# Assignment 2
# COVOLUTION

**Executive Summary:**

Our project's goal is to create a brand-new convolutional neural network from scratch that is intended just for computer vision applications. The "Dog-vs-Cats" dataset that is published on Kaggle is the source of the dataset we are using. Due to the small amount of data we have, creating an efficient model is difficult.

Convolutional neural networks, often known as convnets, are a well-liked class of deep learning models that have excelled in computer vision tasks. Convnets' capacity to recognize and understand spatial patterns in pictures is one of its main advantages. They are hence ideally suited for jobs like segmentation, object identification, and picture recognition.

Considering the scant amount of data, we think that our convnet model can still generate accurate findings. This is because convnets can extract and recognize pertinent characteristics from pictures to learn and generalize from tiny datasets. Our model will be trained on the small dataset, then refined using transfer learning strategies, and its performance will be verified using suitable assessment criteria. Our overall objective is to create a convolutional neural network that can accurately and efficiently categorize photos from the "Dog-vs-Cats" dataset using a small quantity of training data.

**Problem:**

Predicting whether a picture belongs to the dog class or the cat class is the objective of the Cats-vs-Dogs dataset, a binary classification issue.

**Techniques:**

**Dataset:**

The Cats-vs-Dogs dataset is 543MB in size and comprises 25,000 photos of dogs and cats, 12,500 of each type (compressed). As soon as it has been downloaded and uncompressed, we will divide it into three subsets: a training set with 1000 samples of each class, a validation set with 500 samples of each class, and lastly a test set with 500 samples of each class. We need to expand the capacity of our neural network due to the higher picture size and more complicated nature of the challenge we are working on.

We will extend the current Conv2D + MaxPooling2D design by one stage to achieve this. In addition to increasing network bandwidth, this will make the feature maps smaller so that they won't be too big when we get to the Flatten layer. As we go through the network layers, the feature maps steadily get smaller until they are 7x7 just before the Flatten layer. Our input photos are originally 150x150 in size. Although this selection of input size is rather random, it is suitable for the particular case.

**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:**

We plan to employ data augmentation approaches to improve the model's accuracy. Data augmentation allows us to generate more data from the current training samples by random changes, allowing us to get excellent results even with little datasets. As a consequence, during training, the model will never see the same image twice, which enhances its generalization capacity.

We want to randomly apply modifications to the photos in the training set for our particular job, including flipping, rotating, and zooming. By doing this, we may alter the already-existing photos, broadening the dataset's variety and enhancing the model's resilience.

**Pre-trained model:**

A pretrained network may be used as a general model and its characteristics can be used to several different computer vision applications assuming the original dataset was 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.

As an illustration, let's take a look at a sizable convolutional neural network that was trained using the 1.4 million annotated pictures and 1,000 distinct classes that make up the ImageNet dataset. Many animal species are represented in this collection, including different cat and dog breeds. This network's design is known as VGG16, which is a straightforward and popular convnet architecture for ImageNet.

Using a pretrained network may be done in two ways: feature extraction and fine-tuning. In order to get even better results, we will first perform feature extraction without data augmentation and then feature extraction with data augmentation.

**Results:** The accuracy and validation loss for each strategy are displayed in the table below.

**Table For Model from Scratch:**

| **Train Size** | **Test Size** | Validation Size | **Data Augmentation** | **Train Accuracy(%)** | **Validation Accuracy(%)** |
|---|---|---|---|---|---|
| 1000 | 500 | 500 | NO | 81.80 | 73.10 |
| 1000 | 500 | 500 | YES | 66.50 | 70.40 |
| 1500 | 500 | 500 | NO | 85 | 77.10 |
| 1500 | 500 | 500 | YES | 67.40 | 63 |
| 1500 | 1000 | 500 | YES | 85.37 | 73.50 |
| 1500 | 1000 | 500 | NO | 71.47 | 73 |

**Table For Pre-Trained Model:**

| **Data Augmentation** | **Train Accuracy(%)** | **Validation Accuracy(%)** |
|---|---|---|
| **NO** | 99.77 | 97.50 |
| **YES** | 96.97 | 97.50 |

The sample sizes for the train, test, and validation sets are shown in the tables above along with the model settings. Results with and without data augmentation, for models trained with an increase in train size, or for models trained with various train and validation sizes, are included for the model created entirely. We compare the accuracy, validation accuracy, and with and without data augmentation for the pre-trained model.

The results show that the models that were regularly trained with data augmentation were unable to outperform those that were trained without it. The accuracy of the model is also enhanced by expanding the training set or modifying the size of the validation set. By contrasting the pre-trained model with and without data augmentation, we can see that the model's accuracy and validation accuracy did not increase as a result of the data augmentation. Pre-trained models often perform better overall than models that are created from scratch, especially when coping with a dearth of training data.