E9 246 - Advanced Image Processing Assignment 1

Bitupan Arandhara
bitupana@iisc.ac.in

Question 1. Local Binary Pattern (LBP) for Texture Analysis

Methodology

The standard LBP operator was implemented using NumPy with a 3×3 neighborhood. The LBP codes were extracted, and then the corresponding LBP histograms were computed for each image. The histogram feature vectors for all the training images were stored in an numpy array.

Texture classification was performed using 1-Nearest Neighbor (1-NN) classifier with Euclidean Distance. Each test image histogram was compared with training histograms and assigned the label of the closest training image.

Results and Observations

Classification Accuracy using Standard LBP

The classification accuracy obtained using standard LBP on original test images is:

$$\textbf{Accuracy} = \textbf{13.3\%}$$

Effect of Rotation on Standard LBP

Test images were rotated at different angles and classification was repeated for each angle.

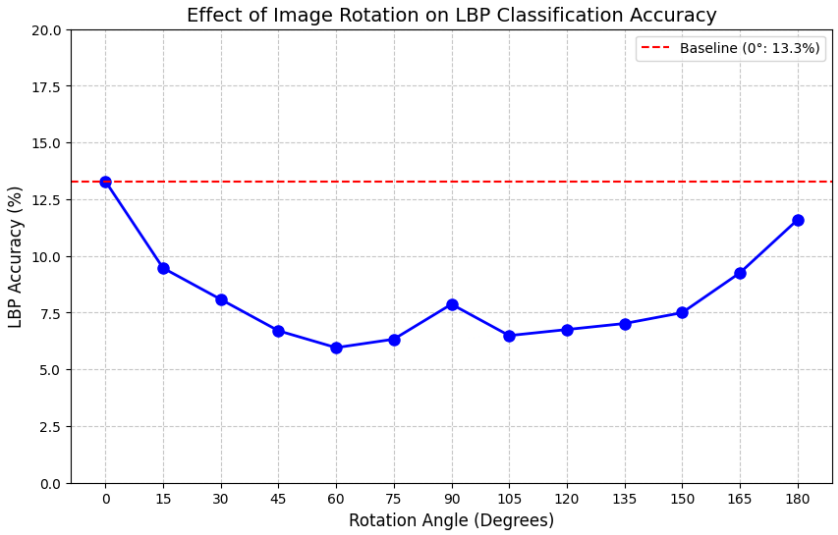


Figure 1: Classification Accuracy vs Rotation Angle using Standard LBP

- **Observations**:
  - The highest accuracy occurs at 0° because the training and testing textures are aligned.
  - Accuracy decreases significantly between 15° to 75°.
  - At 180°, accuracy increases again because the orientation becomes somewhat similar to the original texture.
- **Analysis**:
  - Accuracy decreases when the image is rotated.
  - The standard LBP is sensitive to changes in orientation because it depends on the order of the neighboring pixels.
  - When the image rotates, the relative position of neighbors changes, which changes the binary pattern.

Rotation-Invariant LBP

The Rotation-invariant LBP was implemented as per the given paper.

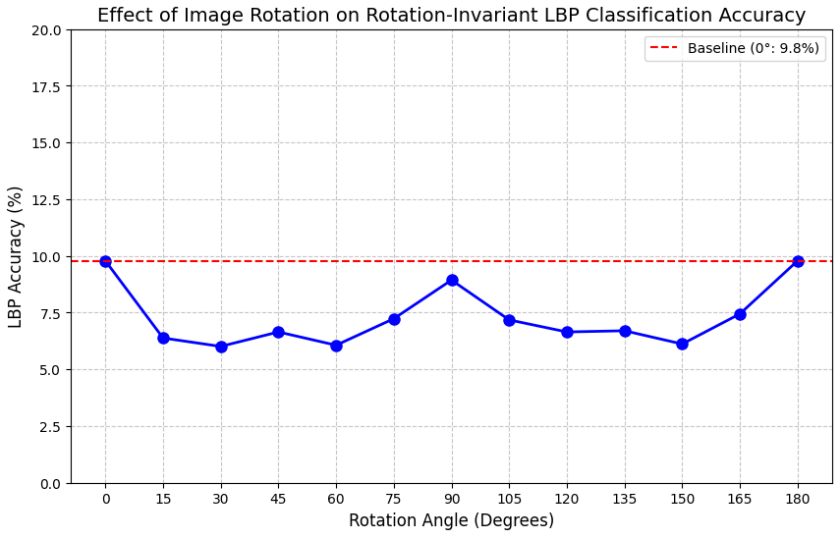


Figure 2: Classification Accuracy vs Rotation Angle using Rotation-Invariant LBP

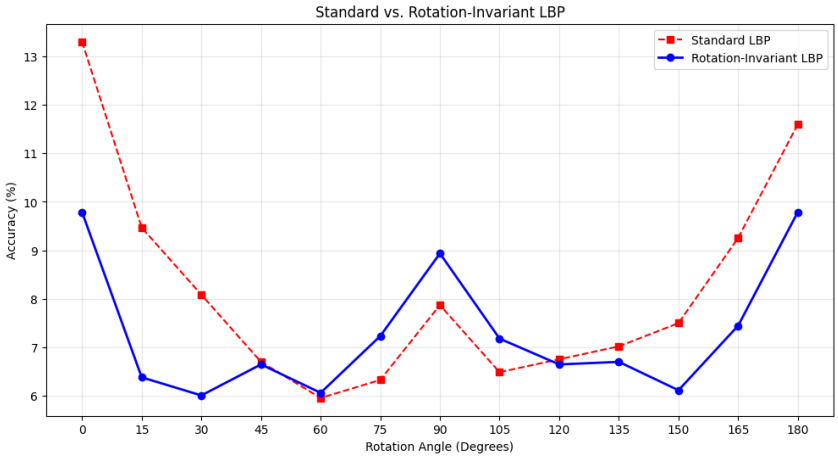


Figure 3: Comparing the Classification Accuracy of Standard LBP and Rotation-Invariant LBP for Different Angles

- **Observations**:
  - Standard LBP performs slightly better when images are not rotated.
  - Rotation-invariant LBP gives more consistent accuracy across different rotation angles.
  - The slight drop in accuracy at 0° happens because rotation-invariant LBP reduces the number of unique patterns, which reduces feature discriminability.

Q2. Adaptation to Sketch Domain

Methodology

A pretrained **ResNet18** model trained on **ImageNet-1K** dataset was used for different fine-tuning tasks. The final fully-connected layer (classification head) was modified to support the 99 classes of the given dataset. Each experiment was trained for 15 epochs. Cross-entropy loss was used for all tasks.

A. Fine-Tuning the Entire Network

Using SGD Optimizer

Stochastic gradient descent with momentum was used as the optimizer with a learning rate of 0.001 and momentum 0.9.

- Best Training Accuracy: 93.02%
- Best Validation Accuracy: 76.77%
- Test Accuracy: 79.30%

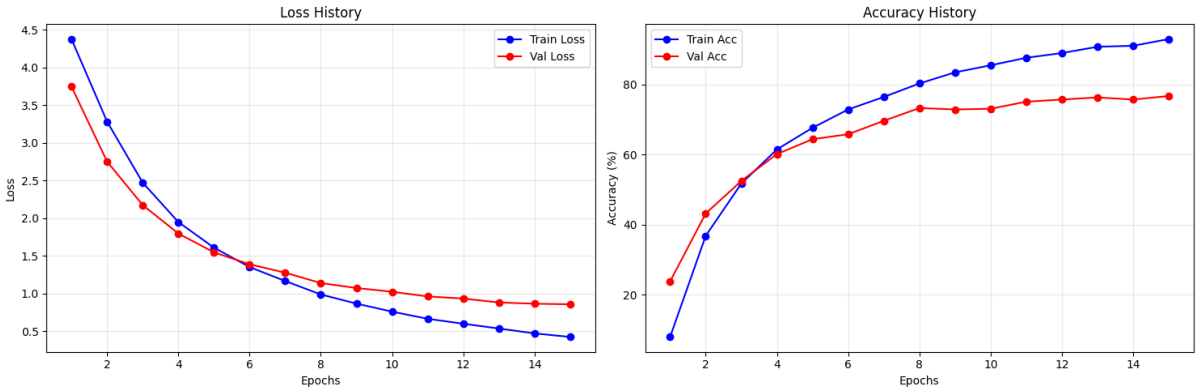


Figure 4: Comparison of Train vs Validation Loss and Accuracy for SGD with Momentum

Using Adam Optimizer

Adam optimizer with a learning rate of 0.0001 was used.

- Best Training Accuracy: 96.05%
- Best Validation Accuracy: 81.62%
- Test Accuracy: 82.62%

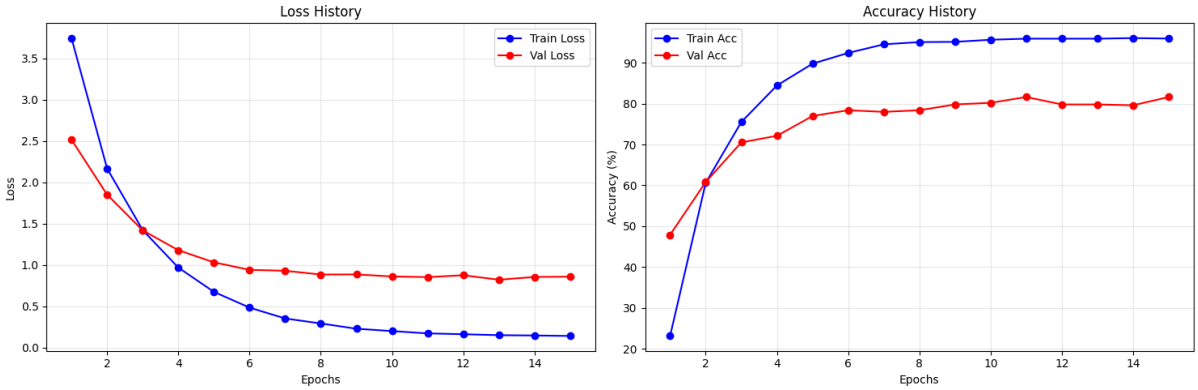


Figure 5: Comparison of Train vs Validation Loss and Accuracy for Adam Optimizer

C. Constrained Fine-Tuning

Fine-Tuning Only Last Two Layers (layer4 and fc)

Adam optimizer with a learning rate of 0.0001 was used.

- Best Training Accuracy: 96.28%
- Best Validation Accuracy: 82.22%
- Test Accuracy: 82.16%

Restricting fine-tuning to the last two layers (layer4 and fc) yielded performance nearly identical to fine-tuning the entire network. This shows that the early and middle layers of ResNet18 are sufficiently general to be reused for our sketch dataset.

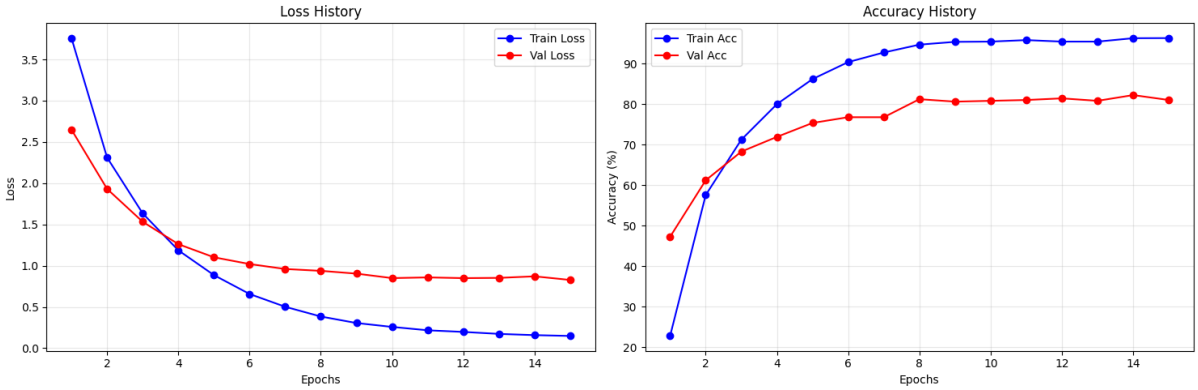


Figure 6: Comparison of Train vs Validation Loss and Accuracy for Fine-Tuning only Last Two Layers

Updating Only Batch Normalization + Fully Connected Layers

- Adam optimizer with a learning rate of 0.0001.

  - Best Training Accuracy: 63.95%
  - Best Validation Accuracy: 58.99%
  - Test Accuracy: 56.56%

- Adam optimizer with a learning rate of 0.001.
  - Best Training Accuracy: 88.80%
  - Best Validation Accuracy: 76.16%
  - Test Accuracy: 78.00%

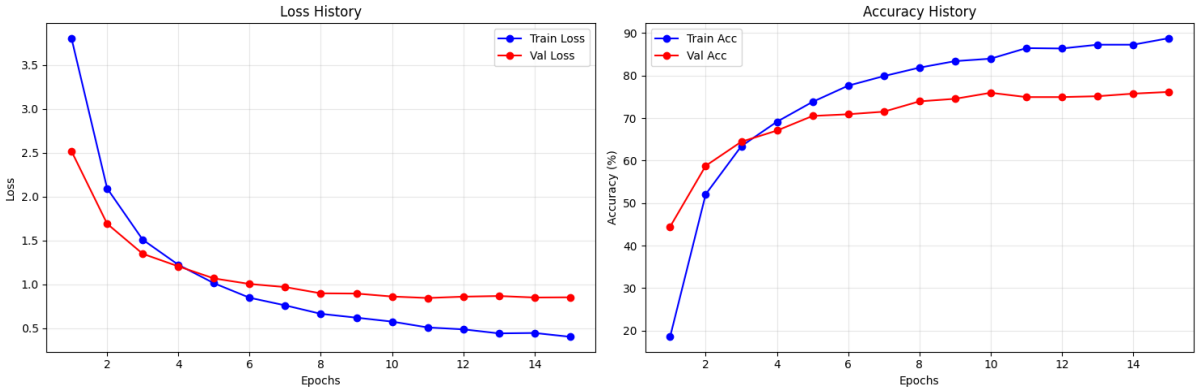


Figure 7: Comparison of Train vs Validation Loss and Accuracy for Fine-Tuning only BN and Final FC layers (with lr 0.001)

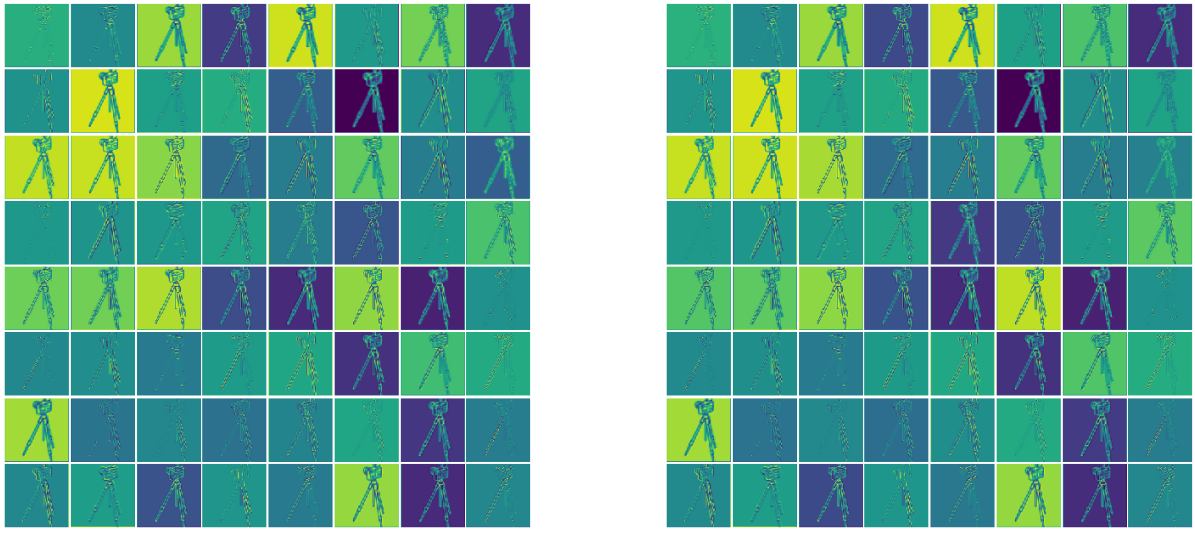
Analysis of the effect of learning rate: The significantly higher accuracy at LR 0.001 indicates that the model requires a larger step size to effectively shift the feature distributions (via BN parameters) from natural images to sketch images. A lower LR of 0.0001 seems insufficient to move these parameters out of their ImageNet-pretrained local minima.

Comparing the performance of Full Fine-Tuned Network to the Batch-Norm Fine-Tuned Network: Fine-tuning the complete network reached a test accuracy of 82.62%, whereas Batch Normalization (BN) layers and Fully Connected (FC) layer fine-tuned model reached 78.00%. BatchNorm layers adjust mean and variance of feature distributions. Updating them helps partially adapt to the feature distribution of sketch images from natural images. But, BatchNorm alone cannot learn new features. Only changes to the convolution filters will allow the model to learn the fundamental features changes inthe new domain. Hence, the complete fine-tuned model outperforms the BN only fine-tuned model.

B. Detailed Feature Map Comparison

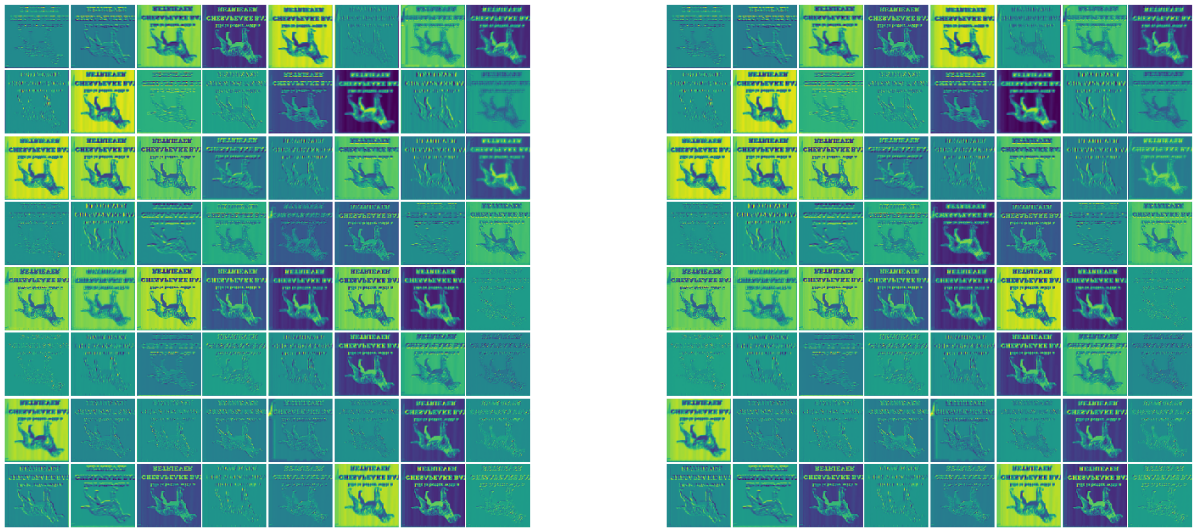
To better understand the effect of domain adaptation, two representative test images were selected where the difference between the pretrained and fine-tuned models is clearly visible. The analysis is performed across early, middle, and deep layers of ResNet18.

Early Layer (conv1)



(a) Original Pretrained Model (b) Fine-Tuned Model

Figure 8: Feature Maps for Image 0 at conv1 Layer



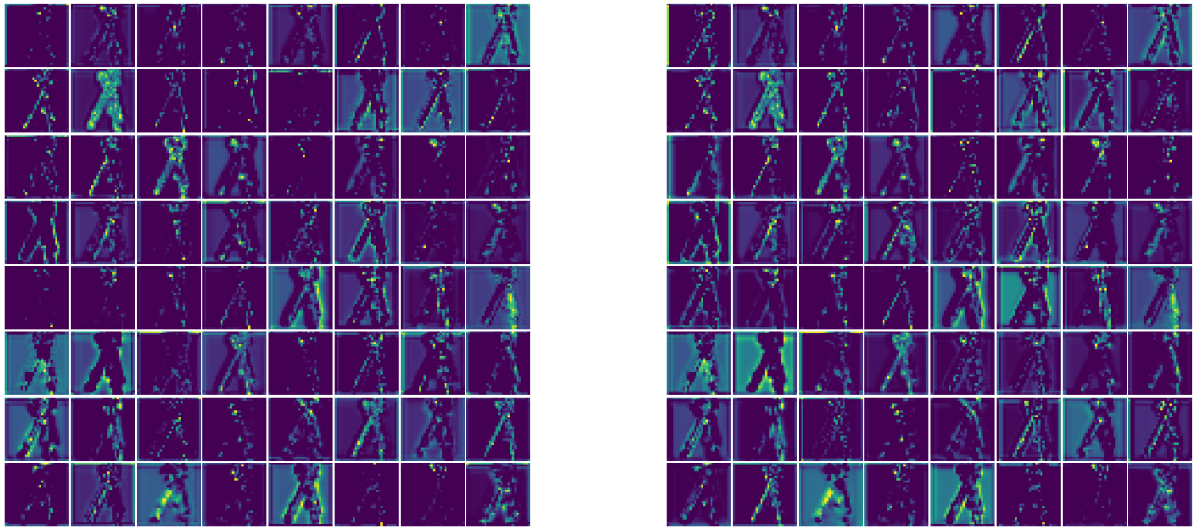
(a) Original Pretrained Model (b) Fine-Tuned Model

Figure 9: Feature Maps for Image 3 at conv1 Layer

Observations:

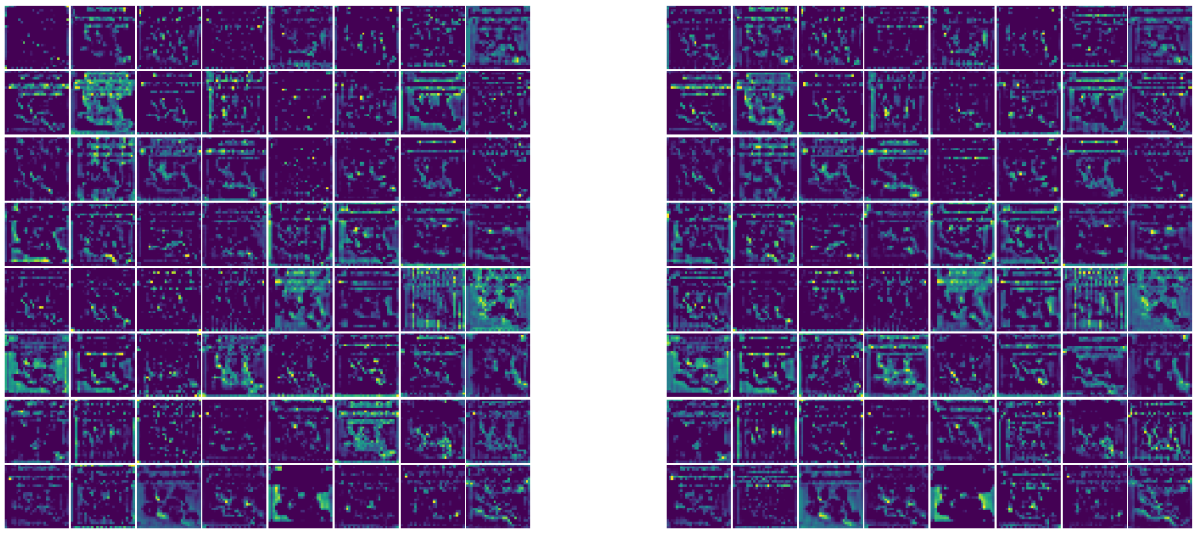
- At the conv1 layer, both the pretrained and fine-tuned models mainly capture low-level visual features such as edges, intensity changes, and simple structural patterns.
- It can be seen that there is not much difference between the intial layer's features.

Middle Layer (layer2)



(a) Original Pretrained Model (b) Fine-Tuned Model

Figure 10: Feature Maps for Image 0 at layer2



(a) Original Pretrained Model (b) Fine-Tuned Model

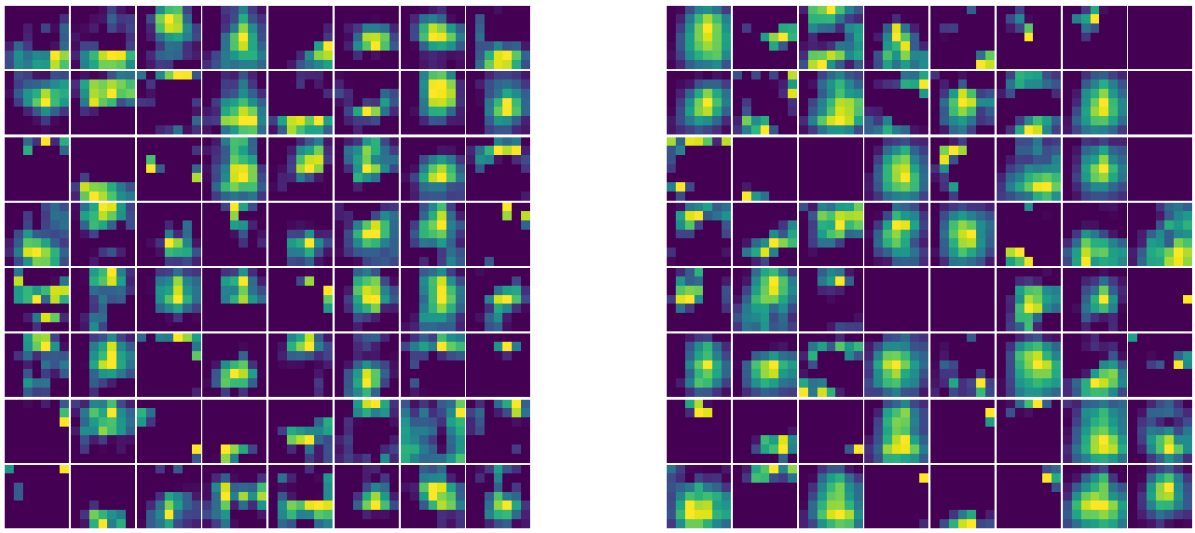
Figure 11: Feature Maps for Image 3 at layer2

Observations:

- At the middle layers, the network transitions from detecting primitive edges to identifying more complex structural components and textures.

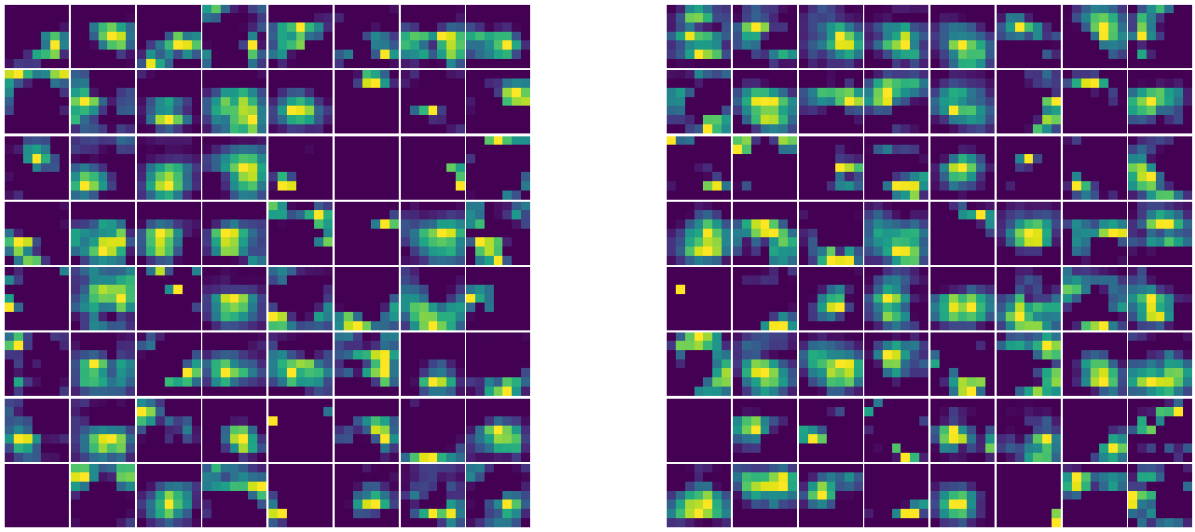
- It can be seen that in the feature maps of the fine-tuned model the activations are more sparse and binary.
- It can be seen that the fine-tuned model is learning more features in the images.
- This suggests the filters have adapted better to the ImageNet-Sketch domain.

Deep Layer (layer4)



(a) Original Pretrained Model (b) Fine-Tuned Model

Figure 12: Feature Maps for Image 0 at layer4



(a) Original Pretrained Model (b) Fine-Tuned Model

Figure 13: Feature Maps for Image 3 at layer4

Observations:

- At this depth, the spatial resolution is 7×7 for a standard 224×224 input, and the receptive field of each neuron covers nearly the entire image.
- The network has transitioned to encoding high-level semantic concepts and global object context of the image.
- The activations are less certain and more distributed in the original feature maps, whereas in the fine-tuned feature maps (especially for $img0$), the activations become more localized and intense.
- In the fine-tuned feature map in Figure 12, we can see that several channels have become completely dark. This confirms that the fine-tuning process has specialized the later-layer filters to ignore the some missing texture and focus entirely on the global arrangement of lines that define the class.