Artsy Classifer

Practical Machine Learning Project

Introduction

In this project, we set out to explore how augmenting a machine learning model with additional features affects its ability to classify paintings by art style. Our challenge was to determine if integrating traditional image processing techniques with advanced machine learning algorithms could enhance the model's performance in handling the complex task of art style classification.

Datasets Used

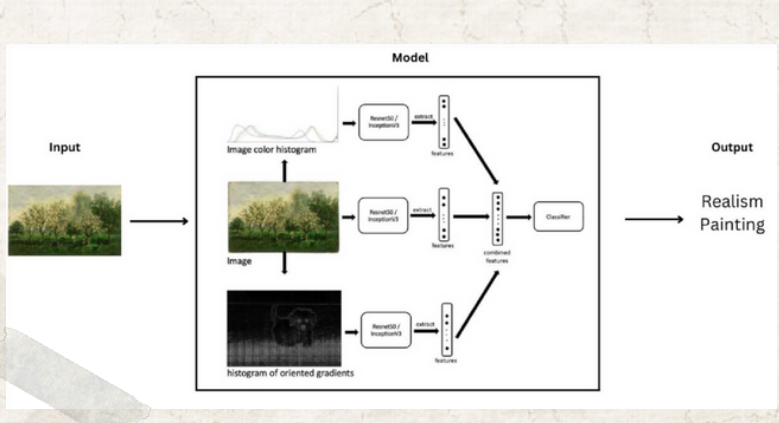
We utilized two major datasets for our analysis: over 1,500 images from Kaggle and 5,000 images from the WikiArt dataset. These datasets include diverse art styles such as Abstract Expressionism, Nouveau Modern, Fauvism, Minimalism, and Rococo, providing a rich basis for training our models.

Methods

Our approach was centered around the ResNet50 deep neural network. To examine the impact of feature augmentation, we configured four different models, each varying in the combination of features used. We experimented with Histogram of Oriented Gradients (HOG), color histograms, both, or none, to assess how each set of features influenced the model's ability to classify art styles accurately.

```python
class FeatureExtractor:
    def __init__(self):
        self.resnet50 = models.resnet50(pretrained=True)
        self.resnet50 = torch.nn.Sequential(*list(self.resnet50.children())[:-1])
        self.resnet50.eval()

    def tensor_to_numpy(self, image_tensor):
        img = transforms.ToPILImage()(image_tensor).convert("RGB")
        return np.array(img)

    def numpy_to_tensor(self, image_np):
        return transforms.ToTensor()(image_np)

    def extract_hog(self, image_np):
        features, _ = hog(image_np, orientations=8, pixels_per_cell=(16, 16),
                          cells_per_block=(1, 1), visualize=True, multichannel=True)
        return features

    def extract_color_histogram(self, image_np, bins=32):
        hist_features = []
        for i in range(3):
            hist, _ = np.histogram(image_np[:, :, i], bins=bins, range=(0, 255))
            hist_features.extend(hist)
        return hist_features

    def extract_resnet50(self, image_tensor):
        with torch.no_grad():
            features = self.resnet50(image_tensor.unsqueeze(0))
        return features.squeeze().numpy()

    def extract_all_features(self, image_tensor):
        image_np = self.tensor_to_numpy(image_tensor)
        hog_features = self.extract_hog(image_np)
        color_hist_features = self.extract_color_histogram(image_np)
        resnet_features = self.extract_resnet50(image_tensor)
        return np.concatenate([hog_features, color_hist_features, resnet_features])

feature_extractor = FeatureExtractor()
```

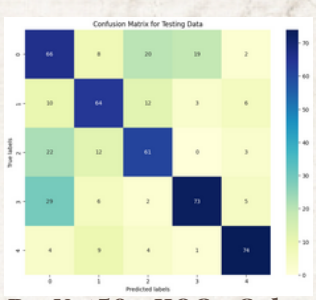
Experiment Details



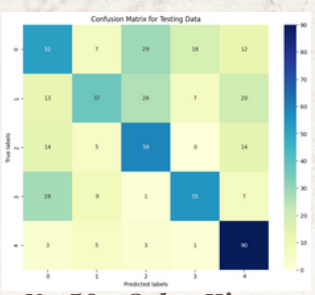
The experiment was structured to compare the performance across the four model variants. Each model underwent the same training regime but differed in the input features: some utilized basic image data, others included HOG, some added color histograms, and one combined both. This setup allowed us to meticulously evaluate the contribution of each feature set to the model's overall accuracy, precision, recall, and F1-score.

Results Highlights

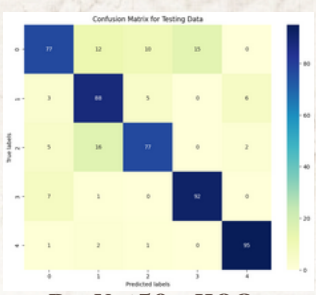
Our findings revealed varied impacts of feature augmentation on model performance. The model using both HOG and basic image data performed the best, achieving an accuracy of 86.02%. This highlighted the significant role of texture captured by HOG features in art style classification. Conversely, models that included color histograms either alone or combined with HOG did not perform as well, indicating that color features alone might not provide sufficient discriminative power for the task at hand.



ResNet50 + HOG + Color Histogram



ResNet50 + Color Histogram



ResNet50 + HOG

ResNet50