Karan Vora (kv2154)
ECE-GY 9143 Introduction to High-Performance Machine Learning Assignment 3

Problem 3):

Solution 3.1):

AlexNet was one of the first CNN to win the ImageNet classification competition of 2012. It consists of 8 layers: 5 Convolutional and 3 Fully-Connected layers with a 1000 class classification as the output layer.

→ Convolutional layer 1:
Input: 227 x 227 image with 3 input channels
96 filters of size 11 x 11 with stride 4 and no padding
Number of parameters: (11 x 11 x 3 x 96) + 96 = 34944

→ Max-Pooling layer 1:
Input: 96 channels with 55 x 55 feature maps
Max-Pooling of size 3 x 3 with stride 2

→ Convolutional layer 2:
Input: 96 channels with 27 x 27 feature maps
256 filters of size 5 x 5 with stride 1 and padding 2
Number of parameters: (5 x 5 x 96 x 256) + 256 = 614656

→ Max-Pooling layer 2:
Input: 256 channels with 13 x 13 feature maps
Max-Pooling pooling of size 3 x 3 with stride 2

→ Convolutional layer 3:
Input: 256 channels with 13 x 13 feature maps
384 filters of size 3 x 3 with stride 1 and padding 1
Number of parameters: (3 x 3 x 256 x 384) + 384 = 885120

→ Convolutional layer 4:
Input: 384 channels with 13 x 13 feature maps
384 filters of size 3 x 3 with stride 1 and padding 1
Number of parameters: (3 x 3 x 384 x 384) + 384 = 1327488

→ Convolutional layer 5:
Input: 384 channels with 13 x 13 feature maps
256 filters of size 3 x 3 with stride 1 and padding 1
Number of parameters: (3 x 3 x 384 x 256) + 256 = 884992

→ Max-Pooling layer 3:
Input: 256 channels with 13 x 13 feature maps
Max-Pooling of size 3 x 3 and stride 2

→ Fully-Connected layer 1:

Input: 9216 (256 x 6 x 6) features
4096 Neurons
Number of parameters: (9216 x 4096) + 4096 = 37752832
→ Fully-Connected layer 2:
Input: 4096 features
4096 neurons
Number of parameters: (4096 x 4096) + 4096 = 16781312

→ Fully-Connected layer 3 (Output layer):
Input: 4096 features
1000 neurons, one for each class in ImageNet dataset
Number of parameters: (4096 x 1000) + 1000 = 4097000

Total number of parameters in AlexNet: 34944 + 614656 + 885120 + 1327488 + 884992 + 37752832 + 16781312 + 4097000 = 61100344

Solution 3.2):

Layer	Number of Activations (Memory)	Parameters (Compute)
Input	224x224x3 = 150K	0
CONV3-64	224x224x64 = 3.2M	(3x3x3)x64 = 1728
CONV3-64	224x224x64 = 3.2M	(3x3x3)x64 = 36864
POOL2	112x112x64 = 800K	0
CONV3-128	112x112x128 = 1.6M	(3x3x64)x128 = 73728
CONV3-128	112x112x128 = 1.6M	(3x3x128)x128 = 147456
POOL2	56x56x128 = 400K	0
CONV3-256	56x56x256 = 800K	(3x3x128)x256 = 294912
CONV3-256	56x56x256 = 800K	(3x3x256)x256 = 589824
CONV3-256	56x56x256 = 800K	(3x3x256)x256 = 589824
CONV3-256	56x56x256 = 800K	(3x3x256)x256 = 589824
POOL2	28x28x256 = 200K	0
CONV3-512	28x28x512 = 400K	(3x3x256)x512 = 1179648
CONV3-512	28x28x512 = 400K	(3x3x512)x512 = 2359296
CONV3-512	28x28x512 = 400K	(3x3x512)x512 = 2359296
CONV3-512	28x28x512 = 400K	(3x3x512)x512 = 2359296
POOL2	14x14x512 = 100K	0
CONV3-512	14x14x512 = 100K	(3x3x512)x512 = 2359296
CONV3-512	14x14x512 = 100K	(3x3x512)x512 = 2359296
CONV3-512	14x14x512 = 100K	(3x3x512)x512 = 2359296
CONV3-512	14x14x512 = 100K	(3x3x512)x512 = 2359296
POOL2	7x7x512 = 25K	0
FC	4096	7x7x512x4096 = 102760448
FC	4096	4096x4096 = 16777216
FC	1000	4096x1000 = 4096000
Total	17144296	143653144

Solution 3.3):

==> For Naive Inception Module,

→ For 1x1 Filter,
Number of Operations = 32 * 32 * 1 * 1 * 128 * 256 = 1048576

→ For 3x3 Filter,
Each 3 x 3 filter operates on all 256 channels of the input volume, which has dimensions of 32 x 32 x 256. The output volume for each filter will be of size 30 x 30 x 1, with the height and width reduced by 2 due to the filter size
Number of Operations = 30 * 30 * 3 * 3 * 192 * 256 = 11940096000

→ For 5x5 Filter,
Each 5 x 5 filter operates on all 256 channels of the input volume, which has dimensions of 32 x 32 x 256. The output volume for each filter will be of size 28 x 28 x 1, with the height and width reduced by 4 due to the filter size
Number of Operations = 28 * 28 * 5 * 5 * 96 * 256 = 5806893760

Total Number of Operations = 1048576 + 11940096000 + 5806893760 = 17748038336

==> For Inception Module with Dimension reduction

→ For 1x1 Filter,
Number of Operations = 32 * 32 * 1 * 1 * 128 * 256 = 1048576

Next,
→ For 1x1 Filter,
Number of Operations = 32 * 32 * 1 * 1 * 128 * 256 = 1048576
→ For the next layer, The output dimensions are 28 x 28 x 128, so for filter size of 3x3
Number of Operations = 30 * 30 * 3 * 3 * 128 * 192 = 199065600

Next,
→ For 1x1 Filter,
Number of Operations = 32 * 32 * 1 * 1 * 32 * 256 = 8388608
→ For the next layer, The output dimensions are 30 x 30 x 32, so for filter size of 5x5
Number of Operations = 28 * 28 * 5 * 5 * 96 * 32 = 60211200

Next,
→ We have a 3x3 max-pooling layer, so the input dimensions of 32 x 32 x 256 will be reduced to 30 x 30 x 256.
→ For next layer the output dimension is 30 x 30 x 64 so for filter size of 1x1
Number of Operations = 30 * 30 * 1 * 1 * 64 * 256 = 14745600

Total Number of Operations = 1048576 + 1048576 + 199065600 + 8388608 + 60211200 + 14745600 = 284508160

From the above mentioned calculation, it is clear that Dimensionality Reduction reduces the required number of operations to perform the inception module by a large factor

Solution 3.4):

Naive architectures for convolutional neural networks (CNNs) typically stack multiple convolutional layers with high numbers of filters to extract features from the input image. However, this approach can lead to two problems:

1. High computational cost: As the number of filters increases in each convolutional layer, the number of parameters and computations required also increases. This can make the model slow and computationally expensive.

2. Information loss: As the input volume passes through multiple convolutional layers, the spatial dimensions reduce while the depth increases. This can lead to a loss of information and may result in the network missing important features.

To address these problems, dimensionality reduction architectures, such as the inception module, have been proposed. Inception modules use multiple filter sizes in parallel to extract features from the input volume at different scales. By doing this, they can capture both fine-grained and coarse-grained features in the input volume.

Specifically, inception modules use 1x1, 3x3, and 5x5 filters in parallel and concatenate their outputs to form the final output of the module. The 1x1 filters are used to reduce the number of input channels and, hence, reduce the computational cost of the subsequent filters. This is known as a bottleneck layer. Additionally, max-pooling is applied before the 1x1 filters to reduce the spatial dimensions of the input volume, which further reduces the computational cost.

By using multiple filter sizes and dimensionality reduction techniques, inception modules can extract features from the input volume in a more efficient and effective way. The use of 1x1 filters for dimensionality reduction significantly reduces the number of computations required, while the use of multiple filter sizes helps capture both fine-grained and coarse-grained features.

The computational saving of the inception module depends on the specific architecture and input volume size, but it can be significant. In some cases, the use of dimensionality reduction architectures like inception modules can reduce the number of computations required by up to 10 times compared to naive architectures with the same number of parameters. This reduction in computational cost makes the model faster and more efficient, which is important for real-world applications with limited computing resources.