DOG BREED CLASSIFIER: A COMPARISON OF CNN AND GPT-4O

Project How does a multimodal Large Language Model compare to a Convolutional Neural Network when classifying dog pictures into breeds?

Hannah Graham [1, 2,†]

1 Northwestern University

2 Code Repo: https://github.com/hannah-r-graham/DogBreedClassifier_whoDidItBetter.git

† Email: Hannahrg24@gmail.com

Abstract

Ever look at a dog and wonder what breed it is? If you could take a picture of the dog and be told its breed, would a Convolutional Neural Network do the job or would an expensive and cutting-edge technology like GPT-4o be necessary? Historically, Convolutional Neural Networks (CNNs) have been an integral tool for visual recognition and image categorization. With the rise of Large Language Models and their ability to intake images, this paper compares a classic Convolutional Neural Network and GPT-4o, a Large Language Model (LLM,) when classifying dog breeds based on an image. Two different prompts used by GPT-4o were used to understand results. On a sample size of 120 images, CNN had the highest accuracy rate of 92.5% while the highest scoring prompt using GPT-4o scored only 83.3%. The CNN model not only had a higher accuracy rate but is also faster to run on the data, is not costly to run, and is much more accessible to run since LLMs require such large compute resources.

Table of Contents

Introduction and Problem Statement

How do we teach a computer to determine a dog's breed just by intaking a picture of the dog? How do we know which model is the most accurate method for classification? This scenario will analyze pictures of various dogs and determine their breed. Classification will be attempted using a Convolutional Neural Network (CNN), a supervised classification model, as well as the Large Language Model GPT-4o, created and maintained by OpenAI.

CNNs were first invented in the 1980's for use with recognizing handwritten digits and later gained popularity for their success with the AlexNet model in 2012 (Kingler 2024). However, Large Language Models recently gained popularity in 2023, with the ability to intake text and images on March 14, 2023 OpenAI (2023). CNNs were made for classification and are a part of the Machine Learning category. GPT-4o is a generative model, attempting to understand user intent and predict the next phrase a person would want. These two technologies are very different in how they are built. This paper explores how well these two models can effectively classify dog breeds by image alone.

Literature Review

Convolutional Neural Networks have been in use for decades and show promising behavior when classifying dog breeds through images, with an accuracy up to 90% even in casual environments. Large language models are also now being employed to perform and assist in image classification. LLMs such as GPT-3 and GPT-4o are used to generate descriptions which are then fed into another model, or in some cases, generate not only a description but also a preliminary category for the image. Although CNNs have proved their worth, LLMs also show quite a bit of potential due to the flexibility in how they are applied to image classification scenarios. How do these two very different models compare with image classification?

Classifying the dog breed in an image via CNN is a popular venture. Subin Thomas (2023) and Tuan Nguyen (2019) performed two separate experiments to obtain the same goal: take images of dogs and classify their breed using a CNN. Thomas was much more successful with a 90% accuracy rate on the validation data set, while Nguyen achieved only an 81% accuracy rate. The difference in accuracy rate could be due to differences in shape, Nguyen chose 224 x 224 while Thomas chose 350 x 350. Both used epochs of 5 and different batch sizes of 32 (Thomas) and 20 (Nguyen). They both used different data sets. All these facts could contribute to the 10% difference in accuracy. Much of the CNN construction of this research leverages the work and code of Subin Thomas, much deserved acknowledgment of their work is recognized here.

While Large Language Models gained more popularity recently than CNNs, LLMs are already being leveraged to support image classification through gathering descriptions at scale, despite being a generative model and not a classification trained model. In a joint effort by University of Washington, Google DeepMind, and ML Collective, the team leveraged LLMs to create expansive descriptions of characteristics of image categories, then an open vocabulary model uses the description as prompts for the classification. These descriptions allowed the model to emphasize certain elements in the image to help determine a more accurate label, giving the model a more specific "direction" to pay attention to, so to speak, when determining an image class according to Pratt et al. (2023). An illustration explaining Pratt et al.'s work directly from their research can be found below in Figure 1. An illustration of Abdelhamed et al.'s interpretation of Pratt et al.'s work can be found in Figure 3.

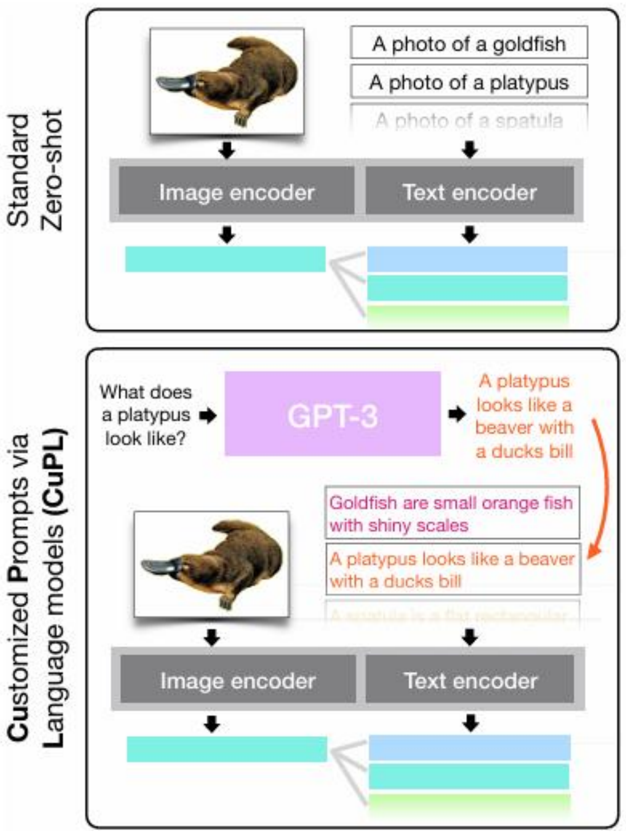

Standard Zero-shot
A photo of a goldfish
A photo of a platypus
A photo of a spatula
Image encoder
Text encoder
Customized Prompts via Language models (CuPL)
What does a platypus look like?
GPT-3
A platypus looks like a beaver with a ducks bill
Goldfish are small orange fish with shiny scales
A platypus looks like a beaver with a ducks bill
Image encoder
Text encoder

Figure 1: A comparison of standard zero shot learning and the work completed by Pratt et al. through Customized Prompts via Language Models (CuPL). Source: Pratt et al. 2023

Columbia University did similar work to Pratt et al. except with Vision-Language models, by using GPT-3 to create descriptions of certain categories and having the VLM check for those descriptors when classifying the image (Menon and Vondrick, 2022). Figure 2 displays how the team leveraged descriptions and weights for those descriptions then compared them with a common visual model CLIP.

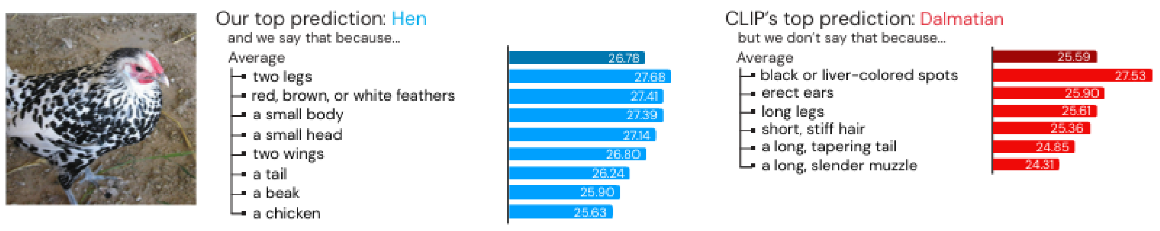


Figure 2: Visualization of Menon and Vondrick, 2022 work of using GPT3 generated descriptions to help with image classification. The figure compares Menon and Vondrick 2022's work with a visual model CLIP. Source: Menon and Vondrick, 2022.

Google Research has taken descriptions with LLMS a step further. They published a study of their findings using LLMs for zero-shot image classification in October 2024, only a month prior to this paper's creation. The team used a single set of prompts that can be used across various datasets to gather descriptions about an image and an initial suggestion of the class of the image. Then the image and the LLM description are then fed into a standard zero-shot linear image classifier. In other words, the LLM is used to create more data to help inform and support the zero-shot image classifier to make a more accurate prediction of the image Abdelhamed et al. (2024). A visualization of their methods as compared to CuPL Pratt et al. 2023 can be seen in Figure 3 below.

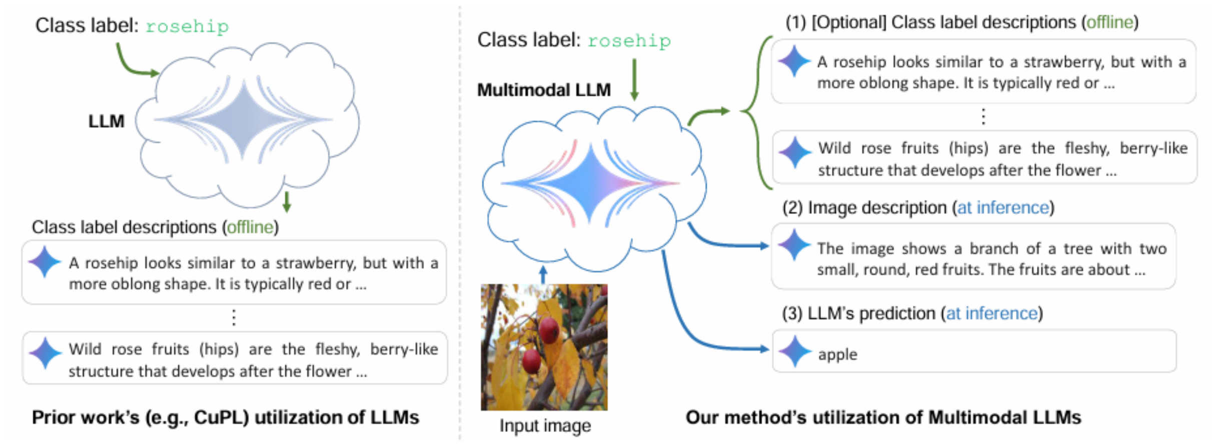

Class label: rosehip
LLM
Class label descriptions (offline)
A rosehip looks similar to a strawberry, but with a more oblong shape. It is typically red or ...
Wild rose fruits (hips) are the fleshy, berry-like structure that develops after the flower ...
Prior work's (e.g., CuPL) utilization of LLMs
Class label: rosehip
Multimodal LLM
Input image
(1) [Optional] Class label descriptions (offline)
A rosehip looks similar to a strawberry, but with a more oblong shape. It is typically red or ...
Wild rose fruits (hips) are the fleshy, berry-like structure that develops after the flower ...
(2) Image description (at inference)
The image shows a branch of a tree with two small, round, red fruits. The fruits are about ...
(3) LLM's prediction (at inference)
apple
Our method's utilization of Multimodal LLMs

Figure 3: A visual representation comparing Pratt et al.'s work and Abdelhamed et al.'s work through the lens of Abdelhamed et al.. The team leveraged LLM generated descriptions in addition to an LLM determined prediction on the item to classify the image where Pratt et al. only used descriptions from the LLM. Source: Abdelhamed et al. 2024.

Data

Source data is from Kaggle, a well-known data science competition website. This data set was originally used in a classification competition in 2018. The data contains 120 breeds of dogs and 10,357 different images of dogs in jpeg format. Files include: one csv that has an image ID and the dog breed label for that image. The second file contains JPEG images of all dogs and the name of the image is the image ID correlating to the first file with breed labels.

Each breed has an average of 85.18 images. The Scottish Deerhound has the most images in the dataset at 126. The Briard has the least at 66.

count	120.00 breeds
mean	85.18 images per breed
std	13.29 images per breed
min	66.00 images per breed

25%	75.00 images per breed
50%	82.00 images per breed
75%	91.25 images per breed
max	126.00 images per breed

Table 1: Count of images per dog breed from Kaggle dataset. These images will be used to train, test, and validate the CNN model during development.

Min width	97 pixels
Max width	3,264 pixels
Min height	103 pixels
Max height	256 pixels

Table 2: Image sizes for the dataset that the CNN will be trained, tested, and validated on.

Methods

Overall process to be expanded in further detail:

1. Develop the CNN:
    a. Pre-process data for CNN to intake
    b. Train the CNN model, evaluate, then save the model.
2. Choose one image per breed, 120 images total to compare the CNN and GPT-4o on.
3. Run the CNN on the 120 images.
4. Using GPT-4o, run prompts one and two on the 120 images.
5. Compare the results from the CNN model and the results from GPT-4o.

Develop the CNN:

Prepare the data for the CNN model:

Using Pandas and the CV2 package, the labels dataset and the images were accessed. For each image, the size of the image was stored then the min and max pixel size for each image was printed. Given the data, images were all resized to 350x350 pixels before being ingested by the CNN and LLM. The images were split into training, test, and validation groups with an 80-10-10 split with a random state value of 42 using the SKLearn package.

Build the model:

The Convolutional Neural Network (CNN) was constructed using the Xception model pre-trained on ImageNet as the base model. The model architecture included a global average pooling layer, a dropout layer with a rate of 0.7, and a dense output layer with softmax activation for classification into the number of unique dog breeds.

Two callbacks were employed during training: EarlyStopping and ModelCheckpoint. EarlyStopping monitored the validation loss and stopped training if it did not improve for four consecutive epochs. ModelCheckpoint saved the model with the best validation loss.

Model training and evaluation:

The model was trained using the fit method with data generators for training, validation, and test sets. The training process included 5 epochs with a batch size of 32. The model's performance was evaluated on the test set, and the results of the model running on the test set can be found below in the appendix section.

The model was saved and ran against 120 images that the LLM will also use. The results of this run and a comparison with the LLM results on the 120 images can be found in the results section of this paper.

LLM methods:

Choosing a Large Language Model:

GPT-4o is OpenAI’s largest and one of their most recent models and therefore the chosen model for this research. Access to this model is through Azure OpenAI subscription, then through API Key and Endpoint accessed by the code through a .env file

Prepare the data:

For cost reasons, only 120 images were inputted into the LLM, the same 120 images as inputted into the CNN model after it was trained. These 120 images were resized to 350x350 pixels, like the CNN model. Using the base64 package, the JPEG images were to base64, ‘ASCII’. UTF-8 was first attempted instead of ASCII, however the token count was 100x more for UTF-8. The Azure AI playground uses ASCII for images, so doing this conversion ahead of time reduced the token count and prevented the code from hitting the current rate and quota limits for Azure OpenAI API.

Determine prompts to try:

Using 2-3 sample images within the Azure OpenAI for testing prompts, one prompt seemed like a good starting point that was the basis for prompts one and two below.

1. Prompt one: Ask GPT-4o to identify the dog breed

   Determine which dog breed is in the image for each image_id. BreedName should be 3 words or less. Your response should be in Json format. And should look like: image_filename, BreedName. Example output: {"image_filename": "filename.jpg", "BreedName": "affenpinscher". {base64_image}}

2. Prompt two: Ask GPT for dog breed and insert list of dog breeds into prompt:

   Determine which dog breed is in the image for each image_id. BreedName should be 3 words or less. Breed names consist of: 'list of 120 breeds' . Your response should be in json format. And should look like: image_filename, BreedName. Example output: {"image_filename": "filename.jpg", "BreedName": "affenpinscher". {base64_image}}

Send images to GPT-4o using Azure OpenAI API:

Using a function with a for loop, each image, image name (also its ID), context, and prompt were sent to GPT-4o along with the following parameters: temperature = 0 (limit creativity of model's response), top_p = 1 (ignore top-p sampling), max tokens = 100 (since dog breeds tend to be short). Each image corresponds to one API call. The standard tier GPT-4o model has sensitive rate limits so the function also requires a sleep time of 20 seconds between API calls. The responses from the API were parsed using the JSON Python package. The results were then saved to a Pandas DataFrame and CSV. Each prompt has its own csv output.

Post processing API response:

The labels dataset included "_" and "-" for the true breed category for each image. The responses from GPT for both prompts and the labels dataset needed the "_" and "-" removed so the script could accurately compare results between the models.

Results

Model/Method	% correctly classified	% incorrectly classified	Time to run
CNN	92.5%	7.5%	15 sec
Prompt One	55.8%	44.2%	41 min
Prompt Two	83.3%	16.7%	41 min

Table 3: Comparing CNN and LLM accuracy on 120 images, representing each dog breed.

Interestingly, there were six images/breeds that were misclassified by the CNN model and both GPT-4o prompts. This will take more investigating and testing, but the common failure rate could mean multiple things. Those images could have been worse quality or had more distracting items in the image. The breeds themselves could be difficult to distinguish from other breeds that are similar in likeness. More analysis and testing is needed, but it is a great signal to determine future work.

The 6 dog breeds that were misclassified by all 3 models:

1. Australian terrier
2. toy terrier
3. eskimo dog
4. walker hound
5. brittany spaniel
6. shetland sheepdog

Discussion

CNN had the highest degree of accuracy out of all three models. The best scoring LLM model had a 9.2% lower accuracy than the CNN model. The CNN also took a fraction of the time to run compared to the LLM prompts since it was able to run locally, and did not require a time delay in the code due to the current quota limitations on the GPT-4o API. Running the LLMs is also a much more expensive venture. Running the 120 images on each LLM prompt costs roughly $2.50. To run on all 10,000+ images it would cost over $700 at the time of writing this paper which is out of budget for this study. The CNN does not cost beyond electricity to run on both the 120 images and the 10,000+ from the original dataset.

It is important to note that prompt one did considerably worse than the others due to data structure. While prompt two was given breed names to choose from explicitly, prompt 1 had to interpret and format the breed name. This leads to prompt one having many cases of being mostly right but added "terrier" to the end of the breed or not including a space. A great example is for image ID: 07192213791150248bfb5bbe6b0b0373. The image is labelled as "basset" by the source data, but prompt one classified it as "basset hound". This makes it difficult to assess accuracy, especially as the number of images increases far beyond 120.

Prompt two excelled at classification compared to prompt one. The accuracy improved 27.5% by simply giving the list of dog breed classes to the prompt. However, prompt two was still much less accurate than the CNN at only 83% accuracy as compared to CNN's 92.5%.

Out of the 120 dog images, the CNN incorrectly labelled 9 breeds, leading to an accuracy of 92.5%. Which aligns with the test result accuracy of 91.9% measured during development displayed in Appendix B. Test result accuracy by epoch can also be found in Appendix B.

The incorrectly classified breeds by the CNN include Australian terrier, toy terrier, eskimo dog, wire haired fox terrier, walker hound, lhasa, brittany spaniel, shetland sheepdog, great dane. Those same 9 breeds were also miscategorized by the first prompt by the LLM. The second prompt, however, only miscategorized 6 out of the 9 the CNN did.

Challenges and Important Improvements:

The biggest challenges for this research came in two categories: data structure/test and access to Azure OpenAI's models. The labelled data was not processed as heavily as it should have been before plugging into the models. Special characters such as "_" and "-" should be removed right away for a more accurate comparison of the LLM output and the target output.

While it is quick to set up an Azure OpenAI service and connect through an API key, quota limitations for tokens and API access were hit quickly. This research was split into several days when interacting with the GPT-4o API because as soon as the quota is hit for the endpoint, the endpoint cannot be reached for another 24 hours. As discussed above, even when working with the endpoint, working with GPT-4o is extremely expensive at $2.50 for every prompt run on 120 images. GPT-4o is the most expensive model operated by OpenAI so one improvement could be using Small Language Models or an older model to compare with the CNN.

Conclusions

The Convolutional Neural Network outperformed the best GPT-4o prompt in every metric including accuracy, speed, availability/access, and cost. While the sample size was small at only 120 images, the CNN still had a similar accuracy rate of 92% when run on 10k+ images. GPT-4o could not be run on all 10k images due to being cost prohibitive. When using a picture to understand what breed a dog is, the Convolutional Neural Network is clearly the best model for this job.

Directions for Future Work

Both the CNN model accuracy and prompt two used with GPT-4o can continuously be improved. Accuracy rates in the 80s and 90s for this research scope are great, but a far cry from where they could be with more iterations on the models and prompts. In addition, an analysis and further testing on the six dog breeds that every model missed is a great place to start. 120 images is also a very small sample size and if given more funding, more images could be run through both models and evaluated.

This research operated under the assumption that all pictures were purebred dogs and not mixed breeds. In the real world, it is uncommon for a dog to be solely one breed so a future direction in this work could be a “breed” recipe predictor, where each image is given a percentage of a breed represented and each image could have one breed or many breeds in it.

Acknowledgements

Data Availability

Data results of this study can be found on GitHub Repo here: https://github.com/hannah-r-graham/DogBreedClassifier_whoDidItBetter.git

Data for original images can be found here: Dog Breed Identification | Kaggle

Code Availability

Code can be found on GitHub Repo: https://github.com/hannah-r-graham/DogBreedClassifier_whoDidItBetter.git

Includes:

- CNN Jupyter notebook
- LLM Jupyter notebook
- Analysis notebook

References

Abdelhamed, Abdelrahman, Mahmoud Afif, Alec Go. 2024. “What do you see? Enhancing zero-shot image classification with multimodal large language models”. *arxiv.org.* (Accessed November 3, 2024; available online at https://arxiv.org/pdf/2405.15668).

Kingler, Nico. 2024. “Convolutional Neural Networks (CNNs): A 2025 Deep Dive”. *Visio.ai.* (Accessed November 1, 2024; available at https://viso.ai/deep-learning/convolutional-neural-networks/).

Menon, Sachit, Carl Vondrick. 2022. “Visual classification via description from large language models”. *arxiv.org.* (Accessed November 1, 2024: available at https://arxiv.org/pdf/2210.07183).

Nguyen, Tuan. 2019. “Build Your First Computer Vision Project — Dog Breed Classification”. *Medium.com.* (Accessed November 1, 2024; available at https://towardsdatascience.com/build-your-first-computer-vision-project-dog-breed-classification-a622d8fc691e).

OpenAI. 2024. “GPT-4”. (Accessed November 1, 2024; available at *https://openai.com/index/gpt-4-research/).*

Pratt, Sarah, Ian Covert, Rosanne Liu, Ali Farhadi. 2023. “What does a platypus look like? Generating customized prompts for zero-shot image classification”. *arxiv.org.* (Accessed November 1, 2024; available at https://arxiv.org/pdf/2209.03320).

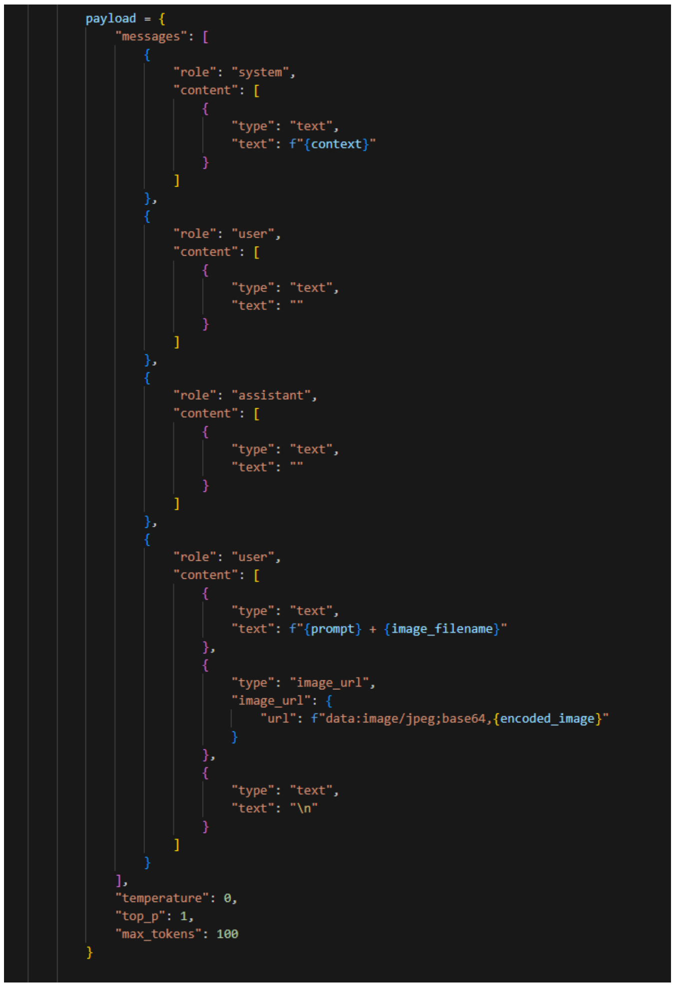
Thomas, Subin. 2023. “Deploying a Dog Breed Classification ML Model: An End-to-End Guide”. *Medium.com.* (Accessed November 1, 2024; available at https://medium.com/@subin60/deploying-a-dog-breed-classification-ml-model-an-end-to-end-guide-fc7c025e13a2).

Cukierski, Will. 2017. “Dog Breed Identification”. *Medium.com.* (Accessed November 1, 2024; available at https://kaggle.com/competitions/dog-breed-identification).

Appendix A

Prompts utilized with query formatting:

Prompt 1:

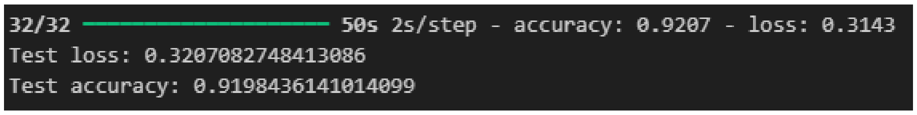
- context = '''Determine which dog breed is in the image for each image_id. BreedName should be 3 words or less. '''
- prompt = '''your response should be in json format. And should look like: image_filename, BreedName. Example output: "image_filename": "filename.jpg", "BreedName": "affenpinscher"''
- 

```
payload = {
    "messages": [
        {
            "role": "system",
            "content": [
                {
                    "type": "text",
                    "text": f"{context}"
                }
            ]
        },
        {
            "role": "user",
            "content": [
                {
                    "type": "text",
                    "text": ""
                }
            ]
        },
        {
            "role": "assistant",
            "content": [
                {
                    "type": "text",
                    "text": ""
                }
            ]
        },
        {
            "role": "user",
            "content": [
                {
                    "type": "text",
                    "text": f"{prompt} + {image_filename}"
                },
                {
                    "type": "image_url",
                    "image_url": {
                        "url": f"data:image/jpeg;base64,{encoded_image}"
                    }
                },
                {
                    "type": "text",
                    "text": "\n"
                }
            ]
        }
    ],
    "temperature": 0,
    "top_p": 1,
    "max_tokens": 100
}
```

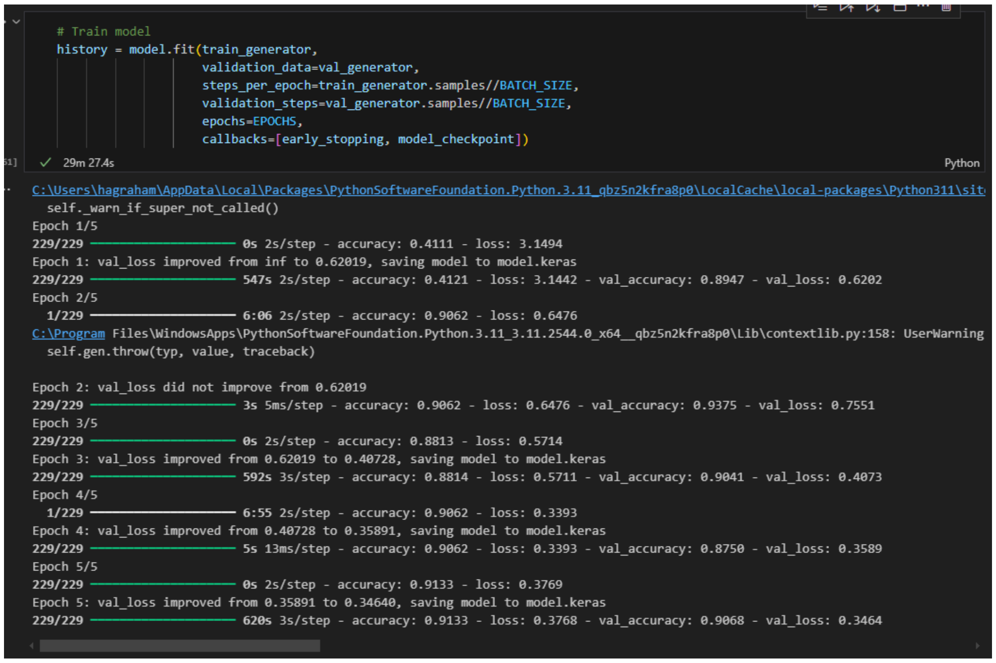
Prompt 2 = prompt 1 + list of dog breeds found in labels.csv

Appendix B

Additional information on the accuracy of the CNN model:



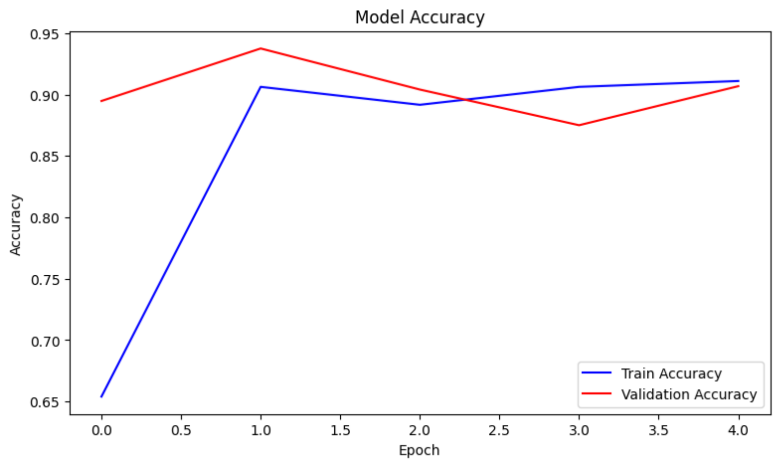
```
32/32 ━━━━━━━━━━━━━━━━━━━━ 50s 2s/step - accuracy: 0.9207 - loss: 0.3143
Test loss: 0.3207082748413086
Test accuracy: 0.9198436141014099
```

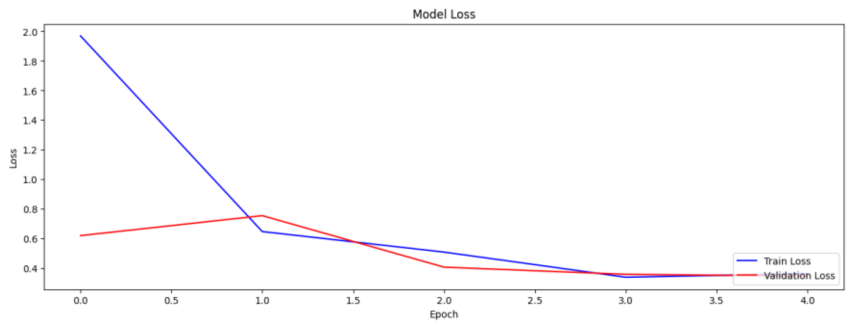


```python
# Train model
history = model.fit(train_generator,
                    validation_data=val_generator,
                    steps_per_epoch=train_generator.samples//BATCH_SIZE,
                    validation_steps=val_generator.samples//BATCH_SIZE,
                    epochs=EPOCHS,
                    callbacks=[early_stopping, model_checkpoint])
```

```
C:\Users\hagraham\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\LocalCache\local-packages\Python311\sit
  self._warn_if_super_not_called()
Epoch 1/5
229/229 ━━━━━━━━━━━━━━━━━━━━ 0s 2s/step - accuracy: 0.4111 - loss: 3.1494
Epoch 1: val_loss improved from inf to 0.62019, saving model to model.keras
229/229 ━━━━━━━━━━━━━━━━━━━━ 547s 2s/step - accuracy: 0.4121 - loss: 3.1442 - val_accuracy: 0.8947 - val_loss: 0.6202
Epoch 2/5
  1/229 ━━━━━━━━━━━━━━━━━━━━ 6:06 2s/step - accuracy: 0.9062 - loss: 0.6476
C:\Program Files\WindowsApps\PythonSoftwareFoundation.Python.3.11_3.11.2544.0_x64__qbz5n2kfra8p0\Lib\contextlib.py:158: UserWarning
  self.gen.throw(typ, value, traceback)

Epoch 2: val_loss did not improve from 0.62019
229/229 ━━━━━━━━━━━━━━━━━━━━ 3s 5ms/step - accuracy: 0.9062 - loss: 0.6476 - val_accuracy: 0.9375 - val_loss: 0.7551
Epoch 3/5
229/229 ━━━━━━━━━━━━━━━━━━━━ 0s 2s/step - accuracy: 0.8813 - loss: 0.5714
Epoch 3: val_loss improved from 0.62019 to 0.40728, saving model to model.keras
229/229 ━━━━━━━━━━━━━━━━━━━━ 592s 3s/step - accuracy: 0.8814 - loss: 0.5711 - val_accuracy: 0.9041 - val_loss: 0.4073
Epoch 4/5
  1/229 ━━━━━━━━━━━━━━━━━━━━ 6:55 2s/step - accuracy: 0.9062 - loss: 0.3393
Epoch 4: val_loss improved from 0.40728 to 0.35891, saving model to model.keras
229/229 ━━━━━━━━━━━━━━━━━━━━ 5s 13ms/step - accuracy: 0.9062 - loss: 0.3393 - val_accuracy: 0.8750 - val_loss: 0.3589
Epoch 5/5
229/229 ━━━━━━━━━━━━━━━━━━━━ 0s 2s/step - accuracy: 0.9133 - loss: 0.3769
Epoch 5: val_loss improved from 0.35891 to 0.34640, saving model to model.keras
229/229 ━━━━━━━━━━━━━━━━━━━━ 620s 3s/step - accuracy: 0.9133 - loss: 0.3768 - val_accuracy: 0.9068 - val_loss: 0.3464
```




Model Loss
Loss
Epoch
Train Loss
Validation Loss
2.0
1.8
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0