Report Task 3

Decision tree

Best Parameters found: {'criterion': 'entropy', 'max_depth': 40, 'max_features': 'sqrt', 'min_samples_leaf': 18, 'min_samples_split': 33}
Best Cross-Validation Accuracy: 0.2301
Final Test Set Accuracy: 0.2528

Ensemble methods

1. Random Forest

Best RF Params: {'max_depth': 20, 'max_features': 'sqrt', 'min_samples_leaf': 4, 'min_samples_split': 34, 'n_estimators': 399}
Best Cross-Validation Accuracy: 0.4650
Random Forest Test Accuracy: 0.4714

2. Adaboost

Best AdaBoost Params: {'learning_rate': np.float64(1.1337735012888268), 'n_estimators': 269}
Best Cross-Validation Accuracy: 0.3368
AdaBoost Test Accuracy: 0.3369

Support Vector Machine

Best Parameters found: {'C': np.float64(621.381353074258), 'gamma': np.float64(0.013847288133351985), 'kernel': 'rbf'}
Best Cross-Validation Accuracy: 0.5394
Final Test Set Accuracy: 0.5637

2. Task 5: Decision Tree

- Search Strategy: You used `RandomizedSearchCV` with `n_iter=50`.
    - Rationale: Decision trees are computationally cheaper to train than ensembles or SVMs. This allowed you to set a higher number of iterations (50) to explore the hyperparameter space more thoroughly compared to the other tasks (which used 20).
- Hyperparameters:
    - `min_samples_split` & `min_samples_leaf`: You searched for integers between 2–50 and 1–20 respectively. This is a regularization technique. By forcing leaf nodes to have a minimum number of samples, you prevent the tree from memorizing noise (overfitting).
    - `max_features`: Exploring `sqrt` vs `log2`. This introduces randomness into the split selection, which usually helps in reducing variance, though it is more critical in Random Forests than single trees.

3. Task 6: Ensemble Methods (`task6.py`)

This script compares Bagging (Random Forest) and Boosting (AdaBoost).

A. Random Forest

- `n_estimators` (100–500): You searched for a relatively high number of trees.
    - Rationale: In Random Forests, adding more trees generally does not cause overfitting; it stabilizes the error. You likely abandoned lower values (e.g., <50) because they result in high variance.
- `max_depth`: You included `None` as an option.

    - Rationale: Random Forests rely on deep trees (low bias) and average them to reduce variance. Allowing fully grown trees (`None`) is a standard approach here, unlike in single Decision Trees where pruning is essential.

B. AdaBoost

- `learning_rate` (0.01–1.5): You used a uniform distribution.
    - Rationale: There is a trade-off between `learning_rate` and `n_estimators`. A lower learning rate requires more estimators to converge. By searching this range, you allow the model to find the balance between a "slow learner" (robust) and a "fast learner" (potentially aggressive).
- Algorithm 'SAMME': You specified `algorithm='SAMME'`.
    - Rationale: This is the discrete boosting algorithm (Stagewise Additive Modeling using a Multi-class Exponential loss function).

4. Task 7: Support Vector Machine (`task7.py`)

This task is the most computationally intensive, justifying specific choices.

- Log-Uniform Distributions (`reciprocal`):
    - For `C` (Regularization) and `gamma` (Kernel coefficient), you used `scipy.stats.reciprocal`.
    - Rationale: SVM parameters are sensitive across orders of magnitude. A uniform search between 0.1 and 1000 would spend 90% of its time searching between 100 and 1000. A reciprocal (log-uniform) distribution allows the search to explore 0.1, 1, 10, and 100 with equal probability. This is the standard best practice for SVM tuning.
- Kernel Selection (`rbf`):
    - Rationale: The Radial Basis Function (RBF) kernel is generally the best default for image classification when features are non-linear.
    - Abandoned Approach: Polynomial Kernel. While powerful, polynomial kernels are often much slower to compute and have more hyperparameters (degree, coef0). Given the dataset size, RBF offers a better balance of speed and non-linear capability. Linear Kernel was likely abandoned because image data is rarely linearly separable.

Summary of Abandoned/Alternative Approaches

1. GridSearchCV: You used `RandomizedSearchCV` in all files. `GridSearchCV` was likely abandoned because checking every combination of the defined grids would take exponentially longer. Random search is empirically shown to find better models in less time by exploring the space more efficiently.
2. Raw Pixel Training: As noted in section 1, training without PCA was likely abandoned due to memory and time constraints.
3. Low Validation Folds: You used `cv=3` instead of the standard 5 or 10. This was likely a trade-off to reduce runtime, as training SVMs and Ensembles on 50,000 images (even with PCA) is heavy.