# Fetching the Right Breed

Dog Breed Image Classification
CS5804 Mini Project

# Group 7 - Team Introduction


Katie Geary
kgeary6@vt.edu
BME PhD student
Interest: AI in medicine

Jonathan Mahoney,
jpmahoney@vt.edu
MEng in CSA
Interest: AI

Meghan Cooke
macooke@vt.edu
MEng in CSA
Interest: AI

Srujan Vithalani
srujan@vt.edu
MEng in CSA
Interest: AI

# Problem Description

What kind of dog is that?!

- Accept images from users
- Determine if a dog is present
- Identify dog breeds accurately
- Tests the assumption that humans look like their dogs

# Background

## Images Classification of Dogs and Cats using Fine-Tuned VGG Models

**Publisher: IEEE** Cite This PDF

Mahardi ; I-Hung Wang ; Kuang-Chyi Lee ; Shinn-Liang Chang **All Authors**



**Abstract**

Document Sections

» Introduction

» Dataset for CDC

» Fine Tuning Models for CDC

» CDC Models Training Result

» Conclusion

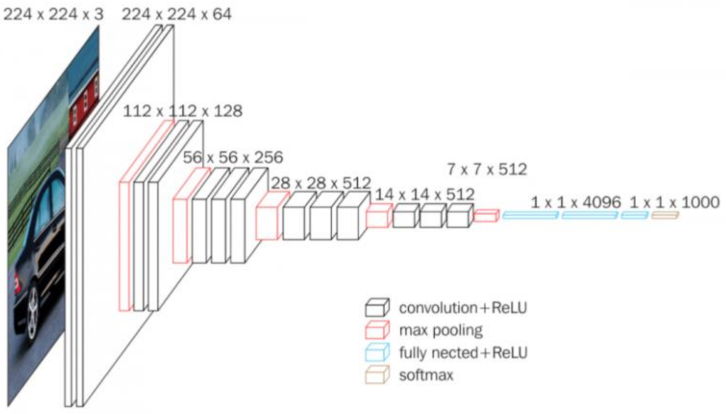
**Abstract:**
Image classification has become more popular as it is the most basic application and implementation of deep learning. Images of dogs and cats are the most common example to train image classifiers as they are relatable. It is easy to classify the image of cats and dogs, but the images of various breeds are difficult to classify with high accuracy. In this paper, we tried to build an image classifier to recognize various breeds of dogs and cats (CDC) using fine-tuned VGG models. Two common models, VGG16 and VGG19 were used to build the classifier. The resulting model from VGG16 has a training accuracy of 98.47%, validation accuracy of 98.56%, and testing accuracy of 83.68%. The model from VGG19 has a training accuracy of 98.59%, validation accuracy of 98.56%, and testing accuracy of 84.07%.

**Published in:** 2020 IEEE Eurasia Conference on IOT, Communication and Engineering (ECICE)


Fig. 4 Prediction result of single image

# Background

VGG16 is a Convolutional Neural Network developed in 2014 out of Oxford University




- Trained on ImageNet
- Convolution, ReLU, max pooling, fully connected and softmax layers
- Very accurate but painful to train
- PyTorch, Tensorflow, and Keras
- Highly adaptable

# Background

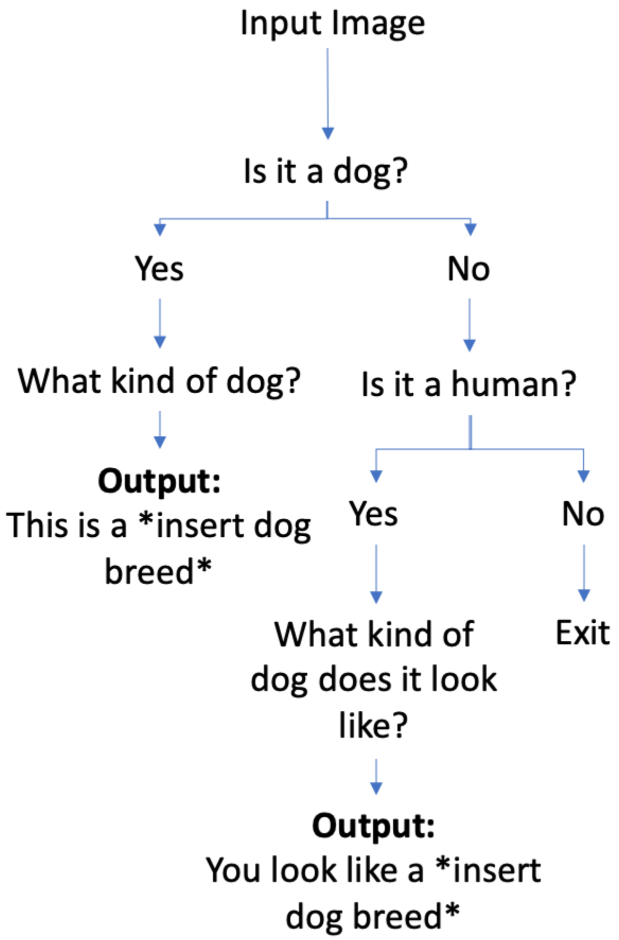
## Stanford Dogs Dataset

- Originally from ImageNet, but refined for fine-grained image categorization
- 20580 images total
- 120 breeds
  - ~150 images of each breed
  - Breeds form all over the world

# Approach

UDACITY

- Obtain pre-trained VGG16 network
- Remove end/output  layers
- Replace to match the framework of what you are predicting
- Retrain on Stanford Dogs dataset



# Image Preprocessing

```
[ ] test_pct = 0.3
    test_size = int(len(dataset)*test_pct)
    train_size = len(dataset) - test_size

    train_size, test_size

    (14406, 6174)
```

70:30 split for training and test data

```
[ ] train_ds, test_ds = random_split(dataset, [train_size, test_size])
    len(train_ds), len(test_ds)

    (14406, 6174)
```

Training image augmentation

```
train_transform = transforms.Compose([
    transforms.Resize((256, 256)),
    transforms.RandomCrop(224, padding=4, padding_mode='reflect'),
    transforms.RandomHorizontalFlip(p=0.3),
    transforms.RandomRotation(degrees=30),
    transforms.ToTensor(),
])

test_transform = transforms.Compose([
    transforms.Resize((224,224)),
    transforms.ToTensor(),
])
```

# Model

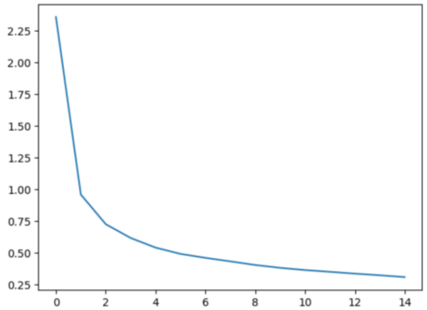
```python
model = models.vgg16(pretrained=True)

for param in model.parameters():
    param.requires_grad = False

model.classifier[-1].requires_grad = True

# Replace the last fully connected layer with a new one
num_features = model.classifier[-1].in_features
model.classifier[-1] = nn.Linear(num_features, 120)

model = model.to(device)
```

# Results

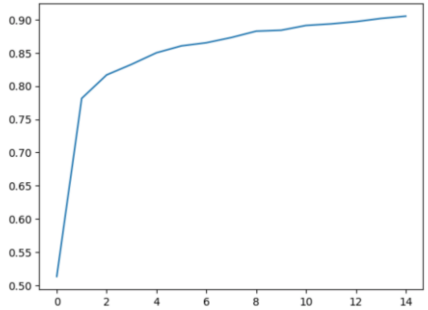
```
train_loss = []
train_acc = []

for epoch in range(EPOCHS):
  trainEpochLoss, trainEpochAcc = Train(model, train_loader, optimizer, criterion)
  train_loss.append(trainEpochLoss)
  train_acc.append(trainEpochAcc)
  print(f'Epoch: {epoch} Train Loss: {trainEpochLoss} Train Acc: {trainEpochAcc}')
```

```
Epoch: 0 Train Loss: 2.355252196523878 Train Acc: 0.5134666562080383
Epoch: 1 Train Loss: 0.9587489938735962 Train Acc: 0.7814105749130249
Epoch: 2 Train Loss: 0.7257632601261139 Train Acc: 0.8167430758476257
Epoch: 3 Train Loss: 0.6171630475256178 Train Acc: 0.832639217376709
Epoch: 4 Train Loss: 0.5415320964654287 Train Acc: 0.8500624895095825
Epoch: 5 Train Loss: 0.4919695989290873 Train Acc: 0.860405445098877
Epoch: 6 Train Loss: 0.4609191241529253 Train Acc: 0.8651256561279297
Epoch: 7 Train Loss: 0.4332262490193049 Train Acc: 0.8729696273803711
Epoch: 8 Train Loss: 0.4048339033789105 Train Acc: 0.882548987865448
Epoch: 9 Train Loss: 0.3825222474336624 Train Acc: 0.8839372992515564
Epoch: 10 Train Loss: 0.3648779981666141 Train Acc: 0.8911564946174622
Epoch: 11 Train Loss: 0.35117199811670513 Train Acc: 0.8935166001319885
Epoch: 12 Train Loss: 0.3367801884147856 Train Acc: 0.8968485593795776
Epoch: 13 Train Loss: 0.3238857471280628 Train Acc: 0.9016382694244385
Epoch: 14 Train Loss: 0.30980095757378473 Train Acc: 0.9051090478897095
```

```
plt.plot(train_loss, label='Train Loss')
plt.show()
```



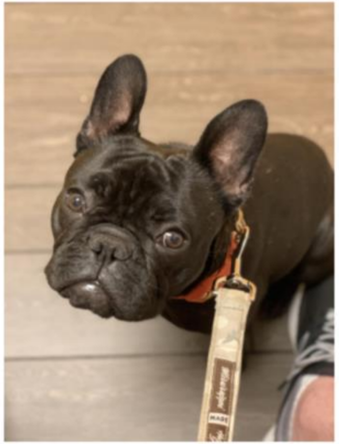

```
plt.plot(cpu_train_arr, label='Train Acc')
plt.show()
```



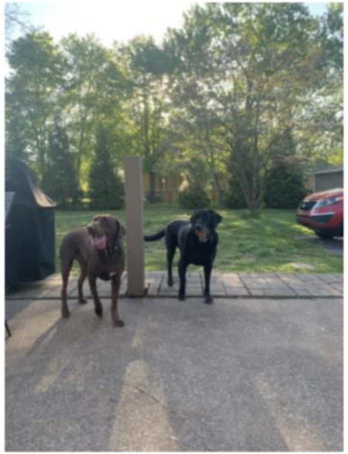

```
predicted, expected = evaluate(model, test_loader, criterion, test=True)
```

```
Scores:
Precision: 0.8370735501684606, Recall: 0.833685805761606, Fscore: 0.8329355043953408, Accuracy: 0.8382161458333334
```

# Results



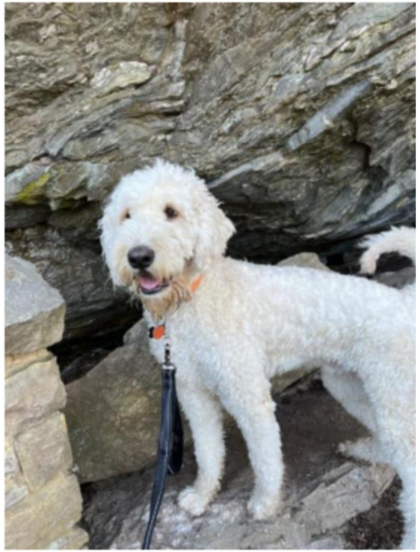
'French bulldog'



'Labrador retriever'

'Labrador retriever'

'briard' Actual: Labradoodle



'Old English sheepdog'

Actual: Sheepadoodle

# Results

Human Detected! You look like a Weimaraner

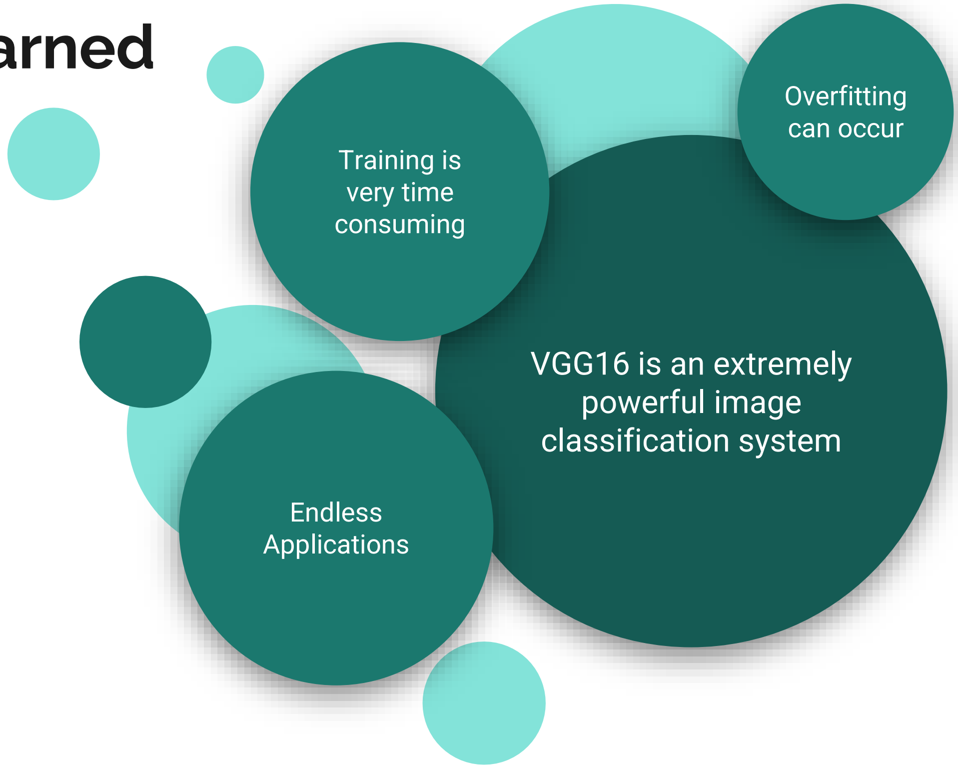
Human Detected! You look like a Lhasa

# Results

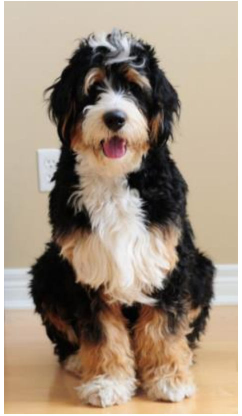
Human Detected! You look like a Sussex spaniel

Human Detected! You look like a Maltese dog

# Lessons Learned



# Future Work



Incorporation into an app for use on the go

Additional breeds

Breed mix prediction

Other animals

# References

[1] Mahardi, I. -H. Wang, K. -C. Lee and S. -L. Chang, "Images Classification of Dogs and Cats using Fine-Tuned VGG Models," 2020 IEEE Eurasia Conference on IOT, Communication and Engineering (ECICE), Yunlin, Taiwan, 2020, pp. 230-233, doi: 10.1109/ECICE50847.2020.9301918.
[2] Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556.*
[3] Udacity, "Deep-learning-V2-pytorch/project-dog-classification at master · udacity/deep-learning-V2-pytorch," GitHub. [Online]. Available: https://github.com/udacity/deep-learning-v2-pytorch/tree/master/project-dog-classification. [Accessed: 23-Mar-2023]
[4] A. Khosla, N. Jayadevaprakash, B. Yao, and L. Fei-Fei, "Stanford Dogs Dataset," *Stanford dogs dataset for fine-grained visual categorization*. [Online]. Available: http://vision.stanford.edu/aditya86/ImageNetDogs/. [Accessed: 23-Mar-2023]