DEEP LEARNING FOR

COMPUTER VISION

**Fall-2023**

**AML Report - Rishitha Reddy Muddasani**

# Executive Summary

Our project's goal is to create a new convolutional neural network from the scratch that is intended only for computer vision applications. The "Dog-vs-Cats" dataset on Kaggle is the source of the dataset we are using. Because of the limited amount of data at our disposal, creating a useful model is a challenging task. Convolutional neural networks, or convnets is a popular deep learning models that have shown to be extremely effective in computer vision tasks. Convnets capacity to recognize and identify spatial patterns in pictures is one of its main advantages. They are therefore excellent at tasks like segmentation, object identification, and picture recognition.

We think our convnet model can still yield good results with the limited amount of accessible data. This might be due to convolutional neural networks capacity to acquire knowledge and generalize from small datasets by extracting and identifying relevant features from images. Our model will be trained using a limited dataset, fine-tune it using transfer learning methods, and its performance will be assessed with suitable assessment criteria. Ultimately, we want to create a convolutional neural network that can efficiently and accurately categorize images from the "Dog-vs-Cats" dataset using a small quantity of input.

# Problem

# The objective of the Cats-vs-Dogs dataset binary classification task is to determine if an image is part of the dog or cat class.

# Techniques

## Dataset:

The Cats-vs-Dogs dataset is 543 MB in size (compressed) and includes 25,000 images of dogs and cats (12,500 from each class). Three subsets will be included in the new dataset we construct after downloading and unzipping it: a training set with 1000 samples of each class, a validation set with 500 samples of each class, and a test set with 500 samples of each class. We have to expand our neural network's size since the problem we are working on is more complicated and requires a larger picture. We will extend our current Conv2D + MaxPooling2D design by one stage to achieve this. This ensures that the feature maps are not too large when we get to the Flatten layer while simultaneously increasing the capacity of the network. First, our input images are 150x150 in size. The feature maps get smaller as we move up the network's layers, reaching 7x7 just before the Flatten layer. The input size selected is a bit arbitrary, but it fits the task at hand.

## Preprocessing:

* Read the picture files.
* Decode the JPEG content to RBG grids of pixels.
* Convert these into floating point tensors.
* Rescale the pixel values (between 0 and 255) to the [0, 1] interval (as you know, neural networks prefer to deal with small input values).

## Data Augmentation:

To enhance our model's accuracy, we plan to employ data augmentation methodologies. Through random changes, data augmentation creates new data from the given training samples, allowing us to get excellent results even with small datasets. Because of this, the model will never see the same image again while being trained, which enhances its capacity for generalization. We want to randomly perform operations to the images in the training set, such as flipping, rotating, and zooming, to accomplish our particular purpose. By doing this, we may produce different versions of the already-existing photos, broadening the dataset's variety and enhancing our model's robustness.

# Pre-trained model:

A pretrained network may be used as a generic model and its features used to a wide range of computer vision applications if the original dataset is large and diverse. One of the main advantages of deep learning over other machine learning methods is its capacity to transfer learnt characteristics across various tasks. Using the ImageNet dataset, which has 1.4 million annotated pictures and 1,000 distinct classes, a large convolutional neural network that has been trained may be analyzed as an example. The dataset comprises many animal classifications, including different dog and cat breeds. VGG16, a popular and straightforward convnet design for ImageNet, is the name of this network's architecture.

Feature extraction and fine-tuning are the two main ways to use a pretrained network. To improve upon the findings, we will employ feature extraction in this case, initially without data augmentation and then with data augmentation.

**Results:** The table below shows the accuracy and validation loss for each approach.

TABLE FOR MODEL FROM SCRATCH

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Train Size | Test Size | Validation Size | Data Augmentation | Train Accuracy(%) | Validation Accuracy  (%) |
| 1000 | 500 | 500 | NO | 74.2 | 69.5 |
| 1000 | 500 | 500 | YES | 69.6 | 66.0 |
| 1500 | 500 | 500 | NO | 84.0 | 74.3 |
| 1500 | 500 | 500 | YES | 71.2 | 69.5 |
| 1500 | 1000 | 500 | YES | 83.7 | 66.2 |
| 1500 | 1000 | 500 | NO | 66.9 | 63.9 |

TABLE FOR PRE-TRAINED MODEL

|  |  |  |
| --- | --- | --- |
| Data Augmentation | Train Accuracy(%) | Validation Accuracy(%) |
| NO | 99.8 | 97.8 |
| YES | 96.2 | 97.4 |

The sample sizes for the train, test, and validation sets are presented in the tables above along with the model settings. We provide findings with and without data augmentation for the scratch-built model, as well as results for models trained with varying train and validation sizes or with an increase in train size. We examine the validation accuracy and accuracy for the pre-trained model with and without data augmentation.

Based on the findings, we can see that the models consistently trained with and without data augmentation were unable to outperform the others. The accuracy of the model can also be increased by expanding the training set or modifying the validation set's dimensions. Data augmentation did not increase the pre-trained model's accuracy or validation accuracy, as can be shown by comparing the two sets of data. When working with a limited amount of training data, pre-trained models often perform better than models created from scratch.