

# Project Proposal

December 28, 2021

## 1 Domain Background

dogs! what are they? how do we define a "dog"? That sounds like an easy question; dogs are animals that have lots of hair and fur and walk on four legs. they vary in size but typically, they have two eyes and a shiny nose. But that's a wolf!



not only a wolf but a bear, a possum, raccoon, ferret, and many other mammals. So how do we define dogs? and even more challenging question, how do we define each breed?

this project is part of the field of object recognition and image classification. For us to find a way to know if this object is a dog and which breed it is, we have to use neural networks and object detection to train a model to distinguish between breeds.

I have an interest in classification based on images. I have multiple projects in mind that would require a model to classify objects. It would be a valuable learning experience to learn how to do it effectively.

## 2 Problem Statement

While we humans have the mental ability to distinguish between animals, computers are a different story. Humans don't need to learn to differentiate between dogs and other animals. It comes naturally to us. Writing an algorithm for computers to do the same is difficult because we struggle to define dogs on paper.

To solve this problem, We used transfer learning with the base model VGG16. VGG16 is trained on thousands of images.

## 3 Datasets and Inputs

The dataset for this project will be provided in Udacity's dog identification app. The dataset contains images of dogs and humans. Each person has a unique folder with at least one photo. Similarly, each dog breed has at least one folder with at least one photo representing it.

Collectively, there are 13233 human images and 8351 dog images. The set is divided into three sections: testing, validation, and training. However, the set is generally small. Although we have more than 8000 dog images, we have 133 breeds. The dataset is spread pretty thin between so many dog breeds.

We will discuss how we solved this problem in the next section.

---

<sup>1</sup>Image from Scientific american : <https://www.scientificamerican.com/article/how-wolf-became-dog/>

## 4 Solution Statement

To find humans, we will use Open CV's facial recognition feature. Open CV is a popular pre-trained model optimized for multiple applications including facial recognition.

As for dogs, we will use a different pre-trained model, VGG-16. VGG-16 uses ImageNet to create a model that can detect objects and animals.

After detecting the dogs/humans in the image, we use convolutional neural networks to train a model to classify images based on dog breeds.

After the model is ready, we will use a python server as an endpoint to connect to the model and create a react front end to communicate with the endpoint.

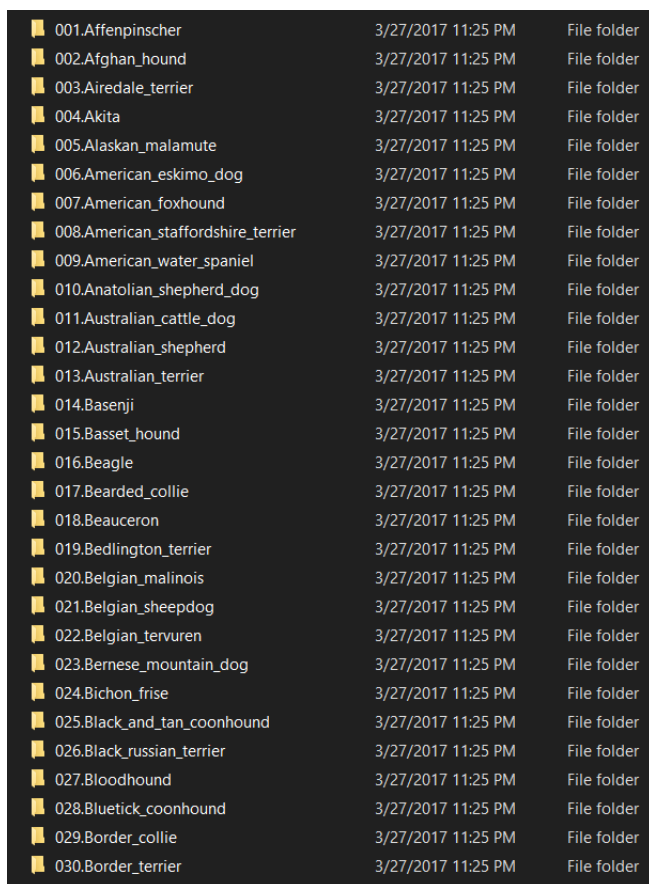
## 5 Project Design

- Explore the dogs and humans dataset
- Use OpenCV to detect humans and produce the trimmed image that contains the human's face
- use VGG-16 to identify images with dogs and produced a trimmed image that contains a dog.
- preprocess image using OpenCV and turn them into a sharp, black and white image.
- create a Convolutional neural network model using Pytorch
- choose a loss function and an optimiser for the Pytorch model
- train the model.
- test the accuracy of the model.
- create a server to server the model and an App to communicate with the model.

## 6 Data exploration

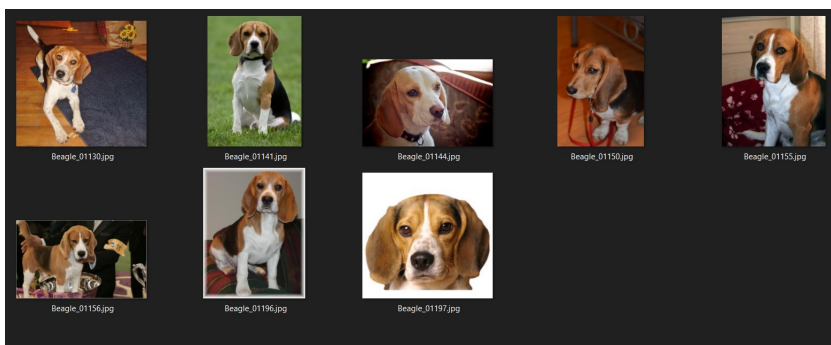
Our dataset is all images. To explore our data, we started by looking at the folder of images and its setup.

The dog images are separated by purpose (test, train, validate). Further, if we open any of the folders, we can see that the images are separated by breed.



001.Affenpinscher	3/27/2017 11:25 PM	File folder
002.Afghan_hound	3/27/2017 11:25 PM	File folder
003.Airedale_terrier	3/27/2017 11:25 PM	File folder
004.Akita	3/27/2017 11:25 PM	File folder
005.Alaskan_malamute	3/27/2017 11:25 PM	File folder
006.American_eskimo_dog	3/27/2017 11:25 PM	File folder
007.American_foxhound	3/27/2017 11:25 PM	File folder
008.American_staffordshire_terrier	3/27/2017 11:25 PM	File folder
009.American_water_spaniel	3/27/2017 11:25 PM	File folder
010.Anatolian_shepherd_dog	3/27/2017 11:25 PM	File folder
011.Australian_cattle_dog	3/27/2017 11:25 PM	File folder
012.Australian_shepherd	3/27/2017 11:25 PM	File folder
013.Australian_terrier	3/27/2017 11:25 PM	File folder
014.Basenji	3/27/2017 11:25 PM	File folder
015.Basset_hound	3/27/2017 11:25 PM	File folder
016.Beagle	3/27/2017 11:25 PM	File folder
017.Bearded_collie	3/27/2017 11:25 PM	File folder
018.Beauceron	3/27/2017 11:25 PM	File folder
019.Bedlington_terrier	3/27/2017 11:25 PM	File folder
020.Belgian_malinois	3/27/2017 11:25 PM	File folder
021.Belgian_sheepdog	3/27/2017 11:25 PM	File folder
022.Belgian_tervuren	3/27/2017 11:25 PM	File folder
023.Bernese_mountain_dog	3/27/2017 11:25 PM	File folder
024.Bichon_frise	3/27/2017 11:25 PM	File folder
025.Black_and_tan_coonhound	3/27/2017 11:25 PM	File folder
026.Black_russian_terrier	3/27/2017 11:25 PM	File folder
027.Bloodhound	3/27/2017 11:25 PM	File folder
028.Bluetick_coonhound	3/27/2017 11:25 PM	File folder
029.Border_collie	3/27/2017 11:25 PM	File folder
030.Border_terrier	3/27/2017 11:25 PM	File folder

Each breed is associated with a unique integer. It allows us to identify each breed without the need to transform data from categorical to numerical. After checking the folders, we then open a few breed folders to examine the images. All images vary in size.



To prepare our images for the model, we need to resize them later. This is because Convolutional models expect a unified tensor size. Given these two observations, we can traverse through the images and create a dictionary that contains the breed and the code associated with it.

## 7 Data Preprocessing

For Preprocessing our images, we begin by looping through the dog images folder to construct an image path array. For each image path, we extract the dog breed and the unique code.

We do this by viewing all folders in the test set and separating their path using a delimiter.

We will utilize our dictionary later when the model predicts a number and we want to find which breed is associated with the prediction.

We use the same idea of using a delimiter to create our dataset Loaders. To construct our data loaders, we use a delimiter to separate the breed from its number. We attach these two bits of information to our image path and return it as a data point.

One data point then would be the breed name, the numerical code of the breed, and the path of the image. We will utilize the image path, the code, and the breed to train the model later

Next, we tackle the size of the dataset. Since the dataset is generally small, we used the technique of Data Augmentation. To do this, we used torchvision's transforms. Transforms can fundamentally change the image so that for our model, this augmented image would seem like a new data point. For each image, we would create six augmented versions with varying effects(perspective warping, color jitter, blur, etc...)

```

image_dim = 50 #50

transform={}

transform["train"] = transforms.Compose([
    transforms.Resize(image_dim),
    transforms.CenterCrop(image_dim),
    transforms.RandomGrayscale(p=0.35),
    transforms.RandomRotation(degrees=(0, 180)),
    transforms.RandomPosterize(bits=2),
    transforms.RandomAdjustSharpness(sharpness_factor=0.1, p=0.5),
    transforms.RandomEqualize(),
    transforms.RandomHorizontalFlip(p=0.6),
    transforms.RandomVerticalFlip(p=0.02),
    transforms.ColorJitter(brightness=0.1),
    transforms.ToTensor(),
    transforms.Normalize(
        mean=[0.485, 0.456, 0.406],
        std=[0.229, 0.224, 0.225]
    )
])

transform["valid"] = transforms.Compose([
    transforms.Resize(image_dim),
    transforms.CenterCrop(image_dim),
    transforms.ToTensor(),
    transforms.Normalize(
        mean=[0.485, 0.456, 0.406],
        std=[0.229, 0.224, 0.225]
    )
])

```

Further, We prepare each image(both augmented and the original) with two main transforms: Resize and Normalization.

The resize transform sets the image to a dimension of 224x244. Its a requirement by the VGG16 model for the input tensor. The normalization layers help the model learn effectively. Through training the model multiple times, a normalized model performed better by at least 5 to 10%

## 8 Model Training

To solve the problem of breed classification, we will use transfer learning with a pre-trained VGG16 model. VGG16 is trained in the detection of objects and animals. To fit it to our purpose, we begin by defining the model. Then, we freeze its parameters and train only the last layer. The last layer is replaced with our layer that produces an output of 133, the number of the breeds we want to classify.

```

import torchvision.models as models
import torch.nn as nn
import time

model_transfer = models.vgg16(pretrained=True) #vgg16
for par in model_transfer.parameters():
    par.requires_grad = False

model_transfer.classifier[6]=nn.Linear(model_transfer.classifier[6].in_features, 133)

```

Next, we define our model training process. This step is relatively straightforward. We begin by defining our loss function and optimizer. We will use Cross-Entropy Loss and AdamW for our loss function and optimizer. Both are well-tested and well-performing functions and yield adequate results. Next, We cycle through our data loaders and load batches to feed into our model. We give the output of our model to our loss function and optimizer and then step.

After a few training cycles, we found that the best hyperparameters are Epochs = 7 and batches = 3.

## 9 Benchmark Model

The number of dog breeds we are classifying is 133 breeds. A random model should have an accuracy of less than 1%

According to Dave Gershgorn, an AI reporter working for Quartz, google created a model that can classify dogs and cats with 70 percent accuracy in 2012<sup>2</sup>.

So our goal was to produce a model that can break 60-70%

## 10 Evaluation Metrics

To check the accuracy of the model, we will split the data to test, train and validation. We will then check how many predictions were correct by the model. the Dataset is generally balanced so no need for resampling.

---

<sup>2</sup>Dave Gershgorn, 2017, "Five years ago, AI was struggling to identify cats. Now its trying to tackle 5000 species", April 11, Quartz, <https://qz.com/954530/five-years-ago-ai-was-struggling-to-identify-cats-now-its-trying-to-tackle-5000-species/>

We set the model for evaluation and then feed it our test data. Then we compare the model results with our expected result.

```
def test(loaders, model, criterion, use_cuda):

    # monitor test loss and accuracy
    test_loss = 0.
    correct = 0.
    total = 0.

    model.eval()
    for batch_idx, (data, target, data_type) in enumerate(loaders['test']):
        # move to GPU
        data_list=[]
        for index, image in enumerate(data):
            image = Image.open(image)
            if data_type[index] == "augmented":
                data_list.append(transform["train"](image).cuda())
            else:
                data_list.append(transform["valid"](image).cuda())

        data = torch.stack(data_list)
        if use_cuda:
            data, target = data.cuda(), target.cuda()
        # forward pass: compute predicted outputs by passing inputs to the model
        output = model(data)
        # calculate the loss

        loss = criterion(output, target)

        # update average test loss
        test_loss = test_loss + ((1 / (batch_idx + 1)) * (loss.data - test_loss))
        # convert output probabilities to predicted class
        pred = output.data.max(1, keepdim=True)[1]
        # compare predictions to true label
        correct += np.sum(np.squeeze(pred.eq(target.data.view_as(pred))).cpu().numpy())
        total += data.size(0)

    print('Test Loss: {:.6f}\n'.format(test_loss))

    print('\nTest Accuracy: %2d%% (%2d/%2d)' % (
        100. * correct / total, correct, total))
```

Our model scored 88%. Much higher than the set goal.

```
In [33]: #model_transfer.load_state_dict(torch.load('model_transfer.pt'))
         model_transfer.load_state_dict(torch.load('model_transfer.pt'))
```

Out[33]: <All keys matched successfully>

### (IMPLEMENTATION) Test the Model

Try out your model on the test dataset of dog images. Use the code cell below to calculate accuracy. The goal is greater than 60%.

```
In [34]: test(loaders_transfer, model_transfer, criterion_transfer, use_cuda)
```

Test Loss: 0.381690

Test Accuracy: 88% (737/836)

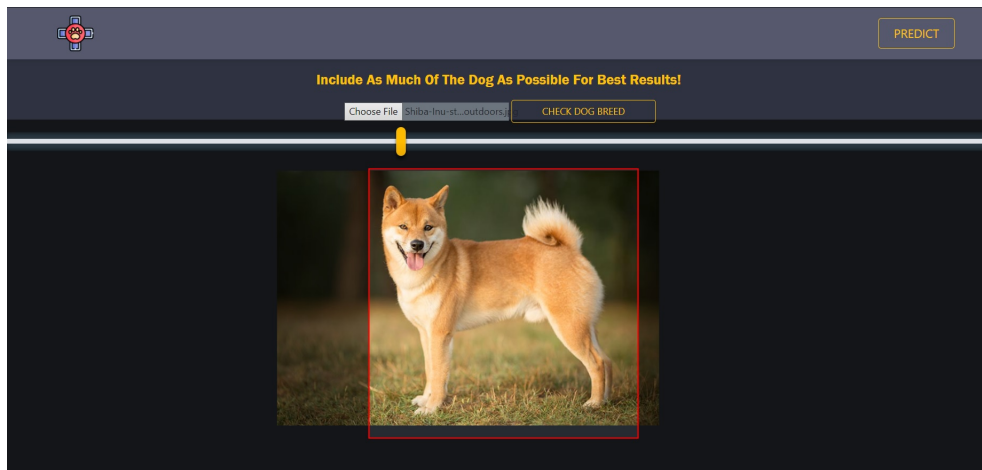


## 11 Breed Classifier React App


To communicate with the model, we Use FastApi python server to load our model into and then create a react App to communicate with the Api. This is the first page of the App.



We then choose an image and then fit it the best we can. Then click "Check Breed"



this will query the model for a prediction which will be used for the next page.




## You Are A Dog !

### Your Predicted Breed Is :

## Finnish Spitz

Want to learn more? Check the wikipedia page below!

GO TO WIKI!




Cute as a Button!

Want to check another image? Click on the button below

LET'S GO! →→

If you want to learn more, you can also click on the wiki button that will send you to a page related to the predicted breed.



WIKIPEDIA  
The Free Encyclopedia

- Main page
- Contents
- Current events
- Random article
- About Wikipedia
- Contact us
- Donate
- Contribute
- Help
- Learn to edit
- Community portal
- Recent changes
- Upload file
- Tools
- What links here
- Related changes
- Special pages
- Permanent link
- Page information
- Cite this page
- Wikidata item
- Print/export
- Download as PDF
- Printable version

Article [Talk](#)

Read [Edit](#) [View history](#)

Search Wikipedia

## Finnish Spitz

From Wikipedia, the free encyclopedia


The **Finnish Spitz** (Finnish: *suomenpystykorva*) is a breed of dog originating in Finland. The breed was originally trained to hunt all types of game from squirrels and other rodents to bears.<sup>[1]</sup> It is a "bark pointer", indicating the position of game by barking, and drawing the game animal's attention to itself, allowing an easier approach for the hunter. Its original game hunting purpose was to point to game that fled into trees, such as grouse, and capercaillies, but it also serves well for hunting elk. Some individuals have even been known to go after a bear. In its native country the breed is still mostly used as a hunting dog. The breed is friendly and in general loves children, so it is suitable for domestic life. The Finnish Spitz has been the national dog of Finland since 1979.

### Contents [hide]

- Lineage
- History
- Appearance
  - 3.1 Coat
  - 3.2 Color
  - 3.3 Height and weight
- Temperament
  - 4.1 Barking
  - 4.2 Training
- Health
- See also
- References
- Further reading
- External links

### Lineage [ edit ]

### Finnish Spitz



**Other names** Finnish Hunting Dog  
Finnish Spets  
Finsk Spets  
Loulou Finnois  
suomalainen pystykorva  
suomenpystykorva

**Origin** Finland

	Traits	[hide]
<b>Height</b>	Dogs: 44 to 50 cm (17 to 20 in) Bitches: 39 to 45 cm (15 to 18 in)	
<b>Weight</b>	12 to 13 kg (26 to 29 lb)	
<b>Coat</b>	Double	
<b>Color</b>	Red, red gold or gold	
<b>Kennel club standards</b>		[hide]
<b>Suomen Kenneliläite</b>	standard	