 224, 64)	36928
max_pooling2d_5 (MaxPooling2D)	(None, 112, 112, 64)	0
conv2d_15 (Conv2D)	(None, 112, 112, 128)	73856
conv2d_16 (Conv2D)	(None, 112, 112, 128)	147584
max_pooling2d_6 (MaxPooling2D)	(None, 56, 56, 128)	0
conv2d_17 (Conv2D)	(None, 56, 56, 256)	295168
conv2d_18 (Conv2D)	(None, 56, 56, 256)	590080
conv2d_19 (Conv2D)	(None, 56, 56, 256)	590080
max_pooling2d_7 (MaxPooling2D)	(None, 28, 28, 256)	0
conv2d_20 (Conv2D)	(None, 28, 28, 512)	1180160
conv2d_21 (Conv2D)	(None, 28, 28, 512)	2359088
conv2d_22 (Conv2D)	(None, 28, 28, 512)	2359088
max_pooling2d_8 (MaxPooling2D)	(None, 14, 14, 512)	0

1s completed at 11:14PM

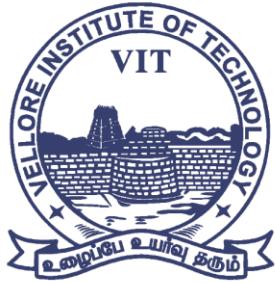
21BAI1364 BCSE332P LAB ASSIGN.5.ipynb

[9] max_pooling2d_7 (MaxPooling2D)

Layer (type)	Output Shape	Param #
conv2d_20 (Conv2D)	(None, 28, 28, 512)	1180160
conv2d_21 (Conv2D)	(None, 28, 28, 512)	2359088
conv2d_22 (Conv2D)	(None, 28, 28, 512)	2359088
max_pooling2d_8 (MaxPooling2D)	(None, 14, 14, 512)	0
conv2d_23 (Conv2D)	(None, 14, 14, 512)	2359088
conv2d_24 (Conv2D)	(None, 14, 14, 512)	2359088
conv2d_25 (Conv2D)	(None, 14, 14, 512)	2359088
max_pooling2d_9 (MaxPooling2D)	(None, 7, 7, 512)	0
flatten_1 (Flatten)	(None, 25088)	0
dense_3 (Dense)	(None, 4096)	102764544
dense_4 (Dense)	(None, 4096)	16781312
dense_5 (Dense)	(None, 1000)	4097000

Total params: 138357544 (527.79 MB)
Trainable params: 138357544 (527.79 MB)
Non-trainable params: 0 (0.00 Byte)

1s completed at 11:14PM



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

The screenshot shows a Google Colab notebook titled "21BAI1364 BCSE332P LAB ASSIGN.5.ipynb". The code cell contains Python code for image recognition using the VGG16 model. The sidebar shows a Google Account profile for "vittavishnu.data2021@vitstudent.ac.in".

```
#IMPLEMENT VGG16 FOR IMAGE RECOGNITION
import tensorflow as tf
from tensorflow.keras.applications import VGG16
from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications.vgg16 import preprocess_input,decode_predictions
import numpy as np
import matplotlib.pyplot as plt
# Load the VGG16 model with pre-trained weights
model = VGG16(weights='imagenet')
# Load and preprocess the input image
img_path = '/content/paint.jpg'
img = image.load_img(img_path, target_size=(224, 224))
x = image.img_to_array(img)
x = np.expand_dims(x, axis=0)
x = preprocess_input(x)
# Perform inference
preds = model.predict(x)
# Decode and print the top-3 predicted labels
print('Predicted:', decode_predictions(preds, top=3)[0])
# Step 4: Visualize the features
plt.imshow(img)
plt.title('Input Image')
plt.axis('off')
plt.show()
```

1/1 [██████████] = 1s 1s/step
Downloading data from https://storage.googleapis.com/download.tensorflow.org/data/imagenet_class_index.json

https://accounts.google.com/SignOutOptions?hl=en&continue=https://colab.research.google.com/drive/14n5SwP1q5zy50pKOwn-VKLW/JClcSdE?usp=chrome_ntp&scrollTo=V5nk3d2Bcppo

The screenshot shows the same Google Colab notebook. The code cell has been run, and the output displays the predicted class and a visualization of the input image. The sidebar shows the same Google Account profile.

```
# Step 4: Visualize the features
plt.imshow(img)
plt.title('Input Image')
plt.axis('off')
plt.show()
```

1/1 [██████████] = 1s 1s/step
Downloading data from https://storage.googleapis.com/download.tensorflow.org/data/imagenet_class_index.json
3536/35363 [██████████] = 0s 0us/step
predicted: [('n0476259', 'tray', 0.2080151), ('n03063599', 'coffee_mug', 0.13549), ('n03876231', 'paintbrush', 0.10177331)]

Input Image

https://accounts.google.com/SignOutOptions?hl=en&continue=https://colab.research.google.com/drive/14n5SwP1q5zy50pKOwn-VKLW/JClcSdE?usp=chrome_ntp&ec=GBRAqQM

CODE FOR PRE-TRAINED ResNet MODEL:



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

```
import tensorflow as tf
from tensorflow.keras import layers, models

def residual_block(x, filters, kernel_size=3, stride=1):
    # Shortcut connection
    shortcut = x

    # First convolution layer
    x = layers.Conv2D(filters, kernel_size, strides=stride,
padding='same')(x)
    x = layers.BatchNormalization()(x)
    x = layers.Activation('relu')(x)

    # Second convolution layer
    x = layers.Conv2D(filters, kernel_size, padding='same')(x)
    x = layers.BatchNormalization()(x)

    # Shortcut connection
    if stride != 1 or shortcut.shape[-1] != filters:
        shortcut = layers.Conv2D(filters, (1, 1), strides=stride,
padding='same')(shortcut)
        shortcut = layers.BatchNormalization()(shortcut)

    # Add the shortcut to the main path
    x = layers.Add()([x, shortcut])
    x = layers.Activation('relu')(x)

    return x

def ResNet(input_shape=(224, 224, 3), num_classes=1000):
    inputs = layers.Input(shape=input_shape)

    # Initial convolution and max pooling
    x = layers.Conv2D(64, (7, 7), strides=2, padding='same')(inputs)
    x = layers.BatchNormalization()(x)
    x = layers.Activation('relu')(x)
    x = layers.MaxPooling2D((3, 3), strides=2, padding='same')(x)

    # Residual blocks
    x = residual_block(x, filters=64)
    x = residual_block(x, filters=64)
    x = residual_block(x, filters=128, stride=2)
```



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

```
x = residual_block(x, filters=128)
x = residual_block(x, filters=256, stride=2)
x = residual_block(x, filters=256)
x = residual_block(x, filters=512, stride=2)
x = residual_block(x, filters=512)

# Global average pooling and dense layer
x = layers.GlobalAveragePooling2D()(x)
x = layers.Dense(num_classes, activation='softmax')(x)

model = models.Model(inputs, x, name='resnet')

return model

# Example usage
input_shape = (224, 224, 3)
num_classes = 1000 # Change this to match your dataset's number of
classes
resnet_model = ResNet(input_shape, num_classes)

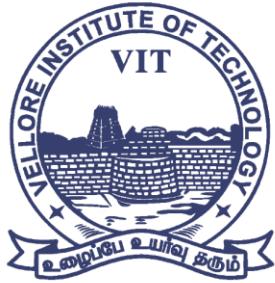
# Print model summary
resnet_model.summary()

import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.applications import ResNet50
from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications.resnet50 import preprocess_input,
decode_predictions

# Step 1: Load the ResNet50 model
model = ResNet50(weights='imagenet', include_top=True)

# Step 2: Load and preprocess the input image
img_path = '/content/monalisa.jpg'
img = image.load_img(img_path, target_size=(224, 224))
x = image.img_to_array(img)
x = np.expand_dims(x, axis=0)
x = preprocess_input(x)

# Step 3: Pass the preprocessed image through the ResNet50 model
```



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

```
features = model.predict(x)

# Step 4: Visualize the input image
plt.imshow(img)
plt.title('Input Image')
plt.axis('off')
plt.show()

# Step 5: Classify the image
preds = decode_predictions(features, top=3)[0]

# Step 6: Display the results
for pred in preds:
    print(f'{pred[1]}: {pred[2]}')
```

SCREENSHOTS OF THE ResNet MODEL:

A screenshot of a Google Colab notebook titled "21BAI1364 BCSE332P LAB ASSIGN.5.ipynb". The code cell contains Python code for a residual block in TensorFlow/Keras. A modal window from Google Account is overlaid on the right side of the screen, displaying a profile picture, the name "Vitta Vishnu Datta!", and options to "Manage your Google Account", "Add account", and "Sign out". The status bar at the bottom shows disk usage, weather (82°F, mostly cloudy), and system icons.



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

The screenshot shows a Google Colab notebook titled "21BAI1364 BCSE332P LAB ASSIGN.5.ipynb". The code implements a ResNet model:

```
# Add the shortcut to the main path
x = layers.Add([x, shortcut])
x = layers.Activation('relu')(x)

return x

def ResNet(input_shape=(224, 224, 3), num_classes=1000):
    inputs = layers.Input(shape=input_shape)

    # Initial convolution and max pooling
    x = layers.Conv2D(64, (7, 7), strides=2, padding='same')(inputs)
    x = layers.BatchNormalization()(x)
    x = layers.Activation('relu')(x)
    x = layers.MaxPooling2D((3, 3), strides=2, padding='same')(x)

    # Residual blocks
    x = residual_block(x, filters=64)
    x = residual_block(x, filters=64)
    x = residual_block(x, filters=128, stride=2)
    x = residual_block(x, filters=128)
    x = residual_block(x, filters=256, stride=2)
    x = residual_block(x, filters=256)
    x = residual_block(x, filters=512, stride=2)
    x = residual_block(x, filters=512)

    # Global average pooling and dense layer
    x = layers.GlobalAveragePooling2D()(x)
    x = layers.Dense(num_classes, activation='softmax')(x)

    model = models.Model(inputs, x, name='resnet')

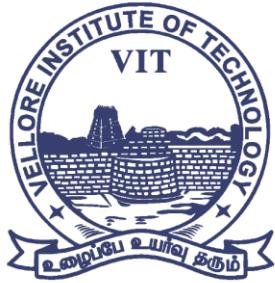
    return model
```

The code is completed at 11:45PM.

The screenshot shows the same Google Colab notebook. The code remains the same, but a "Model: 'resnet'" summary table is displayed below the code cell:

Layer (type)	Output Shape	Param #	Connected to
Input	(None, 224, 224, 3)	0	
Conv2D	(None, 112, 112, 64)	3584	1
Batch Normalization	(None, 112, 112, 64)	256	1
ReLU	(None, 112, 112, 64)	0	1
MaxPooling2D	(None, 56, 56, 64)	0	1
Residual block	(None, 56, 56, 64)	3584	2
Residual block	(None, 56, 56, 64)	3584	3
Residual block	(None, 56, 56, 128)	14336	4
Residual block	(None, 28, 28, 128)	14336	5
Residual block	(None, 28, 28, 256)	28672	6
Residual block	(None, 14, 14, 256)	28672	7
Residual block	(None, 14, 14, 512)	57344	8
Residual block	(None, 14, 14, 512)	57344	9
Global Average Pooling 2D	(None, 512)	0	10
Dense	(None, 1000)	512000	11

The code is completed at 11:45PM.



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

21BAI1364 BCSE332P LAB ASSIGN.5.ipynb

All changes saved

Model: "resnet"

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	[None, 224, 224, 3]	0	[]
conv2d_26 (Conv2D)	(None, 112, 112, 64)	9472	['input_1[0][0]']
batch_normalization (Batch Normalization)	(None, 112, 112, 64)	256	['conv2d_26[0][0]']
activation (Activation)	(None, 112, 112, 64)	0	['batch_normalization[0][0]']
max_pooling2d_10 (MaxPooling2D)	(None, 56, 56, 64)	0	['activation[0][0]']
conv2d_27 (Conv2D)	(None, 56, 56, 64)	36928	['max_pooling2d_10[0]']
batch_normalization_1 (Batch Normalization)	(None, 56, 56, 64)	256	['conv2d_27[0][0]']
activation_1 (Activation)	(None, 56, 56, 64)	0	['batch_normalization_1[0][0]']
conv2d_28 (Conv2D)	(None, 56, 56, 64)	36928	['activation_1[0][0]']
batch_normalization_2 (Batch Normalization)	(None, 56, 56, 64)	256	['conv2d_28[0][0]']
add (Add)	(None, 56, 56, 64)	0	['batch_normalization_2[0][0]', 'max_pooling2d_10[0][0]']
activation_2 (Activation)	(None, 56, 56, 64)	0	['add[0][0]']

vittavishnu.data2021@vitstudent.ac.in
Managed by vitstudent.ac.in

Hi, Vitta Vishnu Datta!
Manage your Google Account

+ Add account | Sign out

Privacy Policy · Terms of Service

82°F Mostly cloudy

Search

23:47 07-03-2024

21BAI1364 BCSE332P LAB ASSIGN.5.ipynb

All changes saved

activation_2 (Activation) (None, 56, 56, 64) 0 ['add[0][0]']

conv2d_29 (Conv2D) (None, 56, 56, 64) 36928 ['activation_2[0][0]']

batch_normalization_3 (Batch Normalization) (None, 56, 56, 64) 256 ['conv2d_29[0][0]']

activation_3 (Activation) (None, 56, 56, 64) 0 ['batch_normalization_3[0][0]']

conv2d_30 (Conv2D) (None, 56, 56, 64) 36928 ['activation_3[0][0]']

batch_normalization_4 (Batch Normalization) (None, 56, 56, 64) 256 ['conv2d_30[0][0]']

add_1 (Add) (None, 56, 56, 64) 0 ['batch_normalization_4[0][0]', 'activation_2[0][0]']

activation_4 (Activation) (None, 56, 56, 64) 0 ['add_1[0][0]']

conv2d_31 (Conv2D) (None, 28, 28, 128) 73856 ['activation_4[0][0]']

batch_normalization_5 (Batch Normalization) (None, 28, 28, 128) 512 ['conv2d_31[0][0]']

activation_5 (Activation) (None, 28, 28, 128) 0 ['batch_normalization_5[0][0]']

conv2d_32 (Conv2D) (None, 28, 28, 128) 147584 ['activation_5[0][0]']

conv2d_33 (Conv2D) (None, 28, 28, 128) 8320 ['activation_4[0][0]']

batch_normalization_6 (Batch Normalization) (None, 28, 28, 128) 512 ['conv2d_32[0][0]']

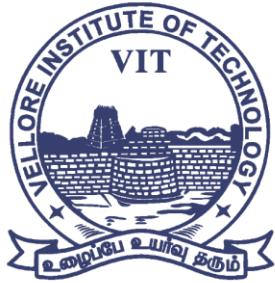
Google Account
Vitta Vishnu Datta 21BAI1364
vittavishnu.data2021@vitstudent.ac.in

https://accounts.google.com/SignOutOptions?hl=en&continue=https://colab.research.google.com/drive/14n5SwP1qSzy50pKOwn-VKLWJClLcSdE#ec=GBRAqQM

82°F Mostly cloudy

Search

23:48 07-03-2024



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

The screenshot shows a Google Colab notebook titled "21BAI1364 BCSE332P LAB ASSIGN.5.ipynb". The code is a list of operations and their corresponding details:

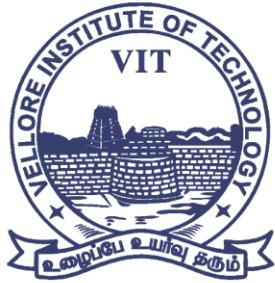
Operation	Details	Count	Code
batch_normalization_6	(Batch Normalization)	512	['conv2d_32[0][0]']
batch_normalization_7	(Batch Normalization)	512	['conv2d_33[0][0]']
add_2	(Add)	0	['batch_normalization_6[0][0]', 'batch_normalization_7[0][0]']
activation_6	(Activation)	0	['add_2[0][0]']
conv2d_34	(Conv2D)	147584	['activation_6[0][0]']
batch_normalization_8	(Batch Normalization)	512	['conv2d_34[0][0]']
activation_7	(Activation)	0	['batch_normalization_8[0][0]']
conv2d_35	(Conv2D)	147584	['activation_7[0][0]']
batch_normalization_9	(Batch Normalization)	512	['conv2d_35[0][0]']
add_3	(Add)	0	['batch_normalization_9[0][0]', 'activation_6[0][0]']
activation_8	(Activation)	0	['add_3[0][0]']
conv2d_36	(Conv2D)	295168	['activation_8[0][0]']
batch_normalization_10	(Batch Normalization)	1024	['conv2d_36[0][0]']

At the bottom, it says "1s completed at 11:45PM". The system tray shows a weather icon (82°F Mostly cloudy), a battery level (80.87 GB available), and a timestamp (07-03-2024 23:48).

The screenshot shows a Google Colab notebook titled "21BAI1364 BCSE332P LAB ASSIGN.5.ipynb". The code is a list of operations and their corresponding details:

Operation	Details	Count	Code
batch_normalization_10	(Batch Normalization)	1024	['conv2d_36[0][0]']
activation_9	(Activation)	0	['batch_normalization_10[0][0]']
conv2d_37	(Conv2D)	590080	['activation_9[0][0]']
conv2d_38	(Conv2D)	33824	['activation_8[0][0]']
batch_normalization_11	(Batch Normalization)	1024	['conv2d_37[0][0]']
batch_normalization_12	(Batch Normalization)	1024	['conv2d_38[0][0]']
add_4	(Add)	0	['batch_normalization_11[0][0]', 'batch_normalization_12[0][0]']
activation_10	(Activation)	0	['add_4[0][0]']
conv2d_39	(Conv2D)	590080	['activation_10[0][0]']
batch_normalization_13	(Batch Normalization)	1024	['conv2d_39[0][0]']
activation_11	(Activation)	0	['batch_normalization_13[0][0]']
conv2d_40	(Conv2D)	590080	['activation_11[0][0]']
batch_normalization_14	(Batch Normalization)	1024	['conv2d_40[0][0]']

At the bottom, it says "1s completed at 11:45PM". The system tray shows a weather icon (82°F Mostly cloudy), a battery level (80.87 GB available), and a timestamp (07-03-2024 23:48).



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

The screenshot shows a Google Colab notebook titled "21BAI1364 BCSE332P LAB ASSIGN.5.ipynb". The code displays a series of operations, likely part of a neural network's forward pass:

```
batch_normalization_14 (Ba (None, 14, 14, 256) 1024 ['conv2d_40[0][0]']
tchNormalization)
add_5 (Add) (None, 14, 14, 256) 0 ['batch_normalization_14[0][0]
activation_10[0][0]']
activation_12 (Activation) (None, 14, 14, 256) 0 ['add_5[0][0]']
conv2d_41 (Conv2D) (None, 7, 7, 512) 1180160 ['activation_12[0][0]']
batch_normalization_15 (Ba (None, 7, 7, 512) 2048 ['conv2d_41[0][0]']
tchNormalization)
activation_13 (Activation) (None, 7, 7, 512) 0 ['batch_normalization_15[0][0]
']
conv2d_42 (Conv2D) (None, 7, 7, 512) 2359888 ['activation_13[0][0]']
conv2d_43 (Conv2D) (None, 7, 7, 512) 131584 ['activation_12[0][0]']
batch_normalization_16 (Ba (None, 7, 7, 512) 2048 ['conv2d_42[0][0]']
tchNormalization)
batch_normalization_17 (Ba (None, 7, 7, 512) 2048 ['conv2d_43[0][0]']
tchNormalization)
add_6 (Add) (None, 7, 7, 512) 0 ['batch_normalization_16[0][0]
batch_normalization_17[0][0]
']
activation_14 (Activation) (None, 7, 7, 512) 0 ['add_6[0][0]']
```

The notebook interface includes a file browser, code editor, and various tools. The status bar at the bottom shows "82°F Mostly cloudy" and the date "07-03-2024".

The screenshot shows the same Google Colab notebook after execution. The code now includes a summary of parameters:

```
activation_14 (Activation) (None, 7, 7, 512) 0 ['add_6[0][0]']
conv2d_44 (Conv2D) (None, 7, 7, 512) 2359888 ['activation_14[0][0]']
batch_normalization_18 (Ba (None, 7, 7, 512) 2048 ['conv2d_44[0][0]']
tchNormalization)
activation_15 (Activation) (None, 7, 7, 512) 0 ['batch_normalization_18[0][0]
']
conv2d_45 (Conv2D) (None, 7, 7, 512) 2359888 ['activation_15[0][0]']
batch_normalization_19 (Ba (None, 7, 7, 512) 2048 ['conv2d_45[0][0]']
tchNormalization)
add_7 (Add) (None, 7, 7, 512) 0 ['batch_normalization_19[0][0]
activation_14[0][0]']
activation_16 (Activation) (None, 7, 7, 512) 0 ['add_7[0][0]']
global_average_pooling2d ( (None, 512) 0 ['activation_16[0][0]']
GlobalAveragePooling2D)
dense_6 (Dense) (None, 1000) 513000 ['global_average_pooling2d[0][0]']

Total params: 11703912 (44.65 MB)
Trainable params: 11694312 (44.61 MB)
Non-trainable params: 9600 (37.50 KB)
```

The notebook interface and system status are identical to the first screenshot.



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

```
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.applications import ResNet50
from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications.resnet50 import preprocess_input, decode_predictions

# Step 1: Load the ResNet50 model
model = ResNet50(weights='imagenet', include_top=True)

# Step 2: Load and preprocess the input image
img_path = '/content/monalisa.jpg'
img = image.load_img(img_path, target_size=(224, 224))
x = image.img_to_array(img)
x = np.expand_dims(x, axis=0)
x = preprocess_input(x)

# Step 3: Pass the preprocessed image through the ResNet50 model
features = model.predict(x)

# Step 4: Visualize the input image
plt.imshow(img)
plt.title('Input Image')
plt.axis('off')
plt.show()

# Step 5: Classify the image
preds = decode_predictions(features, top=3)[0]
```

```
from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications.resnet50 import preprocess_input, decode_predictions

# Step 1: Load the ResNet50 model
model = ResNet50(weights='imagenet', include_top=True)

# Step 2: Load and preprocess the input image
img_path = '/content/monalisa.jpg'
img = image.load_img(img_path, target_size=(224, 224))
x = image.img_to_array(img)
x = np.expand_dims(x, axis=0)
x = preprocess_input(x)

# Step 3: Pass the preprocessed image through the ResNet50 model
features = model.predict(x)

# Step 4: Visualize the input image
plt.imshow(img)
plt.title('Input Image')
plt.axis('off')
plt.show()

# Step 5: Classify the image
preds = decode_predictions(features, top=3)[0]

# Step 6: Display the results
for pred in preds:
    print(f'{pred[1]}: {pred[2]}')
```



VIT[®]

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

The screenshot shows a Jupyter Notebook interface in Google Colab. The code cell contains:

```
for pred in preds:  
    print(f'{pred[1]}: {pred[2]}')
```

The output cell shows the execution progress: 1/1 [██████████] - 1s 837ms/step. Below it is the generated image titled "Input Image", which is a reproduction of the Mona Lisa painting.

At the bottom of the notebook, there is some generated text:

book_jacket: 0.21421214938163757
spotlight: 0.14951694011688232
comic_book: 0.1476392298936844

A sidebar on the right displays a user profile for "vittavishnu.datta2021@vitstudent.ac.in". The profile picture is a cartoon character, and the name is "Hi, Vitta Vishnu Datta!". There are buttons for "Manage your Google Account", "+ Add account", and "Sign out". Below the sidebar, links for "Privacy Policy" and "Terms of Service" are visible.

COMPARING RESULTS OF VGG16 & GOOGLE NET MODELS:

```
import tensorflow as tf
from tensorflow.keras.datasets import cifar10
from tensorflow.keras import layers, models
from tensorflow.keras.applications import VGG16, InceptionV3
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from sklearn.model_selection import GridSearchCV

# Function to define and train a model with early stopping
def train_model(model, train_datagen, train_images, train_labels,
test_images, test_labels, epochs=10):
    # Define early stopping callback
    early_stopping = tf.keras.callbacks.EarlyStopping(monitor='val_loss',
patience=3)
```



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

```
# Train the model
model.fit(train_datagen.flow(train_images, train_labels,
batch_size=64,
            validation_data=(test_images, test_labels),
            epochs=epochs,
            callbacks=[early_stopping])

# Evaluate the model
results = model.evaluate(test_images, test_labels)
return results

# Load and preprocess CIFAR-10 data
(train_images, train_labels), (test_images, test_labels) =
cifar10.load_data()
train_images = tf.image.resize(train_images, (75, 75)) # Resize images
(Option 1)
test_images = tf.image.resize(test_images, (75, 75)) # Resize images
(Option 1)
train_images, test_images = train_images / 255.0, test_images / 255.0

# Define data augmentation for training
train_datagen = ImageDataGenerator(
    rotation_range=15,
    width_shift_range=0.1,
    height_shift_range=0.1,
    horizontal_flip=True,
    vertical_flip=False,
    fill_mode='nearest'
)

# Define common parameters
input_shape = (75, 75, 3) # Updated input shape after resizing (Option 1)
num_classes = 10

# Function to define a model based on a pre-trained model choice
def create_model(model_choice, input_shape=input_shape):
    if model_choice == 'VGG16':
        # VGG16 model
        base_model = VGG16(weights='imagenet', include_top=False,
input_shape=input_shape)
```



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

```
base_model.trainable = False

model = models.Sequential([
    base_model,
    layers.Flatten(),
    layers.Dense(512, activation='relu'),
    layers.Dropout(0.5),
    layers.Dense(num_classes, activation='softmax')
])
elif model_choice == 'InceptionV3':

    # Inception (GoogleNet) model
    base_model = InceptionV3(weights='imagenet', include_top=False,
input_shape=input_shape)
    base_model.trainable = False

model = models.Sequential([
    base_model,
    layers.GlobalAveragePooling2D(),
    layers.Dense(512, activation='relu'),
    layers.Dropout(0.5),
    layers.Dense(num_classes, activation='softmax')
])
else:
    raise ValueError(f"Invalid model choice: {model_choice}")
model.compile(optimizer='adam',
loss='sparse_categorical_crossentropy', metrics=['accuracy'])
return model

# Define models for comparison (modify list to include others)
model_choices = ['VGG16', 'InceptionV3']

# Hyperparameter tuning with GridSearchCV (optional)
param_grid = {
    'epochs': [10, 20, 30],
    'learning_rate': [0.001, 0.0001]
}

for model_choice in model_choices:
    print(f"\nEvaluating Model: {model_choice}")

    # Create the model
```



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

```
model = create_model(model_choice)

# Hyperparameter tuning with GridSearchCV (uncomment to activate)
# grid_search = GridSearchCV(model, param_grid, cv=3)
# grid_search.fit(train_datagen.flow(train_images, train_labels,
batch_size=64), epochs=epochs)
# means = grid_search.cv_results_['mean_test_score']
# std = grid_search.cv_results_['std']
```

SCREENSHOTS OF THE CODE SNIPPET:

The screenshot shows a Google Colab notebook titled "21BAI1364 BCSE332P LAB ASSIGN.5.ipynb". The code cell contains the provided Python script for training a VGG16 model on CIFAR-10 data. The notebook interface includes a sidebar with code and text tabs, and a toolbar with various icons. The status bar at the bottom shows the user's account information, the URL of the notebook, and system status like battery level and network connection.

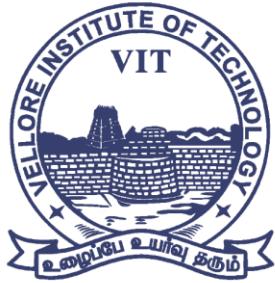
```
import tensorflow as tf
from tensorflow.keras.datasets import cifar10
from tensorflow.keras import layers, models
from tensorflow.keras.applications import VGG16, InceptionV3
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from sklearn.model_selection import GridSearchCV

# Function to define and train a model with early stopping
def train_model(model, train_datagen, train_labels, test_images, test_labels, epochs=10):
    # Define early stopping callback
    early_stopping = tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience=3)

    # Train the model
    model.fit(train_datagen.flow(train_images, train_labels, batch_size=64),
              validation_data=(test_images, test_labels),
              epochs=epochs,
              callbacks=[early_stopping])

    # Evaluate the model
    results = model.evaluate(test_images, test_labels)
    return results

# Load and preprocess CIFAR-10 data
(train_images, train_labels), (test_images, test_labels) = cifar10.load_data()
```



VIT[®]

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

Load and preprocess CIFAR-10 data
(train_images, train_labels) = cifar10.load_data()
train_images = tf.image.resize(train_images, (75, 75)) # Resize images (Option 1)
test_images = tf.image.resize(test_images, (75, 75)) # Resize images (Option 1)
train_images, test_images = train_images / 255.0, test_images / 255.0

Define data augmentation for training
train_datagen = ImageDataGenerator(
 rotation_range=15,
 width_shift_range=0.1,
 height_shift_range=0.1,
 horizontal_flip=True,
 vertical_flip=False,
 fill_mode='nearest'
)

Define common parameters
input_shape = (75, 75, 3) # Updated input shape after resizing (Option 1)
num_classes = 10

Function to define a model based on a pre-trained model choice
def create_model(model_choice, input_shape):
 if model_choice == 'VGG16':
 # VGG16 model
 base_model = VGG16(weights='imagenet', include_top=False, input_shape=input_shape)
 base_model.trainable = False

 model = models.Sequential([
 base_model,
 layers.Flatten(),
 layers.Dense(512, activation='relu'),
 layers.Dropout(0.5),
 layers.Dense(num_classes, activation='softmax')
])
 elif model_choice == 'InceptionV3':
 # Inception (GoogleNet) model
 base_model = InceptionV3(weights='imagenet', include_top=False, input_shape=input_shape)
 base_model.trainable = False

 model = models.Sequential([
 base_model,
 layers.GlobalAveragePooling2D(),
 layers.Dense(512, activation='relu'),
 layers.Dropout(0.5),
 layers.Dense(num_classes, activation='softmax')
])
 else:
 raise ValueError(f"invalid model choice: {model_choice}")
 model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
 return model

Define models for comparison (modify list to include others)
model_choices = ['VGG16', 'InceptionV3']

Hyperparameter tuning with GridSearchCV (optional)
param_grid = {

model = models.Sequential([
 base_model,
 layers.Flatten(),
 layers.Dense(512, activation='relu'),
 layers.Dropout(0.5),
 layers.Dense(num_classes, activation='softmax')
)
elif model_choice == 'InceptionV3':
 # Inception (GoogleNet) model
 base_model = InceptionV3(weights='imagenet', include_top=False, input_shape=input_shape)
 base_model.trainable = False

 model = models.Sequential([
 base_model,
 layers.GlobalAveragePooling2D(),
 layers.Dense(512, activation='relu'),
 layers.Dropout(0.5),
 layers.Dense(num_classes, activation='softmax')
])
else:
 raise ValueError(f"invalid model choice: {model_choice}")
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
return model

Define models for comparison (modify list to include others)
model_choices = ['VGG16', 'InceptionV3']

Hyperparameter tuning with GridSearchCV (optional)
param_grid = {



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

The screenshot shows a Google Colab notebook titled "21BAI1364 BCSE332P LAB ASSIGN.5.ipynb". The code in the cell is for creating a model and performing hyperparameter tuning with GridSearchCV. It includes code to download VGG16 and InceptionV3 weights from Google Cloud Storage. The output shows the progress of the downloads.

```
# create the model
model = create_model(model_choice)

# Hyperparameter tuning with GridSearchCV (uncomment to activate)
# grid_search = GridSearchCV(model, param_grid, cv=3)
# grid_search.fit(train_datagen.flow(train_images, train_labels, batch_size=64), epochs=epochs)
# means = grid_search.cv_results_['mean_test_score']
# std = grid_search.cv_results_['std']

Downloading data from https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz
178498071/178498071 [=====] - 4s 0us/step

Evaluating Model: VGG16
Downloading data from https://storage.googleapis.com/tensorflow-keras-applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels_notop.h5
58889256/58889256 [=====] - 0s 0us/step

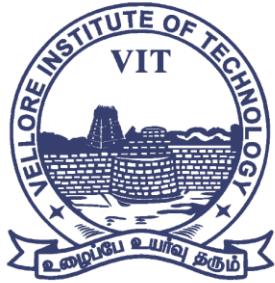
Evaluating Model: InceptionV3
Downloading data from https://storage.googleapis.com/tensorflow-keras-applications/inception_v3/inception_v3_weights_tf_dim_ordering_tf_kernels_notop.h5
87910968/87910968 [=====] - 0s 0us/step
```

```
#COMPARE RESULTS OF VGG16 AND GOOGLENET
# Import necessary libraries
from keras.applications.vgg16 import VGG16, decode_predictions,
preprocess_input as vgg_preprocess_input
from keras.applications.inception_v3 import InceptionV3,
preprocess_input as inception_preprocess_input
from keras.preprocessing import image
import numpy as np

# Load and preprocess the image for VGG16
img_path_vgg = '/content/elephant.jpg'
img_vgg = image.load_img(img_path_vgg, target_size=(224, 224))
x_vgg = image.img_to_array(img_vgg)
x_vgg = np.expand_dims(x_vgg, axis=0)
x_vgg = vgg_preprocess_input(x_vgg)

# Load VGG16 model
vgg_model = VGG16(weights='imagenet', include_top=True)
vgg_output = vgg_model.predict(x_vgg)
vgg_predictions = decode_predictions(vgg_output, top=5)[0]

print("VGG16 Predictions:")
```



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

```
for pred in vgg_predictions:  
    print(pred[1], pred[2])  
  
# Load and preprocess the image for GoogLeNet  
img_path_inception = '/content/elephant.jpg' # Corrected image path  
img_inception = image.load_img(img_path_inception, target_size=(299, 299))  
x_inception = image.img_to_array(img_inception)  
x_inception = np.expand_dims(x_inception, axis=0)  
x_inception = inception_preprocess_input(x_inception)  
  
# Load GoogLeNet model  
inception_model = InceptionV3(weights='imagenet', include_top=True)  
inception_output = inception_model.predict(x_inception)  
inception_predictions = decode_predictions(inception_output, top=5)[0]  
  
print("\nGoogLeNet Predictions:")  
for pred in inception_predictions:  
    print(pred[1], pred[2])
```

The screenshot shows a Google Colab notebook titled "21BAI1364 BCSE332P LAB ASSIGN.5.ipynb". The code cell contains Python code for comparing the results of VGG16 and GoogLeNet models. The code imports necessary libraries, loads and preprocesses images for both models, and prints their respective predictions.

```
#COMPARE RESULTS OF VGG16 AND GOOGLENET  
# Import necessary libraries  
from keras.applications.vgg16 import VGG16, decode_predictions, preprocess_input  
from keras.applications.inception_v3 import InceptionV3, preprocess_input as inception_preprocess_input  
from keras.preprocessing import image  
import numpy as np  
  
# Load and preprocess the image for VGG16  
img_path_vgg = '/content/elephant.jpg'  
img_vgg = image.load_img(img_path_vgg, target_size=(224, 224))  
x_vgg = image.img_to_array(img_vgg)  
x_vgg = np.expand_dims(x_vgg, axis=0)  
x_vgg = vgg_preprocess_input(x_vgg)  
  
# Load VGG16 model  
vgg_model = VGG16(weights='imagenet', include_top=True)  
vgg_output = vgg_model.predict(x_vgg)  
vgg_predictions = decode_predictions(vgg_output, top=5)[0]  
  
print("VGG16 Predictions:")  
for pred in vgg_predictions:  
    print(pred[1], pred[2])  
  
# Load and preprocess the image for GoogLeNet  
img_path_inception = '/content/elephant.jpg' # Corrected image path  
img_inception = image.load_img(img_path_inception, target_size=(299, 299))  
x_inception = image.img_to_array(img_inception)
```



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

A screenshot of a Google Colab notebook titled "21BAI1364 BCSE332P LAB ASSIGN.5.ipynb". The code cell contains Python code for image classification using two pre-trained models: VGG16 and InceptionV3. The code loads images from a local directory, preprocesses them, and then uses the models to predict the classes. The output shows predictions for an elephant image, with VGG16 predicting "African_elephant" and InceptionV3 predicting "tusker".

```
# Load VGG16 model
vgg_model = VGG16(weights='imagenet', include_top=True)
vgg_output = vgg_model.predict(x_vgg)
vgg_predictions = decode_predictions(vgg_output, top=5)[0]

print("\nVGG16 Predictions:")
for pred in vgg_predictions:
    print(pred[1], pred[2])

# Load and preprocess the image for GoogLeNet
img_path_inception = '/content/elephant.jpg' # Corrected image path
img_inception = image.load_img(img_path_inception, target_size=(299, 299))
x_inception = image.img_to_array(img_inception)
x_inception = np.expand_dims(x_inception, axis=0)
x_inception = inception_preprocess_input(x_inception)

# Load GoogLeNet model
inception_model = InceptionV3(weights='imagenet', include_top=True)
inception_output = inception_model.predict(x_inception)
inception_predictions = decode_predictions(inception_output, top=5)[0]

print("\nGoogLeNet Predictions:")
for pred in inception_predictions:
    print(pred[1], pred[2])
```

WARNING:tensorflow:5 out of the last 5 calls to <function Model.make_predict_function.<locals>.predict_function at 0x7a8c6184b400> triggered tf.TooManyGradientsError. 1/1 [=====] - 0s 135ms/step
VGG16 Predictions:
African_elephant 0.7449078
tusker 0.2224986
Indian_elephant 0.032488205
Arabian_camel 3.3451295e-05
standard_poodle 0.00669176e-05
Download data from https://storage.googleapis.com/tensorflow/keras-applications/inception_v3/inception_v3_weights_tf_dim_ordering_tf_kernels_19612376/96112376 [=====] - 0s 0us/step
WARNING:tensorflow:6 out of the last 6 calls to <function Model.make_predict_function.<locals>.predict_function at 0x7a8c2a2b1750> triggered tf.TooManyGradientsError. 1/1 [=====] - 3s 3s/step

Google Account
Vitta Vishnu Datta 21BAI1364
vittavishnu.datta2021@vitstudent.ac.in

https://accounts.google.com/SignOutOptions?hl=en&continue=https://colab.research.google... completed at 10:54 AM

A screenshot of a Google Colab notebook titled "21BAI1364 BCSE332P LAB ASSIGN.5.ipynb". The code cell contains Python code for image classification using the InceptionV3 model. The code loads images from a local directory, preprocesses them, and then uses the model to predict the classes. The output shows predictions for an elephant image, with InceptionV3 predicting "tusker", "African_elephant", "Indian_elephant", "zebra", and "rain_barrel".

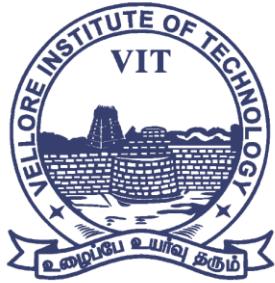
```
inception_predictions = decode_predictions(inception_output, top=5)[0]

print("\nGoogLeNet Predictions:")
for pred in inception_predictions:
    print(pred[1], pred[2])
```

WARNING:tensorflow:5 out of the last 5 calls to <function Model.make_predict_function.<locals>.predict_function at 0x7a8c6184b400> triggered tf.TooManyGradientsError. 1/1 [=====] - 0s 135ms/step
VGG16 Predictions:
African_elephant 0.7449078
tusker 0.2224986
Indian_elephant 0.032488205
Arabian_camel 3.3451295e-05
standard_poodle 0.00669176e-05
Download data from https://storage.googleapis.com/tensorflow/keras-applications/inception_v3/inception_v3_weights_tf_dim_ordering_tf_kernels_19612376/96112376 [=====] - 0s 0us/step
WARNING:tensorflow:6 out of the last 6 calls to <function Model.make_predict_function.<locals>.predict_function at 0x7a8c2a2b1750> triggered tf.TooManyGradientsError. 1/1 [=====] - 3s 3s/step

Google Account
Vitta Vishnu Datta 21BAI1364
vittavishnu.datta2021@vitstudent.ac.in

https://accounts.google.com/SignOutOptions?hl=en&continue=https://colab.research.google.com/drive/14n5SwP1q5zy50pkOwn-VKUWJCILcSdE?usp=chrome_ntp&ec=GBRAQM



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI



PREPARED AND SUBMITTED BY:

VITTA VISHNU DATTA.

21BAI1364 – SCOPE.

VIT – CHENNAI CAMPUS.