1. What are the advantages of a CNN for image classification over a completely linked DNN?

Exploitation of Spatial Locality:

Parameter Sharing:

Hierarchical Feature Extraction:

Reduced Parameter Count:

Translation Invariance:

1. Consider a CNN with three convolutional layers, each of which has three kernels, a stride of two, and SAME padding. The bottom layer generates 100 function maps, the middle layer 200, and the top layer 400. RGB images with a size of 200 x 300 pixels are used as input. How many criteria does the CNN have in total? How much RAM would this network need when making a single instance prediction if we're using 32-bit floats? What if you were to practice on a batch of 50 images?

2,800 (bottom layer) + 180,200 (middle layer) + 720,400 (top layer) = 903,400 parameters

if using 32-bit floats, a single instance prediction would require approximately 4.14 MB of RAM, while training on a batch of 50 images would require approximately 207 MB of RAM.

1. What are five things you might do to fix the problem if your GPU runs out of memory while training a CNN?

Reduce Batch Size:

Use Mixed Precision Training:

Decrease Model Complexity:

Utilize Memory Optimization Techniques:

Employ Model Parallelism or Distributed Training

1. Why would you use a max pooling layer instead with a convolutional layer of the same stride?

Downsampling and Dimension Reduction:

Translation Invariance

Robustness to Small Variations

Feature Selection and Activation Sparsity

Parameter Efficiency:

1. When would a local response normalization layer be useful?

A Local Response Normalization (LRN) layer can be useful in certain scenarios, particularly in convolutional neural networks (CNNs), for

1. In comparison to LeNet-5, what are the main innovations in AlexNet? What about GoogLeNet and ResNet's core innovations?

AlexNet Innovations:

AlexNet introduced several key innovations that significantly advanced the field of deep learning:

Larger Network: AlexNet was a much deeper network compared to previous architectures like LeNet-5, consisting of eight layers, including five convolutional layers and three fully connected layers.

GoogLeNet Innovations:

GoogLeNet (Inception) introduced several innovative components to improve network performance and efficiency:

Inception Module: GoogLeNet employed the Inception module, which consisted of parallel convolutional layers of different filter sizes (1x1, 3x3, 5x5) and pooling operations. This allowed the network to capture features at multiple scales and reduced the number of paramete

1. On MNIST, build your own CNN and strive to achieve the best possible accuracy.

import torch

import torch.nn as nn

import torch.optim as optim

import torchvision.transforms as transforms

from torchvision.datasets import MNIST

from torch.utils.data import DataLoader

# Define the CNN architecture

class SimpleCNN(nn.Module):

def \_\_init\_\_(self):

super(SimpleCNN, self).\_\_init\_\_()

self.conv1 = nn.Conv2d(1, 32, kernel\_size=3, stride=1, padding=1)

self.relu1 = nn.ReLU()

self.pool1 = nn.MaxPool2d(kernel\_size=2)

self.conv2 = nn.Conv2d(32, 64, kernel\_size=3, stride=1, padding=1)

self.relu2 = nn.ReLU()

self.pool2 = nn.MaxPool2d(kernel\_size=2)

self.fc1 = nn.Linear(7 \* 7 \* 64, 128)

self.relu3 = nn.ReLU()

self.fc2 = nn.Linear(128, 10)

def forward(self, x):

x = self.conv1(x)

x = self.relu1(x)

x = self.pool1(x)

x = self.conv2(x)

x = self.relu2(x)

x = self.pool2(x)

x = x.view(x.size(0), -1)

x = self.fc1(x)

x = self.relu3(x)

x = self.fc2(x)

return x

# Set device (CPU or GPU)

device = torch.device("cuda" if torch.cuda.is\_available() else "cpu")

8. Using Inception v3 to classify broad images. a.

Images of different animals can be downloaded. Load them in Python using the matplotlib.image.mpimg.imread() or scipy.misc.imread() functions, for example. Resize and/or crop them to 299 x 299 pixels, and make sure they only have three channels (RGB) and no transparency. The photos used to train the Inception model were preprocessed to have values ranging from -1.0 to 1.0, so make sure yours do as well.

import torch

import torch.nn as nn

import torchvision.transforms as transforms

from torchvision.models import inception\_v3

from PIL import Image

# Load the Inception v3 model

model = inception\_v3(pretrained=True, transform\_input=True)

model.eval()

# Preprocessing transforms

preprocess = transforms.Compose([

transforms.Resize((299, 299)),

transforms.ToTensor(),

transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5)),

])

# Load and preprocess the image

image\_path = 'path\_to\_your\_image.jpg' # Replace with the path to your image

image = Image.open(image\_path).convert('RGB')

image = preprocess(image)

image = image.unsqueeze(0) # Add a batch dimension

# Forward pass through the model

with torch.no\_grad():

logits = model(image)

# Convert logits to probabilities using softmax

probabilities = nn.functional.softmax(logits, dim=1)

# Load the ImageNet class labels

with open('imagenet\_labels.txt') as f:

labels = [line.strip() for line in f.readlines()]

# Get the top 5 predicted labels and their probabilities

top5\_prob, top5\_labels = torch.topk(probabilities, k=5)

top5\_prob = top5\_prob.squeeze().tolist()

top5\_labels = [labels[idx] for idx in top5\_labels.squeeze().tolist()]

# Print the top 5 predicted labels and probabilities

for prob, label in zip(top5\_prob, top5\_labels):

print(f"{label}: {prob\*100:.2f}% confidence")

9. Large-scale image recognition using transfer learning.

a. Make a training set of at least 100 images for each class. You might, for example, identify your own photos based on their position (beach, mountain, area, etc.) or use an existing dataset, such as the flowers dataset or MIT's places dataset (requires registration, and it is huge).

b. Create a preprocessing phase that resizes and crops the image to 299 x 299 pixels while also adding some randomness for data augmentation.

c. Using the previously trained Inception v3 model, freeze all layers up to the bottleneck layer (the last layer before output layer) and replace output layer with appropriate number of outputs for your new classification task (e.g., the flowers dataset has five mutually exclusive classes so the output layer must have five neurons and use softmax activation function).

d. Separate the data into two sets: a training and a test set. The training set is used to train the model, and the test set is used to evaluate it.

Ans :Here's an example of how you can perform large-scale image recognition using transfer learning with the Inception v3 model:

Creating the Training Set:

Reprocessing with Data Augmentation:

Modifying the Inception v3 Model:

Splitting the Data into Training and Test Sets