UNIVERSITY OF
LINCOLN

Adewusi Adedapo Adetomiwa

Student ID 27487532

27487532@students.lincoln.ac.uk

Module : Big Data Analytics & Modelling

Dr. Miao Yu

Table of Contents

Introduction

In the contemporary landscape of digital innovation, the problem of improving the processing and interpreting of vast amounts of visual data and classification has become a cornerstone of technological advancement. Image classification, a transformative application of machine learning, enables computers to identify and categorize objects within images with remarkable precision. This capability has profound implications across various domains, including wildlife conservation, automated monitoring systems, and veterinary diagnostics. This report delves into the realm of big data analytics, leveraging the Kaggle Animal Image Classification Dataset to design, implement, and evaluate scalable ML & Deep learning used in distinguishing between cats, dogs, and snakes based on visual features such as fur, scales, and texture.

The study aims to develop and critically compare five ML models— SVM, Random Forest, AdaBoost, K-Nearest Neighbors (KNN), and Convolutional Neural Network (CNN)—alongside an enhanced CNN utilizing the pre-trained ResNet50 architecture. By harnessing big data analytics principles, this investigation addresses challenges such as image variability, class similarity, and computational scalability. The dataset, comprising 3000 color images evenly distributed across three classes (1000 each of cats, dogs, and snakes), provides a robust foundation for testing model efficacy under diverse real-world conditions. Drawing on foundational and niche research in machine learning and image processing, this report aims to contribute to the evolving field of scalable image classification, offering insights into precision, efficiency, and practical applicability.

Dataset Summary

The Kaggle Animal Image Classification Dataset, curated by Saxena (2023), serves as the empirical backbone of this study. Publicly accessible at it consists of 3000 RGB color images organized into three subfolders corresponding to cats, dogs, and snakes. Each class contains 1000 images, capturing animals in varied settings—domestic environments for pets and natural or controlled habitats for snakes. The dataset's diversity in lighting, backgrounds, and poses mirrors real-world complexities, making it an ideal testbed for evaluating scalable image classification solutions.

The images vary in dimensions, necessitating preprocessing to standardize inputs for modeling. This variability, combined with the balanced class distribution, aligns with benchmarks established in seminal works on image recognition (Krizhevsky et al., 2012). The dataset's structure and scale enable a comprehensive assessment of both traditional ML algorithms and advanced deep learning techniques, providing a realistic scenario for big data analytics applications.

Table 1: Classes and total images of Animal dataset containing Cats, Dogs, and Snakes in their proportion

Classes	Total Images
Cats	1000
Dogs	1000
Snakes	1000

The image below shows a sample of the images of the animal classes

Figure 1: Showing the classes of the animal dataset which are cat, dog and snake

Image classification has evolved significantly with the advent of vast data through machine learning, transitioning from traditional statistical methods to sophisticated neural network architectures. Early approaches, such as those employing SVMs (Vapnik, 1995), relied on hand-crafted features and hyperplane separation in high-dimensional spaces. While effective for smaller datasets, these methods struggle with the scale and complexity of modern image data. Ensemble techniques like Random Forests (Breiman, 2001) introduced robustness to noise and high-dimensional inputs, yet they rely on flattened features which is tantamount to loss of spatial relationships inherent in images.

LeCun et al. (1989) highlighted the deep learning model like CNNs, marked a paradigm shift in image classification. CNNs depends on convolutional layers for extraction, making them exceptionally fit for visual research. Subsequent advancements, such as the development and improvement with pre-trained models with CNN architecture like ResNet50 (He et al., 2016), have further enhanced performance by utilizing transfer learning to adapt features learned from large datasets (e.g., ImageNet) to specific tasks. This approach is particularly valuable for smaller datasets, where training from scratch may lead to overfitting.

Research on animal image classification highlights the challenges of variability in appearance and texture. Rawat and Wang (2017) emphasize the importance of deep learning in distinguishing intricate patterns, such as scales versus fur, while Freund and Schapire (1997) explore boosting techniques like AdaBoost, noting their sensitivity to noise in complex datasets. Less commonly cited studies, such as Biau and Scornet (2016), provide deeper insights into Random Forest limitations, underscoring the need for spatial-aware models in image tasks. This study builds on these foundations, integrating big data tools and niche methodologies to address the specific challenge of classifying cats, dogs, and snakes.

Methodology

Firstly, prepare the downloaded dataset for testing & training. Best practices in image processing employ several preprocessing steps (Chollet, 2017). The image resizing was done, and all images were 64 by 64 pixels to ensure uniformity and reduce computational complexity while preserving the key features of the dataset.

```python
# Set random seed for reproducibility
np.random.seed(42)

# Load dataset from local directory
data_dir = 'C:\Even\Animals'
class_names = ['cats', 'dogs', 'snakes']
image_size = (64, 64)  # Resize images to 64x64 for consistency
X, y = [], []
for label, class_name in enumerate(class_names):
    class_dir = os.path.join(data_dir, class_name)
    for img_name in os.listdir(class_dir):
        img_path = os.path.join(class_dir, img_name)
        try:
            img = load_img(img_path, target_size=image_size)
            img_array = img_to_array(img) / 255.0  # Normalize to [0, 1]
            X.append(img_array)
            y.append(label)
        except:
            print(f"Error loading image: {img_path}")

X = np.array(X)
y = np.array(y)
```

Figure 2: Creating a directory for the dataset, resizing the image dataset to 64 by 64, normalizing to 0,1, and dividing by 255 to accommodate non-spatial models.

Then comes normalization in which the pixel values were scaled to [0,1] by dividing by 255, standardizing inputs for neural network and improving convergence (Goodfellow et al., 2016). Flattening for SVM, Random Forest, and AdaBoost was done and was flattened to 1D vectors (64 by 64 by 3= 12,288 features) to accommodate non-spatial models (Pedregosa et al.,2011) shown in Figure 3.

Enhancing SVM model performance, standardization of flattened features was done (Zero mean, Unit variance), as recommended by Vapnik, 1995. Label Encoding was done, and cats=0, dogs= 1, and snakes= 2 were encoded for neural networks to support multi-class classification (Abadi et al., 2016). The preprocessing steps for the dataset were splits animal dataset: Test (20%) and Training (80%) scores with stratification to maintain class proportion and mitigate potential imbalances (Kohavi, 1995), shown in Figure 3

```python
# Split dataset into training and test sets (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)

# Flatten images for SVM, Random Forest, and AdaBoost
X_train_flat = X_train.reshape(X_train.shape[0], -1)
X_test_flat = X_test.reshape(X_test.shape[0], -1)

# Standardize features for SVM
scaler = StandardScaler()
X_train_flat = scaler.fit_transform(X_train_flat)
X_test_flat = scaler.transform(X_test_flat)

# Encoding labels for neural networks
y_train_cat = to_categorical(y_train, len(class_names))
y_test_cat = to_categorical(y_test, len(class_names))

# Preprocess images for ResNet50 (ImageNet preprocessing)
X_train_resnet = np.array([preprocess_input(img * 255.0) for img in X_train])
X_test_resnet = np.array([preprocess_input(img * 255.0) for img in X_test])
```

Figure 3: Splitting the dataset to 80% Train and 20% Test with flattening of the images, standardizing and encoding it for neural networks (FCNN, CNN), and ResNet50 to improve the CNN.

Before the machine learning model was selected, some metrics were set to evaluate its performance and determine which one is suitable and how to improve it: , Recall, Precision, F1-score and accuracy metrics.

```python
# Function to evaluate and plot confusion matrix
Tabnine | Edit | Test | Explain | Document
def evaluate_model(model, X_test, y_test, model_name, is_keras=False):
    if is_keras:
        y_pred = model.predict(X_test)
        y_pred = np.argmax(y_pred, axis=1)
        y_true = np.argmax(y_test, axis=1)
    else:
        y_pred = model.predict(X_test)
        y_true = y_test

    # Calculate performance metrics
    accuracy = accuracy_score(y_true, y_pred)
    precision = precision_score(y_true, y_pred, average='weighted')
    recall = recall_score(y_true, y_pred, average='weighted')
    f1 = f1_score(y_true, y_pred, average='weighted')

    # Print metrics
    print(f"\n{model_name} Performance:")
    print(f"Accuracy: {accuracy:.4f}")
    print(f"Precision: {precision:.4f}")
    print(f"Recall: {recall:.4f}")
    print(f"F1-Score: {f1:.4f}")
```

Figure 4: Evaluation part with precision, recall, precision and F1 Score and confusion matrix for the models

Five ML models and an enhanced CNN were implemented, leveraging Scikit-Learn for traditional algorithms and TensorFlow/Keras for neural networks:

1. Random Forest: Utilized RandomForestClassifier with parameters optimized via GridSearchCV (max_depth: [10, 20], n_estimators: [100, 200], 3-fold cross-validation). Robust to noise but limited by flattened inputs (Biau & Scornet, 2016).

```python
# 1. Random Forest Model
print("Training Random Forest...")
rf_params = {'n_estimators': [100, 200], 'max_depth': [10, 20]}
rf = GridSearchCV(RandomForestClassifier(random_state=42), rf_params, cv=3, n_jobs=-1)
rf.fit(X_train_flat, y_train)
print("Best Random Forest Parameters:", rf.best_params_)
results['Random Forest'] = evaluate_model(rf, X_test_flat, y_test, "Random Forest")
```

Figure 5: Random Forest Model with a max_depth of 20 and n_estimator of 200 using the flattened vector to show the best parameter used in the model and the metrics detaill.

2. AdaBoost: Implemented with AdaBoostClassifier (number of estimators: [50, 100], 3-fold cross-validation), learning rate: [0.1, 1.0]. Sensitive to noise in image data (Schapire, 2013).

```python
# 2. AdaBoost Model
print("Training AdaBoost...")
ada_params = {'n_estimators': [50, 100], 'learning_rate': [0.1, 1.0]}
ada = GridSearchCV(AdaBoostClassifier(random_state=42), ada_params, cv=3, n_jobs=-1)
ada.fit(X_train_flat, y_train)
print("Best AdaBoost Parameters:", ada.best_params_)
results['AdaBoost'] = evaluate_model(ada, X_test_flat, y_test, "AdaBoost")
```

Figure 6: AdaBoost model was explored with n_estimator, learning rate, and GridSeachCV with 3-fold CV on the test flattened images.

3. **SVM:** Employed SVC with parameters (C: [0.1, 10], using Linear and rbf kernel, coupled with degree of 2, 3, 3-fold cross-validation). Computationally intensive but effective for high-dimensional data (Vapnik, 1995).

```python
# 3. SVM Model
print("Training SVM...")
svm_params = {
    'C': [0.1, 1, 10],
    'kernel': ['rbf', 'linear', 'poly'],
    'degree': [2, 3]  # Only used for poly kernel
}
svm = GridSearchCV(SVC(), svm_params, cv=3, n_jobs=-1)
svm.fit(X_train_flat, y_train)
print("Best SVM Parameters:", svm.best_params_)
results['SVM'] = evaluate_model(svm, X_test_flat, y_test, "SVM")
```

Figure 7: Support Vector Machine model implemented with Kernel, learning rate of 10, kernel (RBF, linear, poly) with degree if it is poly with X_test_flattened images.

4. KNN: Used KNeighborsClassifier (weights: [‘uniform’, ‘distance’], n_neighbors: [3, 5, 7, 9], metric: [‘euclidean’, ‘manhattan’], 5-fold cross-validation). Simple yet sensitive to feature scaling.

```python
# 4. K-Nearest Neighbors (KNN)
print("Training KNN...")
knn = KNeighborsClassifier()
param_grid_knn = {
    'n_neighbors': [3, 5, 7, 9],
    'weights': ['uniform', 'distance'],
    'metric': ['euclidean', 'manhattan']
}
grid_knn = GridSearchCV(knn, param_grid_knn, cv=5, scoring='accuracy', n_jobs=-1)
grid_knn.fit(X_train_flat, y_train)
print(f"Best KNN Parameters: {grid_knn.best_params_}")
results['KNN'] = evaluate_model(grid_knn.best_estimator_, X_test_flat, y_test, "KNN")
```

Figure 8: KNN model built on n_neighbors of 3,5,7,9, weight, and metric is Euclidean, Manhattan, using the flattening Train and KNeighborsclassifier with 5 Cross-validation.

A metric called accuracy is suitable and used in Neural Networks and deep learning like FCNN, CNN, and ResNet50(Pre-trained). Goodfellow et al. (2016) note that accuracy is a “natural choice” for classification tasks owing to its simplicity and the ability to interpret images, especially when communicating results to non-experts (Goodfellow et al.,2016).

5. CNN: Built with two Conv2D layers (32 and 64 filters), two MaxPooling2D layers, a dense layer (128 neurons), ReLU activation, Adam optimizer, 20 epochs, and batch size of 32. Excels at spatial feature extraction (LeCun et al., 1989).

```python
# 5. Convolutional Neural Network (CNN)
print("Training CNN...")
cnn = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(image_size[0], image_size[1], 3)),
    MaxPooling2D((2, 2)),
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Flatten(),
    Dense(128, activation='relu'),
    Dropout(0.5),
    Dense(len(class_names), activation='softmax')
])
# Compile and train the model
cnn.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
cnn.fit(X_train, y_train_cat, epochs=10, batch_size=32, verbose=1)

# Evaluate the results
results['CNN'] = evaluate_model(cnn, X_test, y_test_cat, "CNN", is_keras=True)
```

Figure 9: CNN built on Keras sequential API with Convolutional layer with two MaxPooling2D, 3 neurons, Adam Optimizer, 20 Epoch, and batch size of 32.

CNN + ResNet50: Incorporated pre-trained ResNet50 with ImageNet weights, followed by GlobalAveragePooling2D, a dense layer (128 neurons), ReLU activation, 0.5 dropout, Adam optimizer, 10 epochs, and batch size of 32. Leverages transfer learning for enhanced performance (He et al., 2016)

```python
print("Training ResNet50...")
# Load and configure ResNet50
base_model = ResNet50(weights='imagenet', include_top=False, input_shape=(image_size[0], image_size[1], 3))
base_model.trainable = False  # Freeze base model layers

# Build the model
resnet = Sequential([
    base_model,
    GlobalAveragePooling2D(),
    Dense(128, activation='relu'),
    Dropout(0.5),
    Dense(len(class_names), activation='softmax')
])
# Compile the model
resnet.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
# Train the model with validation data
history = resnet.fit(
    X_train_resnet,
    y_train_cat,
    epochs=10,
    batch_size=32,
    validation_data=(X_test_resnet, y_test_cat),  # Use test set as validation
    verbose=1
)
# Evaluate the results
print("\nResNet50 Evaluation Metrics:")
results['ResNet50'] = evaluate_model(resnet, X_test_resnet, y_test_cat, "ResNet50", is_keras=True)
```

Figure 10: ResNet50 with a raw image of 64 by 64 by 3 pre-processed with ImageNet Normalization with GlobalAveargePooling2D to reduce dimensions using CNN architecture.

Evaluation Metrics

Models were assessed using metrics like, precision, recall, Accuracy, F1-score, and confusion matrices, elucidating view of Model performance across the three classes.

Results

Table 2: The classifier (Random Forest, AdaBoost, SVM, FCNN, CNN, ResNet) and their metrics Accuracy, Recall, and F1-Score and Precision,

Classifier	***Accuracy***	***Precision***	***Recall***	***F1-score***
Random Forest	0.6083	0.6125	0.6083	0.6084
AdaBoost	0.5067	0.5177	0.5067	0.5098
SVM	0.6317	0.6384	0.6317	0.6325
KNN	0.5383	0.5451	0.5383	0.5397
CNN	0.7517	0.7502	0.7517	0.7476
CNN + ResNet50	0.8817	0.8829	0.8817	0.8817

Table 3: The Classifier with the metric in percentage

Classifier	***Accuracy***	***Precision***	***Recall***	***F1-score***
Random Forest	61%	61%	61%	61%
AdaBoost	51%	52%	51%	51%
SVM	63%	64%	63%	63%
KNN	54%	55%	53%	54%
CNN	75%	75%	75%	75%
CNN + ResNet50	88%	88%	88%	88%

Confusion matrices for all Groups are shown in the table below shown

Table 4: Confusion Matrices of all classifiers (Models) with the Three classes of Animals used in the Analysis (Cats, Dogs, Snakes, and their Ranges)

Animal Classes	Random Forest	AdaBoost	SVM	KNN	CNN	CNN + ResNet50
Cats	131, 52,17	103,71,26	133,52,15	118,62,20	137,49,14	173,23,4
Dogs	73,97,30	67,97,36	78,100,22	82,84,34	52,118,30	39,157,4
Snakes	30,33,137	34,62,104	27,27,146	32,47,121	6,9,185	2,8,190

The CNN with ResNet50 achieved the highest accuracy (88.17%), significantly outperforming traditional models. SVM (63.17%) and Random Forest (60.83%) showed moderate success, while AdaBoost (50.67%) and KNN (53.83%) lagged. The standalone CNN (75.17%) demonstrated substantial improvement over traditional methods, highlighting the advantage of spatial feature extraction.

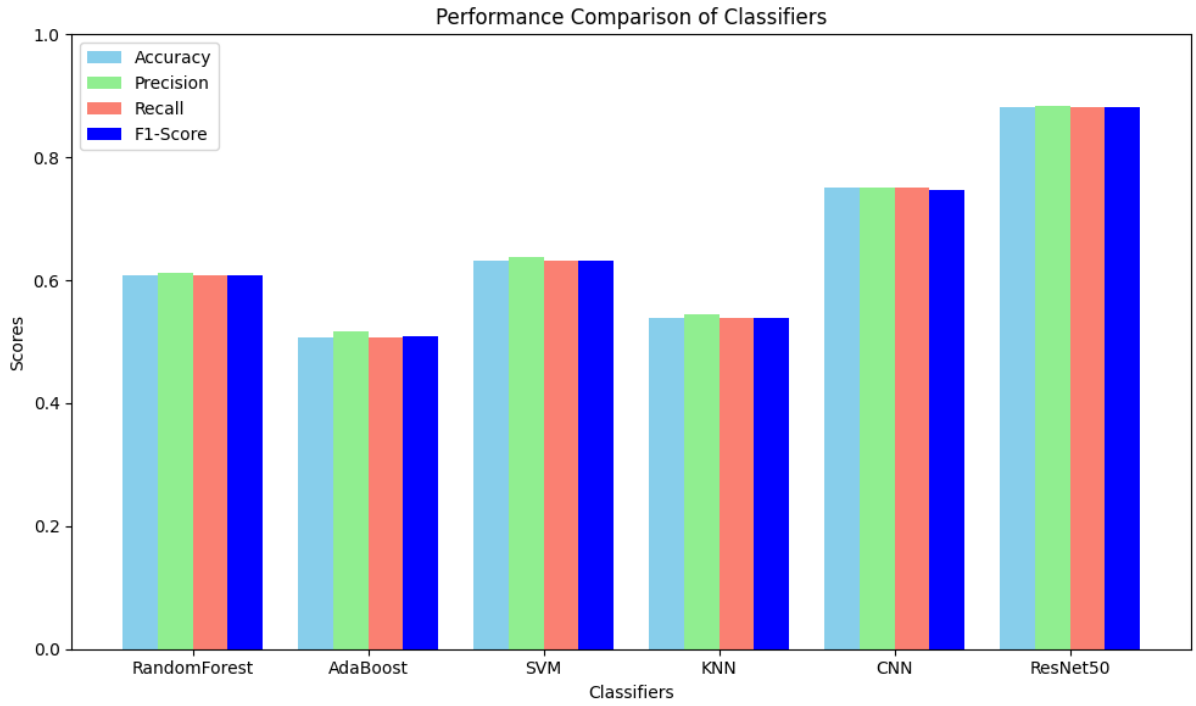


Figure 11: Comparing Metrics with the Classifier, Random Forest, AdaBoost, SVM, KNN, CNN, and ResNet50 on the dataset, showing the best classifier and the metric.

Evaluation and Comparative Analysis

Traditional Models

- **SVM:** Achieved the highest accuracy among traditional models (63.17%), benefiting from GridSearchCV optimization. Its confusion matrix indicates strong performance on snakes (146 correctly classified), likely due to distinct scale patterns, but struggles with cat-dog differentiation (52 cats misclassified as dogs).
- **Random Forest:** Recorded 60.83% accuracy, excelling on snakes (137) and cats (131) but weaker on dogs (97). Loss of spatial context limits its effectiveness (Biau & Scornet, 2016).
- **AdaBoost:** Performed poorly (50.67%), with high misclassification rates across all classes (e.g., 103 cats correct, 71 misclassified as dogs). Noise sensitivity hampers its suitability for image data (Schapire, 2013).
- **KNN:** Attained 53.83% accuracy, performing better than AdaBoost but still limited by high-dimensional flattened inputs. Snakes were classified best (121), reflecting distinct features.

Deep Learning Models

- **CNN:** Achieved 75.17% accuracy, with strong performance on snakes (185) due to convolutional layers capturing spatial patterns (LeCun et al., 1989). Some cat-dog confusion persists (63 misclassifications).
- **CNN + ResNet50:** Outperformed all models at 88.17%, with exceptional results across classes (cats: 173, dogs: 157, snakes: 190). Transfer learning enhances its ability to

handle variability (He et al., 2016). The Pre-trained models help to reduce the misclassification and made it better, shown in Appendix.

The superior performance of deep learning models underscores their capacity to leverage spatial hierarchies, a critical advantage over traditional methods reliant on flattened features. Snakes' distinct textures contribute to high classification rates across models, while cat-dog similarity poses a persistent challenge.

Conclusion

This study harnesses big data analytics to classify animal images, evaluating six models on the Kaggle Animal Image Classification Dataset. The CNN + ResNet50 emerged as the top performer (88% accuracy) shown in table 2, validating deep learning's superiority in visual tasks. Traditional models like SVM (63%) and Random Forest (61%) offer resilience but lack spatial awareness, while AdaBoost (51%) and KNN (54%) underperform due to noise and scalability issues.

These findings contribute to scalable image classification, offering a blueprint for applications in conservation and automation. Future enhancements could push performance further, solidifying big data's role in solving complex visual challenges.

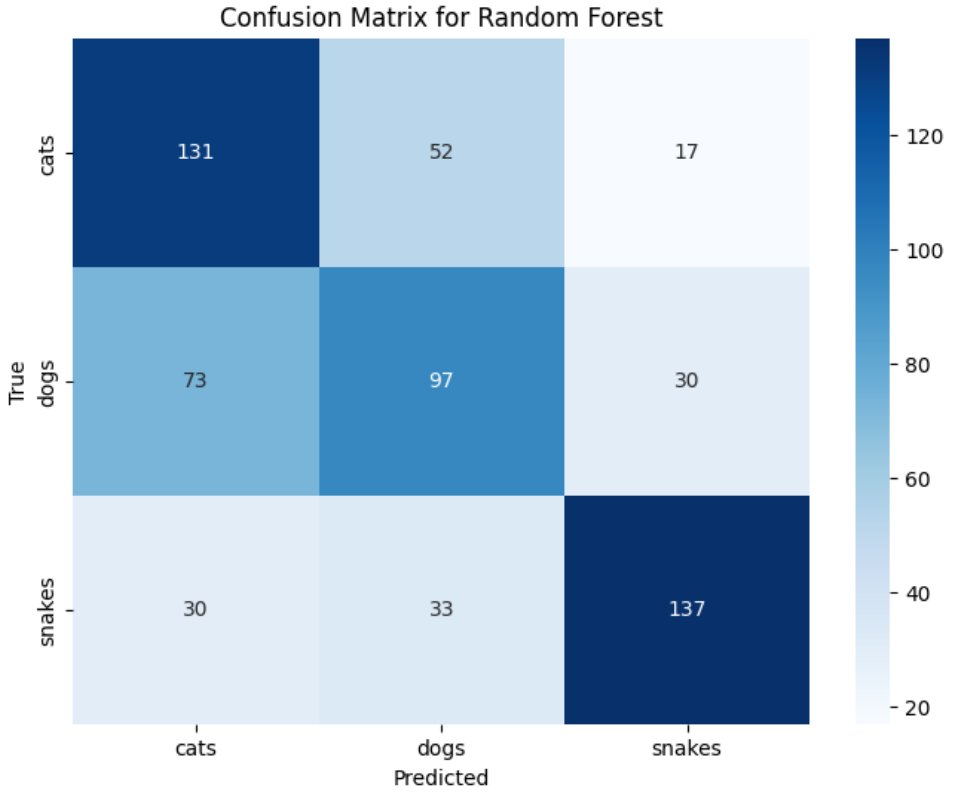
Further Improvement

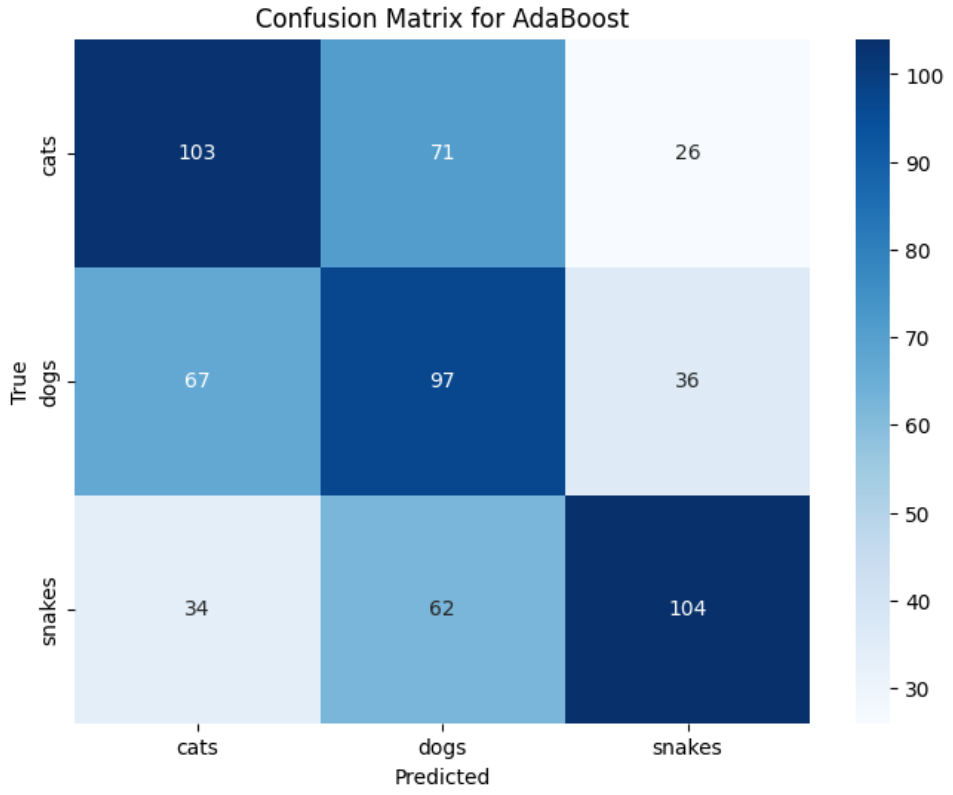
Several strategies can be pursued to elevate the performance of the image classification models and ensure their robustness for real-world applications. First, incorporating data augmentation techniques such as rotation, flipping can enhance model generalization by exposing it to a broader range of image variations, particularly addressing misclassifications due to pose or lighting differences (Shorten & Khoshgoftaar, 2019). For instance, augmenting the dataset could reduce errors like the 52 cats misclassified as dogs by SVM, as noted in the confusion matrix. Second, addressing potential class imbalance—though not present in this balanced dataset (1000 images per class)—can be pre-emptively tackled using techniques like SMOTE (Synthetic Minority Over-sampling Technique) or weighted loss functions, ensuring scalability to less balanced datasets in the future (Chawla et al., 2002). Third, experimenting with advanced architectures like Vision Transformers (ViT) could further improve accuracy, as ViTs have shown superior performance in capturing global image context compared to CNNs, potentially pushing accuracy beyond the current 88% achieved by CNN + ResNet50 (Dosovitskiy et al., 2020). Fourth, training in cluster mode using distributed frameworks like Apache Spark or Dask can enhance scalability, allowing the models to handle larger datasets (e.g., millions of images) efficiently, which is critical for big data applications (Dean et al., 2012). Finally, for practical deployment, optimizing the model for real-time inference on edge devices—such as in wildlife monitoring systems—can be achieved by pruning the CNN + ResNet50 architecture or using quantization techniques, reducing computational overhead while maintaining accuracy (He et al., 2016). These improvements address current limitations and position the models for broader, more impactful applications in automated image classification tasks.

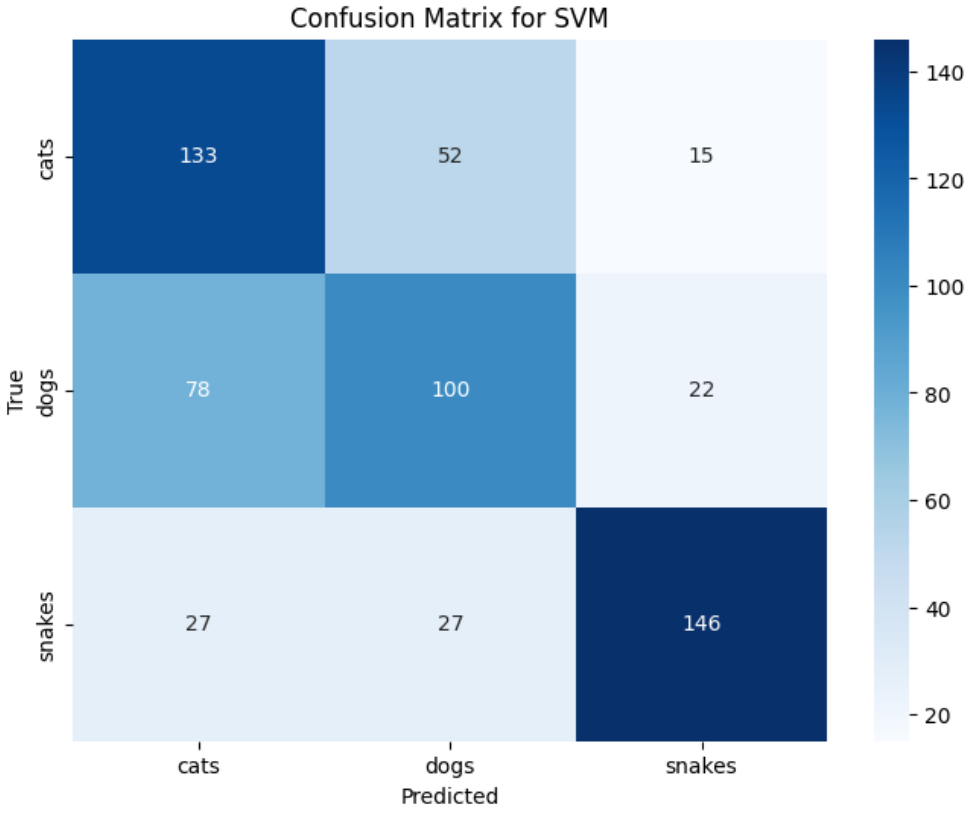
References

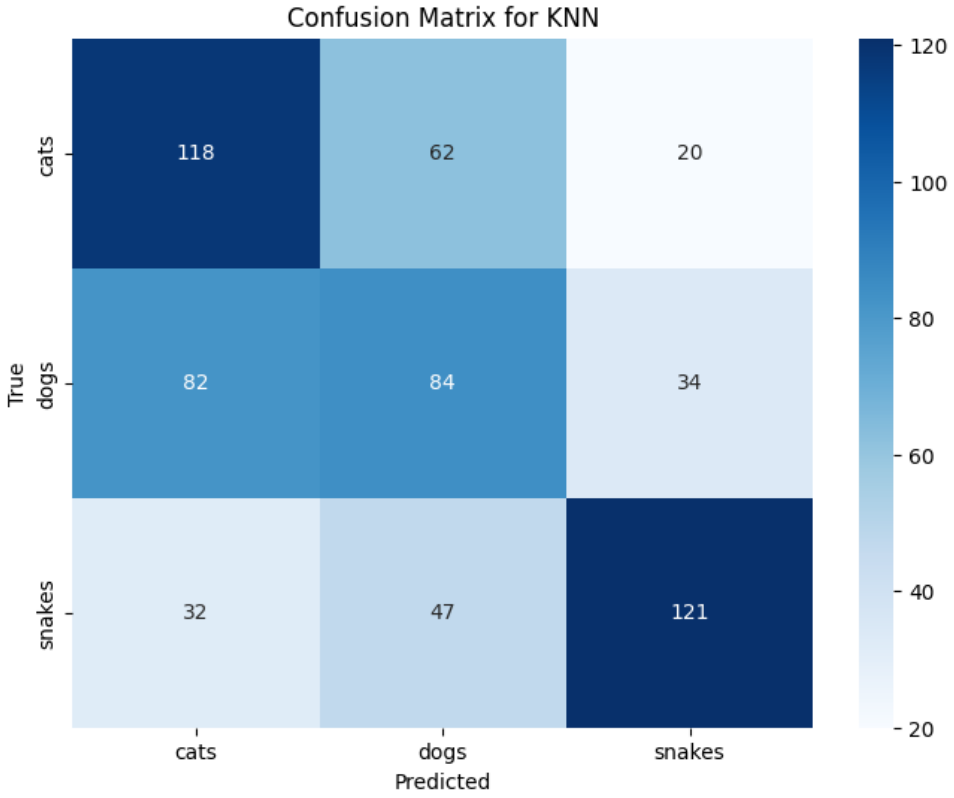
1. Biau, G., & Scornet, E. (2016). A random forest guided tour. *Test, 25*(2), 197-227.
2. Breiman, L. (2001). Random forests. *Machine Learning, 45*(1), 5-32.
3. Dean, J., et al. (2012). Large scale distributed deep networks. *Advances in Neural Information Processing Systems, 25*.
4. Dosovitskiy, A., et al. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.
5. Freund, Y., & Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences, 55*(1), 119-139.
6. He, K., et al. (2016). Deep residual learning for image recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 770-778.
7. LeCun, Y., et al. (1989). Backpropagation applied to handwritten zip code recognition. *Neural Computation, 1*(4), 541-551.
8. Rawat, W., & Wang, Z. (2017). Deep convolutional neural networks for image classification: A comprehensive review. *Neural Computation, 29*(9), 2352-2449.
9. Saxena, A. (2023). Animal Image Classification Dataset. *Kaggle*. Available at: https://www.kaggle.com/datasets/ashishsaxena2209/animal-image-classification-dataset.
10. Shorten, C., & Khoshgoftaar, T. M. (2019). A survey on image data augmentation for deep learning. *Journal of Big Data, 6*(1), 60.
11. Vapnik, V. (1995). *The nature of statistical learning theory*. Springer.

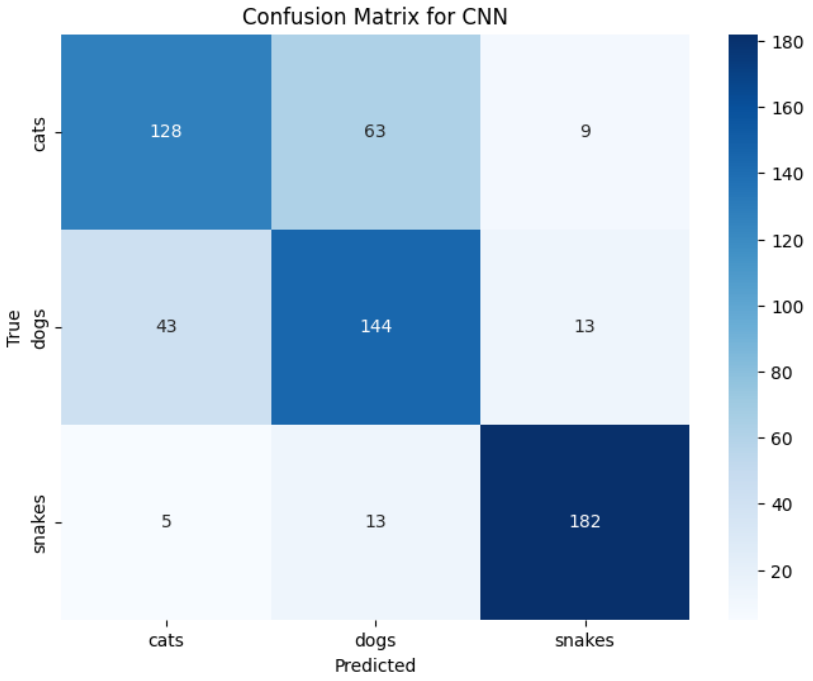
Appendix

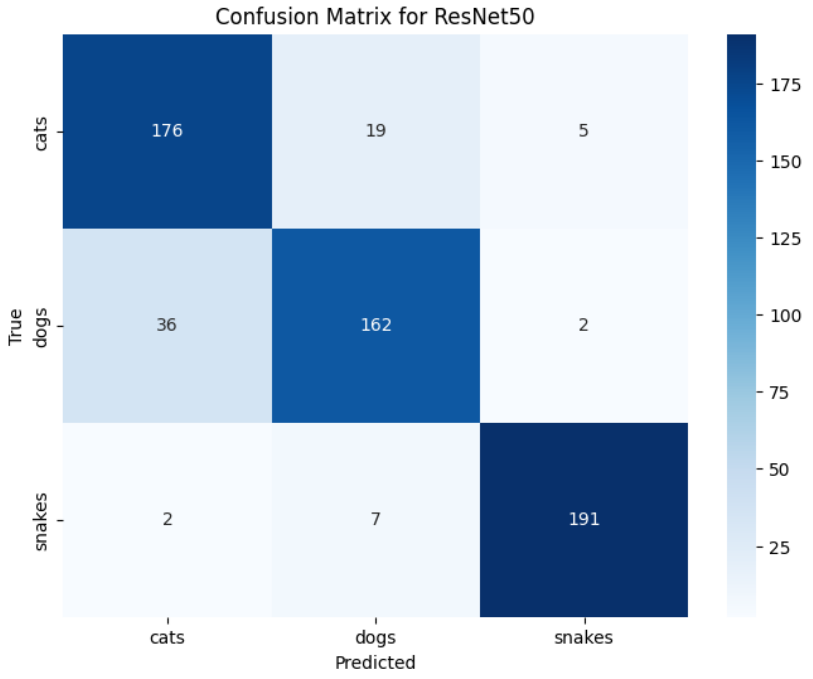

Confusion Matrix for Random Forest
cats
dogs
snakes
True
131
52
17
73
97
30
30
33
137
cats
dogs
snakes
Predicted
120
100
80
60
40
20


Confusion Matrix for AdaBoost
cats
dogs
snakes
True
103
71
26
67
97
36
34
62
104
cats
dogs
snakes
Predicted
100
90
80
70
60
50
40
30


Confusion Matrix for SVM
cats
dogs
snakes
True
133
52
15
78
100
22
27
27
146
cats
dogs
snakes
Predicted
140
120
100
80
60
40
20


Confusion Matrix for KNN
cats
dogs
snakes
True
118
62
20
82
84
34
32
47
121
cats
dogs
snakes
Predicted
120
100
80
60
40
20


Confusion Matrix for CNN
True
cats
dogs
snakes
128
63
9
43
144
13
5
13
182
cats
dogs
snakes
Predicted
180
160
140
120
100
80
60
40
20


Confusion Matrix for ResNet50
True
cats
dogs
snakes
176
19
5
36
162
2
2
7
191
cats
dogs
snakes
Predicted
175
150
125
100
75
50
25

List of Figure

List of Table