A Comparative Analysis of Image Classification Algorithms for Dog Breed Recognition

1st Alexandre Almeida
Stevens Institute of Technology
Department of Computer Science
New Jersey, USA
aalmeid2@stevens.edu

2nd Elisa McGraw
Stevens Institute of Technology
Department of Computer Science
New Jersey, USA
emasiero@stevens.edu

3rd Justin Gallegos
Stevens Institute of Technology
Department of Mathematical Sciences
Colorado, USA
jgallego1@stevens.edu

***Abstract*—This research project focuses on developing an efficient image classification system to accurately recognize and classify diverse dog breeds. The study aims to implement three distinct algorithms, namely Convolutional Neural Network (CNN), K-Nearest Neighbors (KNN), and Random Forest, and subsequently compare their accuracy results. The dataset employed for evaluation is the Stanford Dogs Dataset, encompassing 20,580 high-resolution images distributed across 120 unique dog breeds.**

I. Introduction

Image classification has emerged as a pivotal area of research in computer vision, finding numerous applications in fields such as object recognition, medical imaging, and wildlife conservation. The specific focus of this study lies in dog breed recognition, a challenging task due to the visual complexity and diversity among various breeds. To address this, we propose the development and evaluation of three different algorithms: CNN, KNN, and Random Forest.

II. Related Work

Throughout the evolution of machine learning and artificial intelligence, identification and recognition have emerged as among the most prevalent applications. In China, a remarkable implementation of facial recognition for its citizens has streamlined the process of citizen identification without necessitating physical contact. This innovation played a crucial role in curbing the spread of COVID-19, achieving an impressive accuracy rate of approximately 95% for individuals wearing masks and a staggering 99.5% for those without masks [1].

In a study conducted by Babeş-Bolyai University, the same Stanford Dogs Dataset served as the foundation for dog breed identification. Leveraging the NASNet-A mobile architecture and Inception Resnet V2, they conducted a comparative analysis, yielding accuracy rates of 80.72% and 90.69%, respectively. These architectures, being sophisticated and efficient, were apt choices for the task at hand [2].

In contrast, our implementation adopts a more straightforward approach, focusing on comparing accuracy using different strategies and algorithms. Consequently, our accuracy falls notably short of Babeş-Bolyai's results. It's worth noting that China's extensive and experienced team has invested years in refining their proprietary algorithm, which excels in individual identification rather than generic breed classification.

III. Our Solution

To conduct a fair evaluation, we will split the Stanford Dogs Dataset into training, validation, and testing sets. The training set will be used to train the algorithms, while the validation set will assist in tuning hyperparameters. The final evaluation will be performed on the testing set to assess the accuracy of each model.

A. Evaluating the Dataset

We begin by loading the dataset so that we can perform some basic exploratory data analysis. The Stanford Dogs Dataset offers a comprehensive collection of 20,580 labeled images, featuring 120 distinct dog breeds. Each breed class contains an average of 171.5 images, ensuring sufficient representation for robust evaluation of the algorithms. The dataset's diversity should offer us enough depth to ensure the classification models are capable of being complex and robust.

The dataset was provided with a file structure and labeling system that is rather unintuitive. Inside of the "standford-dogs-dataset" folder our dataset is broken into its two main components. Namely, the "images" and "annotations" folder. Inside of each of these folders is another folder titled "Images" and "Annotations" respectively. If we follow the image path, inside of the Images folder, we observe that the dog breeds are separated by a unique labeling system. The first breed folder is titled "n02085620-Chihuahua", followed by "n02085782-Japanese_spaniel". Notice, there appears to be no order to the numbering assigned to various dog breeds. That is to say, they do not increase incrementally.

Given the messy organization of the dataset, we will opt to restructure the file set later during preprocessing.

B. Machine Learning Algorithms

The proposed approach involves the implementation of three image classification algorithms: CNN, KNN, and Random Forest.

3.1 Random Forest: Random Forest is an ensemble learning method that combines the predictions of multiple decision trees. We will construct a Random Forest model and investigate its performance in dog breed recognition, exploring the impact of various hyperparameters on accuracy.

The Random Forest Algorithm stands out as a powerful and effective solution for image classification tasks. For classifying dog breeds, the first crucial step is preprocessing the dataset. Achieving uniformity in the dataset is essential to ensure consistent and accurate results. By resizing all images to a standard size and converting them to grayscale, we set the foundation for further analysis. However, it is vital to retain color information for this particular task, as certain breeds have distinctive color patterns that play a significant role in their identification. For instance, the iconic black and white coat of a Dalmatian is an unmistakable characteristic that aids in recognizing this breed at a glance.

Feature extraction would then be used to identify the dog breed based on common traits. For example, an affenpinscher and afghan hound are dogs that are considered very furry and therefore the algorithm can remove the breeds such as the bloodhound or great dane as options. In order to make the algorithm "see" the dogs, the images will use local binary patterns (LPB) and color histograms.

To enable the algorithm to "see" and process dog images effectively, we employ two essential techniques: Local Binary Patterns (LBP) and Color Histograms. LBP, a widely used texture descriptor, operates by comparing the intensity of a central pixel with its surrounding neighbors. Particularly prominent in facial recognition and object detection tasks, LBP's ability to handle variations in lighting conditions is crucial for our extensive dog breed database. This adaptability allows the algorithm to navigate through various lighting scenarios, making predictions with enhanced precision. Moreover, the rotational invariance of LBP ensures that even if a dog image is slightly skewed or oriented differently, the algorithm can still accurately identify the breed based on its unique texture and pattern.

While LBP focuses on texture, Color Histograms provide a valuable representation of color intensities present in the image. By dividing the color space into categories or bins and counting pixel occurrences within each bin, we gain valuable insights into the color distribution of the image. This color information is particularly advantageous for identifying dog breeds, as many breeds possess distinct color patterns that are defining features. The color histogram's ability to capture and represent these color patterns further enhances the algorithm's classification accuracy.

The heart of the Random Forest Algorithm lies in its ability to make decisions based on random subsets of image traits. By selecting different subsets of traits for each decision tree, the algorithm creates a diverse ensemble, and the trees work collaboratively to make predictions. This randomness proves beneficial as it prevents the algorithm from getting trapped in patterns that may be caused by noise or irrelevant features in the database. The result is an algorithm that is robust and capable of generalizing well to new data.

However, this advantage comes with a trade-off. The random nature of the algorithm requires careful hyperparameter tuning to optimize accuracy. Striking the right balance between overtraining and undertraining is critical to ensure that the model is flexible enough to handle new inputs while maintaining high accuracy on the training data.

Moreover, the Random Forest Algorithm boasts efficiency, making it an attractive choice for datasets as extensive as ours. It can effectively handle large datasets with numerous features, making it particularly suitable for image data, where the feature space is often high-dimensional. Additionally, the algorithm's low variance ensures stable and reliable results, preventing overfitting and increasing its robustness in handling diverse data.

A final advantage of the Random Forest Algorithm is its ability to handle missing data. If an image is missing certain features, the algorithm can still make predictions based on the available information, contributing to the overall versatility and adaptability of the model.

In conclusion, the implementation of the Random Forest Algorithm for identifying dog breeds presents a compelling approach that capitalizes on the strengths of various techniques, such as feature extraction using LBP and Color Histograms. With its advantages in efficiency, low variance, and handling missing data, the algorithm proves to be a powerful tool for recognizing and classifying diverse dog breeds from our extensive and comprehensive database.

3.2 K-Nearest Neighbors (KNN): KNN is a non-parametric algorithm commonly employed in pattern recognition tasks. In this study, we will explore the application of KNN for dog breed classification, considering different distance metrics and K-values to identify the most appropriate configuration.

The KNN algorithm is a fundamental and straightforward classification technique in the field of machine learning. Its simplicity and effectiveness have made it a popular choice for various tasks, including image classification. KNN is categorized as a lazy learning algorithm, meaning it postpones the learning process until the prediction phase. Instead of learning a specific model during training, KNN stores the entire training dataset in memory and makes predictions on the fly during the testing phase. This characteristic makes KNN highly adaptable to new data and well-suited for applications with large and diverse datasets, such as the image classification problem we aim to solve - recognizing and classifying different dog breeds in the Stanford Dogs Dataset.

The Stanford Dogs Dataset contains a substantial number of images of 120 different dog breeds, making it a valuable resource for training and evaluating an image classification system. For this problem, KNN exhibits several characteristics that make it appropriate.

Images are inherently high-dimensional data with potentially thousands of dimensions (pixel values or extracted features). KNN can efficiently handle such data without requiring dimensionality reduction, which is advantageous for maintaining the integrity of the visual information.

KNN excels at capturing local patterns in the feature space. Dogs of different breeds might share certain global characteristics, but KNN can identify subtle differences specific to each breed. This ability is especially important when distinguishing between visually similar breeds.

KNN's implementation is relatively straightforward, making it accessible to practitioners and researchers without extensive machine learning expertise. This simplicity allows for quick experimentation and iteration in the early stages of the project.

Being a lazy learner, KNN does not build an explicit model during training, and it adapts readily to new data. This real-time adaptation is particularly advantageous when dealing with dynamic datasets or adding new dog breeds to the classification system.

KNN's decision-making process is transparent and interpretable. Since predictions are based on the majority class of the nearest neighbors, it is easy to understand why a particular classification was made.

KNN can serve as a baseline model for more complex algorithms like Support Vector Machines (SVMs) or Deep Neural Networks (DNNs). Comparing the performance of KNN with these models can provide insights into the potential benefits of adopting more sophisticated approaches.

In conclusion, the K-Nearest Neighbors (KNN) algorithm is well-suited for the image classification task of recognizing and classifying dog breeds using the Stanford Dogs Dataset. Its ability to handle high-dimensional data, capture local patterns, adapt to new data easily, interpretability, and simple implementation make it a reasonable choice for the initial stages of the project. However, as with any algorithm, proper data preprocessing, feature extraction, and hyperparameter tuning are critical to maximize KNN's performance and achieve accurate dog breed classification. Furthermore, the exploration of more advanced techniques, such as CNNs, could be considered for potential performance improvements as the project progresses.

3.3 Convolutional Neural Network (CNN): CNNs are mainly used for applications in speech and image recognition. Through the use of convolutional layers, image recognition models perform feature extraction by picking up on key patterns. This feature extraction is akin to dimentionality reduction, and since image data is typically large in dimension this approach proves to be a quintessential use case for a CNN. As such, our design will focus on a CNN architecture tailored for dog breed recognition.

CNNs leverage the spatial structure of images. They recognize that nearby pixels are more likely to be related to each other in the context of an object or pattern. By using pooling layers, CNNs downsample the spatial dimensions while preserving the essential features, enabling the model to focus on more relevant information and reduce computational complexity.

In conclusion, Convolutional Neural Networks make a good approach to solving the image classification problem posed by datasets like the Stanford Dogs Dataset due to their ability to leverage large amounts of data into key insights and characteristics that help classify objects in images.

C. Implementation Details

The initial stages of our project began with research. We believe that in order to successfully implement our algorithms to the task at hand, we needed to fully understand the scope of the problem and what attempts have already been made in regards to solving it. For the Convolutional Nerual Network approach, we found that a lot of previous work on the problem demonstrated an importance for pre-processing the data effectively. [1]

The preprocessing phase aimed to fix the issues identified in the intial analysis. After some consideration, the idea was to load all of the images into memory, reshape and convert them into numpy arrays, and then store them inside of an array. By preloading the images and converting them into arrays, we can ignore the poor file structure of the dataset and load the images into a format that our model will understand. We can also keep track of our labels as we read in photos from each dog breed.

We note that there are 20,580 photos. If we convert each photo to 200x200 pixels, then we will have 20,580 photos * (200*200*3) pixels/photo = 2,469,600,000 pixel values. We can estimate the size of this by simple conversion. If there are 32 bits/pixel * 2,469,600,000 pixels = 79,027,200,000 bits or approximately 9.8 GB. Since most modern computers have at least 16 GB of RAM, we believe this method should be viable and effective.

Model 1 - Convolutional Neural Network:

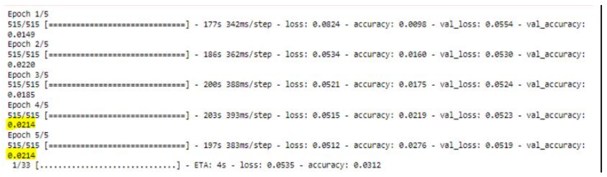
After the preprocessing step, we began building our models. The base architecture for the CNN model consisted of convolutional layers with ReLU activation and max-pooling, capturing the hierarchical features within the images. Dense layers followed, culminating in an output layer using sigmoid activation for initial breed prediction. After some trial and error, it was discovered that we needed to output using softmax activation, since we have a multiclass problem. After fixing the issues caused by an incorrect output layer, we finally had a functioning model. However, the initial model showed an accuracy of 1% on the training data. This suggested that our model was not learning from its training, and thus, we had more debugging to do.

Initial observations suggested that the model was not learning properly. While efforts were made to debug the initial model, it was ultimately decided that the initial code was a good learning point, but perhaps not good enough to be a starting point for our project. Unfortunately, some time on the CNN development was lost due to this decision, but reworking the model from the ground up did prove to be more successful.

The new CNN model was built with a stronger architecture designed for image classification. Each convolutional layer uses the ReLU activation function with He weight initialization, which are generally considered best practices. The model was fit using stochastic gradient descent, and we started with a conservative learning rate and momentum. As mentioned previously, we have a multiclass problem which requires a final prediction for 1 of the 120 breeds. As such, our model is designed with an output layer consisting of 120 nodes (one for each breed) and a softmax activation. Finally, the model was optimized using the categorical cross-entropy loss function.

Our next observations suggested that the model's training accuracy was growing rapidly, while the validation accuracy plateaued. This is a classic sign of overfitting, and these observations were noticeable from as early as our 5th epoch.

In an attempt to correct the overfitting problem, we implemented several strategies. We began by adding dropout layers and attempting to hypertune the dropout rate. Furthermore, we added batch normalization layers to also mitigate overfitting. Most of these attempts showed minimal improvements in our model's ability to learn. We introduced an early stopping



```
Epoch 1/5
515/515 [==============================] - 177s 342ms/step - loss: 0.0824 - accuracy: 0.0098 - val_loss: 0.0554 - val_accuracy:
0.0149
Epoch 2/5
515/515 [==============================] - 186s 362ms/step - loss: 0.0534 - accuracy: 0.0160 - val_loss: 0.0530 - val_accuracy:
0.0220
Epoch 3/5
515/515 [==============================] - 200s 388ms/step - loss: 0.0521 - accuracy: 0.0175 - val_loss: 0.0524 - val_accuracy:
0.0185
Epoch 4/5
515/515 [==============================] - 203s 393ms/step - loss: 0.0515 - accuracy: 0.0219 - val_loss: 0.0523 - val_accuracy:
0.0214
Epoch 5/5
515/515 [==============================] - 197s 383ms/step - loss: 0.0512 - accuracy: 0.0276 - val_loss: 0.0519 - val_accuracy:
0.0214
 1/33 [..............................] - ETA: 4s - loss: 0.0535 - accuracy: 0.0312
```

function to attempt to prevent overfitting in the first place; however, since our model was overfitting as early as the 5th epoch, this also did no significant impact on the model.

After trying to lock down a strong CNN model architecture, we decided that perhaps our image data needs to be reevaluated as well. We attempted image augmentation using ImageDataGenerator from the Keras library. Attempts here also showed little improvement in the overall accuracy of the model, again suggesting the problem lies with the model's architecture or the initial preprocessing.

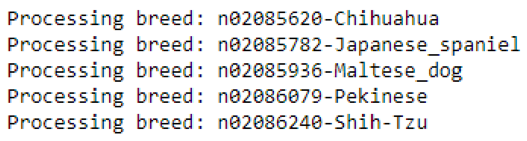
Our final CNN model lacks the robustness required to classify images under such a complex task. We settled on a model which scored a 6.6% on the unseen test data. With time permitting, we would attempt to rework our image preprocessing and potentially introduce transfer learning.

Model 2 - Random Forest:

The random forest algorithm, a cornerstone of machine learning, primarily finds application in classification and regression tasks. It amalgamates predictions from multiple decision trees to make output predictions based on given inputs. Each decision tree operates by splitting inputs based on the most salient feature, introducing randomness through the assignment of inputs to each tree in a random manner. This implies that some features may appear multiple times in a tree, while others may not appear at all [3].

In our implementation, achieving the highest possible accuracy entailed a process of trial and error. Our dataset featured 120 distinct dog breeds, each with over 150 images. To expedite parameter tuning, we initially tested our algorithm on just five breeds.

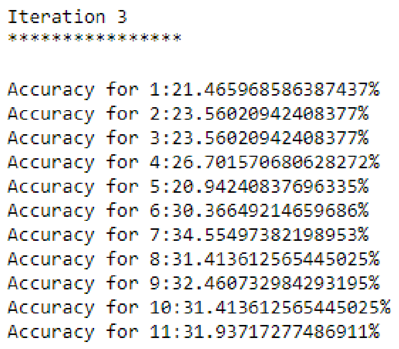
Before implementation, dataset preprocessing was essential. Initially, this entailed resizing each image to 200 by 200 pixels and using vectors. However, subsequent testing and research led us to convert this into a histogram of local binary patterns, enhancing the decision-making capacity of the random forest.



```
Processing breed: n02085620-Chihuahua
Processing breed: n02085782-Japanese_spaniel
Processing breed: n02085936-Maltese_dog
Processing breed: n02086079-Pekinese
Processing breed: n02086240-Shih-Tzu
```

Fine-tuning the random forest involved determining the optimal number of estimators to avoid overfitting or underfitting. We iterated through 10 different shuffles of the dataset, ultimately concluding that around 80 estimators produced the highest average accuracy among the iterations.

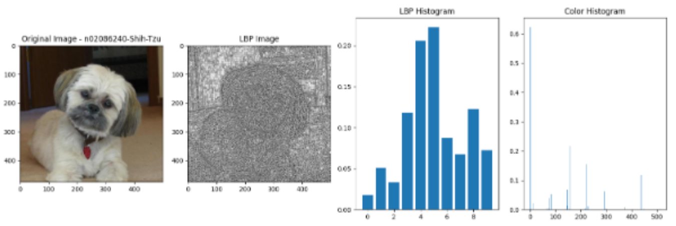
Upon assembling the code and running the algorithm, our model achieved an accuracy of approximately 10% for the dataset. While this accuracy may seem low, considering the



```
Iteration 3
****************

Accuracy for 1:21.465968586387437%
Accuracy for 2:23.56020942408377%
Accuracy for 3:23.56020942408377%
Accuracy for 4:26.701570680628272%
Accuracy for 5:20.94240837696335%
Accuracy for 6:30.36649214659686%
Accuracy for 7:34.55497382198953%
Accuracy for 8:31.41361256544502 5%
Accuracy for 9:32.460732984293195%
Accuracy for 10:31.41361256544502 5%
Accuracy for 11:31.93717277486911%
```

selection from 120 breeds, random chance alone would yield an expected accuracy of around 0.83%. There is substantial room for improvement and further development, with the potential to achieve accuracy rates as high as 99.5% and possibly even individual dog identification rather than breed classification with additional time and refinement.



Model 3 - K-Nearest Neighbors:

In this section, we delve into the intricacies of our dog breed classification model implemented using the K-Nearest Neighbors (KNN) algorithm. Our implementation is divided into two approaches: one employing a subset of 15 dog breeds and the other encompassing all 120 available breeds for training. We'll provide insights into the techniques used to achieve performance and share our experiences in addressing debugging processes.

The implementation initiates with the extraction of dog images from ZIP files, followed by meticulous organization into an appropriate directory structure. This initial step ensures that our data is prepared for further processing. Once our data is set, we load the images from the directory and embark on a comprehensive preprocessing journey. This includes resizing each image to a standardized 227x227-pixel size using the Python Imaging Library (PIL). We then transform these images into numerical arrays, facilitating machine learning analysis. Assigning labels to images based on their respective breeds completes this phase.

With our dataset prepared, we meticulously split it into training and testing sets. In line with common practices, we allocate 20% of the data to the testing set, reserving the rest for training. Our choice for the classification task is the K-Nearest Neighbors (KNN) algorithm. We don't merely employ KNN; we fine-tune critical hyperparameters such as the number of neighbors for optimal performance. Subsequently, we train our model using the training dataset. After the training phase, we employ the trained KNN model to predict labels for the testing data. Model accuracy is calculated by comparing these predictions with the true labels.

In both implementations, we've observed notable differences in accuracy. The initial 15-breed implementation achieved a commendable accuracy of 18.63%, while the comprehensive 120-breed approach using KNN yielded a somewhat reduced accuracy of 5.05%. This performance gap spurred an in-depth debugging process. Handling 120 breeds introduced a complexity spike compared to 15 breeds, demanding the model to distinguish between subtly similar breeds. Additionally, the dataset exhibited an imbalance in the number of images per breed, potentially hindering effective learning. The KNN algorithm's hyperparameters, notably the number of neighbors, required meticulous tuning to adapt to the challenges posed by the larger dataset. Furthermore, while KNN is a straightforward choice, we recognized the need to explore advanced deep learning models like convolutional neural networks (CNNs) to potentially unlock higher accuracy. It's vital to mention that the 120-breed approach notably extended the execution time due to the increased dataset size and complexity.

IV. Comparison

After thorough testing of our models, we observe that the Random Forest model exhibited the highest accuracy among the three algorithms. This suggests that the ensemble methods might be more suitable for our specific classification task. While we believe that the CNN's accuracy could be improved with more advanced architectures, we failed to find a suitable CNN model. Additionally, the KNN model may have a poorer performance when tested against all 120 breeds, since KNNs often struggle with high-dimensionality, which is what we have specifically when working with our image data. This is further supported by the fact that our algorithm had shown promise in classifying dog breeds, achieving an accuracy of 18.63% for a subset of 15 breeds. However, its performance significantly dropped to 5.05% when confronted with the more extensive task of classifying all 120 breeds. KNN excels in simplicity and ease of implementation, but it struggles with the increased complexity of distinguishing a larger variety of breeds. Data imbalance and the need for careful hyperparameter tuning are also prominent challenges in the KNN approach.

The relative performance of these algorithms underscores the intricacies of image classification tasks. While Random Forest exhibited the highest accuracy, continued experimentation and optimization are key to unlocking the full potential of CNN and other advanced models. Understanding the strengths and weaknesses of each algorithm helps guide further improvements and strategies for achieving higher accuracy.

V. Future Directions

The algorithm and current technology are far from reaching their full potential. Future advancements in the field and novel implementations hold the promise of elevating dog breed identification to even higher percentages. Inspired by Babeş-Bolyai University, adopting algorithms akin to ImageNet or NASNet-A, rather than developing a training algorithm from scratch, could prove fruitful for our project [2]. The concept of transfer learning was recently introduced in our course material and would be neat way to continue pushing our project forward.

The CNN model could improve from a few key additions. First off, better image preprocessing could lead to improved training accuracy. Exploring more advanced CNN architecture could also prove to be beneficial. Finally, as mentioned above, transfer learning could be implemented to give the model a solid foundation in which we would fine tune.

To enhance our random forest implementation, techniques such as fine-tuning tree depth, specifying minimum samples per leaf, and optimizing feature selection are pivotal. Increasing tree depth can enable the capture of more intricate patterns, particularly important for extracting crucial features such as a dog's face while minimizing the influence of background noise. Setting the minimum sample threshold aids in reducing both overfitting and underfitting concerns. Meanwhile, feature extraction can help identify and prioritize the most critical and distinctive features of each dog, thereby improving identification accuracy [4].

In light of the observed challenges and insights gained during the transition from a 15-breed to a 120-breed dog classification task using the K-Nearest Neighbors (KNN) algorithm, our future work will prioritize strategies to improve accuracy. The significant drop in accuracy when dealing with a larger number of breeds emphasizes the need for a more refined approach. We plan to tackle the complexity of distinguishing between similar-looking breeds by exploring techniques like data augmentation, which will enhance dataset diversity. Furthermore, we intend to leverage pre-trained models, such as ResNet and Inception, fine-tuned for our classification task. Ensemble methods, combining various models including KNN, and careful hyperparameter optimization for KNN will be essential to ensure our model is tailored to the dataset's intricacies. These strategies collectively aim to elevate the accuracy of our KNN-based dog breed classification model, contributing to the field's advancements.

In conclusion, our implementation of dog breed classification using the K-Nearest Neighbors (KNN) algorithm has provided valuable insights and challenges in the realm of image classification. Initially focusing on a subset of 15 breeds, we achieved an accuracy of 18.63%. However, when extending the task to encompass all 120 breeds, we encountered significant hurdles, leading to a reduced accuracy of 5.05%. This stark drop in accuracy highlighted the increased complexity of distinguishing between a larger variety of breeds, the presence of data imbalance, and the need for meticulous hyperparameter tuning for KNN.

Our future work will center on enhancing the performance of the KNN model for the more comprehensive 120-breed classification task. We plan to address these challenges by employing data augmentation techniques to diversify the dataset, utilizing transfer learning from pre-trained models, exploring ensemble methods, and conducting thorough hyperparameter optimization. These strategies collectively aim to improve the accuracy of our KNN-based dog breed classification model and provide a foundation for more robust and effective image classification in challenging scenarios.

VI. Conclusion

In summary, we have created three different multiclass image classifiers all trained in the task of dog breed recognition. The performance of each classifier, and its specified machine learning algorithm, vary quite a bit. We are far off from saying we have solved our problem or come close to reaching our algorithms maximum potential; however, we also understand the complexity of the problem chosen. Rather than opting for a binary classification task (e.g., is this a dog; yes or no?), we chose a multiclass problem. Furthermore, all of our models demonstrate the ability to classify dogs based on breed better than randomly guessing. As we would expect less than 1% accuracy for random guessing, considering there are 120 breeds to choose from.

With more time, our team believes we would be able to further refine our algorithms and show better improved accuracy scores. We understand that time was lost during several steps of our project. Namely, the initial CNN model was completely re-worked and our attempts at preprocessing the images caused us lost time. We believe that our project encapsulates the spirit of our objective and valuable insights were learned throughout.

References

[1] Pollard, M. *Even mask-wearers can be ID'd, China facial recognition firm says. U.S..*

[2] Raduly, Z., Sulyok, C., Vadaszi, Z., & Zolde, A. *Dog Breed Identification Using Deep Learning. Research Gate..*

[3] Géron, A. *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems. O'Reilly..*

[4] Koehrsen, W. *Hyperparameter tuning the random forest in Python - towards data science. Medium..*