DS 4400: Machine Learning and Data Mining I

Spring 2023

Guidelines

The goal of this course project is to prepare you for some project experience in machine learning & data mining. By the end of the project, we hope that you will have gained some hands-on experience in applying ML to a real-world problem. You can start by browsing through the list of datasets we have suggested in the next section. Pick a dataset that interests you. Then, apply the knowledge that we have gained this semester to analyze this dataset. There are three key dates to keep in mind for the course project:

- By **March 23**, gather a team of up to three students (in total) and decide on a topic. Prepare a few slides and participate in an in-class discussion.
- By **April 13**, set up a public GitHub repository along with documentation (in markdown format), to describe the findings of the project. We will have an in-class discussion to share the project findings (which can be delivered in slides or in a tutorial of the GitHub repository) with the rest of the class.
- By **April 27**, submit the link to the GitHub repository for grading. We will assign the final grade by examining the code repository and its accompanying documentation.

Project proposal presentation (Due March 23)

(In class on Thursday, March 23)

https://www.canva.com/design/DAFd3gYxx48/vUWV_t2ObbIdzB7A1JCsnw/edit?utm_content=DAFd3gYxx48&utm_campaign=designshare&utm_medium=link2&utm_source=sharebutton

The project proposal presentation involves a five-minute presentation that includes all the team members (up to three students) and the topic of your team. As a template for the presentation, answer the following five questions.

Title: TBD

Team members: TBD

P1: What is the problem?

P2: Why is it interesting?

P3: What are the approaches you plan to explore?

P4: Why are these approaches reasonable to consider? Also, include any specific limitations.

P5: What is the expected timeline for conducting the project? Also, including the division of work between every student in the team.

In-class presentation schedule (TBD)

Full project presentation (Due April 13)

Title: Pet Dataset Classification and Segmentation

Abstract: The goal of this project is to be able to recognise the pets (between cats and dogs) and their breeds based on the data given. We use the very interesting and diverse Oxford-IIIT Pet Dataset. The dataset consists of 7384 images, with 37 different breeds of cats and dogs (2371 cats and 4978 dogs). We train the data on 5 different types of models: (i) convolutional neural network, (ii) ensemble convolutional neural network, (iii) Resnet based image classification model (pre-trained with imagenet dataset), (iv) vision transformer (pre-trained with imagenet dataset), and (v) ensemble model based on the pre-trained models (resnet-152, densenet-161 and vgg-19). We trained it using TensorFlow and Keras. The MV2 and Exception model performed the best with an accuracy of about 95%. Where the MV2 model was more consistent. We used the pre-trained model and fit our data onto it. Thus, the system can be used to identify the breed of any cat or dog live. We additionally also tried to predict the bounding boxes on the pets to detect the faces of the pets for classification in a larger frame.

Documentation:

1. Introduction:

The problem we are trying to solve is to detect and evaluate the breed of the pet, and whether it is a cat or a dog. This is interesting because it has practical applications beyond being a thought-provoking problem. It can be used at animal shelters for id'ing lost pets, vet clinics for quick screening, or even social media when curating viewing algorithms for users. Additionally, the dataset is large and diverse, making it a benchmark for evaluating deep learning models in the field of computer vision.

To approach this problem, we first built a segmentation model as a baseline to see how it would perform at classifying the dataset. We then created 3 models off of it: ensemble, vision transformer (ViT), and ResNet. These models each have various specifications that they excel in that we proposed would outperform the segmentation model. We came to this conclusion because they are best at extracting features and recognizing patterns in visual data which is the exact goal of our project.

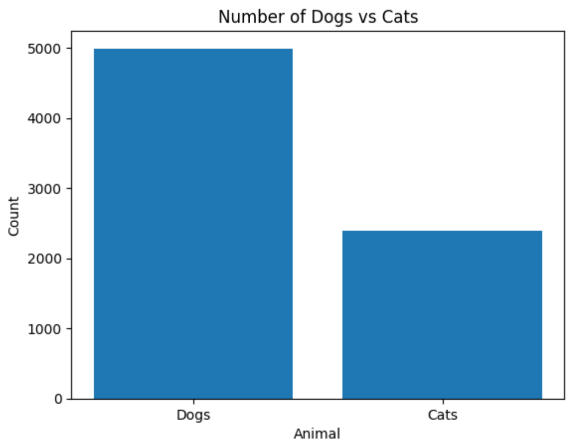
During the early stages of the undertaking when we were researching how to approach the problem and how others had approached similar problems, we saw that there were many previous versions of this problem using regular CNNs. However, there were no previous attempts at this using newer models like ResNet and pre-trained ViT. These state-of-the-art approaches allow us to achieve accuracies that previously would take much larger datasets and much more computational time and power. Additionally, our larger, more diverse models can be combined to create more precise results.

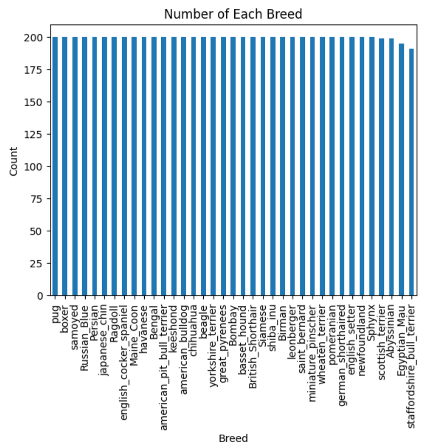
The key components of our approach are the segmentation model as a base, the three other models (ensemble, ViT, and RESNET), and the use of data augmentation techniques to improve the performance of the models. We also combine multiple models in our ensemble learning and improve the accuracy of our predictions. The limitations of our approach include the need for large amounts of training data and the computational resources required to train multiple models.

2. **Setup**

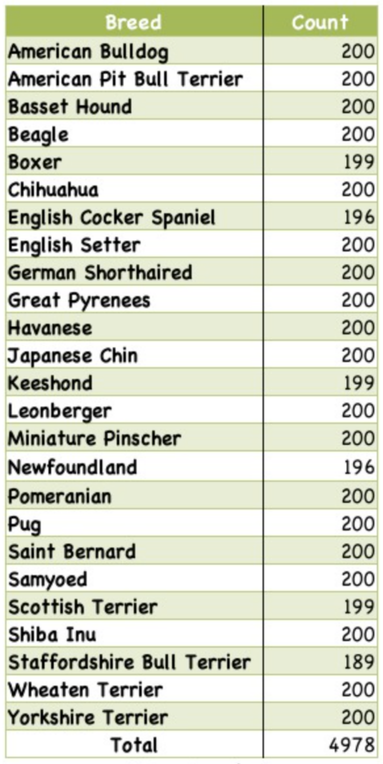
   2.1 Dataset Description

   The dataset that we used for this project is the Oxford-IIIT Pet Dataset.



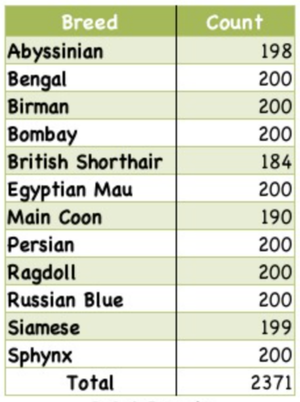


With the following statistics:



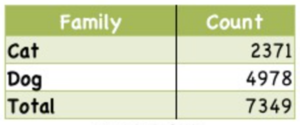
Breed	Count
American Bulldog	200
American Pit Bull Terrier	200
Basset Hound	200
Beagle	200
Boxer	199
Chihuahua	200
English Cocker Spaniel	196
English Setter	200
German Shorthaired	200
Great Pyrenees	200
Havanese	200
Japanese Chin	200
Keeshond	199
Leonberger	200
Miniature Pinscher	200
Newfoundland	196
Pomeranian	200
Pug	200
Saint Bernard	200
Samyoed	200
Scottish Terrier	199
Shiba Inu	200
Staffordshire Bull Terrier	189
Wheaten Terrier	200
Yorkshire Terrier	200
Total	4978

1.Dog Breeds



Breed	Count
Abyssinian	198
Bengal	200
Birman	200
Bombay	200
British Shorthair	184
Egyptian Mau	200
Main Coon	190
Persian	200
Ragdoll	200
Russian Blue	200
Siamese	199
Sphynx	200
Total	2371

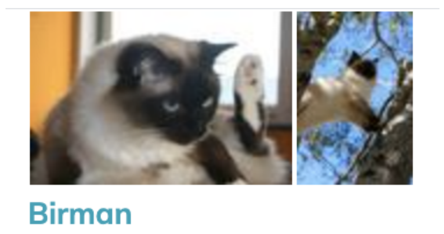
2.Cat Breeds



Family	Count
Cat	2371
Dog	4978
Total	7349

3.Total Pets

The dataset includes the images of the cats and dogs in a variety of different backgrounds, with varied lighting, and objects interfering with the image. The average size of each image was 500x353 pixels and they were all JPEG format.



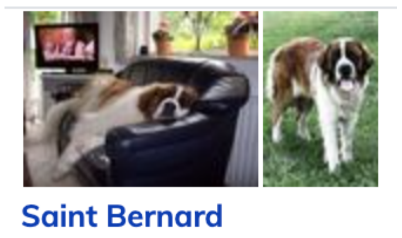


Fig: sample images of the cat and dog in the dataset

<u>2.2 Experimental setup</u>

2.2.1 Setup

We cleaned the data and characterized the information into a dataframe so we can analyze and train the data. The data frame includes the image (as an array - tensor), label (cat or dog), breed (one of 37 mentioned above) and the filename.

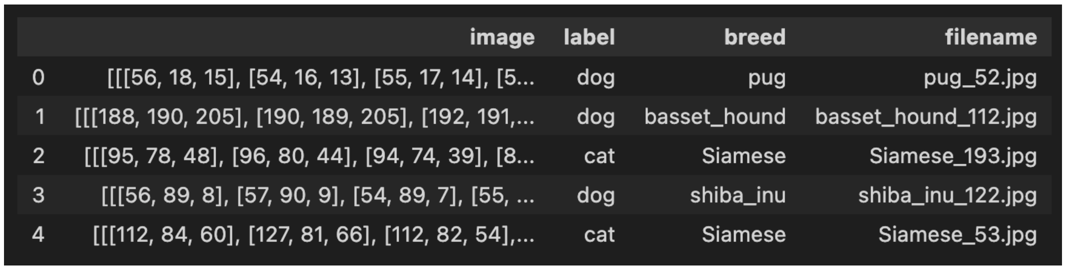
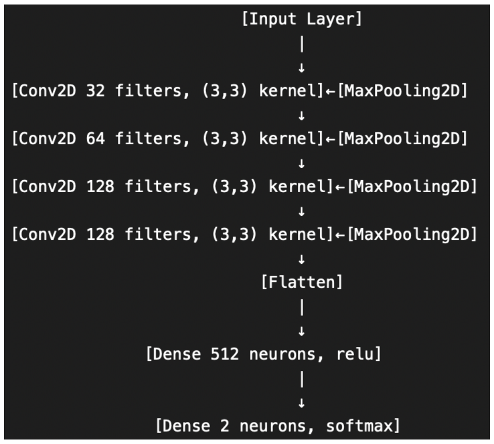


	image	label	breed	filename
0	[[[56, 18, 15], [54, 16, 13], [55, 17, 14], [5...	dog	pug	pug_52.jpg
1	[[[188, 190, 205], [190, 189, 205], [192, 191,...	dog	basset_hound	basset_hound_112.jpg
2	[[[95, 78, 48], [96, 80, 44], [94, 74, 39], [8...	cat	Siamese	Siamese_193.jpg
3	[[[56, 89, 8], [57, 90, 9], [54, 89, 7], [55, ...	dog	shiba_inu	shiba_inu_122.jpg
4	[[[112, 84, 60], [127, 81, 66], [112, 82, 54],...	cat	Siamese	Siamese_53.jpg

We then split the data into the train, test and validation datasets (train: 60, test: 20 and val: 20).

We then defined the base model - CNN with the following architecture:



```
                    [Input Layer]
                          |
                          ↓
[Conv2D 32 filters, (3,3) kernel]←[MaxPooling2D]
                          ↓
[Conv2D 64 filters, (3,3) kernel]←[MaxPooling2D]
                          ↓
[Conv2D 128 filters, (3,3) kernel]←[MaxPooling2D]
                          ↓
[Conv2D 128 filters, (3,3) kernel]←[MaxPooling2D]
                          ↓
                      [Flatten]
                          |
                          ↓
                [Dense 512 neurons, relu]
                          |
                          ↓
                 [Dense 2 neurons, softmax]
```

We created the train and validation generator for the images and fit the model using it.

2.2.2 Models we are going to run

1. **Convolutional Neural Network:** A deep neural network commonly used for image classification and object recognition tasks, which typically consists of convolutional, pooling, and fully connected layers.

2. **Ensemble Convolutional Neural Network:** A combination of multiple convolutional neural networks that work together to improve the accuracy of image classification.
3. **ResNet-based Image Classification Model:** A convolutional neural network architecture that uses residual connections to improve the performance of deep networks for image classification tasks. *(The model is pre-trained with the ImageNet dataset, which contains millions of labeled images.)*
4. **Vision Transformer:** A neural network architecture that uses self-attention mechanisms to process sequential inputs, which has shown impressive performance on various natural language processing tasks. *(This model has been pre-trained with the ImageNet dataset for image classification tasks.)*
5. **Ensemble Model based on Pre-trained Models:** A combination of multiple pre-trained models including ResNet-152, DenseNet-161, and VGG-19, working together to improve the accuracy of image classification.

2.2.3 Parameters

We have varied parameters for each model, specifications are visible in the code itself. However, for the base model and the best model the parameters are mentioned below:
Base Model:

- **Number of filters:** The number of filters used in each convolutional layer. In this model, the first convolutional layer has 32 filters, the second has 64 filters, and the third and fourth convolutional layers have 128 filters each.
- **Filter size:** The size of the filters used in each convolutional layer. In this model, all the filters have a size of 3x3.
- **Activation function:** The activation function used in each layer. In this model, the ReLU activation function is used for all the convolutional layers and the dense layers, while the softmax activation function is used in the final output layer.
- **Input shape:** The shape of the input images. In this model, the input images are expected to have a shape of 224x224 with 3 color channels (RGB).
- **Number of units:** The number of units in the dense layers. In this model, the dense layer following the flattened output has 512 units, while the final output layer has 2 units.
- **Optimizer:** The optimization algorithm used during the training process. In this model, the RMSprop optimizer is used.
- **Loss function:** The loss function used during the training process. In this model, the binary cross-entropy loss function is used.
- **Metric:** The performance metric used to evaluate the model during the training process. In this model, the 'acc' metric (accuracy) is used.

2.2.4 Computing environment

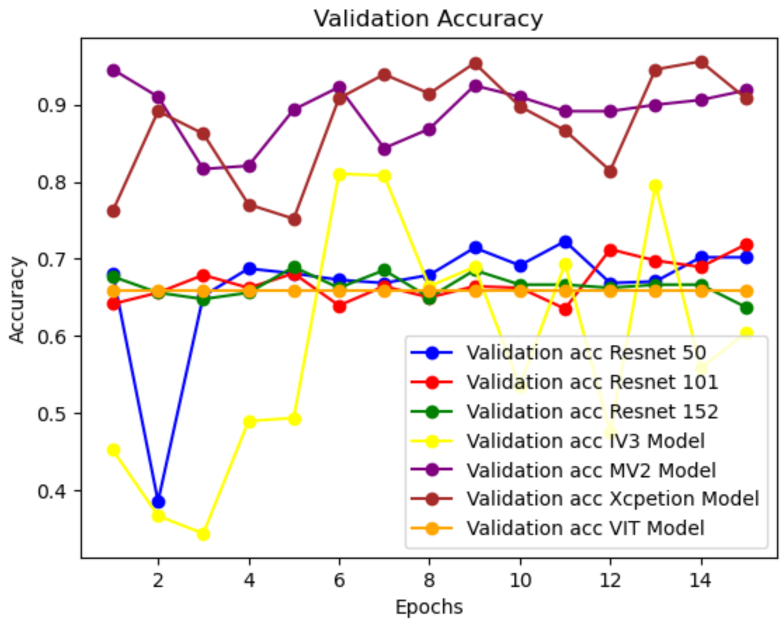
We computed and ran it locally through the Jupyter notebook in the folder. However, it can be run through google colab or discovery cluster (or any other server).

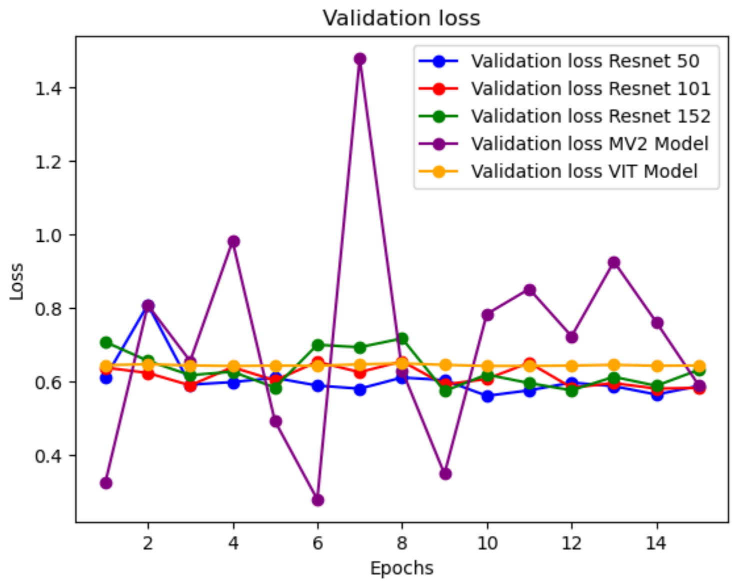
2.3 Problem setup (Network Architecture)

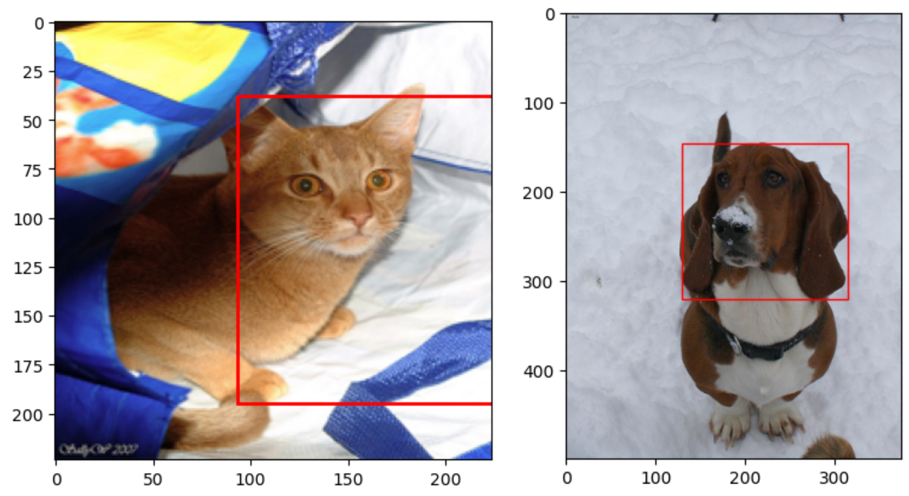
a. For our convolutional neural network, we used 4 convolutional layers with 32, 64, 128, and 128 layers. We then followed this up with 2 dense layers containing 512 and 2 units. We fed our reshaped images into this model and then received our baseline results.
b. In our Vision Transformer model, we first break down the image into a series of patches. We then flatten these patches into a sequence of vectors that are processed by transformer blocks. This process of breaking down the image into patches and recognizing patterns within these smaller parts is what allows the ViT model to recognize subjects of images even when they have been shifted positionally. The feed-forward network of the transformer blocks then applies non-linear transformation to the mechanism, causing further processing of the feature representation of the image.
c. The ResNet model works slightly differently than the ViT model. The ResNet architecture has several residual blocks containing multiple convolutional layers. The output of these blocks is the sum of its input and the output of the convolutional layers it used. Our ResNet model was also pre-trained on data from the ImageNet dataset and then we tuned the model by adding our own output layers.
d. We also have an ensemble model consisting of multiple CNNs and an ensemble model that is a combination of multiple pre-trained models. Both of these work by doing a majority voting of the output probabilities from each model they are composed of and the most popular choice is the predicted output. This output is then a representation of the best guess of all the models combined.
    i. We also fit these pre-trained models on our dataset (fine-tuning them). These were:
        1. ResNet-152: A deep residual neural network architecture that includes 152 layers, designed to address the vanishing gradient problem during training.
        2. DenseNet-161: A densely connected convolutional neural network architecture that includes 161 layers, where each layer receives feature maps from all preceding layers in a feed-forward fashion.
        3. VGG-19: A convolutional neural network architecture with 19 layers, characterized by its simplicity and uniformity, where each layer has a fixed kernel size of 3x3 and a fixed stride of 1.
        4. IV3 (Inception V3): A deep convolutional neural network architecture that employs a combination of parallel convolutional filters with different kernel sizes to capture features at different scales.
        5. MV2 (MobileNetV2): A lightweight convolutional neural network architecture designed for mobile and embedded vision applications, which uses depth wise separable convolutions to reduce the number of parameters and computation.
        6. Exception Model: A deep neural network architecture based on the idea of dynamic routing between capsules, which are groups

of neurons that represent the pose and instantiation parameters of an entity in an image.

3. **Results:** Describe the results from your experiments.
   a. This chart shows the different training accuracies and losses for all the models that we trained our dataset on. The MV2 and Exception models perform the best as they provide the highest accuracy most consistently. This is because MobileNetV2 provides higher accuracies through efficient depthwise separable convolutions and other optimization techniques, while Exception Model provides higher accuracies through dynamic routing between capsules, allowing different parts of an image to interact and influence each other.



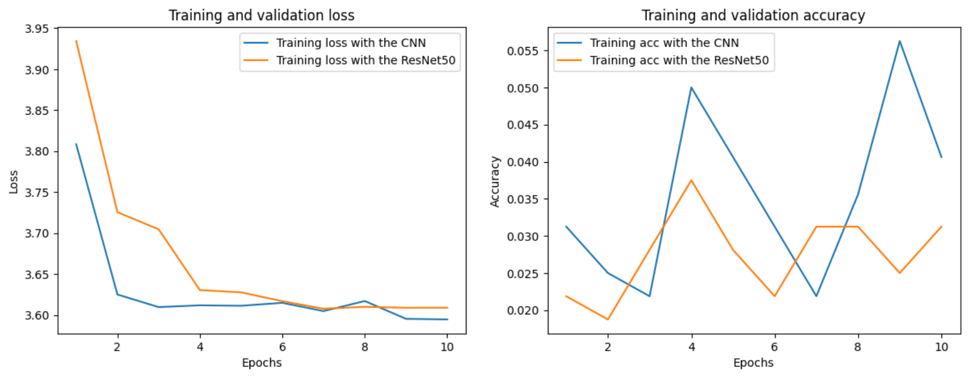




A few attempts at using bounded boxes to capture the body of a cat/dog

b. One of our parameter choices was to resize all inputs images to (224, 224, 3). We made this choice by looking at other similar models and decided that this image shape seemed to be standard. This also allows a smaller input size, and therefore less parameters which means the models we made were computationally efficient to train and less likely to overfit. We then decided to use the ReLu activation function to introduce some non-linearity in our models so it could handle more complex relationships with data. We determined the filter size of 3x3 in our base CNN model based on what seemed to be standard in other models. We also tried varying the number of filters in each layer to see the impact it would have on our model. The loss function we found most appropriate was the binary cross entropy loss since this was a classification problem with two outcomes.

c. We additionally, also looked at the training and validation loss and accuracy with the CNN to get the classification on the 37 classes of the breeds.



4. **Discussion:**

The results of the pretrained Resnet models are fairly accurate with validation accuracy ranging from .6 to .75. Some existing approaches for image classification of dogs and cats is the Asirra CAPTCHA which uses a combination of support-vector machine classifiers trained on input images. This existing approach has an accuracy of 82.7% and specifically focused on the color and texture features of images.

The results of the Vision Transformer model were also fairly accurate with accuracy of about .65 and loss of about .6. This pretrained model might benefit from

The results of the ensemble model created from three CNNs and the other ensemble model created from three pretrained image classifications models are both very poor and performed worse than expected. There might be a problem where the model is overfit on the training data and once the ensemble model tries to combine the predictions of the underlying models on the testing data, it falls apart and can't accurately predict the label of the testing data.

The results of the pretrained model MV2 and Xception model on their own are very good, however, as mentioned before since the ensemble made of these models does very poorly on the testing data, it is likely that these models are overfit and not as accurate as the training and validation accuracy would portray.

The results of the 37 classes of breed prediction was a very difficult problem to solve, especially with the convolutional neural network giving us an accuracy of being in between about 2.5% and about 5%. This makes sense because there are a lot of breeds we could classify into. We additionally tried the classification of the breeds on the pre-trained ResNet model that was trained using the imagenet dataset. The training gave us a similar accuracy of about 3-5%.

Additionally, we tried to predict the bounding boxes on the cats and dogs images and found that the bounding box prediction decreases its loss significantly throughout the training however, it does not most accurately predict the boxes as there is fairly little training data for the bounding boxes which leads to overfitting and incorrect learning predictions. However, we only trained it using CNN. We believe that training it over a pre-trained model to predict the bounding boxes may be far more efficient and improve the accuracy of the model.

5. Conclusion:

We have tried multiple models, to try to classify whether an input image is of a cat or a dog and models to classify into the different breeds of the image provided. Our methods have ranged from simple convolutional neural networks, to an ensemble of CNNs, as well as multiple pre-trained models for image classification. In addition to classifying for the label, we also attempted to work with trimaps of these input images to make bounding boxes for the images to section off the part of the image that is of the face of the cat/dog to make classification further much easier. Upon further fine-tuning of the models, we should be able to find the cat or dog with the bounding boxes, determine whether it is a cat or a dog and finally, predict the breed of the pet.

6. References:

a. https://medium.com/@alexppppp/how-to-train-an-ensemble-of-convolutional-neural-networks-for-image-classification-8fc69b087d3
b. https://viso.ai/deep-learning/vision-transformer-vit/
c. https://arxiv.org/abs/2010.11929

In-class discussion schedule (TBD)

Instructions for in-class discussion

- Present the current progress of the project to the other teams who got assigned to the same group.
- Provide critical feedback to the presenting teams.
- Food will be provided for facilitating the discussion. 🙂

In-class discussion schedule (In class on April 13)

Instructions for in-class discussion

- Present the current progress of the project to the other teams who got assigned to the same group.
  - Either expand on the slides from the in class project presentations or go through the GitHub readme document.
- Provide critical feedback to the presenting teams.
- Snacks will be provided for facilitating the discussion.
- During each discussion group, each group should take turns presenting the project to the other groups in the same discussion group. **As a rule of thumb, each presentation should take about 5 minutes, to allow the other groups enough time to complete their presentations**.

Discussion Part I (11:45am - 12:15pm)

Discussion Group at the Left-Front row of the classroom:

Group 1, Group 2, Group 3, Group 4, Group 5, Group 6

Discussion Group at the Right-Front row of the classroom:

Group 7, Group 8, Group 9, Group 10, Group 11, Group 12

Discussion Group at the Left-Back row of the classroom:

Group 13, Group 14, Group 15, Group 16, Group 17, Group 18

Discussion Group at the Right-Back row of the classroom:

Group 19, Group 20, Group 21, Group 22, Group 23, Group 24

Discussion Part II (12:20pm - 12:50pm)

Discussion Group at the Left-Front row of the classroom:

Group 24, Group 6, Group 7, Group 13, Group 9, Group 1

Discussion Group at the Right-Front row of the classroom:

Group 3, Group 21, Group 17, Group 20, Group 12, Group 8

Discussion Group at the Left-Back row of the classroom:

Group 16, Group 4, Group 2, Group 23, Group 11, Group 19

Discussion Group at the Right-Back row of the classroom:

Group 14, Group 5, Group 15, Group 22, Group 10, Group 18

Discussion Part III (12:50pm - 1:15pm)

Discussion Group at the Left-Front row of the classroom:

Group 5, Group 9, Group 14, Group 23, Group 6, Group 2

Discussion Group at the Right-Front row of the classroom:

Group 12, Group 13, Group 3, Group 17, Group 11, Group 22

Discussion Group at the Left-Back row of the classroom:

Group 7, Group 20, Group 8, Group 19, Group 15, Group 4

Discussion Group at the Right-Back row of the classroom:

Group 21, Group 10, Group 24, Group 1, Group 18, Group 16

Wrap-up (1:15pm - 1:25pm)

Vote for the best presenting team + completing the TRACE survey.

List of groups with student names

Group 1 Yash Bhora and Josh Peirce

Group 2 Angela Rhee, Jasmine Liu, and Ruiming Li

Group 3 Manon Le Donne, Valeria Loria, and Jessica van de Ven

Group 4 Haley Matuszak, Byron Phan, and Jasmine Chiou

Group 5 Abigail Sodergren and Ashley McDermott

Group 6 Allen Ye

Group 7 Caitlin Chao and Srishti Kundu

Group 8 Aditya Chenji, George Bikhazi, Bartosz Mamro

Group 9 Vivek Divakarla and Travis Debruyn

Group 10 Wenting Yue, Xiaofei Xie, and Yuxi Shen

Group 11 Vedanshi Shah, Vedant Bhagat, and Sid Mallareddygari

Group 12 Andrew Miranda

Group 13 Khoa Le

Group 14 Brant Wesley and Emmett Bergeron

Group 15 John Stern and Logan Pfahler

Group 16 Arlo Valiela, Matt Cerillo, and Madisen Hïïîll

Group 17 Amanda Bell, Tim Wang, and Jasmine Wong

Group 18 David Habboosh, Sean Guiry, and Mohamad El Nayal

Group 19 Matt Kim, Marco Hampel, Enson Soo

Group 20 Dave Budhram, Sarah Costa, and Brandon Onyejekwe

Group 21 Marco Tortolani

Group 22 Jeffrey Wang

Group 23 Adam Belfki and Neel Sortur

Group 24 Deion Smith

Group 25 Mara Hubelbank

Final submission for grading (April 27)

Finally, by April 27, refine the GitHub repository and the accompanying documentation. When we evaluate a project, we focus on the following three criteria.

1. **Relevance to DS4400** (⅓ of total grade; We will assign three levels: Good, Average, Below Average). For example, does the project use any ML models studied during the lectures; Does the project build on ideas or questions discussed in any part of the course? Projects that are irrelevant to the syllabus of this course will be given lower priority during grading.
2. **Clarity of documentation and structure of code repository** (⅓ of total grade; We will assign three levels: Good, Average, and Below Average). For example, is the project documentation well-written and well-structured; Is the code repository easy to navigate?
3. **Novelty of the project** (⅓ of the total grade; We will assign three levels: Good, Average, and Below Average). For example, does the project address a problem that has not been properly addressed during the lectures but is still highly relevant? Does it go above and beyond the materials of the lectures?

Datasets

Data Repositories

- The UCI Machine Learning Repository: https://archive.ics.uci.edu/ml/datasets.php
- Kaggle competitions: https://www.kaggle.com/datasets
- OpenML: https://www.openml.org/search?type=data&sort=runs&status=active
- Microsoft malware classification dataset: https://www.kaggle.com/c/malware-classification
- Government datasets: https://catalog.data.gov/dataset
- https://github.com/awesomedata/awesome-public-datasets
- Sports datasets: https://lionbridge.ai/datasets/20-free-sports-datasets-for-machine-learning/

Image Data

- Colored MNIST: https://github.com/facebookresearch/InvariantRiskMinimization/blob/main/code/colored_mnist/main.py

- Street view house numbers: http://ufldl.stanford.edu/housenumbers/
- CIFAR-10 and CIFAR-100: https://www.cs.toronto.edu/~kriz/cifar.html
- Face recognition: http://www.face-rec.org/databases/
- Adience dataset: https://www.kaggle.com/orion99/adience-age-gender-prediction-aligned-faces
- (Image + Text) Flickr 8K dataset: https://www.kaggle.com/adityajn105/flickr8k
- Build a classifier for road signs using a subset of images extracted from the German Traffic Sign Dataset. https://www.kaggle.com/meowmeowmeowmeowmeow/gtsrb-german-traffic-sign
- Build a classifier to distinguish different dog breeds using the Stanford dog dataset: https://www.kaggle.com/jessicali9530/stanford-dogs-dataset
- Facial Expression Detection and Classification: https://github.com/spenceryee/CS229
- Multiclass scene Classification: https://www.kaggle.com/puneet6060/intel-image-classification
- Classify plant and animal species using a subset of the iNaturalist dataset: https://www.kaggle.com/c/inaturalist-2019-fgvc6
- Food image classification: https://github.com/ivanDonadello/Food-Categories-Classification
- Crowd Counting: https://www.kaggle.com/fmena14/crowd-counting?select=frames /
- Gender and age detection using Adience dataset and Convolutional Neural Networks: https://www.kaggle.com/orion99/adience-age-gender-prediction-aligned-faces
- Classifying American Sign Language (ASL) images: https://www.kaggle.com/grassknoted/asl-alphabet
- [More challenging] Investigate algorithms for image segmentation using: https://www.robots.ox.ac.uk/~vgg/data/pets/
- https://en.wikipedia.org/wiki/List_of_datasets_for_machine-learning_research

Natural Language Data

- Sentiment prediction: https://github.com/harvardnlp/sent-conv-torch/tree/master/data
- Hugging Face: https://github.com/huggingface/datasets
- News Article Source Classification - https://components.one/datasets/all-the-news-articles-dataset
- Author Identification- https://www.kaggle.com/c/spooky-author-identification/data
- Predict sentiment analysis on movie reviews, using a Kaggle dataset: https://www.kaggle.com/c/sentiment-analysis-on-movie-reviews/data
- Predict sentiment in reviews or business rating using the Yelp review dataset: https://www.yelp.com/dataset

- Predict the type of news an article is from (using, for example, the AG News dataset from Hugginface)
- Learn to answer questions (using, for example, SQuaD dataset from Hugginface)
- Predict whether a bill comes from California or US Congress (using, for example the billsum dataset from Hugging Face)

Graph Data

- Interesting Datasets for Graph Neural Network (Yelp, Amazon, Flickr): https://medium.com/@khangphysix1997/interesting-dataset-for-graph-neural-network-7a6fc792786e
- Graph Neural Networks on Social Networks: https://www.kaggle.com/code/awadelrahman/tutorial-graph-neural-networks-on-social-networks
- Node Classification with Graph Neural Networks: https://keras.io/examples/graph/gnn_citations/ , https://towardsdatascience.com/graph-convolutional-networks-on-node-classification-2b6bbec1d042
- More graph datasets can be directly loaded from Pytorch-geometric: https://pytorch-geometric.readthedocs.io/en/latest/modules/datasets.html#benchmark-datasets

Healthcare, Financial, and Security Data

Healthcare Data

- Detecting Parkinson's disease in an individual using the UCI dataset using: https://archive.ics.uci.edu/ml/datasets/parkinsons/
- Covid-19 Case Surveillance: https://www.kaggle.com/arashnic/covid19-case-surveillance-public-use-dataset
- Detecting diabetic retinopathy: https://www.kaggle.com/c/diabetic-retinopathy-detection/data
- Abnormality detection based on musculoskeletal radiographs: https://stanfordmlgroup.github.io/competitions/mura
- Prediction of onset of diabetes: https://www.kaggle.com/uciml/pima-indians-diabetes-database
- Chest Xray Image Classification: https://www.kaggle.com/paultimothymooney/chest-xray-pneumonia

Financial Data

- Design a loan default prediction model using the following Kaggle dataset: https://www.kaggle.com/wendykan/lending-club-loan-data
- Credit Card Fraud Detection: https://www.kaggle.com/kartik2112/fraud-detection?select=fraudTrain.csv
- Predict who will get a mortgage using the following dataset: https://www.consumerfinance.gov/data-research/hmda/
- Predict earnings and debt of college graduates using the College Scorecard Data: https://collegescorecard.ed.gov/data/

Security Data

- Evaluate different adversarial attacks against ML classifiers for an application of your choice. There is a library with various implementations of attacks at testing time: https://github.com/tensorflow/cleverhans
- Predict if a machine will be hit by malware using the Microsoft dataset: https://www.kaggle.com/c/microsoft-malware-prediction

Other

- Predict the probability an NBA/NHL player scores when shooting from a given location: https://lionbridge.ai/datasets/20-free-sports-datasets-for-machine-learning/
- Rain prediction in Australia: https://www.kaggle.com/jsphyg/weather-dataset-rattle-package
- Music Genre Classification using audio files, spectrograms and extracted audio features: (https://www.kaggle.com/andradaolteanu/gtzan-dataset-music-genre-classification)
- Tensorflow Speech Recognition: https://www.kaggle.com/c/tensorflow-speech-recognition-challenge
- https://towardsdatascience.com/how-to-create-a-chatbot-with-python-deep-learning-in-less-than-an-hour-56a063bdfc44
- New York City Airbnb price prediction: https://www.kaggle.com/dgomonov/new-york-city-airbnb-open-data
- Video Game Sales Prediction: https://www.kaggle.com/gregorut/videogamesales

Resources

NEU Discovery

Use this form to request access to the Discovery cluster or to update your sponsor information. You must specify the name of your current sponsor and your affiliation with the University. You must also read and accept the terms of use before you can submit the form: https://service.northeastern.edu/tech?id=sc_cat_item&sys_id=0ae24596db535fc075892f17d496199c.

First enter your details, then In Affiliation with Northeastern University checkbox, select Grad Student. In the University Sponsor checkbox, type/select professor's name and in Gaussian select Yes.Then submit the form, within 24 hours you will receive confirmation mail from ITS.

Discovery is a high performance computing resource for the Northeastern Student Community. Discovery hosts multiple GPUs with RAM ranging from 128-512 gb. You can allocate only one GPU per job and need to get approval from the Research Center for additional GPUs. Most projects do not need more than one GPU.

Initializing a job with GPU:

- Use srun or sbatch command to initialize a job and we will demonstrate using srun as it is the simplest among srun and sbatch
- Step 1: Login to discovery using your student credentials as shown here.

- Step 2: Initialize a job using the srun command as shown below
  - `srun --partition=gpu --nodes=1 --pty --export=All --gres=gpu:1 --mem=1G --time=00:30:00 /bin/bash`
  - You must use gres option to allocate a GPU and you can set time for the job using the time options --time with a maximum time limit of 8 hours.

- Step 3: Load cuda and anaconda resources
  - The steps hereafter are specific to the pytorch environment, equivalent steps for a tensorflow environment can be found here.
  - Cuda and Anaconda are pre-installed on discovery and need to be loaded using the following commands
    - module load cuda/11.0
    - module load anaconda3/3.7

- Step 4: Creating a virtual environment using conda
  - `conda create --name pytorch_env python=3.7 anaconda`

- Step 5: Accessing the virtual environment and installing pytorch
  - `conda activate pytorch_env`

- Step 6: Installing pytorch in the virtual environment
  - `conda install pytorch torchvision torchaudio cudatoolkit=11.0 -c pytorch`

- Step 7: Testing the installation

  - After the installation is complete, running the command `python -c 'import torch; print(torch.cuda.is_available())'` should print True.

You need to follow the installation steps only once and then skip steps 4, 6 and 7 for initializing jobs in the future. You can find the documentation for additional resources here.

Google Colab

https://colab.research.google.com/

- To use GPU, go to the "Runtime" dropdown menu, select "change runtime type" and then select GPU in the hardware accelerator drop-down menu.

- For large datasets, we recommend uploading it to google drive. To mount drive simply run the following command-
```
from google.colab import drive
drive.mount('/content/gdrive')
```

- Datasets can also be directly downloaded and unzipped using `!wget and !unzip` commands

- To Install any library use - !pip command
  For example - `!pip3 install torch torchvision`

Tutorial:
1. https://towardsdatascience.com/getting-started-with-google-colab-f2fff97f594c
2. https://www.tutorialspoint.com/google_colab/your_first_colab_notebook.htm