Animal Classification

Sayyed Jilani, Mike Christenson, and Richard Shim

Computer Vision Project

Abstract

In this project, our group classified different animals from the Animal-10 dataset, a dataset from Kaggle that has compiled Google search images of ten distinct animals. To achieve our objective, we extracted diverse feature vectors and concatenated them together to form a comprehensive representation. These feature vectors encompassed color histograms for each channel, including channel means, HSV histograms with means, histograms of oriented gradients (HOG), Local Binary Patterns (LBP), Dense SIFT, Bag of Visual Words (BoVW), and embeddings from pre-trained neural network models such as ResNet101 and EfficientNetB0. Following feature extractions, we applied principal component analysis (PCA) and t-Distributed Stochastic Neighbor Embedding (t-SNE) to reduce dimensionality, enhancing performance and minimizing data loss. Lastly, we trained classification models, such as Logistic Regression, Support Vector Machine, and Multi-Layer Perceptron, on the training subset of our dataset and then further utilized our validation dataset for hyperparameter tuning and test dataset to evaluate the accuracies and efficiencies of our classification models.

1. Introduction

This project utilizes the Animal-10 dataset obtained from Kaggle. This dataset contains roughly 28,000 animal images representing ten classes, each obtained through a Google image search. The images were then verified by a human for label accuracy. Nevertheless, even after the validation process, our group identified instances of misclassifications. These errors include inaccurate pairings of animal images with labels, multiple categories of animals appearing in a single image, and the subject of the image belonging to none of the ten animal classes. As part of the data cleaning process, we systematically removed these identified misclassified images from the dataset. Moreover, the images were of varying resolution, subject framing, and dimensions. For normalization, all images were resized to 256 by 256 pixels. Figure 1 below displays an example image from each class. Before color feature extraction we also normalized each of the color channels to create a consistent frame of comparison across images.

Figure 1: Animal classes from left to right - Dog, Cat, Horse, Sheep, Chicken, Cow, Squirrel. Elephant, Spider, Butterfly. All images have been resized to equal dimensions for this figure.

The distribution of the different classes was considerably varied. To normalize the entire dataset, we opted to sample each class to match the number of images in the smallest class. This meant we ended up working with 1,431 images per class to match the size of the elephant class, the smallest class. The distribution before and after normalization can be seen in Figure 2 below.

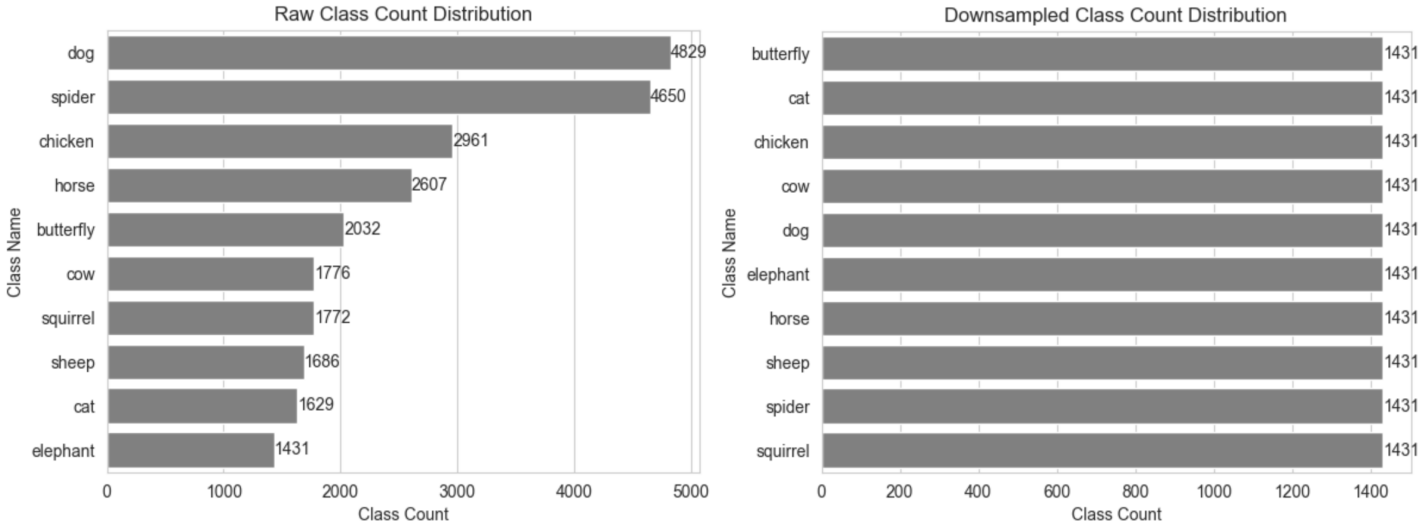


Figure 2: Visualization of image distribution before and after downsampling to equalize class sizes

2. Feature Extraction

In this section, we apply a range of techniques, outlining our motivation, to extract distinctive features from our dataset.

2.1. Color Statistics

The images in our dataset feature various animals, each distinguished by unique attributes that contribute to their differentiation. One prominent factor is the variance in color and color-related patterns among animals. For example, elephants typically exhibit a uniform gray color, while a significant number of cows feature distinct dark spots. To exploit this feature, we analyzed the distribution of pixel intensities within each of the Red, Green, and Blue (RGB) color channels of each image. Additionally, insights into differentiating between animals were drawn from the average color pixel values in each channel across the images. Nevertheless, it's important to highlight that even within each class, there is considerable variation in color-related features and contributions from the background or non-label image subjects can contribute to the color profile. Therefore, extracting colors may not be entirely foolproof. The same rationale should apply to Hue, Saturation, and Value (HSV), in the context of our animal images. Similarly, insights into differentiating between animals were drawn from the distribution of each HSV component, along with the mean HSV values of the image. Overall, the mean HSV saturation offered the best visual distinction between classes in the mean value histogram.

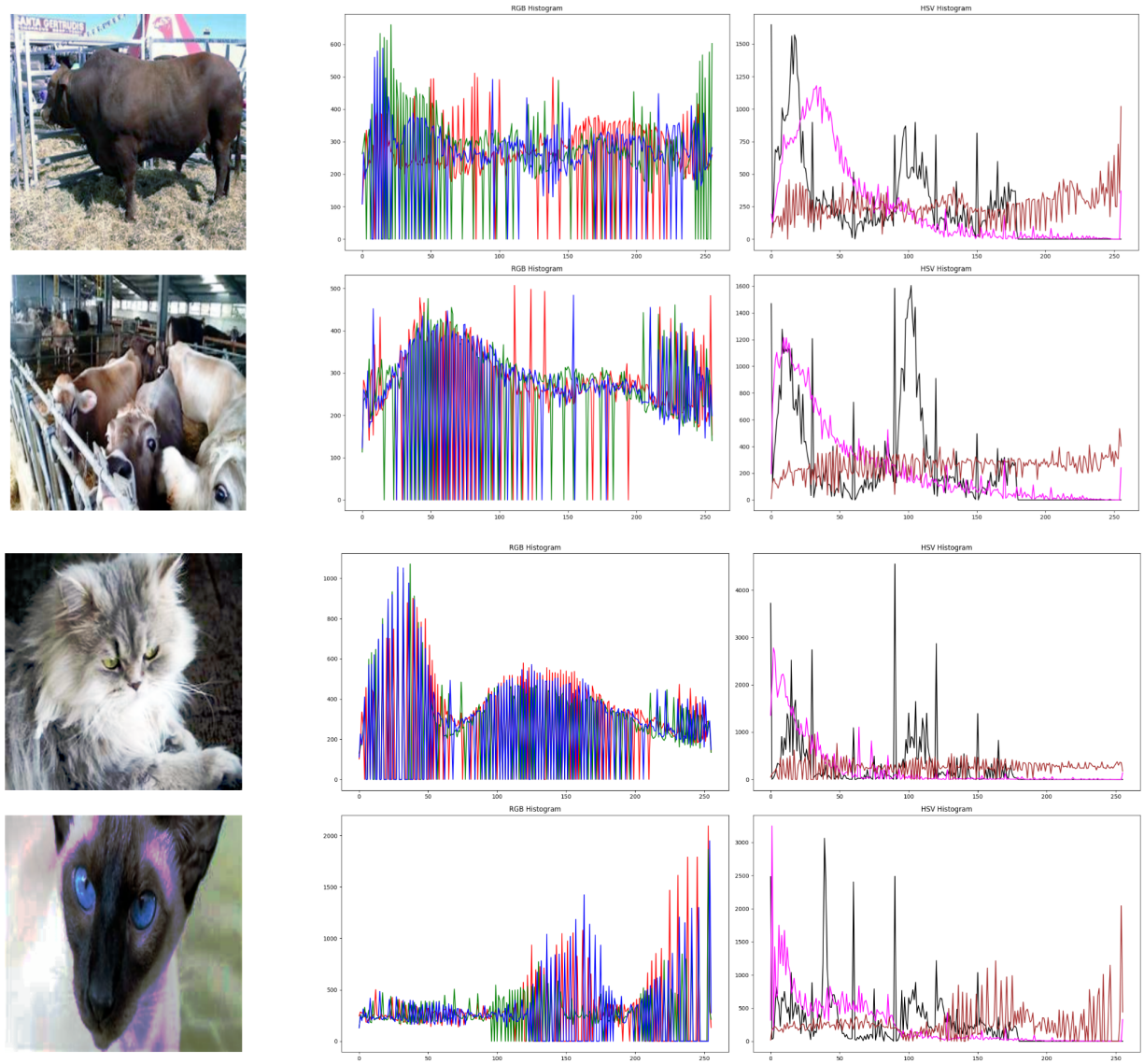


Figure 3: Examples of RGB histograms (middle) and HSV histograms (right) for two cow images and two cat images. The distribution of these histograms is distinct both within a class and among the different classes. This issue strictly appears due to the massive variation in framing, and scaling within our dataset and could put into question the viability of this feature.

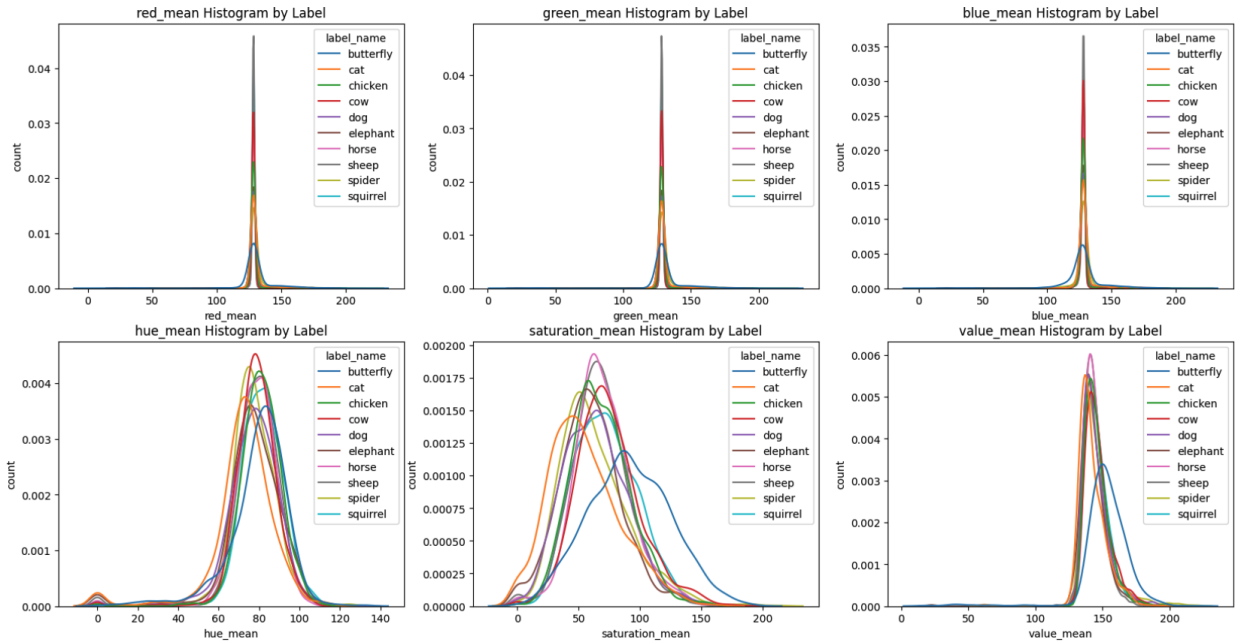
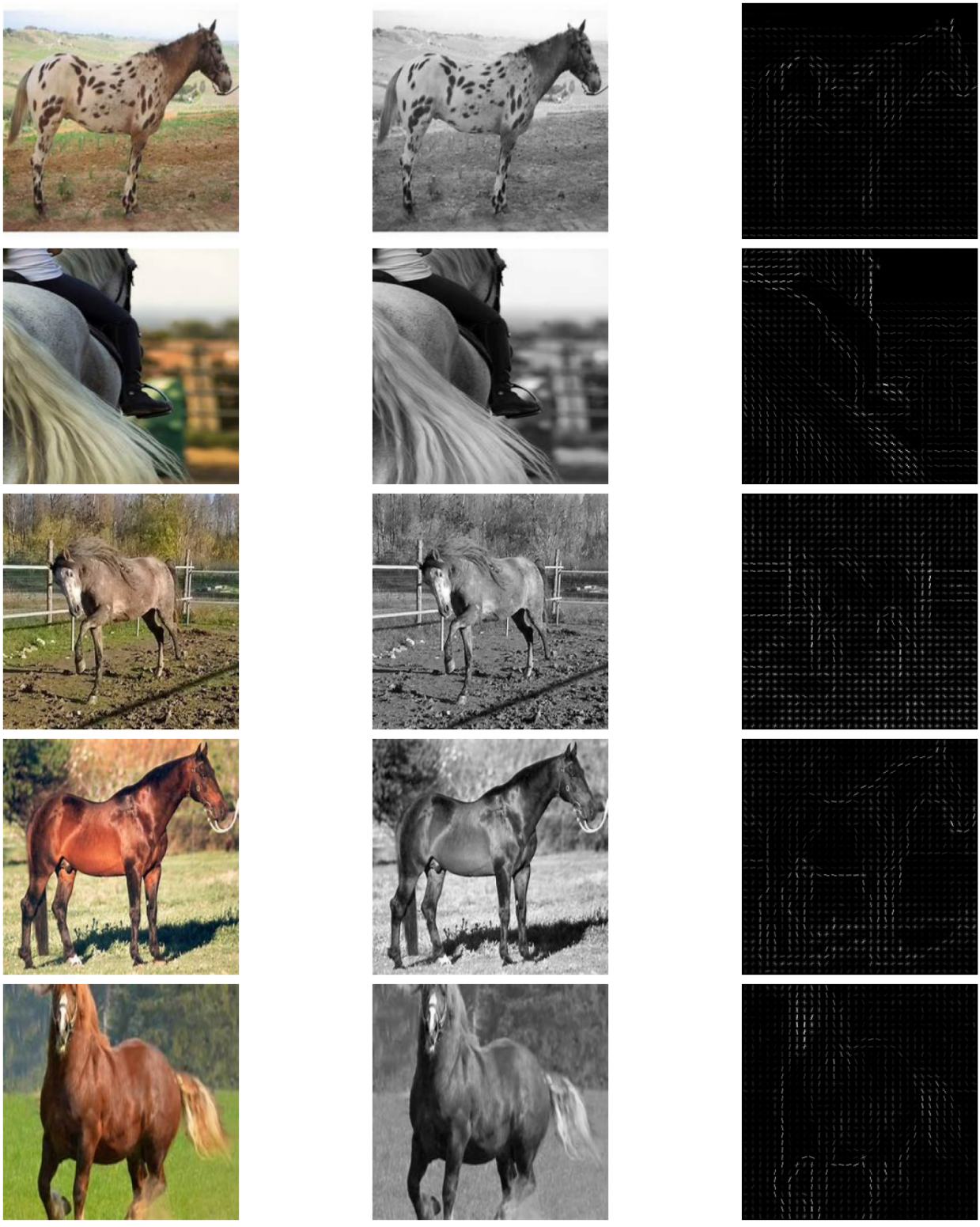


Figure 4: Distribution of mean values in each color channel (first row) and HSV values (second row). The mean distribution for all classes appears to be not quite distinct enough for almost all of the cases, with the saturation values offering the greatest variability in distribution among the different classes.

Based on the distribution of pixel intensities in the different RGB color channels and HSV histograms in Figure 3 of two distinct cow images, it is evident that the distribution can be quite varied even within the same class. Furthermore, based on the mean distribution in Figure 4**,** the mean values across all 10 classes can be quite similar.

2.2. Histogram of Oriented Gradients (HOG)

Another valuable feature that can be harnessed to distinguish between various animals is the inherent structural and textural dissimilarities present in their images. Histogram of Oriented Gradients (HOG) proves to be a potent tool for this purpose. HOG operates by extracting information about different oriented edges within pixel blocks of images. In essence, it analyzes the distribution of gradient orientations, capturing the unique structural patterns and textural details inherent to each animal. HOG can be particularly effective in differentiating substantially distinct structures including but not limited to elephants, butterflies, spiders, and chickens among the 10 classes. The HOG features were extracted with (8 by 8) pixels per cell and (1 by 1) cells per block, with the goal being to capture larger patterns and structures in an image by moderately large dimensions of pixels per cell.



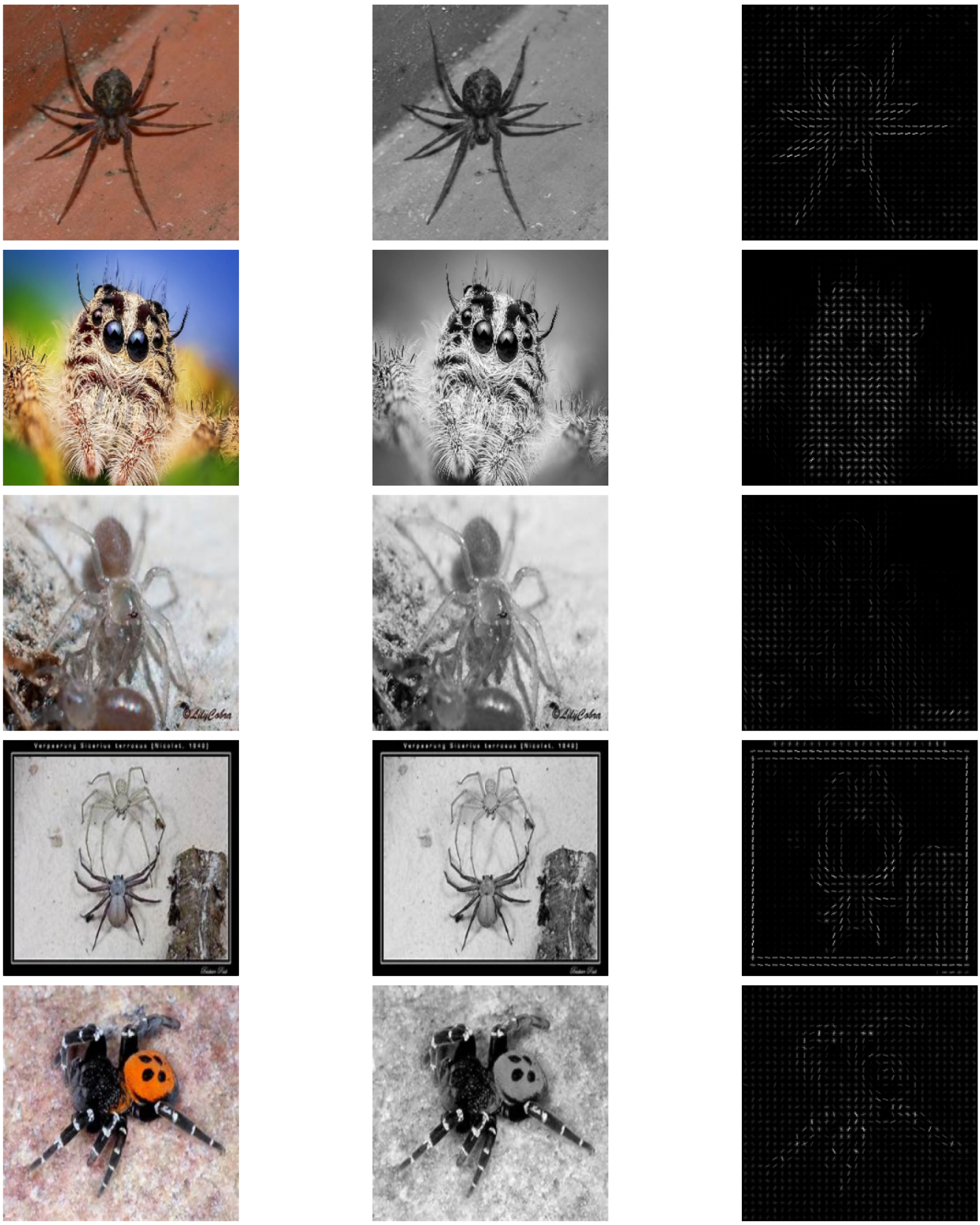


Figure 5: Examples of HOGs from Spider and Horse classes. Each set includes the original resized image, its grayscale version, and the extracted HOG representation. The changes in framing and background noise appear to be significantly affecting the features extracted.

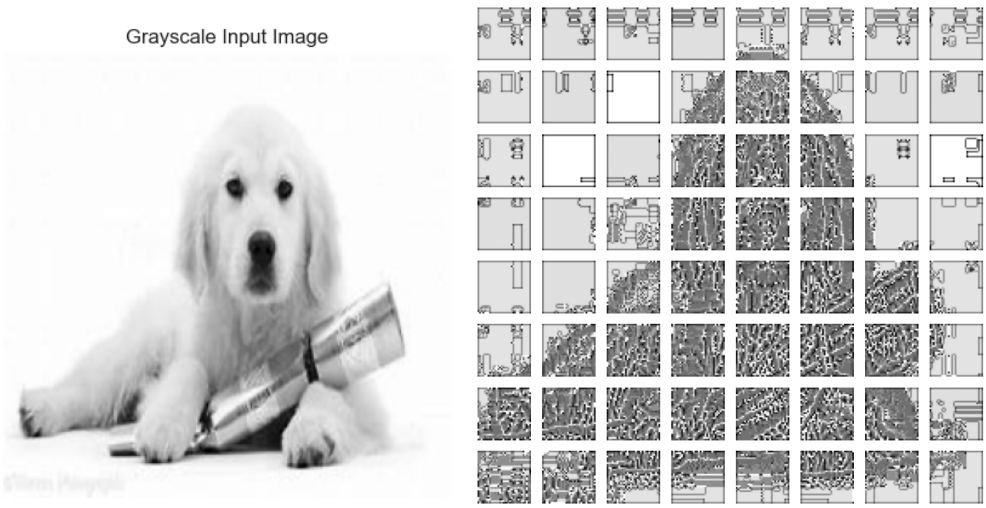
Local Binary Pattern (LBP) and Dense Scale-Invariant Feature Transform (SIFT)

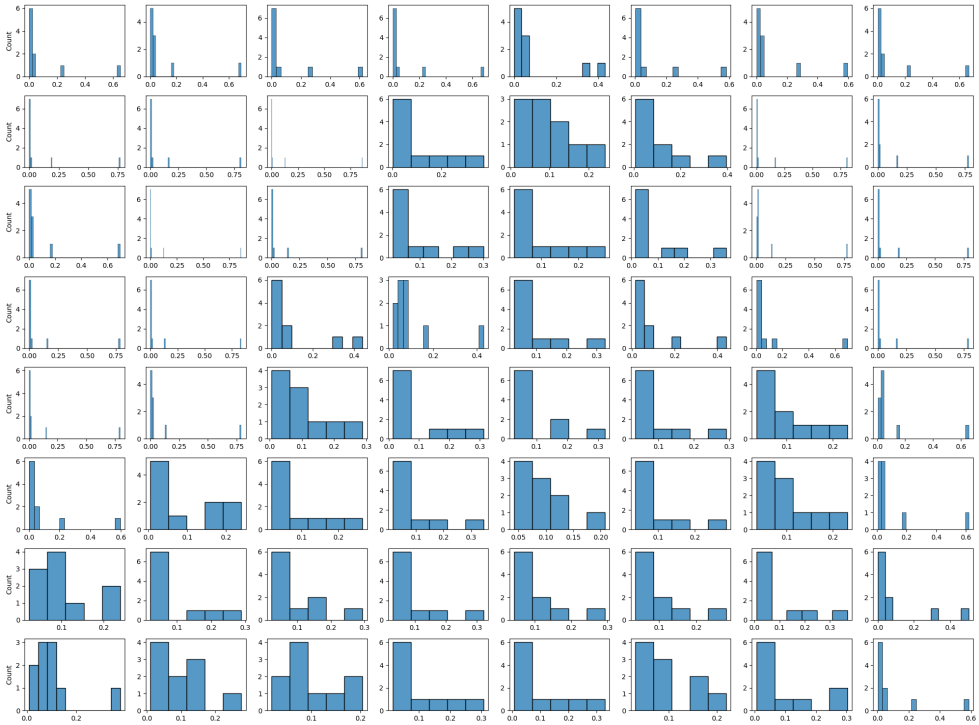
Inspired by the work published in the EURASIP Journal on Image and Video Processing (Yu et al.), ‘Automated identification of animal species in camera trap images’ incorporates the use of Local Binary Patterns (LBP) combined with Dense Scale-Invariant Feature Transform (SIFT) to identify animals captured in camera trap images. Given the function of their work having similarities with our task, utilizing aspects of their work could prove to be beneficial. We incorporate their methods with some adjustments to key parameters and exclusions of certain steps that led to very large feature vectors resulting in computation bottlenecks and steps that appeared to be outside the scope of this project. These adjustments include excluding step sizes to extract texture and microstructure data from overlapping regions of the different blocks. Instead, we opted to extract features only from non-overlapping blocks, which would potentially prevent us from obtaining a more comprehensive selection of feature vectors as we could be missing out on certain textures and microstructures at the boundaries of the blocks. In addition, we also excluded dictionary learning and weighted sparse coding for both features.

2.3. Local Binary Pattern (LBP)

Local Binary Patterns (LBP) serve as a vital texture descriptor in image analysis, comparing each pixel with its neighbors and encoding their relationships as binary patterns. LBP operates by examining the binary values of the intensity levels of neighboring pixels in a defined local neighborhood around each pixel. Specifically, it labels each neighbor as '1' if its intensity is greater than or equal to that of the central pixel and '0' otherwise. These binary labels are then concatenated to form a binary pattern, capturing the local texture information at that particular pixel.

This technique excels in capturing local texture variations, proving indispensable for image classification and object recognition. Widely applied in tasks such as face and pedestrian detection, LBP can be equally adept at capturing global textures through implementations. Our implementation involves dividing each image into 32 by 32-pixel blocks, extracting LBP histograms within each block to represent local texture descriptions for each part of the image. The final step concatenates the histograms from each block of the image, providing a comprehensive global representation of textures across the entire image. LBP effectively captures local spatial patterns and the grayscale contrast under moderate non-linear luminance and remains invariant to image rotation. In each localized block from our images, we can effectively identify local textures that may be common within the global representations among the different animal images. This approach is extremely beneficial for cases where local texture variations are significant, and our hope in utilizing this method is to differentiate the different textures of each animal.





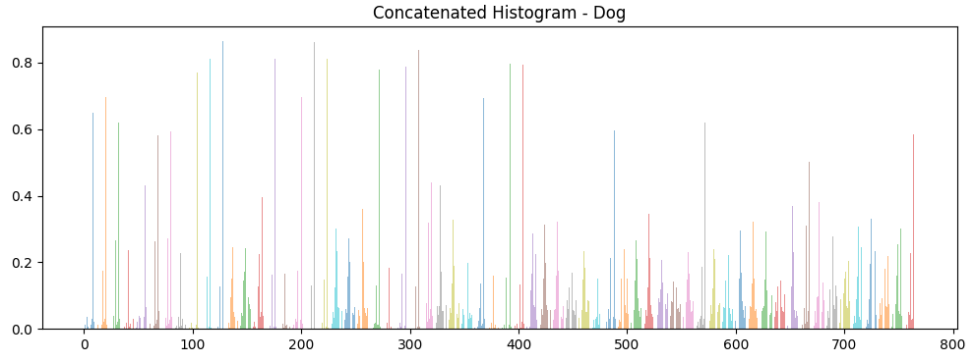

Concatenated Histogram - Dog
0.0
0.2
0.4
0.6
0.8
0
100
200
300
400
500
600
700
800

Figure 6: LBP feature extraction process from dog and elephant images

Figure 6 outlines the process for extracting LBP features from single dog and elephant images. Each image is split into 32 by 32 pixel-blocks, the LBP of each block is computed, and the corresponding texture histograms are recorded. These histograms are concatenated to form the global representation feature vector. There is a difference in patterns being extracted from local regions across the two images and the presence of these features in the global representation can provide the distinct variation needed to identify the different animals. However, it is important to note that the order of these features in the global representation may vary due to differences in image framing. This variability could potentially pose a challenge in certain applications, and careful consideration should be given to address or mitigate such issues in future image analysis tasks.

2.4. Dense SIFT

Dense SIFT is a variation of SIFT, which identifies interest points in an image that become the keypoints, around which the orientation and gradient of each pixel are determined. The resulting descriptor generated by SIFT encapsulates this information around each of the keypoints in a 128-dimensional vector. Dense SIFT, on the other hand, densely samples keypoints across an image at regular intervals in a grid-like pattern, instead of relying on interest points as often determined by Lowe's algorithm. The functionality around each point remains the same, and the resulting descriptors are generated to provide insight into local gradients and orientations across the points at regular grid intervals. These localized features will allow for the identification of localized features in the animals, like patterns and microstructures, while being robust to changes in framing, and rotation (serves similar functionality to LBP above). Furthermore, SIFT is also invariant to moderate scaling, so it can recognize animals of various sizes due to their framing. This would be especially beneficial in our dataset, where there often are a lot of these changes. However, unlike LBP, it does not function as well during nonlinear illuminance changes within an image.

We utilized a variation of the method of Dense SIFT discussed in the paper (Yu et al.), which takes in 16 by 16 blocks and creates keypoints within this interval (every 16 pixels). However, we did not implement their sliding patch at a smaller step size method as it would increase the density of the keypoints and increase the computation power needed to extract the features from our entire dataset. The resulting descriptor extracted information about orientation and gradient magnitudes around each keypoint.

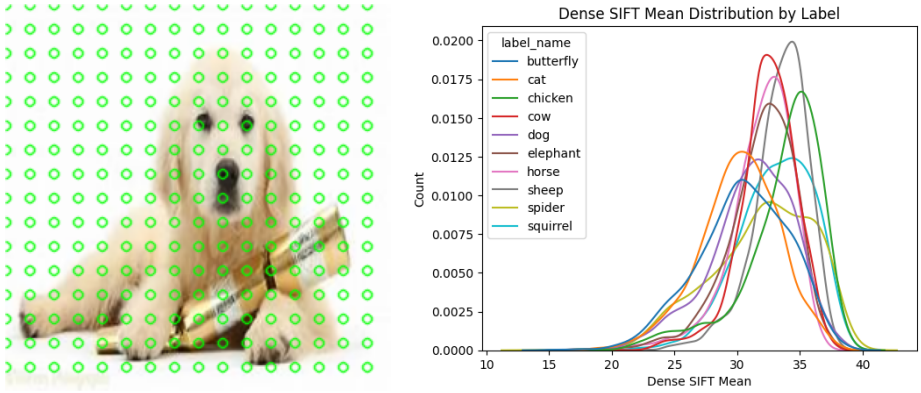


Figure 7: Densely sampled key points and distribution of the SIFT descriptor means for example dog image

Figure 7 depicts an example dog image with the densely sampled keypoints as determined by our 16x16 block size, which will be the same regardless of image or class. Within each keypoint area, the resulting SIFT descriptor is extracted. The plot on the right in Figure 7 outlines the distribution of mean SIFT descriptors for the densely sampled keypoints for all animals. The distribution of means for each class is varied enough to demonstrate the viability of this feature on its own.

2.5. Bag of Visual Words (BoVW)

BoVW involves extracting local features from images, clustering them to create visual vocabulary, and representing each image as a histogram of the frequencies of these visual words. BoVW especially excels in its ability to capture distinctive patterns while reducing the dimensionality of the feature space, making it computationally efficient. While other feature extraction techniques focus on different aspects such as color information, gradient orientations, texture patterns, and key points, BoVW focuses on local features and their spatial distribution.

The 2021 research paper (Sitaula et al) that successfully used BoVW features to classify chest x-ray images for COVID-19 diagnosis as well as older works like the 2013 Bio-image classification paper (Godel et al) provide a strong basis for the adoption of this technique to our classification task.

For BoVW implementation, first, key descriptors are extracted from images using SIFT. This feature identifier looks for points of interest within the image and represents them with 128-element embeddings. Subsequently, these individual descriptor embeddings are compiled across all images and clustered into a vocabulary using K-means clustering, establishing unique visual words. Each image is then analyzed and a histogram is generated based on the frequency of occurrence of visual words from the vocabulary. Each descriptor identified in the image is assigned to its closest visual word within the established vocabulary, contributing to the corresponding visual word's histogram count. The histograms for each image, reflecting the distribution of visual words, are then combined through concatenation to produce the final feature vector.

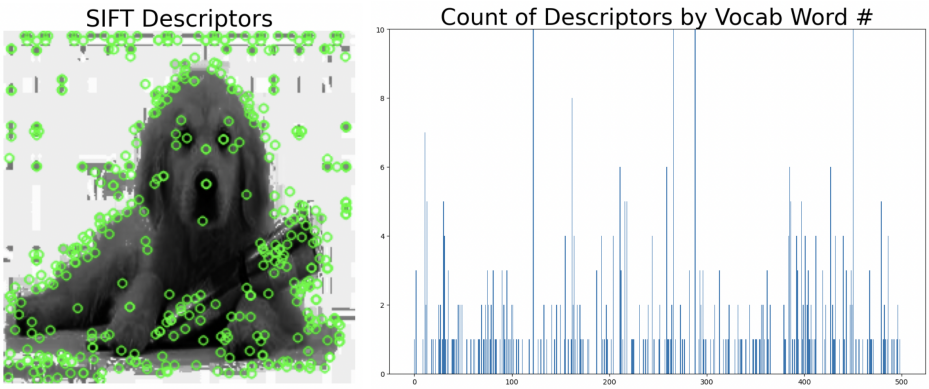


Figure 8. The figure on the left displays an example image with SIFT descriptors highlighted. The figure on the right illustrates a histogram showing the count of descriptors assigned to each visual word in the vocabulary.

While predicting the performance of Bag-of-Visual-Words (BoVW) solely based on histograms can be challenging, Figure 9 demonstrates that the distribution patterns of the top 5 words in the BoVW representation vary significantly across different animal classes. The distinctiveness in the distribution of these visual words underscores the potential discriminative power of BoVW features for distinguishing between various animal categories.

There are several key hyperparameters for the initial SIFT classification and subsequent K-means clustering that we were not able to fully explore due to the computational complexity of the K-means algorithm combined with a large number of SIFT features. The contrast and edge threshold values as well as the maximum number of features per image are important features to dial in to ensure the SIFT algorithm focuses on key points in the actual subject of the image rather than blurry background information that may not provide insight into the classification task. We also did not apply inverse weighting or other techniques to the counts of words to help decrease the importance of common ‘words’ in the vocabulary which may help explain why this feature did not perform as well as we expected.

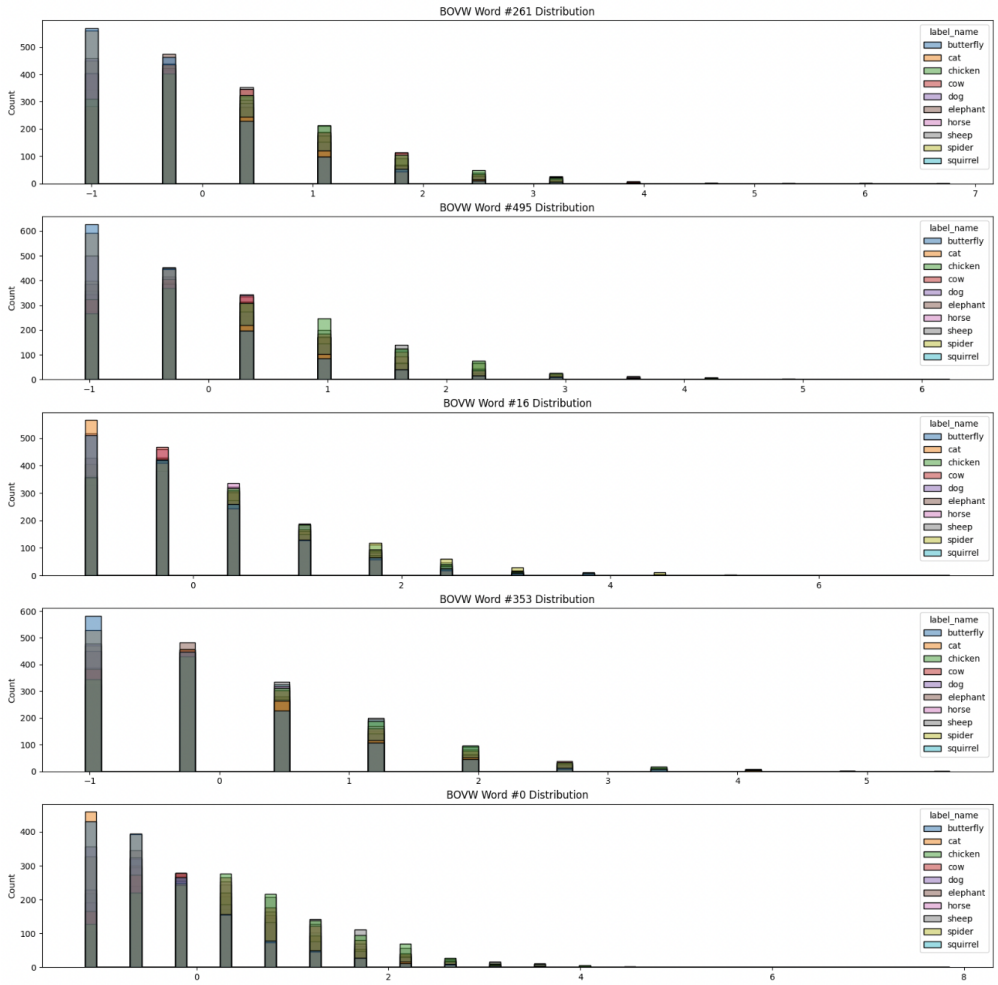


Figure 9. Distribution of top 5 words in BoWW representation by label

2.6. Complex features from pre-trained Neural Networks

Our next approach took into consideration the potential benefits of pre-trained convolutional neural networks, which would provide us the benefit of extracting features from models that were trained on datasets similar to our own.

ResNet101:
ResNet101 was trained on the ImageNet dataset, which is a very similar data set in nature to the Animals-10 subset and, this pre-training on the dataset will allow ResNet101 to identify common visual cues that could contribute to identifying animals. ResNet101 employs a Convolutional Neural Network architecture that progressively reduces the size of the input image with kernel convolution operations that learn kernel weights to extract meaningful information from different aspects of the image. By extracting the outputs of these CNN outputs at one or more stages of the ResNet sequence, we can harness the hierarchical features learned by the model to understand and classify intricate patterns within an image. In our case, we passed preprocessed images to meet the requirements of ResNet101, images resized to 224 by 224 and fed them to the model after removing the classification layer to extract embeddings from the network, which provides a high-level representation of each image in the resulting feature vector. Each dimension of the feature vector extracted captures different aspects of the image content. These aspects can range from simple textures and patterns in early layers to more complex and abstract features in deeper layers.

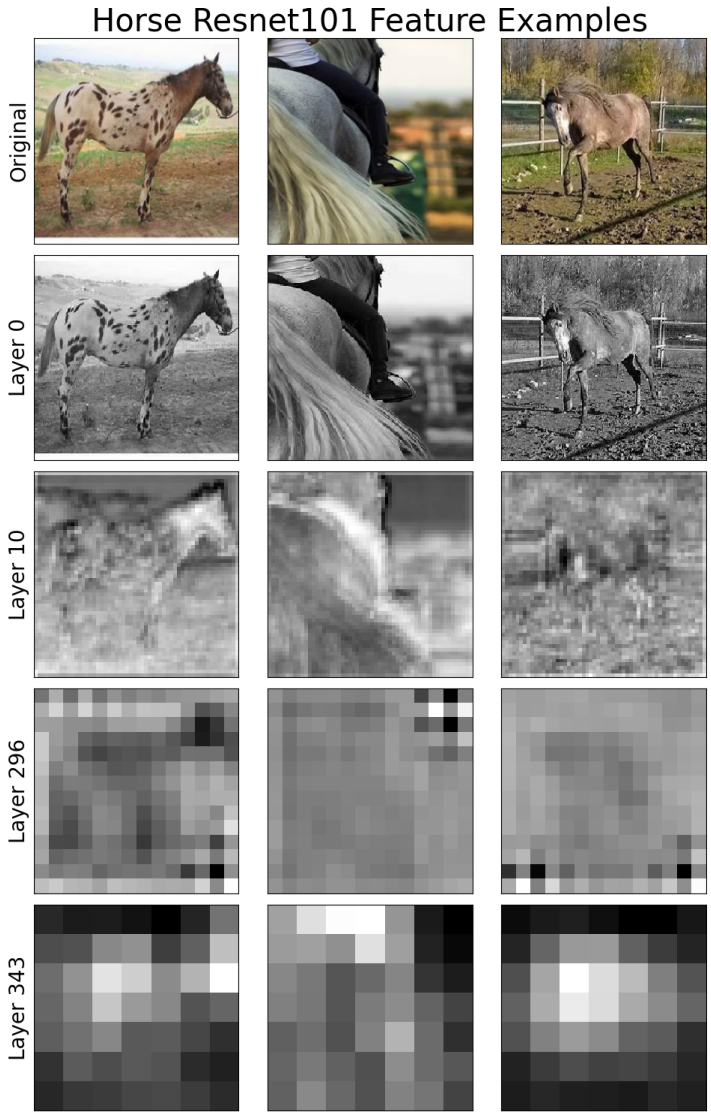


Figure 10: Feature examples from ResNet101 model at layer 0, layer 10, layer 296, and layer 343 for 3 different horse images

EfficientNet:

EffecientNet was developed by Google and includes various models of increasing computational complexities that allow adaptations to different hardware setups. The goal of EffecientNet is to be efficient - achieve optimal performance with the minimum number of parameters. It does this by simultaneously scaling the depth, width, and resolution of the neural network. Similar to ResNet101, EfficientNet employs a convolutional neural network and is pre-trained on datasets including ImageNet. EfficientNet outperforms ResNet101 in efficiency through its adoption of compound scaling, which uniformly adjusts network depth, width, and resolution. This balanced scaling approach allows EfficientNet to achieve competitive accuracy while being computationally efficient, making it perfect for resource-constrained scenarios. Also, the use of lightweight depthwise separable convolutions further contributes to reducing parameters and computations, enhancing the overall efficiency of EfficientNet compared to the more parameter-intensive ResNet101.

Figure 11: Feature examples from EfficientNetB0 model at layer 0, layer 1, layer 10, layer -50, and layer 7 for 3 different horse images

2.7. Principal Component Analysis (PCA)

PCA is a widely employed dimensionality reduction technique, particularly valuable for image classification. Its application involves feature extraction and dimensionality reduction, proving especially valuable when confronted with high-dimensional datasets such as image datasets.

PCA is completed by following a set of steps, including transforming pixel-based features into orthogonal principal components to capture and retain the most significant variations in the data. This process involves choosing the number of principal components to retain based on the desired level of information preservation. The resulting reduction in dimensionality not only simplifies computational complexity but also helps prevent overfitting. This simplified information is then utilized in classifiers, improving the efficiency and accuracy of image classification models.

We initiated the analysis by implementing PCA to reduce the dimensionality of each of the aforementioned features. Then, we graphically depicted the relationship between explained variance and the number of principal components to gain a more insightful perspective on the impact of PCA on these features. Based on the outcomes of the PCA analysis on individual features, a decision was made to combine certain features, followed by a rerun of PCA on the combined set. The outcome of this comprehensive PCA analysis is illustrated in Figure 12.

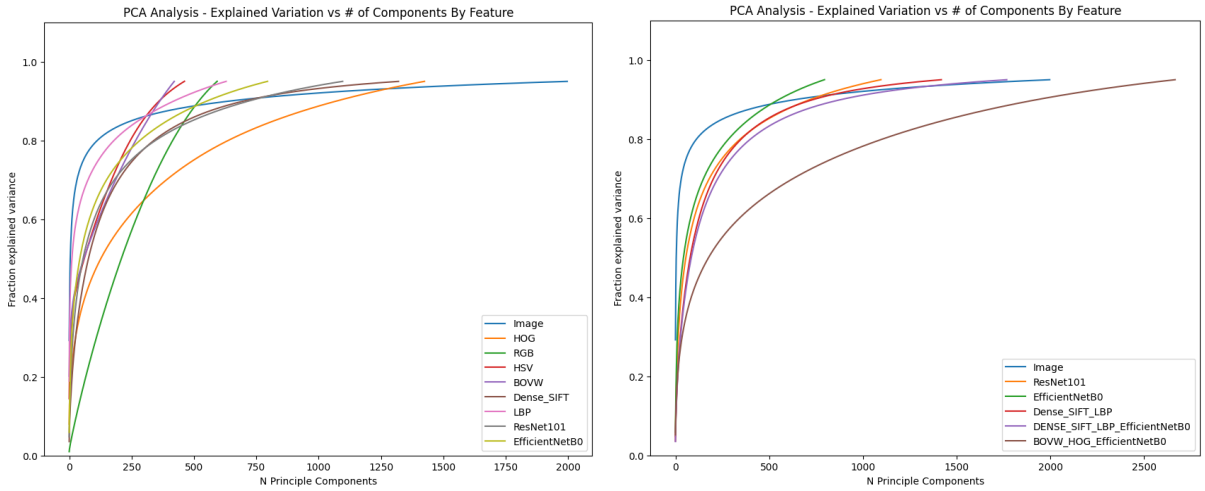


Figure 12: The fraction of explained variance for different feature sets as the number of principal components changes.

From looking at the result, a couple of trends were observed. First, EfficientNetB0 reached the 95% explained variance with less number of components compared to ResNet101. This might explain the more efficient nature of EfficientNetB0 when compared to ResNet101. Second, HOG reached the 95% explained variance noticeably slowly when compared to other simple features such as RGB, HSV, and BoVW. This might be because HOG, focusing on gradients and edge orientations, may require more components to capture the intricacies of complex structures in animal images in the dataset.

2.8. t-distributed Stochastic Neighbor Embedding (t-SNE)

t-SNE is a technique often used for image classification, particularly in scenarios where visualizing high-dimensional data is important. Unlike PCA, which focuses on linear transformations, t-SNE works by capturing complex, non-linear relationships within the data. In the context of image classification, t-SNE is used to reduce the dimensionality of image features while preserving the local similarities between data points. This method is very useful for visualizing clusters and patterns in high-dimensional image datasets, aiding in the identification of distinct classes. By transforming pixel-based features into a lower-dimensional space, t-SNE improves the interpretability of image data, facilitating improved understanding and classification accuracy when integrated into machine learning models.

Figure 13 displays the t-SNE visualizations for distinct features, including HOG, Dense SIFT, BoVW, EfficientNetB0, and ResNet10. It also illustrates combined features, such as Dense SIFT + LBP, Dense SIFT + LBP + EfficientNetB0, and BoVW + HOG + EfficientNetB0.

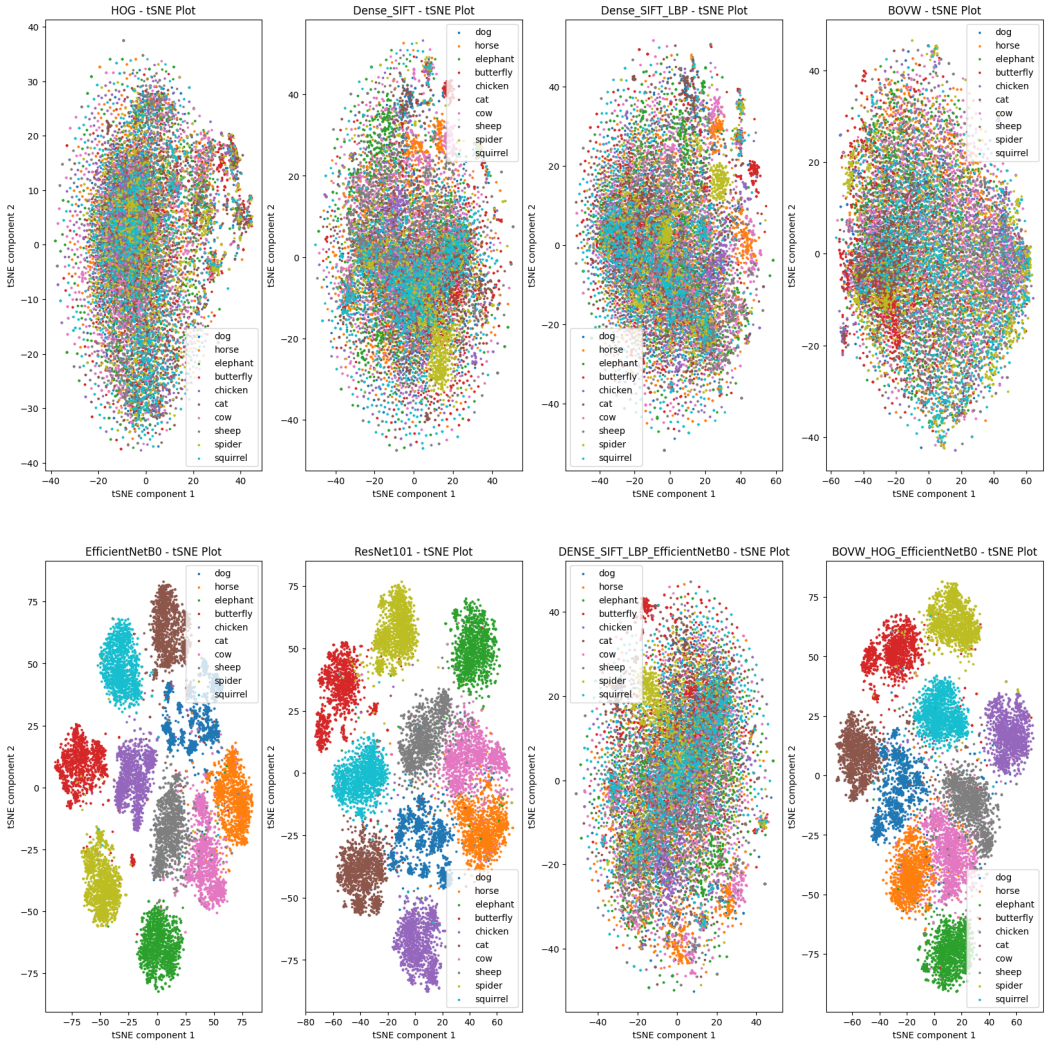


Figure 13: t-SNE visualization of 8 different features

In the t-SNE visualization analysis, it is quite evident that EfficientNetB0, ResNet101, and the combined feature set BoVW + HOG + EfficientNetB0 effectively separated the 10 animal classes in the reduced dimensional space. This success can be attributed to the fact that EfficientNetB0 and ResNet101 are pre-trained models that have been exposed to a diverse range of images, enabling them to capture nuanced complexities in animal images in our dataset effectively. Nevertheless, the combined feature set Dense SIFT + LBP + EfficientNetB0 displayed reduced classification efficacy in t-SNE visualization, despite the inclusion of EfficientNetB0. This discrepancy may arise from the fact that t-SNE visualization quality does not always directly reflect classification performance. Alternatively, the intricate interactions among features within Dense SIFT + LBP + EfficientNetB0, leading to complexities like redundancy and high dimensionality, could contribute to this difference. These challenges may reduce the model's pattern recognition performance in animal classification. Further exploration of feature interdependencies can offer insights to improve feature combination strategies.

3. Classification

In this section, we aim to demonstrate the effectiveness of the extracted features and various feature combinations in training different models to accurately classify the ten distinct classes of animals within our dataset.

In our classification approach, we used Logistic Regression, Linear SVM, Non-Linear SVM, and Multi-Layer Perceptron (MLP) as our classifiers. Each classifier was trained using both individual features and various combinations of these features. Feature combinations were created by either pairing two or more simple features with one of the complex features, by pairing simple features based on previous efficacy in literature, or by maximizing the combination of simple features.

Following the training of our diverse models, we opted to utilize validation accuracy as the primary metric for evaluating their performance. Choosing validation accuracy as our performance metric allows us to assess the performance of our models on unseen data and to take the necessary steps for hyperparameter tuning to enhance the model accuracy while preventing overfitting.

Results

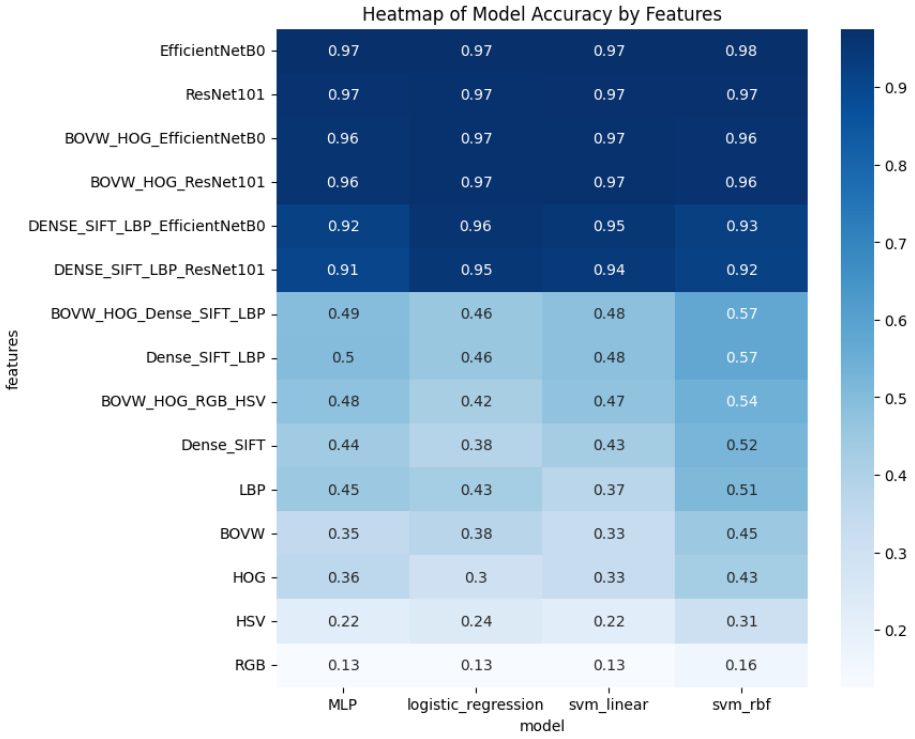


Figure 14: Validation accuracies heatmap for untuned MLP, logistic regression, linear SVM, and non-linear SVM models across diverse feature vector combinations

As seen in Figure 14, the overall performance comparison reveals that, in general, non-linear SVM outperformed the other models across various combinations of feature vectors (was especially the case with simple features). Additionally, the CNN pre-trained models demonstrated a significant superiority over simple features. EfficientNetB0 on its own demonstrated exceptional performance, achieving the highest validation accuracy at 98% when using non-linear SVM as the classifier.

Upon closer examination of simple features, their performance varied. For example, utilizing Dense SIFT in isolation achieved a maximum validation accuracy of 52% for Non-Linear SVM. Combining two or more simple features moderately increased validation accuracy, yet was not substantial enough for practical deployment. The most successful combinations of simple features involved Dense SIFT paired with LBP, BoVW and Hog, and Dense

SIFT paired with LBP. However, these combinations did not surpass the validation accuracy of 57% for Non-Linear SVM and had substantially lower validation accuracies than complex features and combinations involving complex features.

Furthermore, the integration of simple features alongside the CNN features resulted in a modest decrease in accuracy, contrary to the anticipated improvement. This result is not entirely surprising, given the nature of the dataset and the inherent characteristics of the models. Considering the limited success of simple features when used independently, their incorporation into CNN features most likely introduced more noise than benefit, impeding the training of the more fitting CNN features.

Models and Features Evaluation

Logistic regression: Logistic Regression is a maximum likelihood classifier, which predicts the probability of an input belonging to one of two classes. It builds upon linear regression by feeding the inputs through a sigmoidal function, which effectively constrains the output between 0 and 1. This value corresponds to the probability of the input belonging to the positive class. Logistic regression's use can also be extended to multiple classes, especially relevant for our case, and it does so by adopting the 'one vs all' approach. In this approach, multiple binary logistic regression classifiers are compared, and the classification is chosen based on its maximum likelihood.

Although it is a powerful and quite popular algorithm, linear regression does come with some limitations, with the most significant one being its assumption of a linear relationship between the independent and dependent variables. This limits the classifier from capturing the more complex relationships between the input and its label, which may especially be the case for BoVW, Dense SIFT, and LBP. These features capture complex texture and structural patterns that have high dimensionality that may not be easily captured by a linear decision boundary. Even simpler features such as HSV and RGB may have very limited discriminative power due to the variation in the quality of the images due to inconsistent framing and backgrounds.

Logistic regression demonstrated outstanding performance when utilizing ResNet101 and EfficientNetB0 features, which is not unsurprising. These models were trained on millions of images with the goal to generalize well on diverse image datasets. Furthermore, the robust architecture of ResNet101 and EfficientNetB0 is designed to handle variations effectively, a characteristic that aligns well with the challenges present in our dataset. Consequently, it can be inferred that logistic regression, when fed these robust features, can achieve better generalization, leading to better validation accuracy.

Linear SVM: Linear Support Vector Machine is another supervised learning algorithm that has the objective of finding the hyperplane that best separates the data points of different classes in the feature space while maximizing the margin between the classes as it helps improve the model's generalization.

Linear SVM can potentially fall into the same traps as logistic regression, as it can handle data quite well when it is linearly separable and struggles to capture complex relationships when the data is not linearly separable. To that effect, simple features with high dimensionality like BoVW, LBP, and Dense SIFT had a lot of complexity, making Linear SVM a relatively weak classifier for these particular features. In addition, certain feature extraction methods, like LBP, are sensitive to scale changes, while others, such as Dense SIFT, are robust against variations in illuminance within an image. While the combination of features, such as LBP and Dense SIFT, aimed to mitigate individual limitations and enhance performance, the observed gains were not as substantial as anticipated. This outcome can be attributed to other complex and intricate variations, such as changes in background limiting the substantial effect on the generalizability of these features.

RGB and HSV histograms demonstrated even poorer performance, likely because of the inherent limitations arising from the considerable variation in the images. Furthermore, too much diversity in colors within each animal class posed a major challenge for linear SVM to establish a substantial distinction, preventing their ability to generalize effectively.

Non-Linear SVM: Non-Linear SVM can perform better, in comparison to their linear counterpart, when the underlying relationships in the data are complex, and a linear decision boundary is insufficient to accurately separate different classes. By utilizing different kernel functions, Non-Linear SVM can create more flexible and complex decision boundaries which can quite effectively capture the Non-linear relationships.

As seen in Figure 14, in general, Non-Linear SVM outperformed the rest of the classifiers (in some cases almost an increase by 10%). This can be attributed to the high dimensional complexity arising from the Non-linear nature of our feature vectors. Certain features like RGB and HSV histograms continued to perform poorly, which continues to highlight the lack of variability these features offer to identify different animals, and within the context of our dataset, is an all-around poor feature to utilize.

Our other more complex simple features, including HOG, LBP, Dense SIFT, BOVW, and combinations of these features also saw some moderate increase in performances, but they still trailed behind the complex features. As mentioned in the previous models, the variation within our dataset, including background noise, subject posing, and framing, significantly contributed to the quality of these features. Furthermore pairing two or more simple features that are robust to different variations, although improved some performance, was not enough to prevent the effects of some variations.

Our complex features continued to outperform the rest of our simple features quite significantly. The deep architectures of ResNet101 and EffiecntNetB0 are designed to learn hierarchical features at different levels of abstraction. Lower layers capture simple patterns like edges and textures, while higher layers capture even more complex and abstract features that our simple features cannot capture. Given that the images from our dataset reflect the complexity of real-world images, as they contain tons of complex variations, our simple features are unable to capture unique patterns under these complexities.

Multi-Layer Perceptron (MLP): MLP is a type of artificial feedforward neural network made up of multiple layers of neurons arranged in interconnected layers. The architecture itself consists of an input layer, potentially multiple hidden layers, and an output layer. Each connected neuron has a weight associated (which gets learned through backpropagation to minimize the difference between the predicted output and the actual label) along with an associated non-linear function which introduces non-linearity in the model to learn complex relationships.

The biggest limitation of MLP, and which is highly applicable to our case, is that it is quite prone to overfitting, meaning, it learns patterns that are not quite generalizable. This has been seen among all cases of training MLP on all combinations of feature vectors as they were overtrained to have a training accuracy of 100%. The effect of overtraining had a considerable effect on the simple features extracted due to the lower validation accuracy, but not substantially different than the other models. The more complex feature continued to perform exceptionally. The reasoning behind this difference can again be attributed to the nature of the different features as mentioned in the previous cases.

To summarize, the images feature animals in a wide range of poses and diverse background conditions, often accompanied by people or other non-class animals, along with additional elements such as image labels, texts, and artifacts. Features like HOG, which excel in face classification with more standardized input images, face challenges in such diverse contexts.

To remedy these issues, more steps could have been taken to improve the associated performances from these simple features. While BoVW offers potential for fine-tuning parameters like contrast, alpha, and vocabulary size to enhance the resulting visual vocabulary for better key points, the computational intensity of the vocabulary extraction process, particularly for medium-sized vocabularies, makes fine-tuning impractical for this exercise. Our method of utilizing LBP and Dense SIFT extracted texture and microstructure data from non-overlapping equal blocks. This methodology specifically omitted some substantial spatial information (not all patterns are fully contained within a single block). In LBP's cases, as it is sensitive to local texture variations, it misses the transitions and variations occurring near block boundaries, leading to less discriminative feature extraction. This issue can be resolved by following the methodology outlined in the paper 'Automated identification of animal species in camera trap images' (Yu et al.) by incorporating a step size that moves the blocks in patches. However, the biggest drawback of implementing this method is that it would require a lot more computation power to extract even higher dimensional vectors.

4. Generalizability

In developing our models, we devoted careful attention to making sure they were generalizable. The dataset was split into train, validation, and test subsets, with the distribution being 70%, 15%, and 15% respectively. As seen in Figure 14, we took special care in utilizing the validation subset to evaluate our models and also utilized the validation subset for hyperparameter tuning through a grid search. Following the refinement of our models, the test subset was employed to obtain the most equitable and unbiased evaluation.

After conducting an initial assessment through the validation accuracies of each model and feature pairing, it became apparent that most simple feature vectors did not yield satisfactory accuracy results. The exception was the combination of Dense SIFT and LBP, which achieved a validation accuracy exceeding 50%. Despite various attempts to explore different combinations of simple feature vectors, none proved to be quite effective. To optimize computational efficiency during the hyperparameter tuning phase through grid search, we opted to concentrate on fine-tuning hyperparameters for the top-performing candidates. In this context, all model feature combinations with a validation accuracy below 50% were judiciously discarded from the initial phase, leaving only those with demonstrated promise, as detailed in Table 1. This focused approach allowed us to direct our resources toward refining and optimizing the most viable configurations for subsequent phases of analysis.

Table 1 below demonstrates the utilization of grid search for hyperparameter tuning. For each model and feature combination, we outline the best parameter found through grid search, the tuned validation accuracy, and the original validation accuracy as showcased in the heatmap in Figure 14. We focused the grid search on regularization parameters with the hope of more evenly balancing our training and validation accuracy. It should be noted that in some cases the tuned parameters performed slightly worse than the original validation accuracy. This suggests that, in these cases, our initial hyperparameters yielded better results than the best parameters identified through grid search. Furthermore, this implies that the optimal parameters for these models were not part of the set of parameters being tested in our grid search.

Table 1: Grid search result for hyperparameter tuning

	Model	Features	Best Parameters	Tuned Train Acc	Tuned Val Acc	Original Train Acc	Original Val Acc	Tuned Train Time	Tuned Inference Time
0	Logistic Regression	BOVW_HOG_EfficientNetB0	{'C': 0.1, 'penalty': 'l2'}	1.000000	0.968328	1.0	0.966931	272.635703	0.015872
1	Logistic Regression	EfficientNetB0	{'C': 10, 'penalty': 'l2'}	1.000000	0.971588	1.0	0.972986	90.939025	0.004226
2	Logistic Regression	DENSE_SIFT_LBP_EfficientNetB0	{'C': 0.001, 'penalty': 'l2'}	0.992014	0.973451	1.0	0.960876	381.988567	0.009024
3	SVM	BOVW_HOG_EfficientNetB0	{'C': 0.1, 'kernel': 'linear'}	1.000000	0.966465	1.0	0.966465	680.286910	10.731169
4	SVM	EfficientNetB0	{'C': 0.001, 'kernel': 'linear'}	0.996706	0.974383	1.0	0.972054	177.907824	1.489358
5	SVM	DENSE_SIFT_LBP_EfficientNetB0	{'C': 0.0001, 'kernel': 'linear'}	0.998103	0.956684	1.0	0.950163	643.975927	11.013361
6	MLP	BOVW_HOG_EfficientNetB0	{'hidden_layer_sizes': 64}	1.000000	0.959013	1.0	0.961341	41.258962	0.023042
7	MLP	EfficientNetB0	{'hidden_layer_sizes': 128}	1.000000	0.973917	1.0	0.974849	18.069267	0.007065
8	MLP	DENSE_SIFT_LBP_EfficientNetB0	{'hidden_layer_sizes': (128, 64, 32)}	1.000000	0.921751	1.0	0.912436	34.942928	0.015482

Based on the outcomes of grid search, during training, all models exhibited a degree of overfitting to the training subset, as evidenced by consistently slightly higher training accuracies compared to validation accuracies. Ideally, both sets of accuracies would be near-identical. To mitigate overfitting, several steps can be taken, such as utilizing ensemble techniques. This methodology involves combining predictions from multiple models, which as a result reduces the biases from individual models. Furthermore, using a larger training dataset would add even more diversity to our dataset, enabling models to learn from a more extensive set of images.

Yet another viable approach, one we actively utilized, involved dimensionality reduction. In a high-dimensional space, data points can become more sparsely distributed and potentially fit the noise in the training data. This phenomenon, known as ‘the curse of dimensionality’, can make models less generalizable. We selected our best-performing model, as indicated in Figure 14, to investigate the effect of reduced dimensionality on both training and validation accuracies.

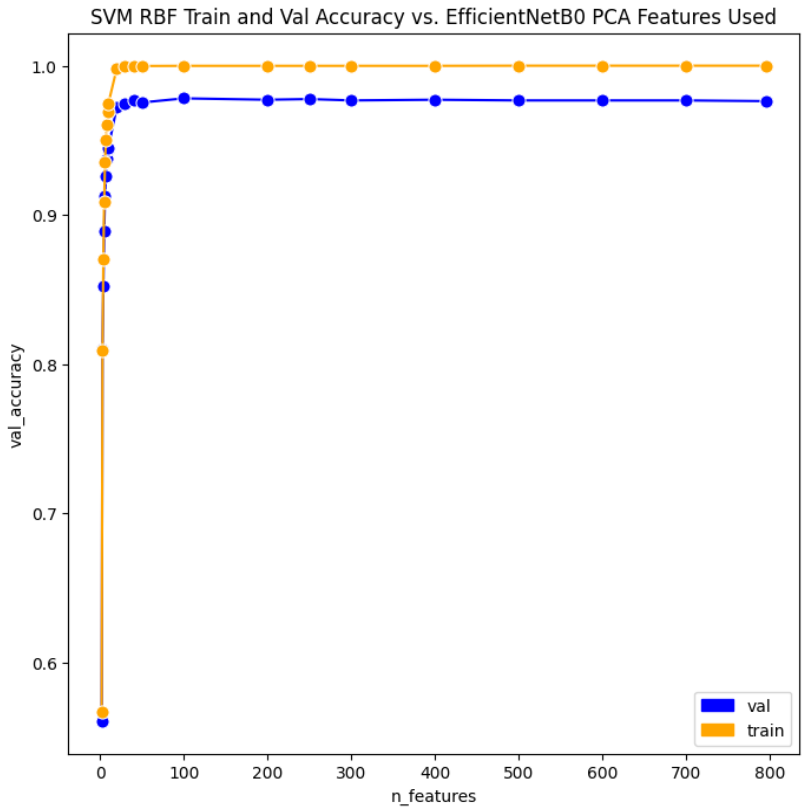


Figure 15: Effect of dimensionality reduction through PCA on training and validation accuracy for the EfficientNetB0 feature vector

As seen in Figure 15, as our model was trained on more features, we observed an increase in validation and training accuracies, but only up to a certain point where the accuracies reached a plateau. Furthermore, as our model was trained on a higher dimensional feature vector, we observed overfitting. Once our model began to overfit, the model itself did not perform any better in terms of generalizability and this trend occurred right after the inflection point for both validation and training accuracies. Based on our analysis results, it can be concluded that the model achieves the highest generalizability when the feature vector has a dimensionality of less than 10.

5. Efficiency vs Accuracy

In a production environment, it is especially important to understand the trade-off between accuracy and efficiency. Providing accurate results, although quite important, would not be useful if the users are unable to wait for the models to make their predictions. Similarly, during development, super complex models may take a considerable amount of time in training before being properly evaluated. In time-sensitive scenarios, this could potentially hurt the organizations that have been developing and deploying these models. As seen in Figure 15 below, we gauge the validation accuracy of all of our models in terms of train time and inference time, while utilizing different combinations of feature vectors.

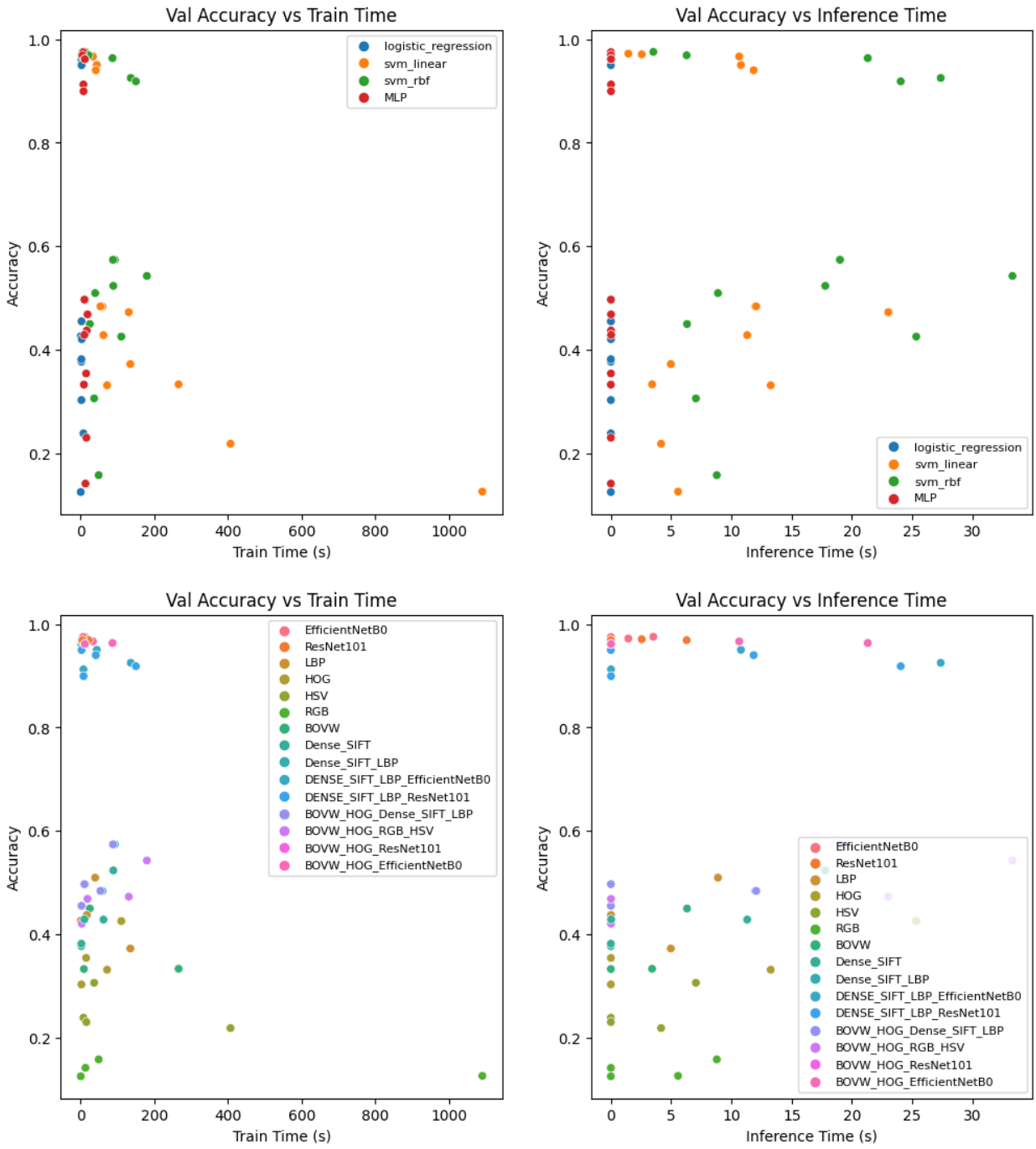


Figure 16: Scatter plots showing the validation accuracy and the associated train time (left) and inference time (right) for different combinations of models. The top row shows the models filtered by classifiers and the bottom row shows the models filtered by feature inputs.

Optimized Model

EfficientNetB0 by itself with an SVM using RBF kernel was the best-performing model in terms of accuracy. It also had a comparable or superior training and inference time to similarly performing feature sets and models (ResNet101 and EfficientNetB0 with simple feature combos). To further optimize the efficiency, we took the base best-performing model and the PCA transformed features truncated at 95% of explained variance and continued reducing features sequentially while retraining the model. We discovered that superior accuracy to our very complex simple feature vectors could be achieved with less than 10 feature vectors and the accuracy plateaued at our full-feature best accuracy of ~97.6% quickly as seen in Figure 15.

For our efficiency-focused model, we noticed that the accuracy plateaus (Figure 15) very quickly, and the training and inference time increase linearly with the number of features as seen in Figure 16. The validation accuracy is approximately equal to the training accuracy at 6 to 7 total features suggesting this is the inflection point for the model becoming over-trained. The minimum training and inference time is achieved with the inclusion of 10 features. After the inclusion of 20 features, the model training and inference time are practically equivalent to the 10 feature model but the validation accuracy rises above 97% for the first time. Therefore the 20-feature model is the most efficient model in terms of train and inference time without too much trade-off on accuracy.

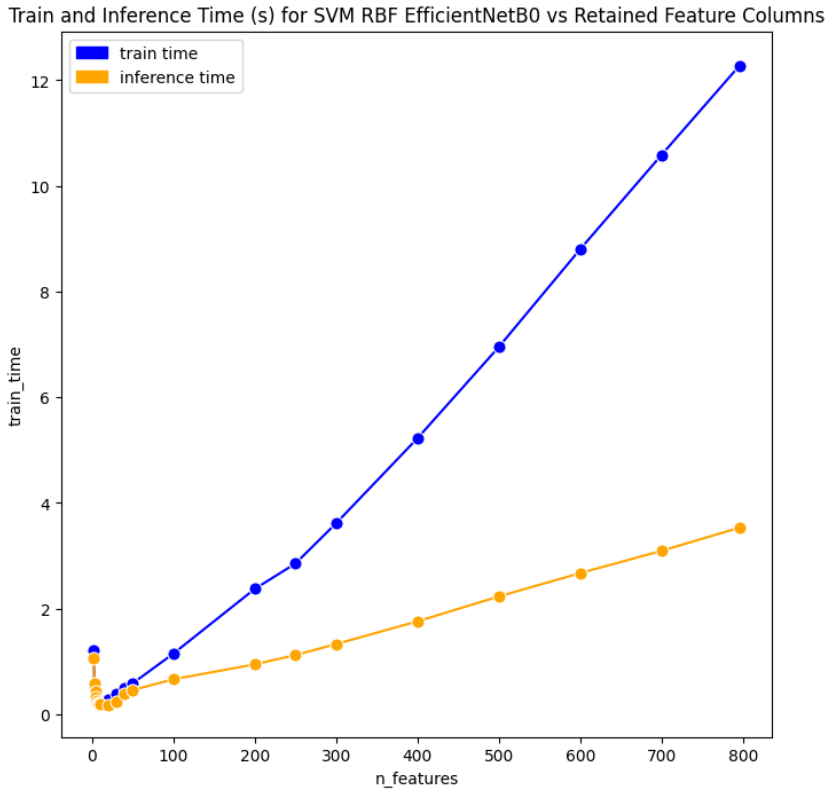


Figure 16: The plot acting as a companion plot to Figure 15, showing how additional dimensionality affects the train and inference times

After optimizing our model for efficiency, we evaluated its performance on our test dataset. Figure 17 below shows the confusion matrix that evaluates the model's performance at the individual image level and provides insight on the kinds of misclassifications found. Figure 18 shows a few example misclassifications occurring by our efficient model when the true label was a cow.

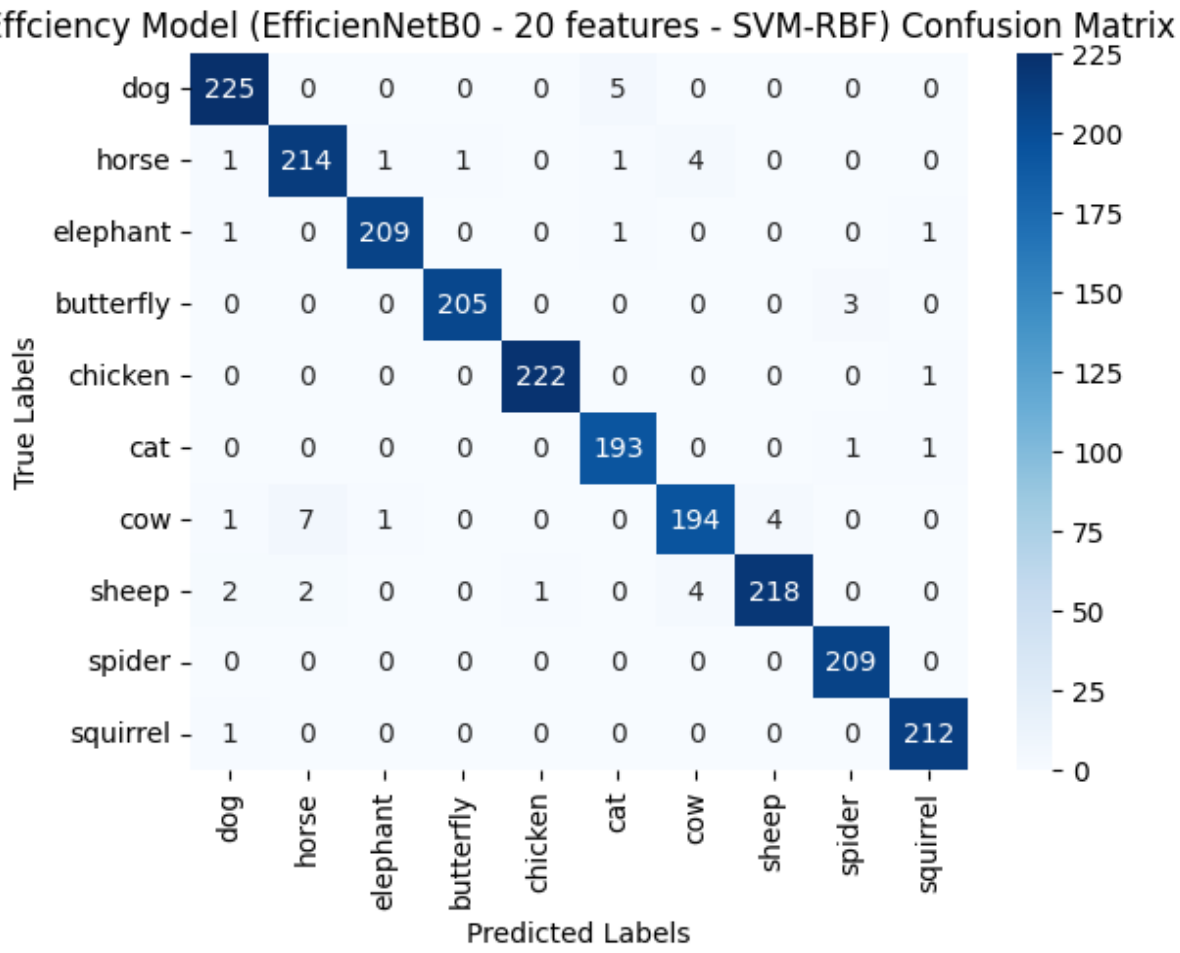


Figure 17: Confusion matrix depicting the performance of our efficient model on our test subset.

Figure 18: The images depicting the misclassification made by our efficient model when the true label was a cow.

The best model overall for validation accuracy throughout original feature down selection and hyperparameter tuning was the SVM-RBF model with only the EfficientNetB0 features. However, in terms of overall maximum validation set accuracy the truncated EfficientNetB0 features with SVM-RBF achieved a slightly higher accuracy at 0.978 using only 100 features compared to 0.975 with the full 800 features as seen in Figure 15. Figure 19 below shows the confusion matrix for our accurate model and Figure 20 shows a few example images of cow misclassifications.

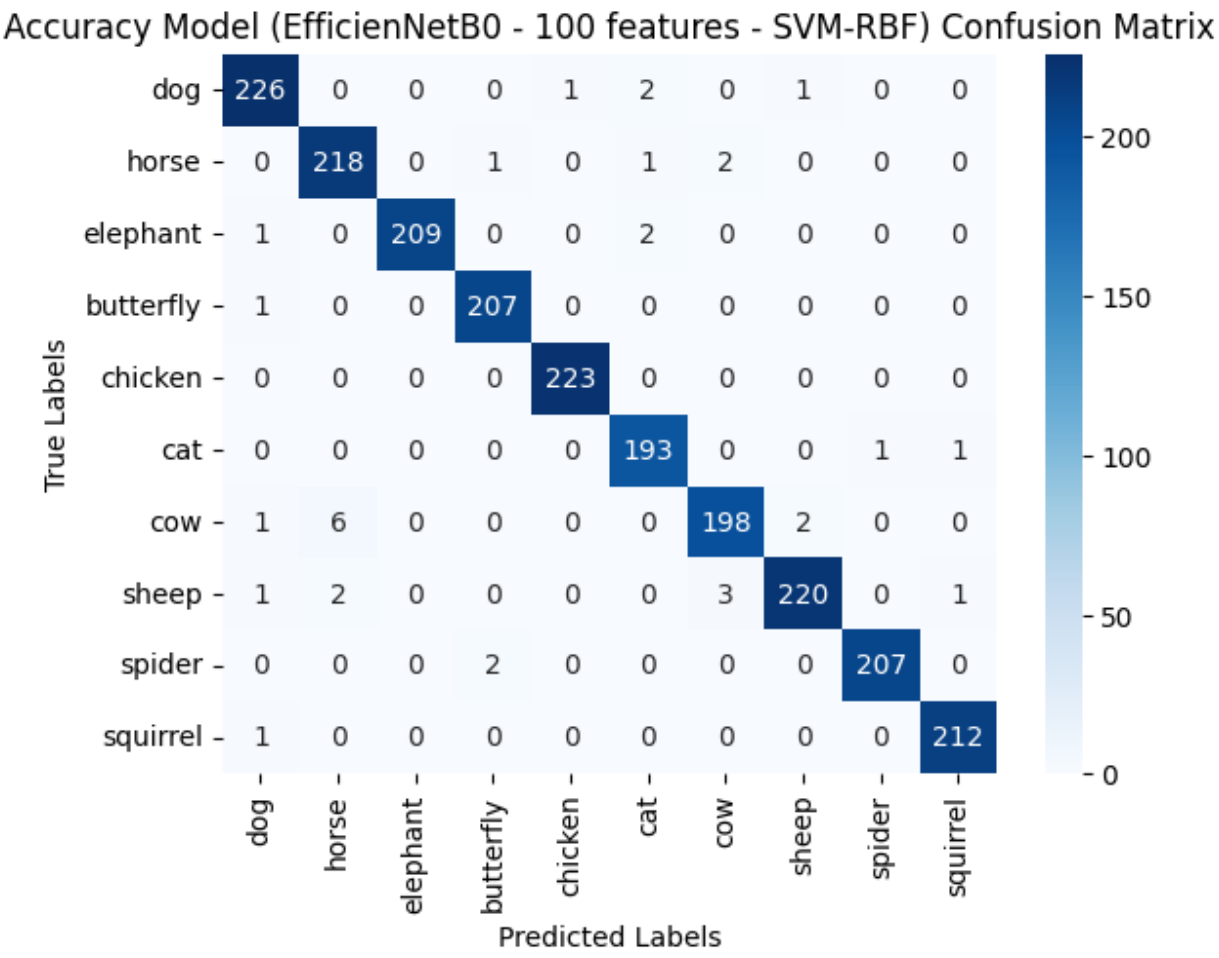


Figure 19: The confusion matrix showing the correct predictions and the misclassifications made by our most accurate model on our test subset.

Figure 20: The images showing the misclassification made by our accurate model when the true label was a cow.

Comparing the results of our two models specifically on the cow class, our more efficient model made 13 misclassifications as opposed to the 9 misclassifications made by our more accurate model. Furthermore, there was a substantial overlap in the images that were being misclassified.

Final Test Accuracy by Class:

	class	Efficent Model	Accurate Model
0	dog	0.978261	0.982609
1	horse	0.963964	0.981982
2	elephant	0.985849	0.985849
3	butterfly	0.985577	0.995192
4	chicken	0.995516	1.000000
5	cat	0.989744	0.989744
6	cow	0.937198	0.956522
7	sheep	0.960352	0.969163
8	spider	1.000000	0.990431
9	squirrel	0.995305	0.995305

```
Overall Test Accuracy:
Accurate:       0.98462
Efficient:      0.97903


Train Time (s):
Accurate:       1.14957
Efficient:      0.27723


Inference Time (s):
Accurate:       0.66375
Efficient:      0.18182
```

Figure 21: Direct comparison of two models (efficient and accurate) on our test dataset

Additionally, our more accurate model only performed better by ~0.5% than our more efficient model. Furthermore, the training time and the inference times, although lower for our more efficient model, would not be considered to be significant in a real-world setting as both models performed quite well in terms of efficiency.

6. Conclusion

In summary, although our finalized classification model worked quite well by utilizing complex features, we do believe that several additional steps could have been taken to increase the viability of some of the simple features extracted. Also, considering the diverse nature of our dataset, characterized by a substantial amount of variation, there is an opportunity for even more careful consideration in combining features. Although certain features may lack robustness to all variations when considered individually, combining them has the potential to enhance overall robustness and effectiveness. We did take a few careful considerations, for example, by combining Dense SIFT and LBP as they were both robust to opposing variations. However, due to computation limitations, we lost some potentially useful spatial features.

Beyond these considerations, for potential next steps in this project, it's worth noting that the Non-Linear SVM exhibited the best performance when solely utilizing EfficientNetB0 as the feature vector. This finding suggests a promising opportunity for even further exploration and optimization in the project's future steps. For example, to extend the accuracy of our model further, we can consider upgrading to a higher parameter EfficientNet model such as B7. Additionally, we can also take steps to further optimize the model for accuracy and efficiency.

References

1. Yu, Xiaoyuan, et al. “Automated Identification of Animal Species in Camera Trap Images - Eurasip Journal on Image and Video Processing.” *SpringerOpen*, Springer International Publishing, 4 Sept. 2013, jivp-eurasipjournals.springeropen.com/articles/10.1186/1687-5281-2013-52
2. Sitaula, Chiranjibi, and Sunil Aryal. “New bag of deep visual words based features to classify chest x-ray images for COVID-19 diagnosis.” Health information science and systems vol. 9,1 24. 18 Jun. 2021, doi:10.1007/s13755-021-00152-w
3. Godil, A. and Wagan, A. (2013), Exploring Local Features and the Bag-of-Visual-Words Approach for BioImage Classification, ACM Conference on Bioinformatics, Computational Biology and Biomedical Informatics (ACM BCB), 20013, Washington, DC, DC, [online],