Dog Image Classification Report

CMPSC 445

Professor Liaw

Mohammed Albattah, Emme Baldwin, Connor Campbell, Alexander Lilly

Table of Contents

Introduction

Purpose

This project aims to classify images of dogs by their age, gender, and breed. One practical use of this project would be when finding a dog and needing to find out information about the dog.

Current Solutions

Most image classifications are done using Convolutional Neural Networks (CNN). One study completed by Mazurek et al., "Canine Age Classification Using Deep Learning as a Step Toward Preventive Medicine in Animals," used a hybrid approach of CNN with a vision transformer (ViF) to address this problem. Unfortunately, their model did not yield accurate results.

Another study completed by Zamansky et al., "Automatic Estimation of Dog Age: The DogAge Dataset and Challenge Announcement," used two types of CNN architectures with six different classification techniques. The CNN architectures used were SqueezeNet and Inception v3. The classification techniques were Support Vector Machine (SVM), k-Nearest Neighbors (k-NN), Logistic Regression (LR), Naive Bayes (NB), Random Forest (RF), and feed-forward neural network (NN). However, this study did not use any preprocessing techniques on the images and called their findings baseline results. Figure 1 summarizes this study's findings using three metrics: Categorization Accuracy (CA), Average Recall (RA), and Mean Absolute Error (MA).

	SVM	kNN	LR	NB	RF	NN
Squeezenet						
CA	0.31	0.26	0.31	0.33	0.32	0.32
RA	0.31	0.26	0.32	0.33	0.31	0.32
MAE	3.20	3.73	3.17	3.05	3.24	3.21
Inception v3						
CA	0.28	0.30	0.26	0.31	0.32	0.34
RA	0.32	0.33	0.31	0.32	0.32	0.34
MAE	3.24	3.04	3.43	3.15	3.21	3.04

Figure 1: Baseline Results of Zamansky et al study for Dog Age Image Classification.

Introduction to Our Approach

We will use machine learning techniques to build a model to classify the dog images by age, gender, and breed. We used the previous study as a starting point and decided to implement multiple preprocessing techniques to attempt to improve the accuracy of the model.

We decided to use SVM as our classification technique and chose 3 different CNN architectures: VGGNet-16, VGGNet-19, and ZFNet. We discovered these CNN architectures through research and found that they are suitable options for image classification. To measure the performance of our model, we will be comparing the number of images classified correctly to those that didn't.

Assumptions

During this project, we made a few assumptions about the data. One key assumption was grouping certain dog breeds under a single "primary" breed label when they shared a common classification. For example, some dogs were labeled Chihuahua Short Hair, Chihuahua Long Hair, and Chihuahua. We consolidated these into one label: Chihuahua. These assumptions are easily verified by a quick Google search.

Additionally, we assumed that color was less important than edges and contrast when determining a dog's breed, age, or gender. This is based on the belief that a dog is more identifiable by its shape. We used this assumption when preprocessing the images.

Data Exploration

Obtaining Data

We found our data source on kaggle.com through a dataset titled “DogAge Dataset” which is open for public usage. This dataset consists of two parts, expert data and petfinder. We chose to use only the data in the expert data section because the images found in the Petfinder section had many problems. These problems included human hands blocking parts of the dog, the dog not looking toward the camera, and more. The expert data consists of 1088 images. However, we expanded the total number of images through image augmentation. Figure 2 shows a sample of the original dataset with labels.

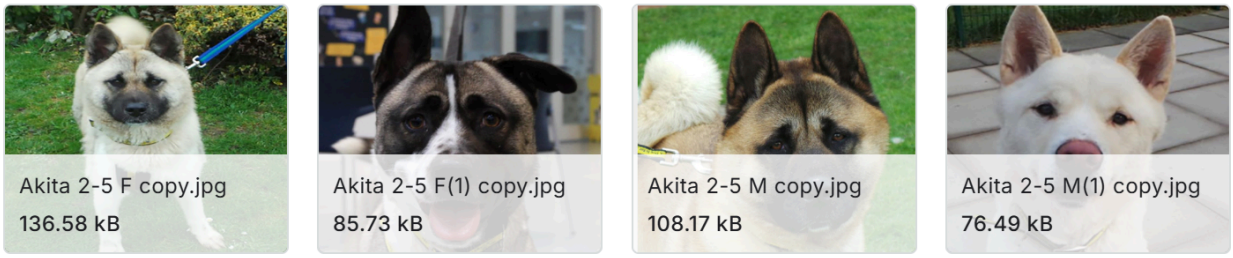

Figure 2: Sample of Images from the Expert Dataset with image labels.

Data Manipulation

To use the data, we created a data frame holding all the images and created variables for age, gender, and breed. To do this, we first removed any unnecessary words such as “copy” and anything else that didn’t belong from each of the labels. Then, we converted everything to uppercase to prevent any casing errors.

Next, we manipulated each dog's age to fit into certain categories. The original dataset had some dogs’ ages labeled in ranges but others with their exact age. For consistency, we converted all ages into a range.

Lastly, we had to look at the dogs’ breeds. We had to thoroughly look at each of the breeds, as many had typos. After careful consideration, we decided to remove any dog that had a breed whose total breed count was under 5. The following page shows the distributions of age, gender, and breed of the remaining dogs.

Variable Exploration

There are three variables of interest used in this project: age, gender, and breed. Each of these variables is categorical. Each dog's gender is labeled as M or F. The dogs' age falls into one of the following categories 0-1, 1-2, 2-5, 5-7, or 8+. Lastly, the dog's breed can fall into 32 different categories. See the Breed Distribution Graph for more details. Luckily none of the dogs had any missing information, so the percentage of missingness is 0%.

Data visualization figures are shown below:

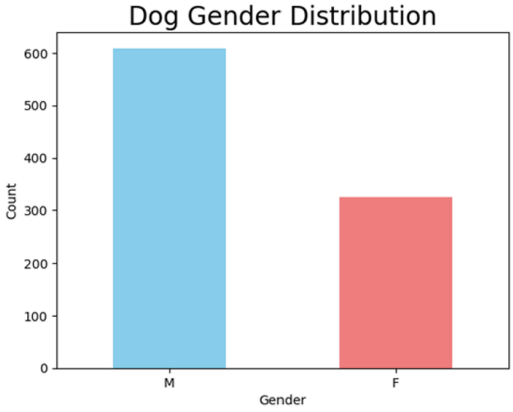

Figure 3: Distribution of the dog's gender

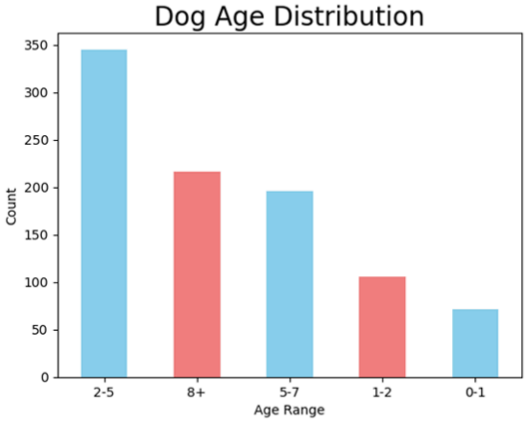

Figure 5: Distribution of the dog's age

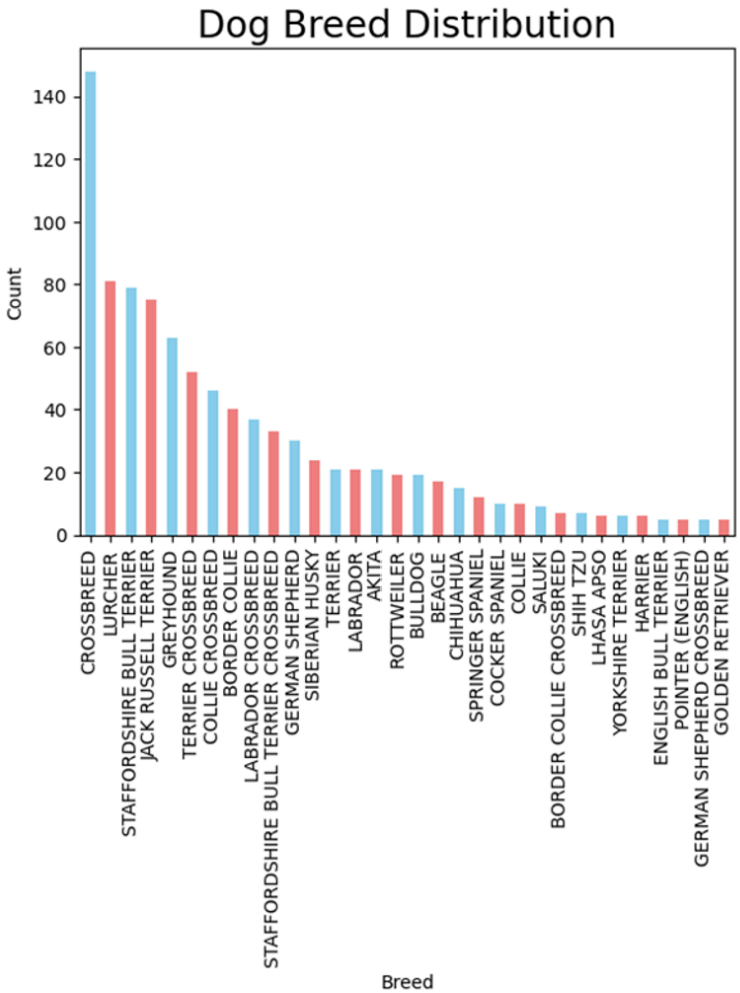

Figure 4: Distribution of the dog's breed

Data Preparation

Preprocessing Data

Before feeding the photos into the machine learning model, we first process them. The intermediate steps of the processing can be seen in Figure 6. First, the images are scaled to a uniform size of 256x256 pixels. This is done so all inputs are uniform in size for the CNN. Additionally, the size of 256x256 was chosen to reduce the complexity and time needed for the images to be processed while maintaining enough finer detail for the dogs to be distinguishable.

Next, the rescaled images are converted from color to black and white. We believed color would be less important than other characteristics, such as edges and contrast, in this problem. Furthermore, reducing the color values from three per pixel, RGB, to just one reduces the complexity of the data, thus reducing the complexity of the model and the computation time.

Finally, the black-and-white images are edge-enhanced and then normalized. Edge enhancement is performed to help restore some details that may have been lost during rescaling. Furthermore, it is done to improve feature extraction. By enhancing the edges, these features become more prominent, making it easier for the model to recognize patterns and distinguish between objects. Normalization is done to ensure all images have a similar range of intensity values. Leveling out intensities reduces the effects of uneven lighting within the photos while increasing the visibility of important dog features. This helps ensure the model is not overly sensitive to environmental effects but instead focuses on the dog's features.

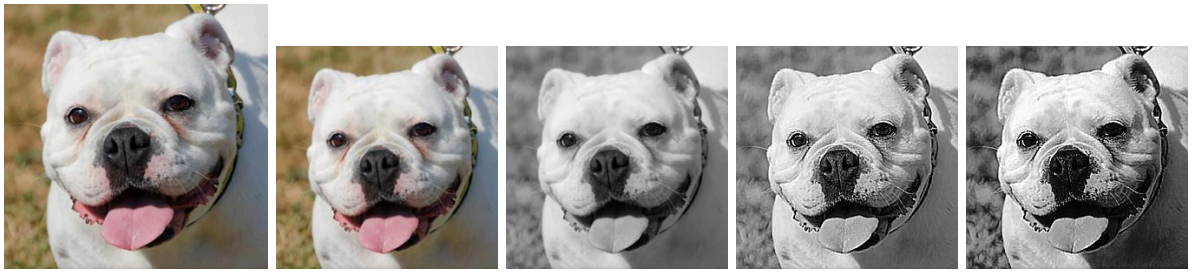

Figure 6: Preprocessing steps (left to right); Original, Resized, Black and White, Edge Enhanced, Normalized.

Background Blurring

Alternative preprocessing was attempted in the form of background blurring. We wanted to blur out the background to reduce the chance of our model focusing on the dog's environment rather than the dog itself. Several attempts were made at blurring the background with the main problem being identifying the background. Short of going through each of the 1,373 images manually, we would need an algorithm. We experimented with prebuilt models with limited success, so ultimately, we decided to create our own. To simplify the process, we opted for an algorithm rather than another machine learning model. The best algorithm we developed relied on the contrast of the photo.

The background detection algorithm would break the original image into its three color bands: red, blue, and green. These bands would then be tested to see which provided the highest contrast, or the highest standard deviation between intensity values. Next the selected color band would be used to create a mask. First, a Gaussian blur was applied then the band was thresholded to create a bit mask of the image. The threshold was chosen to be 127, halfway between 0 (black) and 255 (white), any pixel above this value was set to white, while anything below was set to black. Then the pixels within the middle of the image, a box half the size of the image, were counted. If there were more black pixels than white, the mask was inverted. This was done under the assumption that the dog’s face would be roughly within the center of the image and that black pixels would represent the background pixels. Next, a box blur is applied to the mask to smooth out the edges. This results in less blurring of eyes, noses, and mouths later on. Finally, a copy of the original image is blurred and then combined with the original according to the mask.

The results of this algorithm vary greatly, as it depends on the dog having a high contrast from its surroundings. Due to this, images may have no effect, a positive effect, or a negative effect. No effect is classified as the image before and after being practically identical. Positive effect is when the final image has appropriate background blur and negative effect is when the dog is blurred. Examples of this can be seen in Figure 7. Despite the results having varying accuracy, we ultimately decided to include these images in addition to the preprocessed ones to augment our dataset. This doubled the original size of 1,373, resulting in 2,746 images.

Original Background Removed

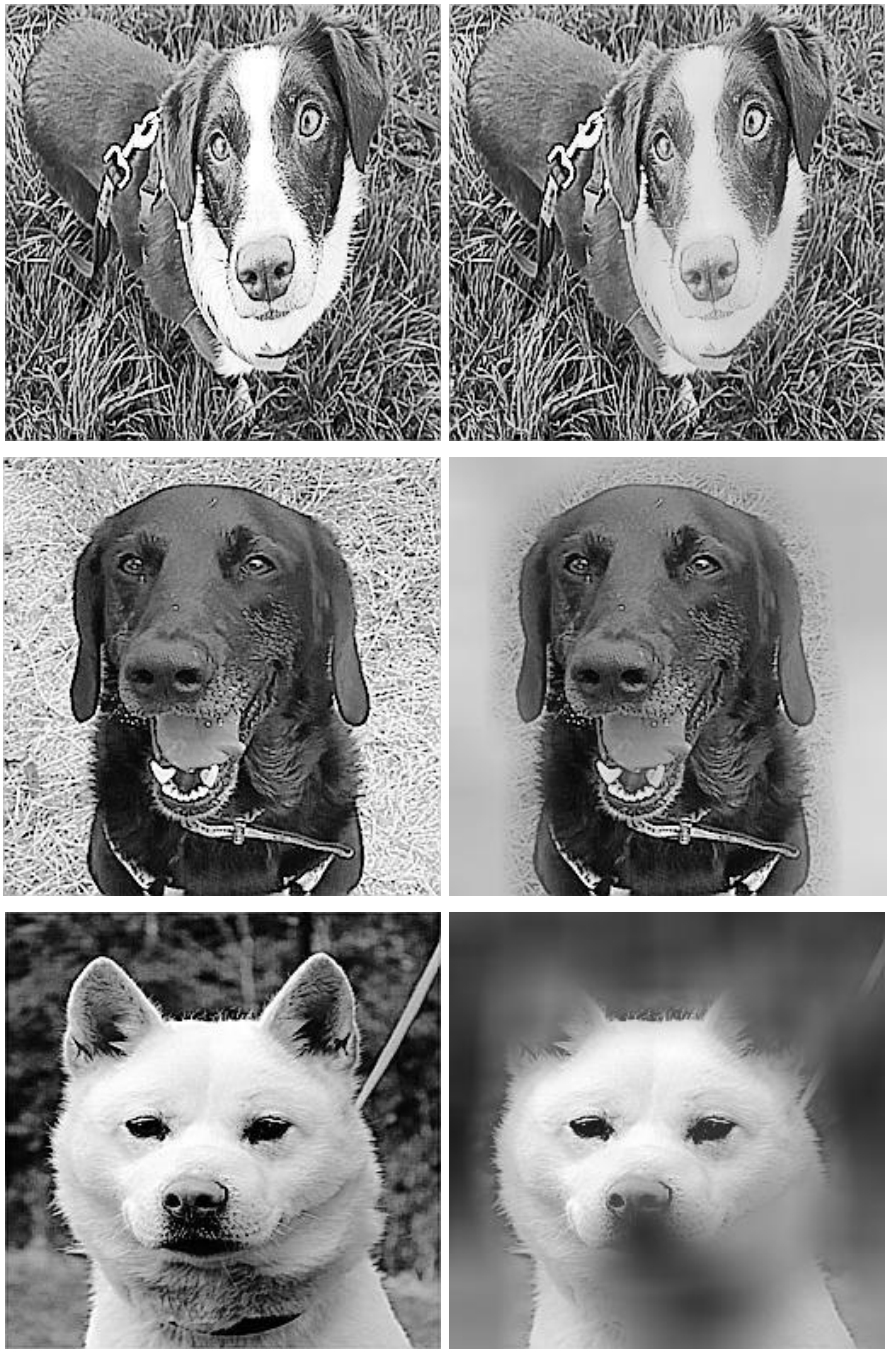

Figure 7: Background Blurring Examples (top to bottom); No Effect, Positive Effect, Negative Effect.

Image Augmentation

Considering our relatively small dataset, we chose to artificially expand it through image augmentation. This was done through transposing, rotating, and adding noise to each image in the dataset, examples can be seen below in Figure 8. As a result, the original dataset increased eightfold, going from 1,373 images to 10,984 images. Adding the background removal images gives a total of 12,357 images. In addition to enlarging our data set, including augmented images helps to reduce overfitting within the model and increases accuracy, leading to a robust and generalizable machine learning model.

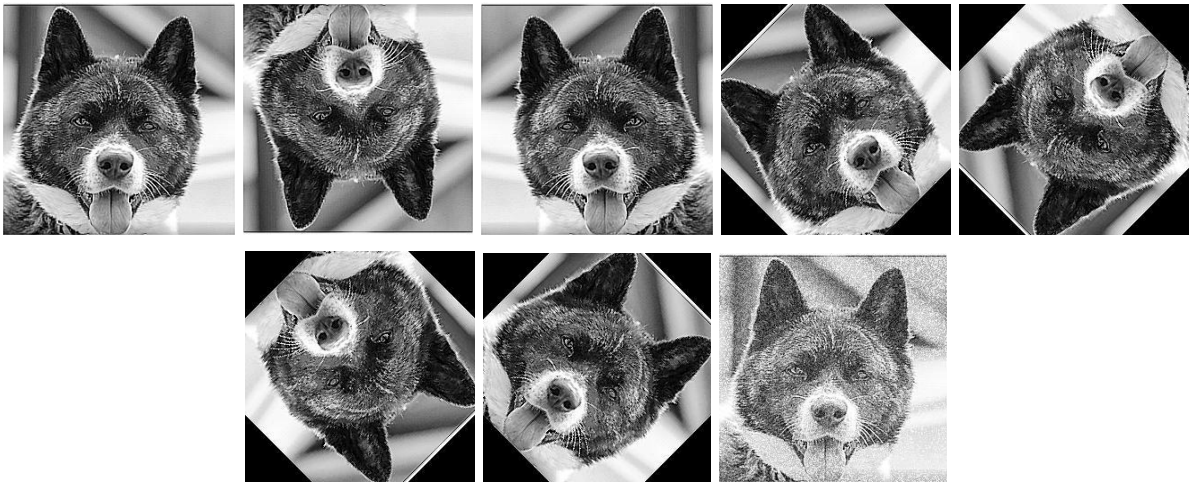

Figure 8: Augmented Image Types (left to right, top to bottom); Original, Flipped Vertically, Flipped Horizontally, Rotated 45°, Rotated 135°, Rotated 225°, Rotated 315°, Noise Infused.

Machine Learning Model

Model Choices

We decided to use three different CNN architectures for this project: VGGNet-16, VGGNet-19, and ZFNet. Both VGGNet-16 and VGGNet-19 were chosen because they are well-known for their strong feature extraction capabilities and compatibility with transfer learning, which allowed us to save time and computational resources. Given its historical success in image classification tasks, ZFNet was included as a baseline for comparison.

Initially, we also planned to integrate an SVM classifier to enhance classification accuracy by using the CNN output as features for the SVM. However, due to implementation challenges, we could not complete this step and had to set it aside for future work.

Tools and Libraries

We relied heavily on TensorFlow (Keras) for building, training, and managing the CNN models. Its built-in support for transfer learning and efficient data pipelines made it an ideal choice. We also used scikit-learn for tasks like splitting the dataset into training and validation sets and TensorFlow's tf.data API for managing the preprocessing pipeline.

Why These Choices Were Made

Data Constraints: Given the relatively small dataset (1,373 original images, expanded to 12,357 with augmentation), using models capable of generalizing with limited data was crucial.

Transfer Learning: Pre-trained models like VGGNet-16 provided a head start by leveraging knowledge from large-scale datasets like ImageNet, helping us perform better with less training.

Preprocessing Alignment: VGGNet-16's strength in recognizing edges and textures aligned well with our preprocessing steps, such as grayscale conversion and edge enhancement.

Challenges and Difficulties

Breed Classification Issues: Predicting dog breeds was particularly challenging due to class imbalance and the subtle visual differences between breeds, resulting in low accuracy (<1%).

SVM Integration: While we had high hopes for using an SVM as a secondary classifier, aligning its inputs with the CNN's outputs proved more complex than anticipated.

High-Dimensional Data: Preprocessing reduced but didn't eliminate computational demands.

Project Implementation

Training Process

To prepare the dataset for training, we followed several key steps:

Image Preprocessing:

- Resized all images to 256x256 pixels to standardize input size.
- Converted images to grayscale to reduce complexity while retaining important features.
- Enhanced edges to make key features more prominent.
- Normalized pixel values to ensure consistent intensity across images.
- Experimented with background blurring, yielding mixed results. Some images improved focus on the dog, while others blurred the dog itself.
- Used image augmentation (rotation, flipping, noise infusion) to expand the dataset to 12,357 images, improving model generalization and reducing overfitting.

Efficient Data Pipeline:

We leveraged TensorFlow's tf.data.Dataset API to handle data loading, preprocessing, batching, and caching. This optimized training performance by allowing for parallel processing.

Model Training

We built a CNN model using VGGNet-16, freezing pre-trained weights and adding custom layers for age, breed, and gender classification. It was compiled with the Adam optimizer and sparse categorical cross-entropy loss.

Training Setup:

- Training and validation sets were split 80-20 using scikit-learn.
- Batch size: 32.
- Epochs: 10, balancing computational efficiency with model convergence.

Hyperparameter Tuning

We experimented with various hyperparameters to fine-tune the model:

- Dropout rate: Set at 0.5 to reduce overfitting in the fully connected layers.
- Learning rate: Default settings for the Adam optimizer provided stable and reliable results.

Testing and Results

After training, we evaluated the model's performance:

- Age Prediction: Achieved 50% accuracy, significantly improving over the 32% reported in the DogAge study.
- Gender Prediction: Performed well with 74% accuracy.
- Breed Prediction: Accuracy was less than 1%, likely due to class imbalance and insufficient fine-grained breed-specific features.

While age and gender predictions were fairly successful, the model struggled with breed classification, highlighting a key area for improvement.

Challenges

Class Imbalance: The dataset included far fewer examples of some breeds, which affected the model's ability to generalize.

Preprocessing Limitations: Although edge enhancement improved some predictions, it didn't adequately capture the subtle differences needed for breed classification.

Future Improvements

- Implement a feature extractor pipeline to feed CNN features into an SVM classifier for more precise control over classification.
- Use advanced preprocessing techniques, such as segmentation, to better highlight breed-specific attributes.
- Balance the dataset by adding more images of underrepresented breeds, either through targeted augmentation or additional data collection.

Assessment

Enhanced Prediction

From the VGGNet-19-based model, we were able to significantly improve the accuracy of age prediction when compared to the neural network in the DogAge estimation study. This was due to our model being a CNN, which allowed us to achieve better efficiency and improved accuracy due to reduced connections and nodes compared to an NN, instead extracting features from the image and refining them as the network progresses.

Accuracy

Accuracy for the age prediction improved from 32% in the DogAge study to around 50%, which is a successful improvement of roughly 66%. Accuracy for the other classes, which did not have prior studies, returned mixed results. For the Breed prediction, the model failed to correctly predict any breeds with an accuracy of >1%. However, with the Gender prediction, the model was moderately successful, with a score of 74%, exceeding our expectations.

Improvements

One major improvement that could be made in the future, which was originally planned in the beginning, would be to reconfigure the base model CNN into a feature extractor. This would provide input for an SVM Classifier, refining the classification that the base CNN already possesses and allowing more control over the parameters related to classification.

Without this significant improvement, the product lacks reliability and should undergo further testing and production. Nevertheless, the results that the CNN model produces offer promising possibilities for what a refined SVM Classifier could add.

Solution Presentation

Code Environment

Our code requires an Anaconda environment with the following packages installed:

- TensorFlow
- Pandas
- sklearn
- NumPy
- Pillow

Data Set

Our dataset of 10,984 256x256 images is much too large to include in this report in its entirety, so it will be included with the code delivery.

Conclusion

Achieving the Objective

Our solution aimed to classify dog images by age, gender, and breed using a Convolutional Neural Network (CNN). While our VGGNet-19-based model improved age prediction accuracy compared to previous studies, it struggled with breed prediction due to class imbalance and subtle visual differences. Despite these challenges, our model provided a solid foundation for multi-label image classification, showing potential for improvement in efficiency and accuracy.

Credits

- Mohammed: Led the programming and implementation of the CNN architectures, ensuring the model was both robust and efficient.
- Connor: Focused on preprocessing the data to maximize efficiency and accuracy, refining the dataset to enhance the model's performance.
- Alex: Conducted research to find the most effective methods and technologies for our predictive model. He also attempted to build the SVM classification technique, helping guide our approach.
- Emme: Contributed to data visualization, helping refine the predictive model and understand its performance better.

Incorporation of Feedback

We did not receive much input after our final presentation, but the feedback we did get helped us pinpoint key areas for improvement. We worked on enhancing breed prediction accuracy by refining our data preprocessing techniques and adjusting our model architecture. These changes were crucial in overcoming some of our initial challenges.

Acknowledgment

We thank Professor Liaw for his guidance and feedback throughout this project. His insights helped shape our approach and drive our progress. We also acknowledge the resources provided by the DogAge dataset and Kaggle, which allowed us to explore and develop our solution effectively.

Sources

The DogAge dataset. (2020, May 28). Kaggle. https://www.kaggle.com/datasets/user164919/the-dogage-dataset

Mazurek, S., Wielgosz, M., Caputa, J., Frączek, R., Karwatowski, M., Grzeszczyk, J., Łukasik, D., Śmiech, A., Russek, P., Jamro, E., Dąbrowska-Boruch, A., Pietroń, M., Koryciak, S., & Wiatr, K. (2022). Canine age classification using Deep Learning as a step towards preventive medicine in animals. *Annals of Computer Science and Information Systems*, *30*, 169–172. https://doi.org/10.15439/2022f226

Patel, M. (2023, October 23). The complete guide to image preprocessing techniques in Python. *Medium*.https://medium.com/@maahip1304/the-complete-guide-to-image-preprocessing-techniques-in-python-dca30804550c

Rimmelasghar. (2023, March 28). *Getting Started with Image Preprocessing in Python*. Kaggle. https://www.kaggle.com/code/rimmelasghar/getting-started-with-image-preprocessing-in-python

Zamansky, A., Sinitca, A. M., Kaplun, D. I., Dutra, L. M. L., & Young, R. J. (2019). Automatic Estimation of Dog Age: The DogAge Dataset and Challenge. In *Lecture notes in computer science* (pp. 421–426). https://doi.org/10.1007/978-3-030-30508-6_34