Project Report: Dog Breed Identification

Team
Lyn Wang (fulingw@berkeley.edu), Jenny Park (jennycpark@berkeley.edu),
Ryan Castillo (ryan_castillo@berkeley.edu), Rodney Tang (rtang920@berkeley.edu)
Github Repo: https://github.com/jennycpark-ucb/mids-w207-foursigma

Abstract

Dog breed classification has practical relevance beyond casual interest, aiding in veterinary diagnostics and shelter placements. Yet even experts struggle to differentiate visually similar breeds, prompting us to explore whether machine learning can assist. We evaluated a series of models of increasing complexity — logistic regression, fully connected neural networks, convolutional neural networks (CNNs), and transfer learning. Performance improved sharply with model sophistication: logistic regression failed at the task, while CNNs captured spatial patterns, yielding better F1 score. Transfer learning dramatically outperformed all prior approaches, achieving a 0.31 F1-score with tightly clustered train, validation, and test accuracy (~31%), indicating strong generalization. Key insights include the importance of using the right complexity — CNNs for spatial hierarchies and pre-trained models for fine-grained visual distinctions. Overall, the classification task proved difficult for untrained models due to limited training images (~100 per class) and the large label space (120 breeds). To improve future performance, we recommend reducing label granularity and enriching image diversity through external datasets.

Introduction

Identifying dog breeds isn't just a matter of curiosity for us dog lovers, but also has practical importance for shelters and veterinarians. Animal shelters may need to identify dog breeds for appropriate care or match them to adoption preferences, whilst vets may need to anticipate breed-specific health conditions. However, even experts can struggle to distinguish between visually similar breeds, such as the Belgian Malinois and German Shepard. In this project, we attempt to apply a machine learning approach to this problem, with the overall goal of identifying dog breeds from our dataset. The input to our algorithm is an image of a dog, and the output is a predicted label of a dog breed. We explore and compare multiple models for this task, including logistic regression, fully connected neural networks, convolutional neural networks (CNNs), and transfer learning with pretrained models. Our objective is to evaluate how well each model performs and to understand which approaches are best suited to this complex visual recognition task.

Related Work

One of the earliest works in this space is from Khosla et al., who introduced the Stanford Dogs dataset. They used a traditional approach with deformable part models and bounding box annotations to try to localize key dog features like ears or snouts. While this method was a good starting point, it relied heavily on manually crafted features. In more recent years, deep learning has taken over. Ráduly et al. applied convolutional neural networks (CNNs) and transfer learning, getting much better results than traditional methods. They were able to demonstrate how pretrained models can save time and improve accuracy even when training data is limited. However, at the same time they began running into more issues with overfitting as we saw too. Recently there's been a lot of excitement around transformer-based models like Vision Transformers (ViTs), which treat images more like sequences of patches. ViTs pretrained on huge datasets can outperform CNNs on fine-grained classification tasks, but need a lot more compute and data

to work well. While we didn't explore ViTs here, they're a promising next step if we had more time and resources. Our project builds on all of this by comparing a range of models from logistic regression to CNNs and transfer learning, all on the same dataset. Instead of going straight to the most complex model, we wanted to understand how each approach performs and where the trade-offs are.

Dataset

For this analysis, we used the Stanford Dogs Dataset, a collection of 20,580 images of size 64x64px spanning 120 dog breeds from around the world. This dataset was built using images and annotations from ImageNet and was pre-split into 12,000 training and 8,580 test examples. The training set contained a perfectly even class distribution, with 100 examples per breed. The test set, however, contained between 50 and 150 examples per breed. Due to this variation, we performed a stratified random split on the test set to create a validation set that was half its size. After the split, both the validation and remaining test sets contained 4,290 examples.

To prepare the data for effective model training, we applied a series of preprocessing and augmentation steps.. Since many images contained non-dog elements, we used the provided annotation data to mask out all non-dog regions in the 20,580 images—reducing visual noise and directing the model's attention to the relevant features. To further improve generalization, we initially set out to create a single, augmented dataset to use across all models. However, through experimentation we found that different models responded quite differently to various augmentation applications. As a result, we ended up using different augmentation techniques for each model. Across models, the augmentation included varying degrees of random horizontal flips, rotations, and/or adjustments to zoom, contrast, and brightness. Transformations were used either to augment the dataset by increasing its size or simply to enhance variability without changing the total number of examples. Additionally, we created a 10-breed subset of the training and validation data to start small and iteratively refine our models. 10 breeds were randomly selected to form this subset. No separate subsetted test set was generated, as our initial goal was not to compare test performance between datasets, but to support early experimentation and model development.

Methods

Multiclass Logistic Regression

A multiclass logistic regression model takes a linear combination of input variables and outputs probabilities for each target class using a softmax function. While this approach is not well-suited for capturing the spatial structure of image data, we chose it as an initial baseline to benchmark future neural network models. Due to the high number of classes and their fairly balanced distribution, majority class and random guessing baselines are quite low (1.77% and 0.68% accuracies, respectively). Therefore, we aimed to use the performance of our logistic regression model (primarily the produced F1 score) to establish a more meaningful baseline for evaluating models that are better equipped to learn from image data.

Fully Connected Neural Network

As a second improvement, we implemented a fully connected neural network in TensorFlow to better capture the complex patterns in our image data. While logistic regression tries to draw a straight line to separate classes, it struggles when the data has non-linear relationships — which is often the case with images. Like before, we flatten the 64×64×3 image into a long vector of pixel values. However, instead of connecting that directly to the output, we pass it through one or more hidden layers. Each neuron in a hidden layer multiplies the input by learned weights, adds a bias term, and applies an activation function (typically ReLU, which keeps only positive values). This allows the model to learn complex combinations

of pixel values, such as particular color patterns or global texture cues. However, because spatial structure is lost during flattening, fully connected networks may struggle to detect local visual patterns like edges or shapes — which convolutional neural networks are better at.

Convolutional Neural Network

Compared to previous two models, CNN is better-suited for image classification because they are specifically designed to capture spatial hierarchies and local patterns in image data. Unlike multi-class logistic regression, which relies on manually engineered features, CNN can automatically learn rich, multi-layer feature representations directly from the pixels. Compared to fully connected neural networks, which treat all input pixels as independent features, CNNs use convolutional layers to preserve the spatial structure of images and learn translation-invariant features like edges, textures, and shapes. This makes them more efficient and effective for image tasks.

Transfer Learning with EfficientNet B4

For our fourth model we wanted to test out the power of Transfer Learning where we essentially use a pre-trained model that had already learned to recognize basic visual features from millions of images, then taught it specifically about dog breeds. This approach is particularly powerful when you have limited training data since the model doesn't need to learn basic fundamental concepts like edges, textures, and shapes from scratch. We tested several pre-trained architectures including ResNet50, MobileNetV2, and EfficientNetB0. EfficientNetB0 proved optimal because it uses compound scaling, a technique that simultaneously optimizes a network's depth, width, and input resolution, giving us strong balanced results without requiring massive computational resources. Our final model combines the pre-trained EfficientNetB0 architecture with custom layers designed specifically for 120-breed classification. By fine-tuning all layers with a lower learning rate, we are able to preserve the valuable features the model had already learned while adapting them to distinguish between different dog breeds.

Experiments, Results & Discussion

Multiclass Logistic Regression

To establish a performance baseline and better contextualize the strengths of more complex models, we began our experiments with a multiclass logistic regression model. For this model, we tried different learning rates, batch sizes, optimizers, and kernel and bias initializers. The model was trained using spares categorical crossentropy loss, and was evaluated both on accuracy scores and weighted F1 scores. The model was tuned before augmentation steps and after, to investigate the impact of augmentation. Before applying any data augmentation, the model achieved a training accuracy of 22.3%, but validation and test accuracies remained low (4.4% and 4.5%, respectively), but above our majority accuracy benchmark of 1.77%. F1 scores however, were pretty low at 0.03. When training on the augmented training set, we saw a much smaller generalization gap (reduced from 17.9% to 4.5%) but an overall decline in accuracy and a persistent F1 score of 0.03. This indicates that the model struggled to learn effectively from the augmented images. This suggests that due to this model's limited capacity to capture spatial features, its performance suffers when introducing more complexity in the images. Despite this, we were still able to obtain a higher accuracy than our majority class benchmark, and produced baseline metrics to evaluate the next set of models.

Fully Connected Neural Network

Following the multiclass logistic regression, we implemented a fully connected neural network (FCNN), also trained using sparse categorical cross-entropy loss. Hyperparameter tuning was performed both manually and using Keras Tuner, where we explored variations in hidden layer size (1 to 3 layers), learning rates (1e-3 to 1e-4), activation functions (ReLU vs. tanh), and optimizers (Adam vs. SGD). A key

challenge was overfitting, due to the relatively small training set of approximately 100 images per class. To mitigate this, we introduced two strategies: first, a data augmentation pipeline that applied random flips, rotations, and zooms during training to increase image variability; and second, an early stopping mechanism to halt training when validation loss stopped improving. These methods successfully reduced the generalization gap and improved test accuracy by approximately 2%. The final model achieved a test accuracy of ~7% — well above the majority class baseline of 1.77% and a modest improvement over logistic regression. Performance was also evaluated using a weighted F1 score, which reached 0.05, indicating that while the model captured some nonlinear structure, its ability to generalize fine-grained visual differences remained limited.

Convolutional Neural Network

As we transitioned to convolutional neural networks (CNNs), we began with a simple architecture: one convolutional layer, max pooling, dropout, flattening, and a dense output. Using Keras Tuner with random search, we optimized five key hyperparameters—filters (32), kernel size (4×4), dropout (0.4), activation (tanh), and learning rate (1e-3)—on a 10-class subset for easier diagnosis. This achieved 93.58% training accuracy but only 40.21% validation accuracy, with a 54-point gap and validation loss increasing after 3 epochs, indicating severe overfitting. To improve generalization, we added L1/L2 regularization to convolutional layers, inserted batch normalization after pooling, and stacked two “conv + pool + batch norm” blocks to increase capacity with control. We also embedded a Random Augmentation layer at the model input so each epoch saw different augmented data. Once the model generalized well, we deepened the dense layers (two extra dense layers + dropout), expanding the hyperparameter set to 13. Given the larger search space, we tuned manually. This revised 10-class model reached 45% validation accuracy after 200 epochs, with only a 3-point training–validation gap. We then applied the architecture to the 120-class dataset, but performance stalled—training loss plateaued after 50 epochs. We identified overly strong L1/L2 penalties and aggressive learning rate decay as likely causes. Reducing L1/L2, slowing learning rate decay from 0.8 to 0.9, and increasing batch size improved gradient stability and convergence speed. The final model achieved 0.13 weighted F1 score. Architecture details are included in the appendix.

Transfer Learning with EfficientNet B4

Building on the insights from previous models, we implemented transfer learning using pre-trained architectures to leverage features learned from large-scale image datasets. We began by evaluating three popular pre-trained models: ResNet50, MobileNetV2, and EfficientNetB0 which are all trained on ImageNet. ResNet50 achieved reasonable accuracy, but proved computationally expensive with 25 million parameters and slow training times. MobileNetV2 offered faster training but produced lower quality features for fine-grained classification tasks. EfficientNetB0 emerged as the optimal choice due to its compound scaling design which dynamically and systemically adjusted network depth, width, and input resolution to achieve superior performance with only 5 million parameters.

Our architecture combined the EfficientNetB0 backbone with a custom classification head consisting of global average pooling, batch normalization, and three dense layers and dropout rates. We followed with a final softmax layer for 120-class classification. Initially, we experimented with freezing the backbone layers and training only the classification head, but this approach yielded suboptimal results. We then enabled fine-tuning of all layers using a reduced learning rate (1e-3) to preserve valuable pre-trained features while adapting them to dog breed recognition. A key challenge was preventing overfitting when fine-tuning such a powerful pre-trained model on our relatively small dataset. We mimicked similar comprehensive regularization strategies including aggressive data augmentation, L2 penalties, early stopping based on validation accuracy. The final model achieved strong performance with a training accuracy of 33.5%, validation accuracy of 31.4%, and test accuracy of 31.5%. This demonstrated powerful generalization with only a 2.1% overfitting gap. Lastly the model reached a weighted F1 score of

0.31 which was more than double the CNN performance and a significant improvement over our logistic regression model. This improvement shows the effectiveness of transfer learning for complex visual classification tasks & limited training data.

Subgroup Analysis
When investigating subgroup performance across models, we found the following insights:

- Highest inaccuracies between classes of the same color - samoyed confused for maltese or westie, giant schnauzer confused for groendael as they both have dark coats.
- Size differences of dogs are ignored, presumably because it is hard to understand scale with the particular image data we have.
- The model struggled with classifying breeds that had a high amount of variability in the images.

We believe that much of this outcome was due to our limited dataset, and having more examples per breed, higher resolution of images, and a model that helps to first distinguish scale, could help the model distinguish visually similar classes.

Conclusions

Across the four progressively richer models,Transfer Learning gave the most substantial improvement, achieving an F1 score of approximately 0.31 and an overall test accuracy of 31.5% on the full 120-breed dataset. While still modest by modern standards, this represents a 30x improvement over the majority class baseline, largely due to the power of a sophisticated deep learning model and the efficiency of starting with existing knowledge rather than learning everything from zero. This allowed us to capture complex spatial hierarchies in image data. The task remained challenging for our untrained models due to the combination of a limited sample size (only ~100 images per class) and a large output space (120 distinct breeds). Unsurprisingly, these models frequently confuse breeds that are visually similar even to the human eye—e.g., Maltese vs. Westie. To improve performance, we recommend two directions. First, reduce class granularity by grouping visually similar breeds into broader categories (terriers, retrievers). Second, enhance data quality and diversity by sourcing additional images from datasets like ImageNet. Overall, this project offered a valuable introduction to deep learning for image classification, highlighting the critical importance of data quality, model architecture, and thoughtful evaluation in fine-grained recognition tasks.

Contributions *(See README.md in repo for specific .ipynb contributions)*

Jenny: *Wrote dataset section and sections on Multiclass Logistic Regression. Worked on data prep (eda, data cleaning, splitting into validation sets and 10 breed subsets) and LR model development.*

Lyn: *All the CNN notebooks including training, tuning and model evaluation, as well as pre-processing stage 1 (data loading and train test split). Wrote abstract, CNN model and model comparison.*

Ryan: *Wrote introduction/conclusion/research question/discussion, developed and wrote fully connected neural network, developed pre-processing stage 3 (implementing bounding box data), suggested project topic and planned models, added references.*

Rodney: *Tuned, trained, and evaluated the EfficientNetB0 transfer learning model. Further pre-processed data with more robust augmentation adjustments and early stoppage to prevent overfitting. Wrote the sections for the model as well as the related works and comparison summaries in our slide deck.*

References

1. Khosla, A., Jayadevaprakash, N., Yao, B., & Li, F. F. (2011, June). Novel dataset for fine-grained image categorization: Stanford dogs. In Proc. CVPR workshop on fine-grained visual categorization (FGVC) (Vol. 2, No. 1).
2. Ráduly, Z., Sulyok, C., Vadászi, Z., & Zölde, A. (2018, September). Dog breed identification using deep learning. In 2018 IEEE 16th International Symposium on Intelligent Systems and Informatics (SISY) (pp. 000271-000276). IEEE.
3. Li, S., Song, W., Fang, L., Chen, Y., Ghamisi, P., & Benediktsson, J. A. (2019). Deep learning for hyperspectral image classification: An overview. IEEE transactions on geoscience and remote sensing, 57(9), 6690-670

Appendix

1. Dataset sizes pre and post processing

	Original Dataset	120-breed dataset (after preprocessing steps)	10-breed dataset (after preprocessing steps)
Training Size	12,000	12,000-72,000*	6,000
Validation Size	-	4,290	388
Test Size	8,580	4,290	-

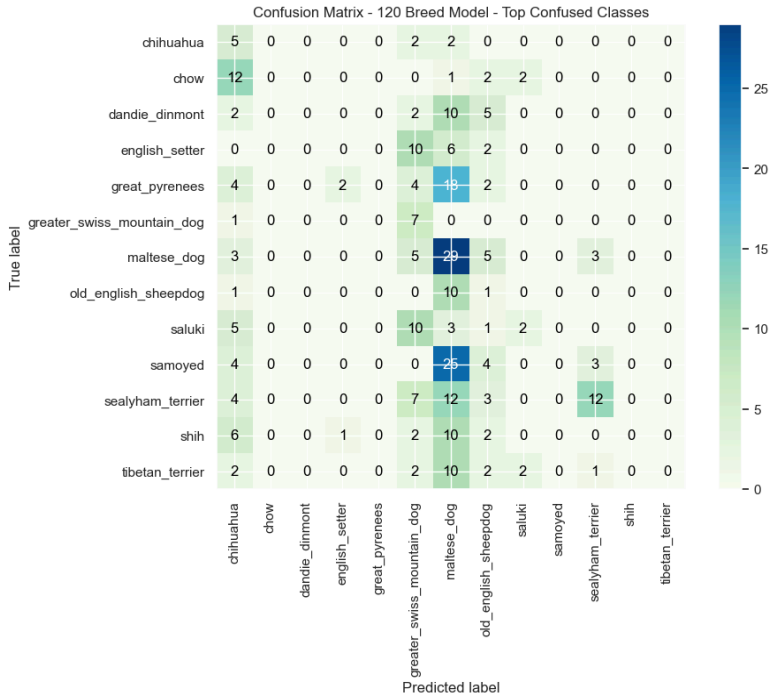
2. Performance Metrics for Sub Dataset (10 Breeds) & Full Dataset (120 Breeds)

Model	**Train Accuracy**	**Val Accuracy**	**Overfit Gap**	**Test Accuracy**	**F1-Score (Weighted-Avg)**	**Baseline Accuracy**
Logistic Regression (Sub Dataset)	44.9%	30.9%	14.0%	N/A	N/A	19.59%
Fully Connected NN (*Sub Dataset*)	41.5%	37.9%	3.61%	N/A	0.30	
CNN (Sub Dataset)	44.44%	42.96%	1.48%	N/A	0.41	
Logistic Regression (Full Dataset)	8.4%	3.7%	4.7%	3.5%	0.03	1.77%
Fully Connected NN (*Full Dataset*)	9.23%	6.97%	2.26%	7.09%	0.051	
CNN (Full Dataset)	15.60%	13.66%	1.94%	16.67%	0.13	
Transfer Learning (Full Dataset)	33.5%	31.4%	2.1%	31.5%	0.31	

3. CNN architecture & Hyperparameters:

	output shape	hyperparameters	regularization
convo 2d	(32, 64, 64, 8)	`filter size` = (8,8), `num_filter` = 8	`L1` = 1e-5, `L2` = 1e-4
max pooling	(32, 32, 32, 8)	`max_pool` = 2	
batch normalization	(32, 32, 32, 8)		
convo 2d	(32, 32, 32, 16)	`filter size` = (4,4), `num_filter` = 16	`L1` = 1e-5, `L2` = 1e-4
max pooling	(32, 16, 16, 16)	`max_pool` = 2	
batch normalization	(32, 16, 16, 16)		
convo 2d	(32, 16, 16, 16)	`filter size` = (2,2), `num_filter` = 16	`L1` = 1e-5, `L2` = 1e-4
convo 2d	(32, 16, 16, 16)	`filter size` = (2,2), `num_filter` = 16	`L1` = 1e-5, `L2` = 1e-4
max pooling	(32, 8, 8, 16)	`max_pool` = 2	
flatten	(32, 1024)		
dense1	(32, 128)	128	
dense2	(32, 128)	128	
dropout			0.25
dense3	(32, 120)	120	

4. *MLR CM for most confused classes*



5. MLR mistakes table for most confused classes

	mistakes
samoyed	36
great_pyrenees	30
sealyham_terrier	26
shih	21
dandie_dinmont	19
saluki	19
tibetan_terrier	19
english_setter	18
chow	17
maltese_dog	16
old_english_sheepdog	11
chihuahua	4
greater_swiss_mountain_dog	1

6. Breeds with top 5 F1 Scores: ['leonberg', 'affenpinscher', 'maltese_dog', 'komondor', 'sealyham_terrier']

Fully Connected

Top 5 F1

	Breed	F1 Score	Precision	Recall	Support
98	sealyham_terrier	0.340659	0.236641	0.607843	51.0
42	entlebucher	0.171717	0.115646	0.333333	51.0
39	english_foxhound	0.170213	0.121212	0.285714	28.0
82	old_english_sheepdog	0.168421	0.133333	0.228571	35.0
94	samoyed	0.160000	0.120690	0.237288	59.0

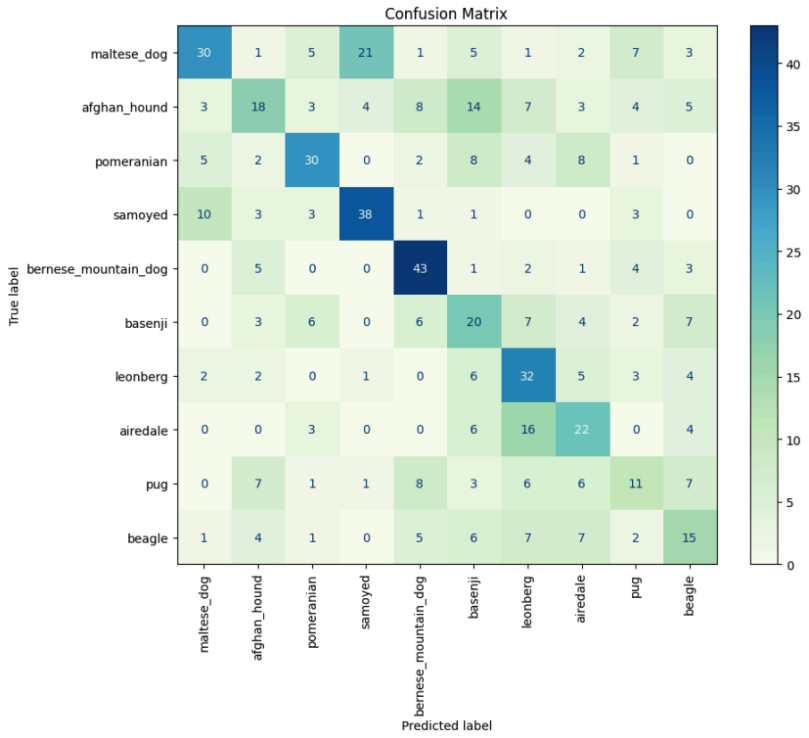
Bottom 5 F1

	Breed	F1 Score	Precision	Recall	Support
4	american_staffordshire_terrier	0.0	0.0	0.0	32.0
12	black	0.0	0.0	0.0	29.0
15	bluetick	0.0	0.0	0.0	36.0
14	bloodhound	0.0	0.0	0.0	43.0
20	bouvier_des_flandres	0.0	0.0	0.0	25.0

Most Confused

	True Class	Top Wrong Class	Count	Support	Recall (%)
73	maltese_dog	samoyed	16	76	9.210526
69	leonberg	brabancon_griffon	15	55	16.363636
94	samoyed	west_highland_white_terrier	14	59	23.728814
97	scottish_deerhound	entlebucher	13	66	4.545455
52	great_pyrenees	kuvasz	12	57	0.000000
78	newfoundland	groenendael	11	47	23.404255
84	papillon	english_springer	9	48	6.250000
10	bedlington_terrier	samoyed	9	41	2.439024
48	giant_schnauzer	groenendael	8	29	0.000000
14	bloodhound	irish_setter	8	43	0.000000
100	shih	sealyham_terrier	7	57	1.754386
59	irish_wolfhound	great_dane	7	59	0.000000
2	african_hunting_dog	leonberg	7	35	8.571429
11	bernese_mountain_dog	entlebucher	7	59	1.694915
117	whippet	leonberg	7	43	0.000000

CNN Confusion matrix of 10-class model (manually selected 10-breed)



5. Subgroup variance of the full model

Top 5 accuracy

	precision	recall	f1-score	support
greater_swiss_mountain_dog	0.354167	0.500000	0.414634	34.0
sealyham_terrier	0.352941	0.470588	0.403361	51.0
japanese_spaniel	0.386364	0.395349	0.390805	43.0
kerry_blue_terrier	0.307692	0.307692	0.307692	39.0
pomeranian	0.288136	0.288136	0.288136	59.0

Breeds with 0 accuracy

	precision	recall	f1-score	support
collie	0.0	0.0	0.0	27.0
cocker_spaniel	0.0	0.0	0.0	30.0
cardigan	0.0	0.0	0.0	27.0
chihuahua	0.0	0.0	0.0	26.0
great_dane	0.0	0.0	0.0	28.0
french_bulldog	0.0	0.0	0.0	29.0
german_short	0.0	0.0	0.0	26.0
golden_retriever	0.0	0.0	0.0	25.0
miniature_schnauzer	0.0	0.0	0.0	27.0
labrador_retriever	0.0	0.0	0.0	35.0
italian_greyhound	0.0	0.0	0.0	41.0
miniature_poodle	0.0	0.0	0.0	27.0
pug	0.0	0.0	0.0	50.0
pekinese	0.0	0.0	0.0	25.0
old_english_sheepdog	0.0	0.0	0.0	35.0
standard_poodle	0.0	0.0	0.0	29.0
toy_poodle	0.0	0.0	0.0	25.0
tibetan_terrier	0.0	0.0	0.0	53.0
tibetan_mastiff	0.0	0.0	0.0	26.0
wire	0.0	0.0	0.0	29.0