E9 246 Advanced Image Processing
Assignment 1

February 5, 2026

Dwaipayan Haldar

---

1. Local Binary Pattern (LBP) for Texture Observation

(a)

Ans: The standard Local Binary Pattern (LBP) operator was applied to each image using a 3×3 neighborhood. For each pixel, the 8 neighboring pixels are compared with the center pixel, and a binary code is generated (1 if neighbor $\geq$ center, 0 otherwise). This produces a 256-bin histogram for each image.

Using 1-Nearest Neighbor (1-NN) classification with Euclidean distance between histograms:

Classification Accuracy: 12.77%

The relatively low accuracy is expected given the large number of texture classes (47) and the sensitivity of standard LBP to geometric transformations. Fig.1 gives the confusion matrix of the same.

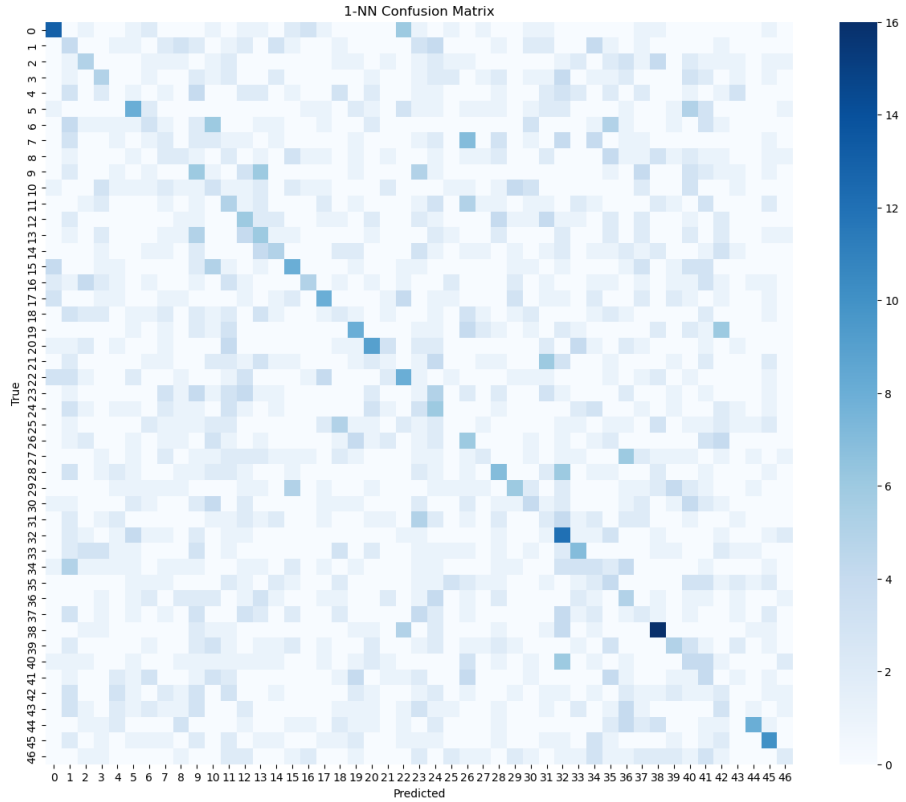

Figure 1: Confusion matrix for 1-NN classification using standard LBP histograms on the DTD dataset (47 classes).

(b)

Ans: The test images were rotated by multiple angles (0°, 15°, 30°, 45°, 60°, 90°, 105°, 120°, 135°, 150°, 165°, 180°) and classification was performed using the same 1-NN approach.

Rotation Angle	Accuracy
0°	12.77%
15°	5.53%
30°	3.67%
45°	2.29%
60°	2.93%
90°	4.41%
105°	3.24%
120°	2.82%
135°	2.18%
150°	2.98%
165°	4.68%
180°	12.50%

Table 1: Classification accuracy vs rotation angle for standard LBP.

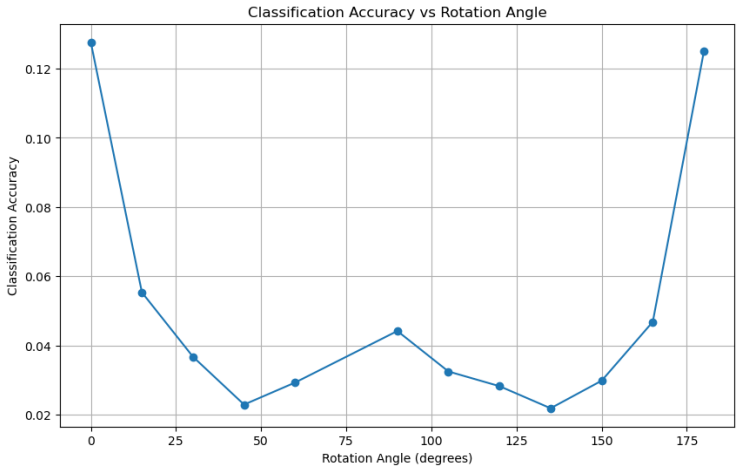

Figure 2: Effect of rotation on standard LBP-based texture classification accuracy.

Observation: The results clearly demonstrate that standard LBP is highly sensitive to rotation. The accuracy drops significantly for rotated images, with minimum accuracy at 45° and 135° rotations. Notably, 0° and 180° rotations maintain similar accuracy since 180° rotation preserves the relative ordering of some neighbor pairs. The pattern shows approximate symmetry around 90°.

(c)

Ans: The rotation-invariant uniform LBP (LBP^{riu2}) descriptor was implemented. Consider $(01100110)_2$ and $(00110011)_2$, they are the same binary string under rotation. Those are grouped into one. So, there are 36 unique patterns like this. Further, we can consider those binary strings which has less than equal to 2 transitions as uniform and there are 9 such unique representations and rest all non-uniform representations into one. So there are finally 10 bins in the rotation invariant uniform LBP. Table.2 gives the comparison of the standard LBP with the rotation invariant LBP. Fig.3 presents the data visually.

Observation: The rotation-invariant LBP shows more consistent performance across different rotation angles compared to standard LBP. While the absolute accuracy at 0° is lower (due to reduced discriminative power from fewer histogram bins), the RIU-LBP demonstrates improved stability under rotation. The accuracy variance across angles is smaller for RIU-LBP, confirming its rotation-invariant property. The trade-off is between rotation invariance and discriminative capability.

Rotation Angle	**Standard LBP**	**RIU-LBP**
0°	12.77%	8.99%
15°	5.53%	4.10%
30°	3.67%	2.87%
45°	2.29%	2.82%
60°	2.93%	3.09%
90°	4.41%	4.47%
105°	3.24%	3.67%
120°	2.82%	3.09%
135°	2.18%	2.98%
150°	2.98%	2.98%
165°	4.68%	3.83%
180°	12.50%	9.57%

Table 2: Comparison of standard LBP and rotation-invariant LBP accuracy.

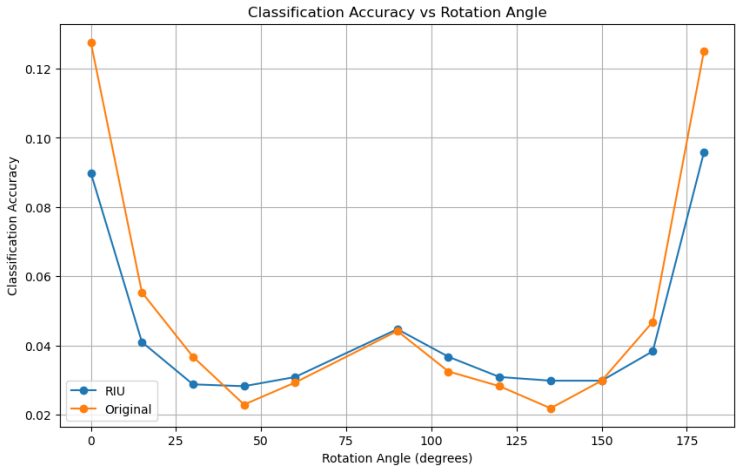

Figure 3: Comparison of standard LBP and rotation-invariant LBP under various rotations.

2. Adaptation to Sketch Domain

(a)

Ans: A ResNet18 model pretrained on ImageNet was fine-tuned on the ImageNet Sketch subset (100 classes). All layers were trainable during fine-tuning. The model achieves strong performance on the sketch domain. Fig.4 gives the loss curves and the accuracy curves per epoch for the data.

Training Configuration:

- Optimizer: Adam (lr=0.0001, weight_decay=1e-4)
- Scheduler: StepLR (step_size=5, gamma=0.1)
- Epochs: 15
- Batch size: 32
- Data augmentation: Random affine transformations

Metric	Accuracy
Training Accuracy	96.48%
Validation Accuracy	80.00%
Test Accuracy	**80.04%**

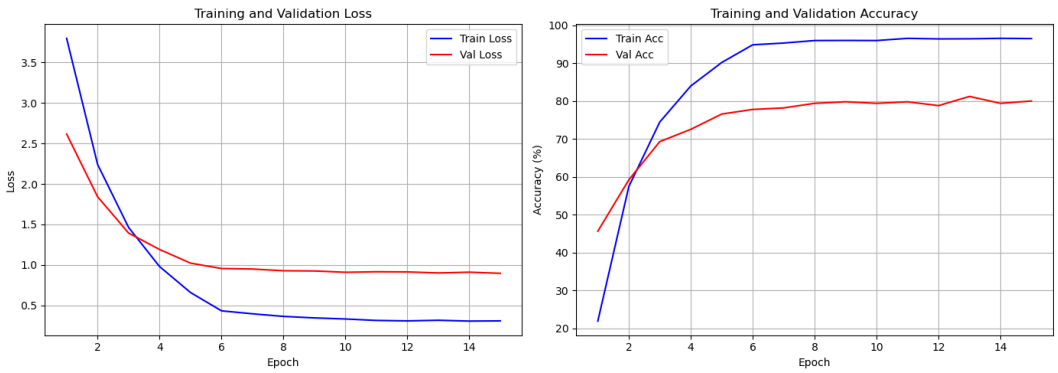

Figure 4: Training and validation loss/accuracy curves for full fine-tuning of ResNet18 on ImageNet-Sketch.

(b)

Ans: Feature maps from layers 1-4 of ResNet18 were visualized for 5 randomly selected test images, comparing the ImageNet-pretrained model (before fine-tuning) with the fine-tuned model.

For each image, the top row shows feature maps from the **pretrained model** and the bottom row shows feature maps from the **fine-tuned model**. Each grid displays 64 feature map channels for each layer.

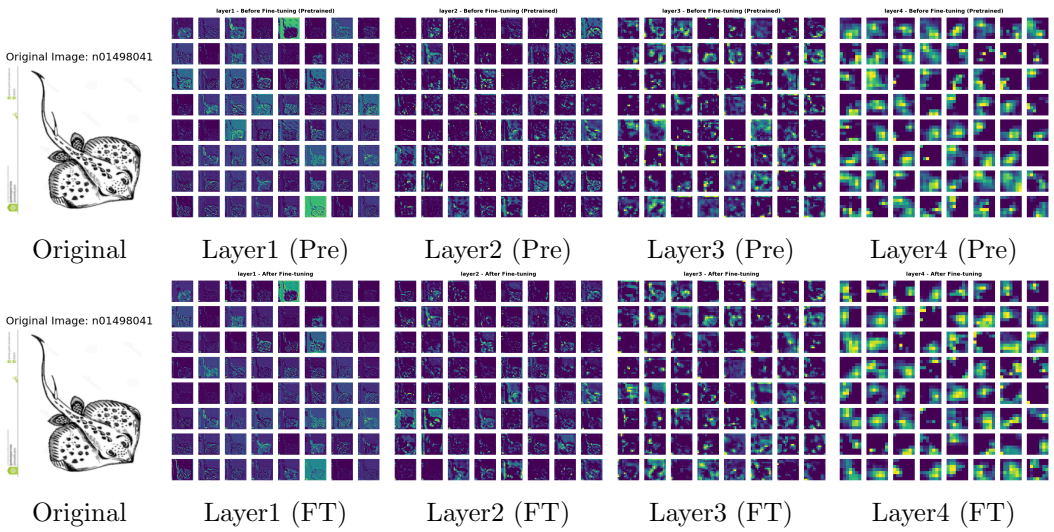

Figure 5: Image 1: Feature maps comparison (Pre=Pretrained, FT=Fine-tuned)

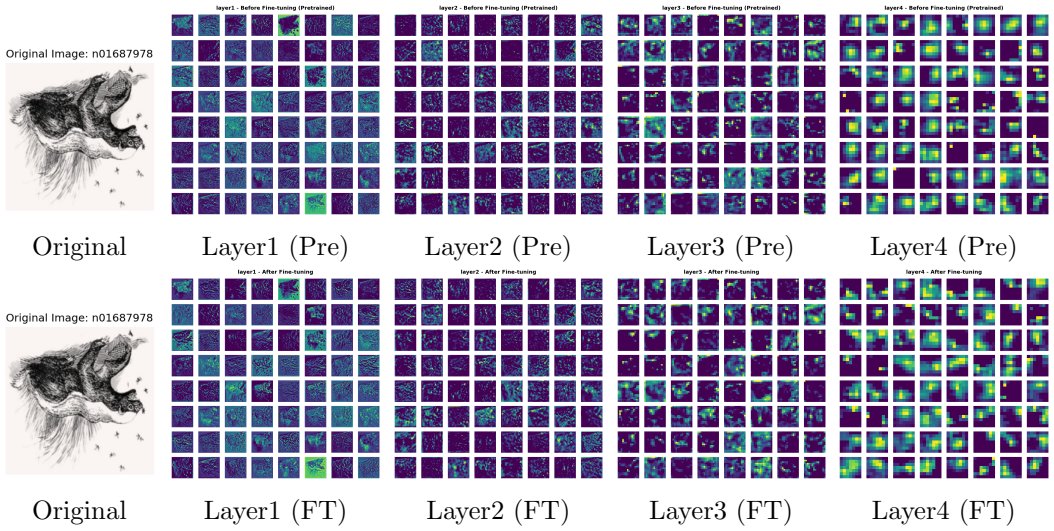

Figure 6: Image 2: Feature maps comparison (Pre=Pretrained, FT=Fine-tuned)

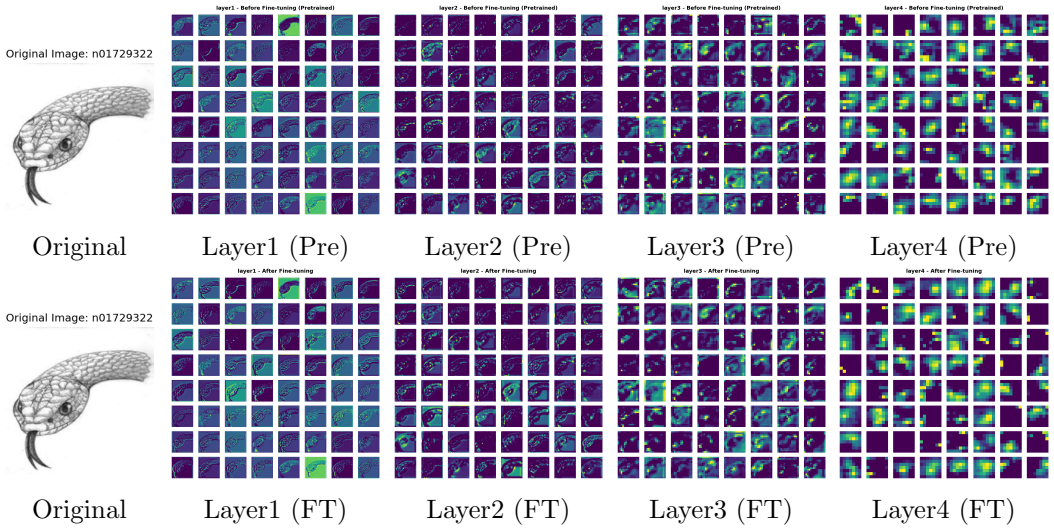

Figure 7: Image 3: Feature maps comparison (Pre=Pretrained, FT=Fine-tuned)

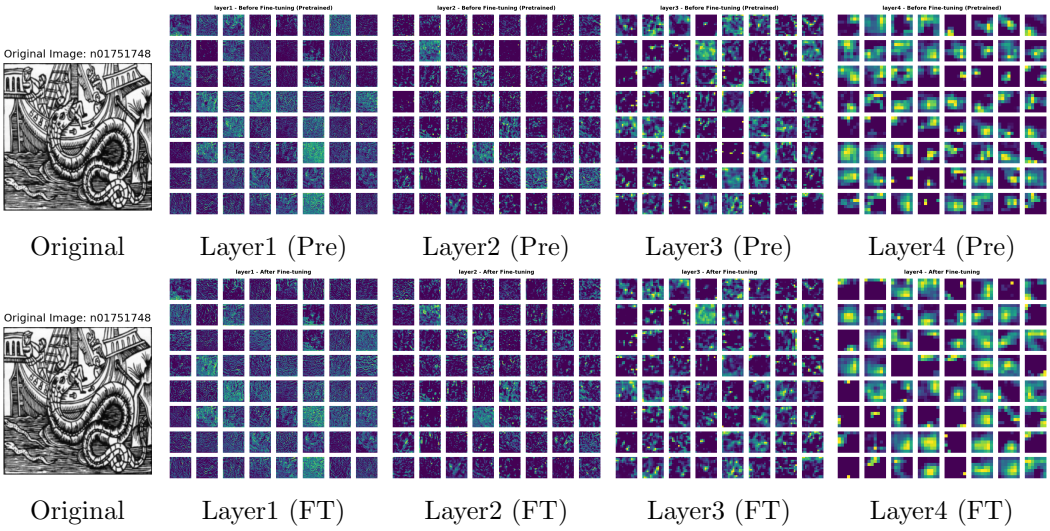

Figure 8: Image 4: Feature maps comparison (Pre=Pretrained, FT=Fine-tuned)

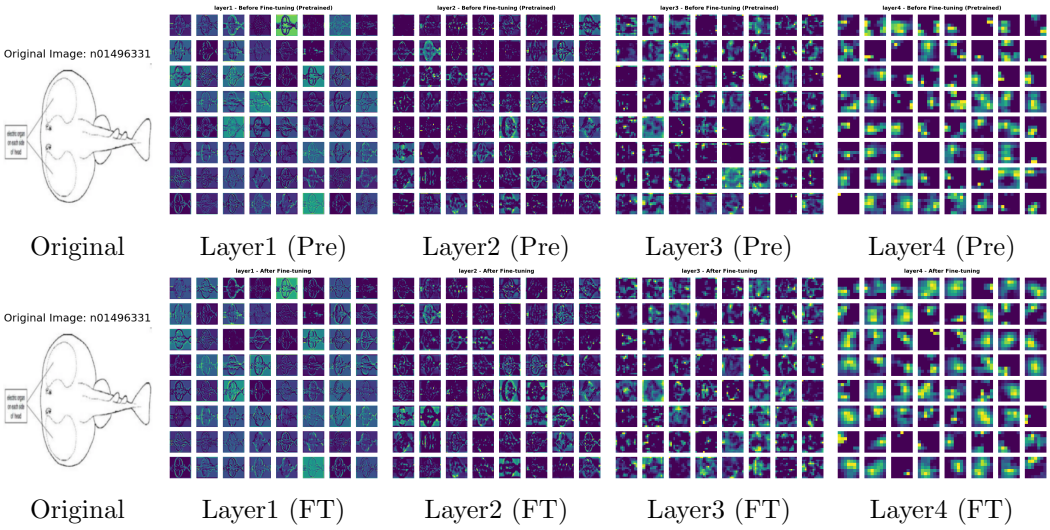

Figure 9: Image 5: Feature maps comparison (Pre=Pretrained, FT=Fine-tuned)

Observations:

- **Early Layers (Layer1, Layer2):** Feature maps are relatively similar between pretrained and fine-tuned models. These layers detect low-level features like edges and basic shapes, which are useful for both natural images and sketches.

- **Last Layer (Layer3, Layer4):** Fine-tuned models show sharper, more localized activations on sketch contours, while pretrained models have more diffuse activations due to their training on textured natural images.

(c)

Ans: ResNet18 was fine-tuned with all convolutional and fully-connected layer parameters frozen, updating only BatchNorm parameters (γ and β) and the final classification layer.

Training Configuration:

- Optimizer: Adam (lr=0.001, weight_decay=1e-4)
- Scheduler: StepLR (step_size=5, gamma=0.1)
- Epochs: 15
- Batch size: 32
- Data augmentation: Random affine transformations

Trainable Parameters:

- BatchNorm parameters: ∼9,600 parameters
- Final FC layer: ∼51,300 parameters
- Total trainable: ∼60,900 parameters (∼0.5% of total)

Results:

Metric	Full Fine-tuning	BatchNorm Only
Training Accuracy	96.48%	84.07%
Validation Accuracy	80.00%	72.93%
Test Accuracy	**80.04%**	**74.21%**

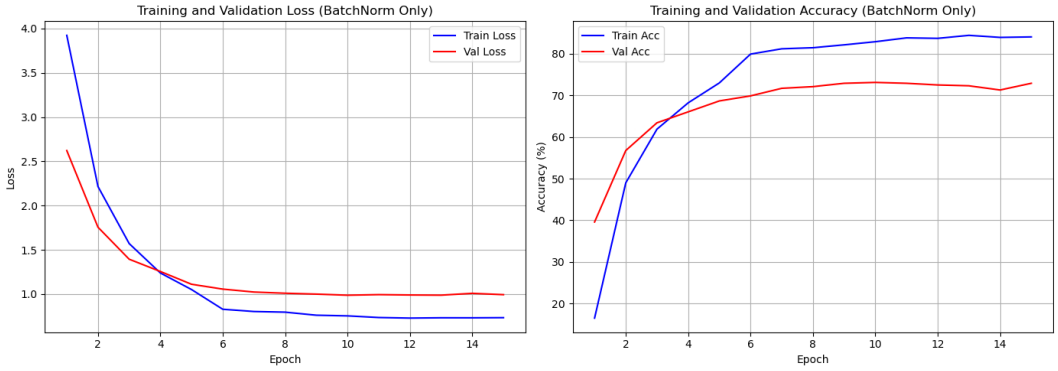

Figure 10: Training curves for BatchNorm-only fine-tuning.

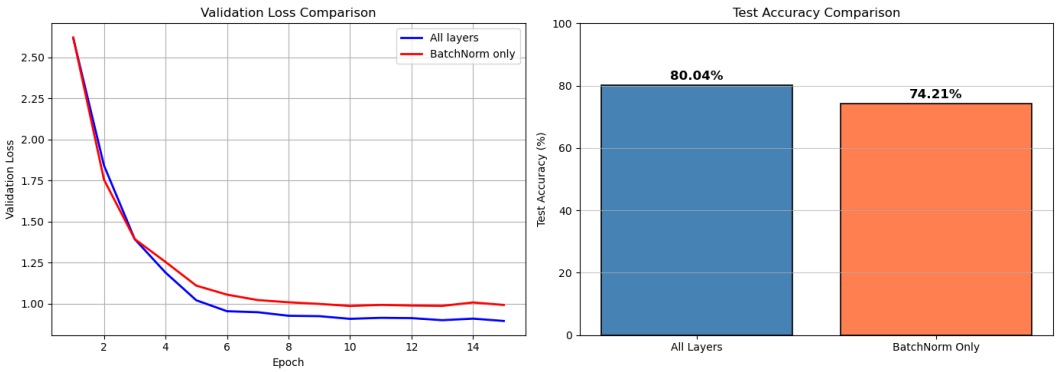

Figure 11: Comparison of full fine-tuning vs BatchNorm-only fine-tuning.

(d)

Ans: Observations:

BatchNorm-only fine-tuning achieves surprisingly good performance (74.21%) despite training only ∼0.5% of the parameters. The performance gap with full fine-tuning is only ∼6%. With few hyper paramater tuning, it may be better than finetuning the whole model sometimes

Why BatchNorm Adaptation Works:

1. **Domain Shift is Statistical:** The primary difference between natural images and sketches lies in their statistical properties:
   - Natural images: Rich colors, textures, varying intensities

- Sketches: Binary-like (black lines on white), sparse activations, different intensity distributions

2. **BatchNorm's Role:** BatchNorm layers normalize activations using learned parameters:

$$\hat{x} = \gamma \cdot \frac{x - \mu}{\sqrt{\sigma^2 + \epsilon}} + \beta \tag{1}$$

   By updating γ (scale) and β (shift), the network recalibrates feature responses for the new domain. This effectively "re-centers" and "re-scales" features to match sketch statistics.

3. **Feature Reuse:** The convolutional filters learned from ImageNet still detect meaningful patterns (edges, shapes) in sketches. BatchNorm adaptation allows the network to use these features with adjusted magnitudes appropriate for the sketch domain.

4. **Efficient Adaptation:** BatchNorm parameters are very few ($\sim$9K), making this an extremely parameter-efficient transfer learning technique.

Conclusion: BatchNorm adaptation is effective for domain adaptation when the domains share similar low-level features and the main difference is in feature statistics. For sketches, both approaches work well because sketches still contain edge-like structures that ImageNet-trained filters can detect.