**REPORT**

***Executive Summary:***

The target of this project will be to craft, train and use a specific convolutional neural network that will be employed in computer vision applications. The dataset I have used is actually a subset of Kaggle‘s “The Dog-vs-Cats” dataset. Inadequate data is another main barrier that can limit the usability of a predictive model as desired.

Convolutional neural networks, or CNNs for short, are currently one of the most utilized deep-learning models that have demonstrated particularly high efficiency in machine perception issues. Recognition and memorization of spatial patterns in the images is the advantage in which they excel the most. This is why they are often chosen for image recognition, object detection, and segmenting tasks. Data deficiency is still hanging out there, yet, the authors have faith that the model will provide accurate and good results. First, this particularity is based upon the fact that the model of learning is capable of learning from the smaller datasets as it identifies and comprehends image features that are relevant. Training of the model with built-in training function based on small datasets, transfer learning techniques and metrics for evaluation are some steps that are to be used.

A main goal here is to train and develop a convolutional neural network system which can categorize "Dog-vs-Cats" images with high accuracy even using the limited dataset.

# Problem

The Cats-vs-Dogs dataset is a binary classification problem where the goal is to predict whether an image belongs to the dog class or a cat class.  
  
**Techniques**

**Dataset Details:**

The Cats-vs-Dogs dataset has 25,000 pictures, with half being dogs and the other half cats. When you download and unzip the dataset, it takes up 543 mb of space. I sorted this dataset into three groups:

* A group for training, with 1,000 pictures of each type (dogs and cats).
* A group for checking how well the model is doing, with 500 pictures of each type.
* A group for final testing, also with 500 pictures of each type.

Because the pictures are big and the problem is kind of complicated, I decided to make the neural network bigger. I added another layer to the design, which includes a combination of Conv2D and MaxPooling2D. This not only increases the network's power but also makes the picture pieces smaller as they go through the layers. By the time they reach the Flatten layer, they're just the right size, about 7x7. The starting pictures are 150x150 in size, which might seem random, but it works well for this problem.

**Preprocessing:**

Image File Processing:

* Look at the picture files.
* Turn the JPEG images into grids of colored dots.
* Change these grids into numbers with decimal points.
* Make sure the numbers are in a good range for computers to understand, like going from really dark to really bright. We do this by making sure the numbers go from 0 to 1 instead of 0 to 255. This helps the computer work better with the pictures.

**Data Augmentation:**

To make the model better at its job, we can use something called data augmentation. It's like giving the model more examples to learn from, even if we don't have many pictures to start with. We do this by making changes to the pictures we already have, like flipping them or zooming in. This way, the model gets to see lots of different versions of the same pictures while it's learning.

For this task, I plan to mix things up by randomly changing the pictures in our training set. I'll rotate them, flip them, and zoom in on some of them. This will give the model a bigger variety of pictures to learn from, making it better at figuring out what's in a picture.

**Pre-trained model:**

A pre-trained model is like a ready-made tool that's already learned a lot from a big and diverse dataset. You can use this model for different computer vision jobs because it already knows about lots of different things. This is one of the cool things about deep learning - it can share what it's learned across different tasks.

For example, imagine there's this big neural network that's been trained on a massive dataset called ImageNet. This dataset has millions of pictures labeled with different categories, like different types of animals. One of the popular neural network, architecture for ImageNet is called VGG16, which is very simple and widely-known.

Now, we have two main ways that we can use a pre-trained model for fine-tuning and feature extraction. In my case, I decided to use feature extraction. First, I used the model without changing anything, and then I tried adding some extra tricks to make it even better.

**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 | 79.90 | 71.80 |
| 1000 | 500 | 500 | YES | 70.30 | 70.00 |
| 1500 | 500 | 500 | NO | 86.03 | 70.80 |
| 1500 | 500 | 500 | YES | 72.30 | 72.0 |
| 1500 | 1000 | 500 | YES | 84.87 | 74.9 |
| 1500 | 1000 | 500 | NO | 49.67 | 50.0 |

TABLE FOR PRE-TRAINED MODEL

|  |  |  |
| --- | --- | --- |
| Data Augmentation | Train Accuracy(%) | Validation Accuracy(%) |
| NO | 99.8 | 96.6 |
| YES | 96.87 | 97.15 |

In the tables above, you can see how many examples we used for training, testing, and checking our model's performance. We also tried different setups to see which one works best.

We tested our model in two ways: one with extra help called "data augmentation" and one without. Data augmentation refers adding variations to the training data to help model learn better.

Models trained with data augmentation didn't always do better than those without it. Surprisingly, simply having more training examples or adjusting the size of the validation set helped improve accuracy.

We also compared our model to one that was already partly trained, called a pre-trained model. We did this with and without data augmentation. But no matter what, the pre-trained model didn't do any better. Usually, pre-trained models do better than ones we start from scratch, especially when we don't have a lot of training data.