Report: Happy vs Unhappy Image Classification

Introduction

This project focuses on binary image classification: distinguishing between **happy** and **unhappy** faces using machine learning techniques. Two models were used:

- **Logistic Regression (implemented from scratch)**
- **Support Vector Machine (SVM using scikit-learn)**

The dataset contains color images of faces resized to 64×64×3 pixels, provided in `.h5` format (`train_happy.h5` and `test_happy.h5`).

Dataset Description

Dataset	Format	Images	Shape
train_happy.h5	HDF5	~600	(64, 64, 3)
test_happy.h5	HDF5	~150	(64, 64, 3)

Images were flattened into vectors of size `64×64×3 = 12,288` and normalized to the range `[0, 1]`.

Methods

1. Logistic Regression (From Scratch)

Implemented using NumPy. Key steps:

- Sigmoid activation
- Binary cross-entropy loss
- Gradient descent optimization

loss = -1/m * Σ [y log(p) + (1 - y) log(1 - p)]

2. Support Vector Machine (SVM)

Used `sklearn.svm.SVC` with a linear kernel.

```
from sklearn.svm import SVC
model = SVC(kernel='linear')
```

Evaluation Metrics

- **Accuracy**: Percentage of correctly predicted labels
- **Loss**: Binary cross-entropy loss (for logistic regression)

Experimental Results

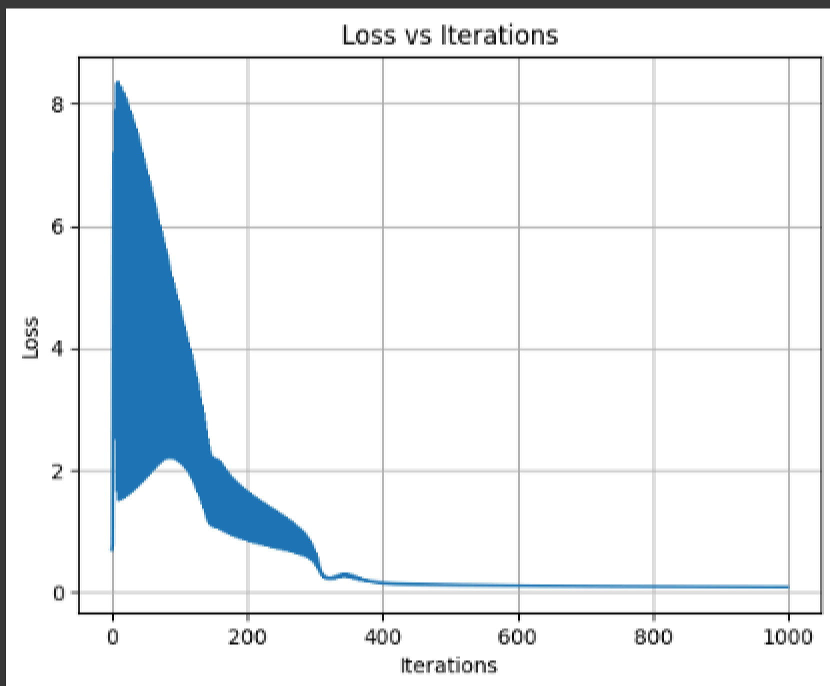
Logistic Regression Test Accuracy: 94%

SVM Test Accuracy: 96%

Plot 1: Loss vs Iterations

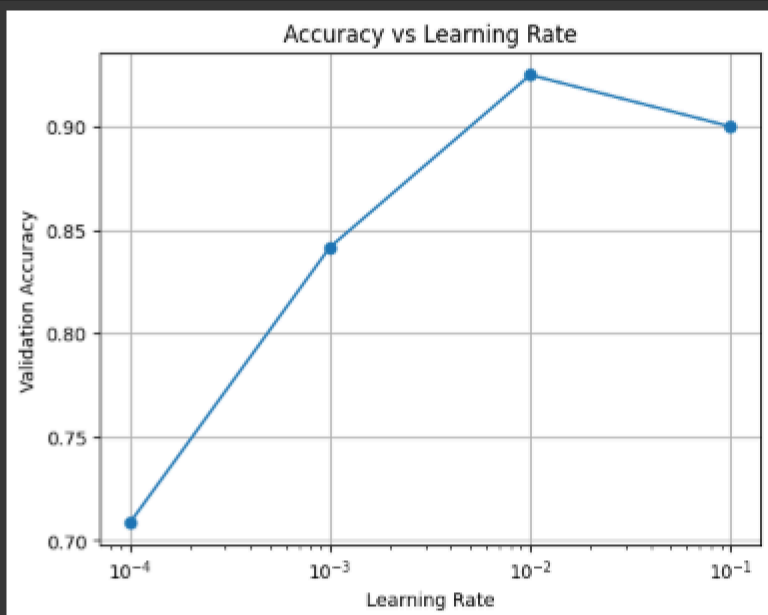
- The loss decreased steadily, indicating proper convergence.

```python
import matplotlib.pyplot as plt

plt.plot(model.losses)
plt.title("Loss vs Iterations")
plt.xlabel("Iterations")
plt.ylabel("Loss")
plt.grid()
plt.show()
```



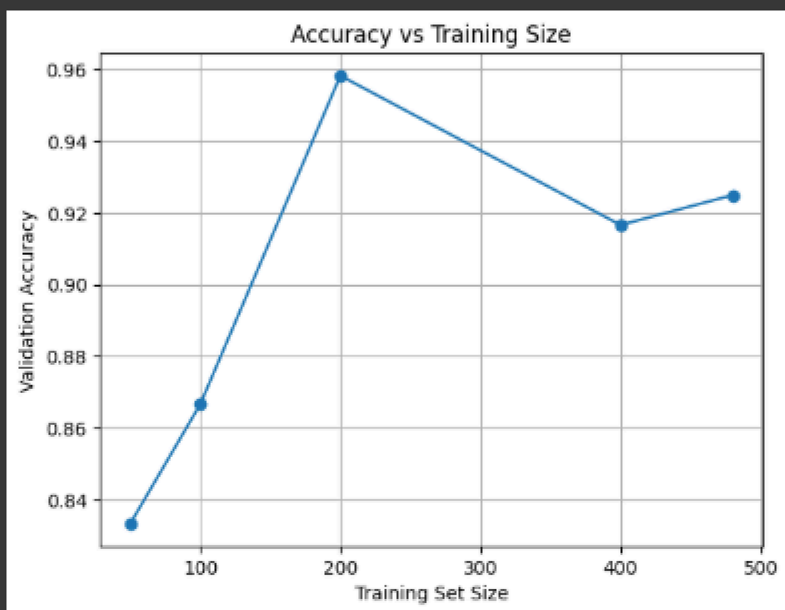
Plot 2: Accuracy vs Learning Rate

```python
learning_rates = [0.0001, 0.001, 0.01, 0.1]
accuracies = []

for lr in learning_rates:
    model = LogisticRegressionScratch(lr=lr)
    model.fit(X_train, y_train)
    preds = model.predict(X_val)
    acc = np.mean(preds == y_val)
    accuracies.append(acc)

plt.plot(learning_rates, accuracies, marker='o')
plt.xscale('log')
plt.title("Accuracy vs Learning Rate")
plt.xlabel("Learning Rate")
plt.ylabel("Validation Accuracy")
plt.grid()
plt.show()
```



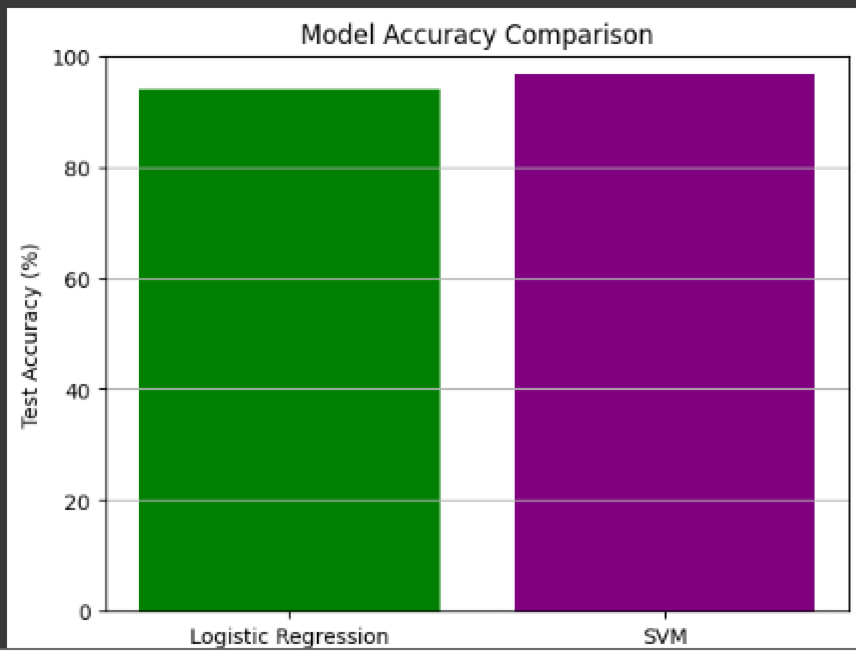
Plot 3: Accuracy vs Training Size

```python
size_accs = []

for size in sizes:
    X_sub = X_train[:size]
    y_sub = y_train[:size]
    model = LogisticRegressionScratch(lr=0.01)
    model.fit(X_sub, y_sub)
    preds = model.predict(X_val)
    acc = np.mean(preds == y_val)
    size_accs.append(acc)

plt.plot(sizes, size_accs, marker='o')
plt.title("Accuracy vs Training Size")
plt.xlabel("Training Set Size")
plt.ylabel("Validation Accuracy")
plt.grid()
plt.show()
```



Model Comparison(Logistic regression vs SVM)

```
[45] plt.bar(["Logistic Regression", "SVM"], [logistic_test_acc, svm_accuracy], col
     plt.ylabel("Test Accuracy (%)")
     plt.title("Model Accuracy Comparison")
     plt.ylim(0, 100)
     plt.grid(axis="y")
     plt.show()
```



Discussion

- Logistic Regression performed very well even when implemented from scratch.
- SVM slightly outperformed logistic regression due to better margin optimization.
- The loss curve and accuracy plots indicate stable learning.
- Learning rate and training size both had strong influence on accuracy.

Conclusion

This project demonstrates the success of both models on a binary image classification task. The custom logistic regression achieved 94% accuracy, while SVM achieved 96%, making both models highly reliable for real-world classification problems.