Chihuahua or Muffins?

1 Problem Statement

In the realm of computer vision and machine learning, the ability to distinguish between visually similar objects poses a fascinating yet challenging problem. The "Chihuahua or Muffin" meme, while unquestionably entertaining, serves as a vivid and engaging illustration of the intricate challenges that underlie computer vision and object recognition. This meme first gained notoriety when Karen Zack posted a collage on Twitter (Figure 1), juxtaposing images of Chihuahuas alongside those of blueberry muffins.

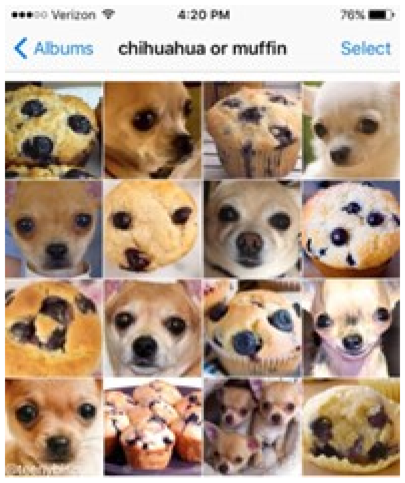


Figure 1: Chihuahua or muffins meme

Deep learning algorithms, with their convolutional neural networks (CNNs) and intricate architectures, have become synonymous with cutting-edge image classification. Nevertheless, this project confronts the prevailing notion by investigating whether traditional machine learning methods, often considered less sophisticated, can hold their ground in discerning between visually ambiguous categories. The inherent challenge lies in ascertaining if classic classifiers can leverage feature engineering and robust training to achieve comparable accuracy in a task typically dominated by deep learning.

This project focuses on the classification of images depicting Chihuahuas, the diminutive canine breed, and muffins, delectable baked goods. Despite belonging to vastly different domains, Chihuahuas and muffins share certain visual characteristics that can confound traditional classifiers. The challenge lies in training a model to discern subtle differences between them and enabling it to accurately categorize these seemingly unrelated entities.

2 Methodology

The project comprises of four key phases to effectively address the challenge of classifying visually similar entities - Chihuahuas and muffins.

- Data Preprocessing and Train-Test Split:

  The initial phase involves standardizing the images to a uniform dimension to ensure consistency in feature representation. Subsequently, the dataset will be partitioned into training and testing sets.

- Dimensionality Reduction and Model Evaluation:

  Following data preparation, both linear and non-linear dimensionality reduction techniques will be applied. The reduced components obtained from these techniques will serve as input features for various classifier models. The performance of these models will be systematically evaluated to discern the most effective configuration.

- Hyperparameter Tuning:

  Upon identifying promising models, the third phase involves fine-tuning their hyperparameters to further improve the models' accuracy.

- Final Model Selection:

  The final evaluation on the test set aims to determine the model that not only excels in the training phase but also generalizes well to new, unseen instances, ensuring robust performance beyond the training data.

3 Data

3.1 Data Source

The dataset, sourced from Kaggle[1] comes pre-divided into a training set and a test set, streamlining the initial stages of model development. Within the training set, there are 2559 images of Chihuahuas and 2174 images of muffins. Similarly, the test set comprises of 640 Chihuahua images and 544 muffin images.

Although the numbers may not be identical, the relative balance in class sizes helps to mitigate biases, promoting fair and effective model training across both categories.

3.2 Data Preprocessing

To ensure uniformity in the dataset, images of varying sizes were standardized by converting them to grayscale and resizing to 224 by 224 pixels. Subsequently, each image was flattened into a row vector containing 50176 elements and standardized by dividing each pixel value by 255. The following samples from the training (Figure 2) and testing sets (Figure 3) showcase the transformed images:

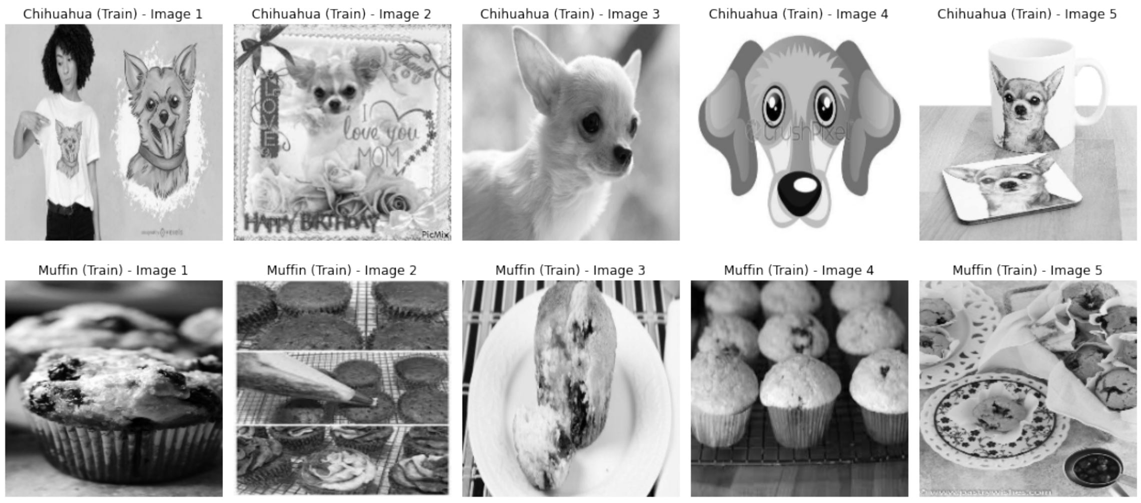

Chihuahua (Train) - Image 1
Chihuahua (Train) - Image 2
Chihuahua (Train) - Image 3
Chihuahua (Train) - Image 4
Chihuahua (Train) - Image 5
Muffin (Train) - Image 1
Muffin (Train) - Image 2
Muffin (Train) - Image 3
Muffin (Train) - Image 4
Muffin (Train) - Image 5

Figure 2: Sample training set images

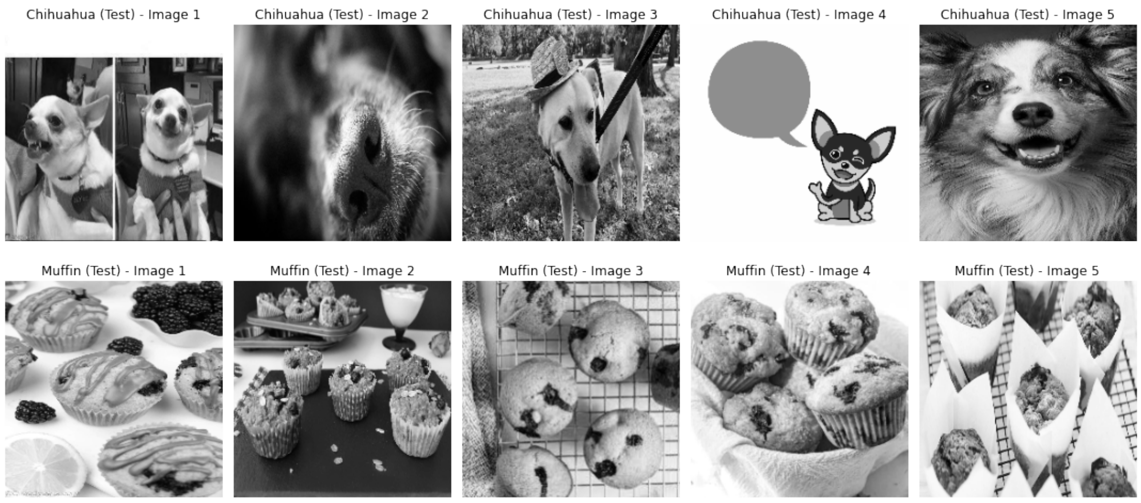

Chihuahua (Test) - Image 1
Chihuahua (Test) - Image 2
Chihuahua (Test) - Image 3
Chihuahua (Test) - Image 4
Chihuahua (Test) - Image 5
Muffin (Test) - Image 1
Muffin (Test) - Image 2
Muffin (Test) - Image 3
Muffin (Test) - Image 4
Muffin (Test) - Image 5

Figure 3: Sample test set images

The dataset exhibits diversity, encompassing not just conventional Chihuahua images but also computer-generated ones. While this diversity can aid model generalization, a potential challenge arises if there is an imbalance between real and cartoon Chihuahua images in the training set. Such an imbalance could confound the model during training, emphasizing the importance of a balanced representation for optimal performance.

4 Results

4.1 Dimensionality Reduction

To streamline model training, two dimensionality reduction techniques, Principal Component Analysis (PCA) and isomap, were investigated. Figure 4 displays plots of the first two components.

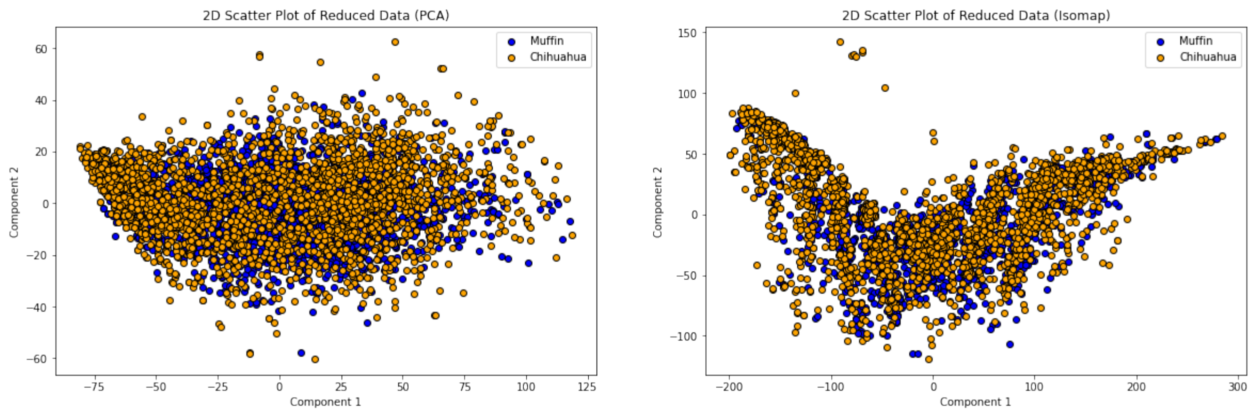


Figure 4: Left: Scatter plot of the first two components of PCA. Right: Scatter plot of the first two components of isomap

However, these components, for both PCA and isomap, prove inadequate for effective class separation. It's evident that higher dimensions are essential for facilitating a meaningful distinction between classes.

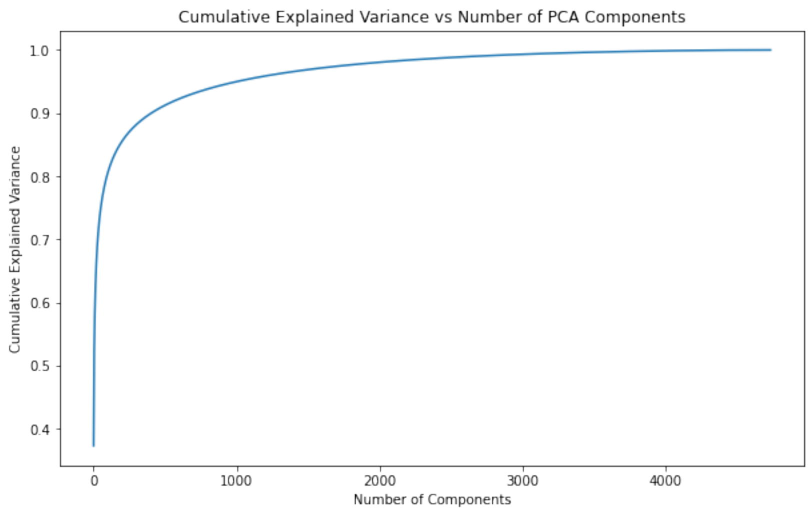


Figure 5: Left: Cumulative explained variance with number of PCA components

Number of PCA Components	Cummulative Explained Variance
2	0.42
5	0.52
51	0.75
405	0.90
1003	0.95
2659	0.99

Table 1: Number of PCA components and the cummulative explained variance

From Figure 5, it is evident that the cumulative explained variance remains notably low when using a lower number of components. Table 1 reinforces this observation, showing that the first 1003 PCA components are required to attain a cumulative explained variance of 95%. This highlights the imperative of adopting a higher-dimensional strategy, underscoring its significance in capturing a more thorough representation of the dataset.

Two widely employed dimensionality reduction techniques in convolutional neural networks are max pooling and average pooling. These methods involve using an n by n sized kernel to identify the maximum or average value of the pixels within the kernel. Figure 6 illustrates a basic example using a 2 by 2 kernel for both max and average pooling.

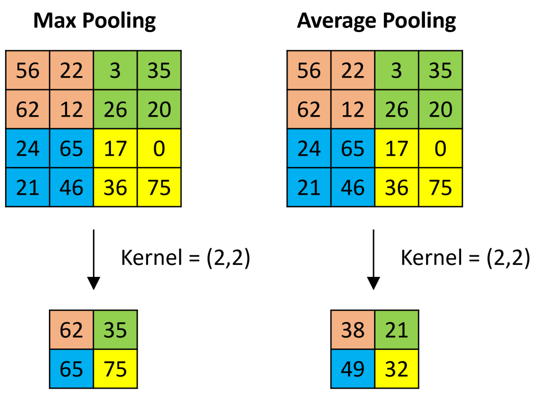
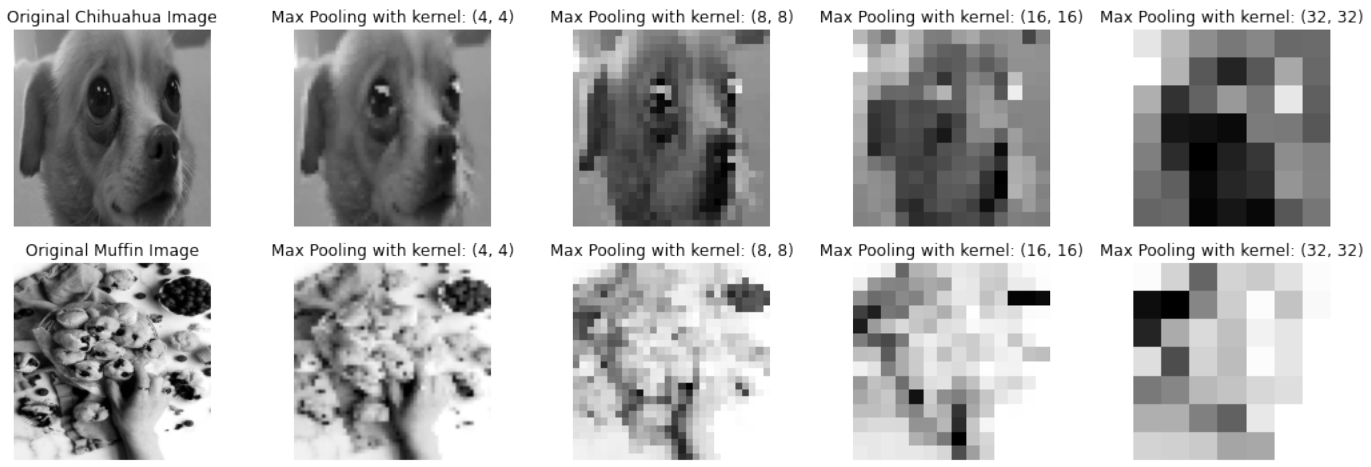


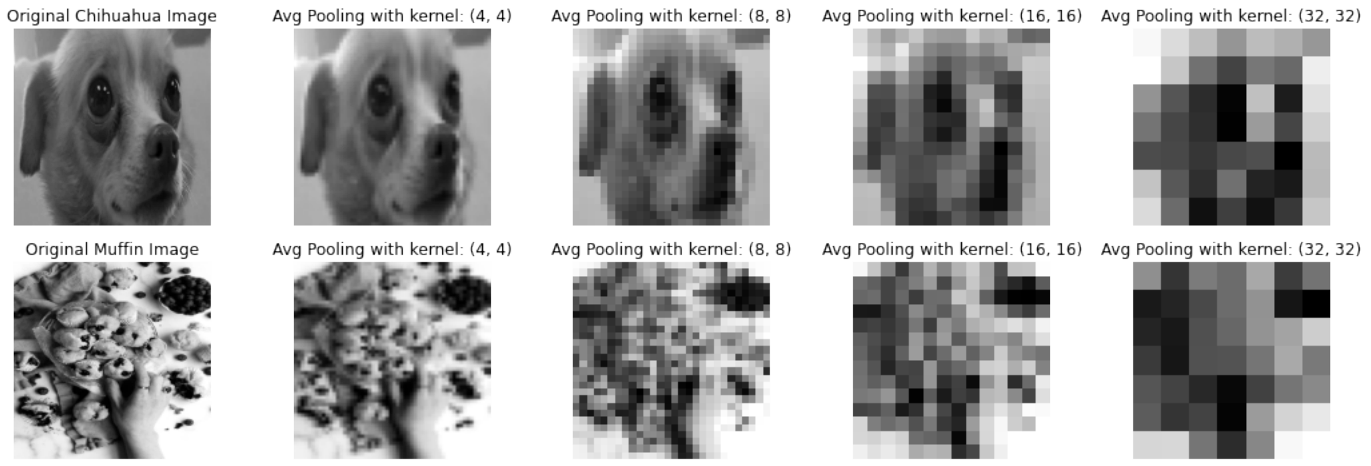
Figure 6: Max and average pooling

This technique is subsequently applied to the training dataset, with Figure 7 and Figure 8 showcasing the resultant example images using max and average pooling using different kernels, respectively.



Original Chihuahua Image
Max Pooling with kernel: (4, 4)
Max Pooling with kernel: (8, 8)
Max Pooling with kernel: (16, 16)
Max Pooling with kernel: (32, 32)
Original Muffin Image
Max Pooling with kernel: (4, 4)
Max Pooling with kernel: (8, 8)
Max Pooling with kernel: (16, 16)
Max Pooling with kernel: (32, 32)

Figure 7: Max pooling with different kernels



Original Chihuahua Image
Avg Pooling with kernel: (4, 4)
Avg Pooling with kernel: (8, 8)
Avg Pooling with kernel: (16, 16)
Avg Pooling with kernel: (32, 32)
Original Muffin Image
Avg Pooling with kernel: (4, 4)
Avg Pooling with kernel: (8, 8)
Avg Pooling with kernel: (16, 16)
Avg Pooling with kernel: (32, 32)

Figure 8: Average pooling with different kernels

Comparing the two image sets, it is apparent that the images resulting from average pooling bear a slightly closer resemblance to the original image compared to those from max pooling. This suggests that average pooling retains more features of the image than max pooling, and this nuance might influence the subsequent model fitting process.

4.2 Model Evaluation on Reduced Data

Leveraging the cumulative explained variance results from PCA, various vanilla classifiers were fitted using the lower-dimensional representations of the training data for model training. Employing 10-fold cross-validation, the mean accuracy was tabulated in Table 2.

Classifier	**Mean Accuracy Using N PCA Components**					
	$N = 2$	$N = 5$	$N = 51$	$N = 405$	$N = 1003$	$N = 2659$
Bayes Classifier	0.637	0.659	0.666	0.657	0.664	0.698
k-Nearest Neighbors	0.622	0.673	0.664	0.584	0.567	0.559
Logistic Regression	0.558	0.582	0.608	0.607	0.594	0.590
Support Vector Machine (RBF)	0.658	0.702	0.753	0.762	0.765	0.772
Neural Network	0.650	0.690	0.744	0.729	0.701	0.687
Decision Tree	0.585	0.644	0.640	0.633	0.650	0.629
AdaBoost	0.645	0.660	0.680	0.695	0.698	0.703
Naive Bayes	0.637	0.659	0.666	0.657	0.664	0.698
Quadratic Discriminant Analysis	0.637	0.687	0.747	0.776	0.728	0.580

Table 2: Mean accuracy of 10-fold cross validation using various numbers of PCA components

Linear Support Vector Machine was intentionally omitted due to its long evaluation time. Training exceeded 2.5 hours when using the first 1003 components. Generally, an increase in the number of components correlated with a rise in mean accuracy for most classifiers. Notably, Quadratic Discriminant Analysis (QDA) demonstrated an exception, achieving peak performance with 405 components with a mean accuracy of 78% but significantly under-performing with 2659 components with a mean accuracy of 58%, suggesting a potential issue of over-fitting.

A parallel assessment was conducted with varying numbers of isomap components, with results presented in Table 3. Consistent trend emerged as the number of components increased, although the mean accuracy of classifiers generally lagged behind those utilizing PCA components.

Classifier	**Time Taken (s) Using N Isomap Components**				
	$N = 2$	$N = 10$	$N = 100$	$N = 1000$	$N = 2000$
Bayes Classifier	0.625	0.644	0.576	0.604	0.637
k-Nearest Neighbors	0.658	0.684	0.685	0.667	0.646
Logistic Regression	0.613	0.637	0.700	0.663	0.602
Support Vector Machine (RBF)	0.646	0.673	0.720	0.731	0.735
Neural Network	0.629	0.667	0.701	0.669	0.675
Decision Tree	0.615	0.645	0.640	0.619	0.613
AdaBoost	0.645	0.664	0.686	0.673	0.661
Naive Bayes	0.625	0.644	0.576	0.604	0.637
Quadratic Discriminant Analysis	0.611	0.627	0.684	0.704	0.541

Table 3: Mean accuracy of 10-fold cross validation using various numbers of Isomap components

The results using max and average pooling are shown in Table 4 and Table 5, respectively.

Classifier	**Mean Accuracy of Max Pooling**			
	(32,32)	**(16,16)**	**(8,8)**	**(4,4)**
Bayes Classifier	0.586	0.582	0.573	0.650
k-Nearest Neighbors	0.691	0.625	0.577	0.558
Logistic Regression	0.555	0.566	0.568	0.586
Support Vector Machine (RBF)	0.741	0.750	0.758	0.766
Neural Network	0.685	0.694	0.691	0.637
Decision Tree	0.680	0.666	0.671	0.662
AdaBoost	0.726	0.710	0.689	0.685
Naive Bayes	0.586	0.582	0.573	0.650
Quadratic Discriminant Analysis	0.606	0.739	0.734	0.537

Table 4: Mean accuracy of 10-fold cross validation using max pooling with different kernel sizes

Classifier	Mean Accuracy of Average Pooling			
	(32,32)	**(16,16)**	**(8,8)**	**(4,4)**
Bayes Classifier	0.622	0.610	0.595	0.585
k-Nearest Neighbors	0.667	0.620	0.592	0.569
Logistic Regression	0.611	0.615	0.605	0.593
Support Vector Machine (RBF)	0.741	0.756	0.760	0.766
Neural Network	0.745	0.740	0.729	0.683
Decision Tree	0.655	0.655	0.678	0.667
AdaBoost	0.676	0.665	0.675	0.663
Naive Bayes	0.622	0.610	0.595	0.585
Quadratic Discriminant Analysis	0.749	0.766	0.745	0.554

Table 5: Mean accuracy of 10-fold cross validation using average pooling with different kernel sizes

In evaluating all four dimensionality reduction techniques, it becomes evident that average pooling outperforms the others significantly. Comparing the pooling methods with PCA and Isomap, pooling methods stand out in retaining a substantial portion of the original image features, as observed in the transformed images depicted in Figures 7 and 8.

Notably, certain classifiers, such as the Support Vector Machine (SVM) with a Radial Basis Function (RBF) kernel, exhibit commendable performance despite larger kernel sizes. For instance, the evaluation time for SVM with a kernel size of (32,32) was 5.49 seconds, whereas the (4,4) kernel took 178.41 seconds for average pooling, with a marginal 2% difference in accuracy. A comprehensive compilation of the time taken for model evaluation is provided in the Annex. Similar observation can also be seen for Quadratic Discriminant Analysis (QDA) although it performed much worst with a smaller kernel size likely due to over-fitting.

Considering both accuracy scores and model evaluation times, SVM and QDA employing average pooling with a kernel size of (8,8) were chosen to undergo hyperparameter tuning.

4.3 Hyperparameter Tuning

Following hyperparameter tuning, the accuracy of both Support Vector Machine (SVM) and Quadratic Discriminant Analysis (QDA) models did not exhibit significant improvement. The vanilla models already demonstrated commendable performance initially.

Hyperparameters	Accuracy
$C = 10$, gamma = 0.01	0.768
$C = 1000$, gamma = 0.01	0.745
$C = 100$, gamma = 0.01	0745
$C = 1$, gamma = 0.01	0.744
$C = 1000$, gamma = 0.1	0.740

Table 6: Top 5 mean accuracy obtained from 10-fold cross validation using SVM and average pooling with different hyperparameters

Hyperparameters	Accuracy
reg_param = 0	0.745
reg_param = 0.8	0.644
reg_param = 0.9	0.635
reg_param = 0.7	0.634
reg_param = 0.6	0.622

Table 7: Top 5 mean accuracy obtained from 10-fold cross validation using QDA with different hyperparameters

4.4 Final Models Evaluation

Subsequently, both models underwent testing on the test set, yielding accuracy rates of 77.2% for SVM and 76.1% for QDA. The consistency between training and testing results suggests there was no issue of overfitting during the training process.

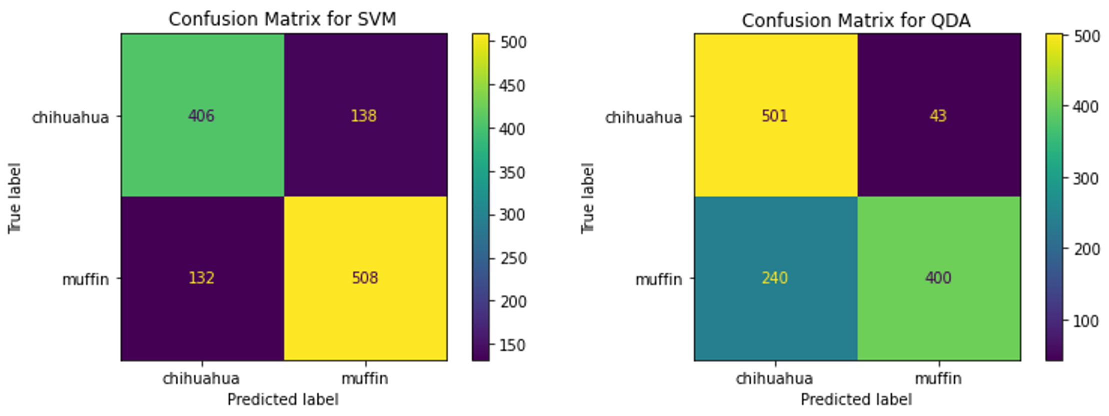
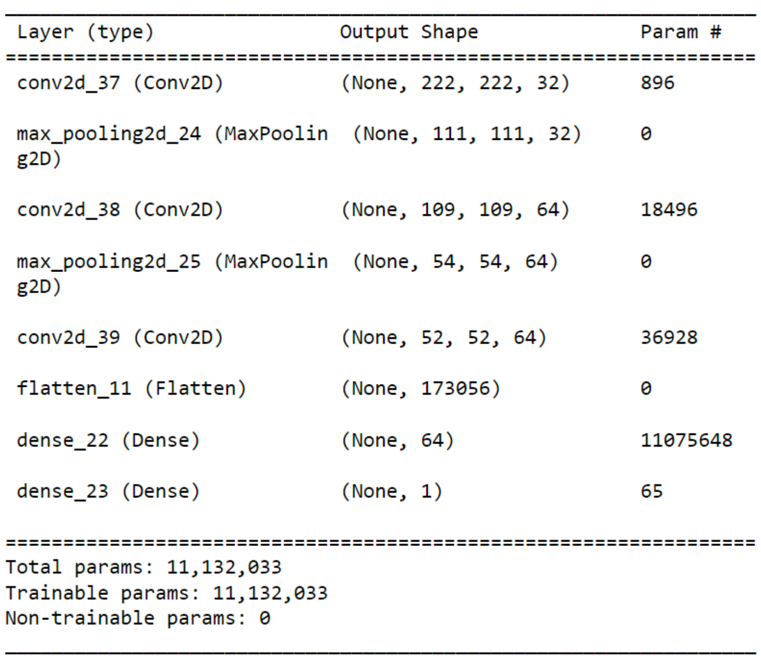


Figure 9: Left: Confusion matrix for SVM model. Right: Confusion matrix for QDA model

Figure 9's confusion matrices provided insights into the strengths and weaknesses inherent in each model. Notably, the SVM model demonstrated a balanced performance by misclassifying both Chihuahua and muffin images with a comparable frequency of around 130 instances each. In contrast, the QDA model exhibited a more nuanced pattern, misclassifying only 43 Chihuahua images but significantly more muffin images (240 images).

In a final comparison, a Convolutional Neural Network (CNN) was trained and tested:


```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_37 (Conv2D)          (None, 222, 222, 32)      896

 max_pooling2d_24 (MaxPoolin  (None, 111, 111, 32)     0
 g2D)

 conv2d_38 (Conv2D)          (None, 109, 109, 64)      18496

 max_pooling2d_25 (MaxPoolin  (None, 54, 54, 64)       0
 g2D)

 conv2d_39 (Conv2D)          (None, 52, 52, 64)        36928

 flatten_11 (Flatten)        (None, 173056)            0

 dense_22 (Dense)            (None, 64)                11075648

 dense_23 (Dense)            (None, 1)                 65

=================================================================
Total params: 11,132,033
Trainable params: 11,132,033
Non-trainable params: 0
_________________________________________________________________
```

Figure 10: CNN model architecture

Despite its simple architecture (Figure 10) with merely 10 training epochs, the CNN model demonstrated a commendable test accuracy of 86%. This comparison underscores the prowess of deep learning, particularly in handling intricate data such as images, showcasing its superiority over conventional machine learning algorithms.

5 Conclusion

In summary, this Chihuahua vs. Muffin classification project subjected conventional machine learning algorithms to rigorous testing, resulting in commendable performance levels of approximately 70%. The exploration of various dimensionality reduction techniques revealed that pooling methods slightly outperformed PCA and Isomap. Despite the final models achieving an accuracy of around 77%, a simple Convolutional Neural Network

(CNN) exceeded these results with a remarkable 10% increase in accuracy. This outcome underscores the prowess of deep learning, particularly in the realm of image classification.

6 References

[1] Kaggle 2022, accessed 5 November 2023, ⟨https://www.kaggle.com/datasets/samuelcortinhas/muffin-vs-chihuahua-image-classification/data⟩.

7 Annex

Classifier	**Time Taken (s) Using N PCA Components**					
	$N = 2$	$N = 5$	$N = 51$	$N = 405$	$N = 1003$	$N = 2659$
Bayes Classifier	0.02	0.03	0.05	0.28	0.67	1.71
k-Nearest Neighbors	0.15	0.20	0.20	0.34	0.59	1.19
Logistic Regression	0.06	0.13	0.29	1.58	10.67	97.44
Support Vector Machine (RBF)	6.21	5.16	7.58	29.97	68.93	176.43
Neural Network	7.71	20.40	94.56	42.73	50.46	122.76
Decision Tree	0.13	0.25	2.35	19.91	52.67	162.38
AdaBoost	1.48	2.07	12.19	95.76	236.10	629.81
Naive Bayes	0.02	0.02	0.05	0.28	0.67	1.75
Quadratic Discriminant Analysis	0.02	0.02	0.31	6.00	26.75	119.88

Table 8: Time taken in seconds for 10-fold cross validation using various numbers of PCA components

Classifier	**Time Taken (s) Using N Isomap Components**				
	$N = 2$	$N = 10$	$N = 100$	$N = 1000$	$N = 2000$
Bayes Classifier	0.02	0.02	0.07	0.65	1.35
k-Nearest Neighbors	0.11	0.18	0.19	0.58	1.20
Logistic Regression	0.06	0.15	0.41	5.84	117.46
Support Vector Machine (RBF)	5.12	4.87	6.75	54.30	110.28
Neural Network	4.25	16.12	36.98	49.49	105.08
Decision Tree	0.11	0.40	3.55	40.42	91.83
AdaBoost	1.16	2.65	20.40	205.01	418.89
Naive Bayes	0.02	0.01	0.06	0.56	1.29
Quadratic Discriminant Analysis	0.03	0.02	0.91	24.92	77.37

Table 9: Time taken in seconds for 10-fold cross validation using various numbers of Isomap components

Classifier	**Time Taken for Max Pooling (s)**			
	(32,32)	**(16,16)**	**(8,8)**	**(4,4)**
Bayes Classifier	0.03	0.11	0.48	1.80
k-Nearest Neighbors	0.29	0.25	0.48	1.36
Logistic Regression	0.54	1.51	12.13	123.79
Support Vector Machine (RBF)	5.70	8.44	43.79	179.28
Neural Network	58.78	102.26	186.96	714.10
Decision Tree	1.04	4.72	22.33	96.03
AdaBoost	4.82	19.77	86.02	360.43
Naive Bayes	0.03	0.11	0.45	1.81
Quadratic Discriminant Analysis	0.24	1.82	14.50	128.77

Table 10: Time taken in seconds for 10-fold cross validation using max pooling with different kernel sizes

Classifier	**Time Taken for Average Pooling (s)**			
	(32,32)	**(16,16)**	**(8,8)**	**(4,4)**
Bayes Classifier	0.04	0.11	0.47	1.83
k-Nearest Neighbors	0.20	0.25	0.52	1.31
Logistic Regression	0.52	1.43	12.18	115.56
Support Vector Machine (RBF)	5.49	8.10	43.35	178.41
Neural Network	114.53	128.64	234.01	857.65
Decision Tree	1.88	7.56	29.87	126.20
AdaBoost	10.08	38.86	151.26	584.44
Naive Bayes	0.03	0.11	0.47	2.03
Quadratic Discriminant Analysis	0.23	1.82	14.40	133.37

Table 11: Time taken in seconds for 10-fold cross validation using average pooling with different kernel sizes