Methodology

Data augumentation:

1. Flip
2. Color(Contrast, Saturation, Gray Scale)
3. Adding noise

Feature extraction:

1. Histogram of Orientated Gradient (HOG)

When dealing with low-resolution images like 32x32, the traditional HOG feature extraction approach may not be as effective due to the limited amount of information available. However, you can still apply some modifications to make the HOG features more meaningful in such cases. Here are a few suggestions:

**Increase the cell size**: Since the image resolution is low, using smaller cell sizes may result in insufficient information for effective feature extraction. Increasing the cell size can help capture more meaningful gradients and improve the representation. You can experiment with larger cell sizes, such as 8x8 or 16x16.

**Decrease the block size**: In the standard HOG approach, blocks of cells are used for normalization. However, in low-resolution images, smaller block sizes may be more suitable to avoid excessive averaging. You can try reducing the block size, such as using a 1x1 or 2x2 block.

**Adjust the number of orientation bins**: The number of orientation bins in the HOG feature extraction process determines the level of detail captured in the gradients. In low-resolution images, a smaller number of bins may be sufficient to capture the main orientation information. You can experiment with reducing the number of bins, such as using 4 or 6 bins instead of the typical 9 or 12.

1. color-based features (Top Color Extractor) (Color Entropy)
2. SIFT
3. texture descriptors (Haralick Texture)
4. Local Binary Patterns (LBP)

Classifier:

1. Linear SVM
2. Kernel SVM
3. Logistic regression
4. Random forest
5. KNN

Experiment Results:

Logistic Regression (C=1) + raw\_feature = 0.4017

Logistic Regression (C=0.1) + raw\_feature = 0.3985

Logistic Regression (C=0.01) + raw\_feature = 0.4004

Logistic Regression (C=0.001) + raw\_feature = 0.3998

**This series of experiment compares *num\_bins***

Logistic Regression (C=1) +

HOG\_feature (win\_size = (32, 32), block\_size = (32, 32), block\_stride = (8, 8), cell\_size = (16, 16), num\_bins = 6) = 0.3979

Logistic Regression (C=1) +

HOG\_feature (win\_size = (32, 32), block\_size = (32, 32), block\_stride = (8, 8), cell\_size = (16, 16), num\_bins = 9) = 0.4227

Logistic Regression (C=1) +

HOG\_feature (win\_size = (32, 32), block\_size = (32, 32), block\_stride = (8, 8), cell\_size = (16, 16), num\_bins = 12) = 0.4245

**This series of experiment compares *cell\_size***

Logistic Regression (C=1) +

HOG\_feature (win\_size = (32, 32), block\_size = (32, 32), block\_stride = (8, 8), cell\_size = (16, 16), num\_bins = 12) = 0.4245

Logistic Regression (C=1) +

HOG\_feature (win\_size = (32, 32), block\_size = (32, 32), block\_stride = (8, 8), cell\_size = (8, 8), num\_bins = 12) = 0.52

Logistic Regression (C=1) +

HOG\_feature (win\_size = (32, 32), block\_size = (32, 32), block\_stride = (8, 8), cell\_size = (4, 4), num\_bins = 12) = 0.5413

**This series of experiment compares *block\_size***

Logistic Regression (C=1) +

HOG\_feature (win\_size = (32, 32), block\_size = (32, 32), block\_stride = (8, 8), cell\_size = (4, 4), num\_bins = 12) = 0.5413

Logistic Regression (C=1) +

HOG\_feature (win\_size = (32, 32), block\_size = (24, 24), block\_stride = (8, 8), cell\_size = (4, 4), num\_bins = 12) = 0.5589

Logistic Regression (C=1) +

HOG\_feature (win\_size = (32, 32), block\_size = (16, 16), block\_stride = (8, 8), cell\_size = (4, 4), num\_bins = 12) = 0.5624

**This series of experiment compares *block\_stride***

Logistic Regression (C=1) +

HOG\_feature (win\_size = (32, 32), block\_size = (16, 16), block\_stride = (8, 8), cell\_size = (4, 4), num\_bins = 12) = 0.5624

Logistic Regression (C=1) +

HOG\_feature (win\_size = (32, 32), block\_size = (16, 16), block\_stride = (4, 4), cell\_size = (4, 4), num\_bins = 12) = 0.5639

SVC(kernel='rbf', max\_iter=3000) + HOG\_feature + TopColor\_feature = 0.6626

Logistic Regression (C=1) +

HOG\_feature (win\_size = (32, 32), block\_size = (16, 16), block\_stride = (8, 8), cell\_size = (4, 4), num\_bins = 12) + TopColor\_feature= 0.5722

**This series of experiment compares *LinearSVM and kernel SVM***

SVC(kernel='rbf', max\_iter=3000) + HOG\_feature + TopColor\_feature = 0.6626

LinearSVC(multi\_class='ovr', max\_iter=3000, C=1) + HOG\_feature + TopColor\_feature = 0.5488

SVC(kernel='linear', max\_iter=3000) + HOG\_feature + TopColor\_feature = 0.5034

**Gamma**

model = SVC(kernel='rbf', max\_iter=3000, gamma=1) + HOG\_feature + TopColor = 0.6004

model = SVC(kernel='rbf', max\_iter=3000, gamma=0.3) + HOG\_feature + TopColor = 0.6109

model = SVC(kernel='rbf', max\_iter=3000, gamma=0.1) + HOG\_feature + TopColor = 0.6764

model = SVC(kernel='rbf', max\_iter=3000, gamma=0.08) + HOG\_feature + TopColor = 0.6755

model = SVC(kernel='rbf', max\_iter=3000, gamma=0.01) + HOG\_feature + TopColor = 0.596

model = SVC(kernel='rbf', max\_iter=3000, gamma=0.001) + HOG\_feature + TopColor= 0.5302

model = SVC(kernel='rbf', max\_iter=3000, gamma=scale) + HOG\_feature + TopColor = 0.6638

**Whether TopColor is useful?**

model = SVC(kernel='rbf', max\_iter=3000, gamma=1, C=1) + HOG\_feature= 0.6009

model = SVC(kernel='rbf', max\_iter=3000, gamma=1, C=1) + HOG\_feature + TopColor = 0.6004

**Analysis**

Why SVM is better than logistical regression. Logistical regression is vulnerable under occurrence of outliers.