Varit Kobutra

Professor Patricia McManus

Computer Vision ITAI 1378: 12461

20 June 2024

Reflective Journal: L05 Image Classification with SVM

This lab introduced me to the fundamentals of image classification using Support Vector Machines (SVM) and the CIFAR-10 dataset. It deepened my understanding of machine learning principles and illuminated the practical challenges inherent in model training. Key takeaways include:

Understanding SVM and Image Classification

- The SVM algorithm is a robust tool for identifying optimal decision boundaries between different classes of data (Burges 121).
- In image classification, SVM's strength lies in its ability to transform image data into a higher-dimensional space, revealing intricate patterns and relationships that are otherwise difficult to discern (Cortes and Vapnik 273).

Data Preparation, Model Training, and Evaluation

- The CIFAR-10 dataset was loaded, preprocessed (visualization, grayscale conversion, flattening), and split into training and testing sets.
- Challenges were encountered with long training times, both locally and on Google Colab, preventing full training as per the provided notebook instructions.
- To address this, I explored alternative approaches on Kaggle and GitHub, gaining valuable insights into hyperparameter optimization and model training strategies.
- Model performance was assessed using accuracy on the testing set, and predictions were visualized to understand strengths and weaknesses.

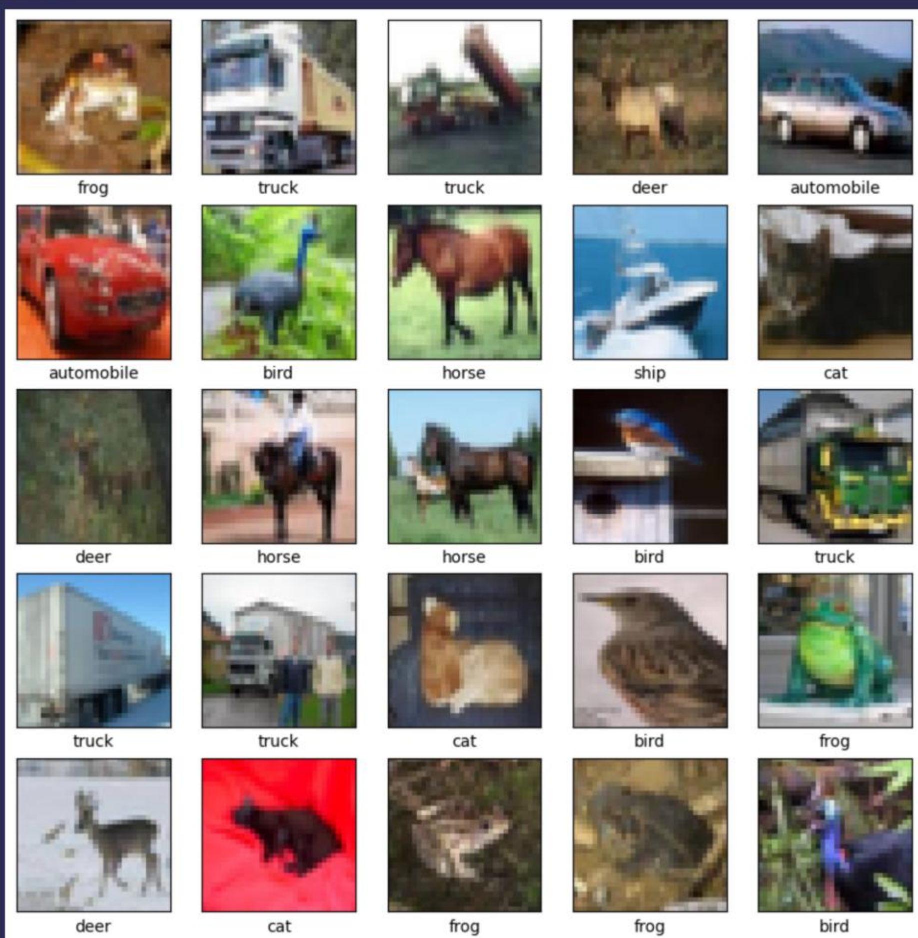
Challenges and Insights

- Hyperparameter tuning was explored to optimize accuracy, emphasizing the importance of understanding their impact on model performance.
- Principal Component Analysis (PCA) was applied to reduce dimensionality and accelerate training, highlighting the trade-off between computational efficiency and model accuracy (resulting accuracy: 31.39%).

Proof of Work

```python
# Visualize some images
plt.figure(figsize=(10, 10))
for i in range(25):
    plt.subplot(5, 5, i+1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(x_train[i])
    plt.xlabel(class_names[y_train[i][0]])
plt.show()
```

✓ 0.2s



```python
# Standardize the data
scaler = StandardScaler()
x_train_scaled = scaler.fit_transform(x_train_split)
x_val_scaled = scaler.transform(x_val_split)
x_test_scaled = scaler.transform(x_test_flat)

# Apply PCA for dimensionality reduction
pca = PCA(n_components=100)  # Reduce to 100 principal components
x_train_pca = pca.fit_transform(x_train_scaled)
x_val_pca = pca.transform(x_val_scaled)
x_test_pca = pca.transform(x_test_scaled)
```

✓ 14.2s

```python
# Train the SVM classifier
svm_classifier = SVC(kernel='linear')  # Linear kernel for faster training
svm_classifier.fit(x_train_pca, y_train_split.ravel())
```

✓ 25m 36.0s

```
SVC
SVC(kernel='linear')
```

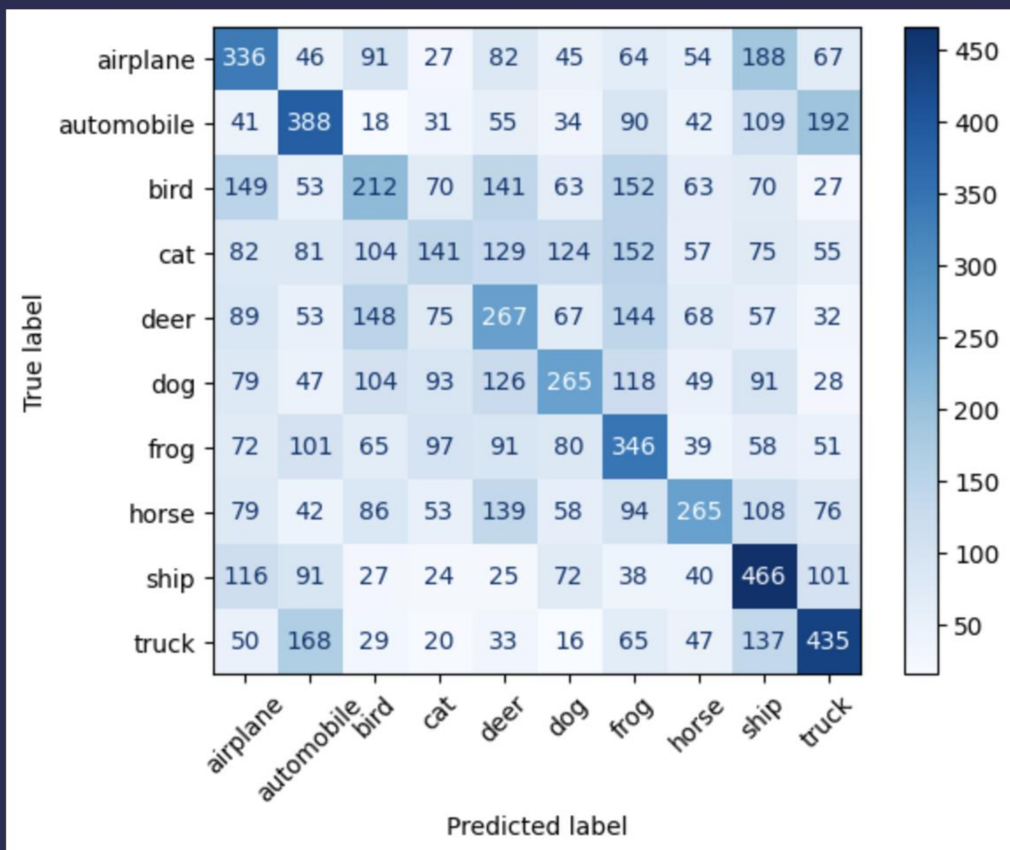
```python
# Make predictions on the testing set
y_pred = svm_classifier.predict(x_test_pca)

# Evaluate the model's performance
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy * 100:.2f}%")
print("Classification Report:\n", classification_report(y_test, y_pred))
```

✓ 14.3s

```
Accuracy: 31.21%
Classification Report:
               precision    recall  f1-score   support

           0       0.31      0.34      0.32      1000
           1       0.36      0.39      0.37      1000
           2       0.24      0.21      0.23      1000
           3       0.22      0.14      0.17      1000
           4       0.25      0.27      0.26      1000
           5       0.32      0.27      0.29      1000
           6       0.27      0.35      0.31      1000
           7       0.37      0.27      0.31      1000
           8       0.34      0.47      0.40      1000
           9       0.41      0.43      0.42      1000

    accuracy                           0.31     10000
   macro avg       0.31      0.31      0.31     10000
weighted avg       0.31      0.31      0.31     10000
```

```python
# Compute the confusion matrix
cm = confusion_matrix(y_test, y_pred)


# Plot the confusion matrix
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=class_names)
disp.plot(cmap=plt.cm.Blues)
plt.xticks(rotation=45)  # Rotate the x labels for better readability
plt.show()
```

✓ 0.1s



Works Cited

Burges, Christopher J.C. "A Tutorial on Support Vector Machines for Pattern Recognition." Data Mining and Knowledge Discovery, vol. 2, no. 2, 1998, pp. 121-167.

Cortes, Corinna, and Vladimir Vapnik. "Support-Vector Networks." Machine Learning, vol. 20, no. 3, 1995, pp. 273-297.

Krizhevsky, Alex. Learning Multiple Layers of Features from Tiny Images. Tech report, University of Toronto, 2009. https://www.cs.toronto.edu/~kriz/cifar.html