# Cat & Dog Classification using Convolutional Neural Network in Python

[**Convolutional Neural Networks (CNNs)**](https://www.geeksforgeeks.org/convolutional-neural-network-cnn-in-machine-learning/) are a type of deep learning model specifically designed for processing images. Unlike traditional neural networks CNNs uses **convolutional layers** to automatically and efficiently extract features such as edges, textures and patterns from images. This makes them highly effective for tasks like image classification, object detection and segmentation.

**Implementing Cat & Dog Classification using CNN**

**1. Importing Libraries**

We will import numpy, pandas, matplotlib and scikit learn for this. Also we will import:

* [OpenCV](https://www.geeksforgeeks.org/opencv-python-tutorial/): This is an open-source library mainly focused on image processing and handling.
* **Tensorflow**: It provides a range of functions to achieve complex functionalities.
* **ImageDataGenerator**: Helps in loading and preprocessing images for training.

\_from\_directory

**i**

**2. Importing Dataset**

We will be using Kaggle dataset for this which is in the format of a zip file containing 2 folders : Cat and Dog. Further each folder contains 12500 images of respective animals. So to import and unzip the file we you can run the below code.

**3. Data Visualization**

We will try to understand and visualize some images which have been provided to us to build the classifier for each class. But before that lets see classes present in this dataset.

*Representation of Images in the Dataset*

* The code extracts image paths and loads them using Matplotlib.
* A grid visualization is created using **subplot()**.

**4. Splitting Dataset**

We split the dataset into **training and validation sets**.

* **image\_dataset\_from\_directory:** is used for data augmentation and scaling images.
* The dataset is split into **90% training and 10% validation**.
* **target\_size=(200, 200):**Resizes images to 200×200 pixels.
* **batch\_size=32:** Defines the number of images per batch.

**5. Model Architecture**

The model will contain the following Layers:

* Four [Convolutional](https://www.geeksforgeeks.org/convolutional-block-for-image-recognition/) Layers followed by [MaxPooling](https://www.geeksforgeeks.org/cnn-introduction-to-pooling-layer/) Layers.
* [Flatten](https://www.geeksforgeeks.org/impact-of-image-flattening/) layer to flatten the output of the convolutional layer.
* Then we will have three fully connected layers followed by the output of the flattened layer.
* We have included some [BatchNormalization](https://www.geeksforgeeks.org/what-is-batch-normalization-in-deep-learning/" \t "_blank) layers to enable stable and fast training and a [Dropout](https://www.geeksforgeeks.org/dropout-regularization-in-deep-learning/) layer before the final layer to avoid any possibility of overfitting.
* Final layer is the output layer which has the activation function sigmoid to classify the results into two classes.
* **Conv2D layers:** extract image features like edges, shapes and textures.
* **MaxPooling2D:** reduces image dimensions while retaining important information.
* **BatchNormalization:** helps stabilize training and speed up convergence.
* **Dropout layers:** prevent overfitting.
* **sigmoid activation:** outputs a binary classification as Cat or Dog.

Let’s print the summary of the model’s architecture:

model.summary()

*Model Summary*

This gives us a brief about our model.

**6. Model Compilation & Training**

Now we will compile and train our model.

* We used Binary Crossentropy Loss Function for binary classification problems with Adam optimizer.

The model is working fine with epochs = 10 but we can fine tune hyperparameter for better results.

**7. Model Evaluation**

Let’s visualize the training and validation accuracy with each epoch.

*training and validation accuracy*

The **loss** graph shows fluctuating training and validation losses with a spike in validation loss around epoch 3 showing potential overfitting. The **accuracy** graph reveals that training accuracy improves steadily. Validation accuracy fluctuates indicating that the model may not be generalizing well.

**8. Model Testing and Prediction**

Let’s check the model for random images.

We can see that our model is able to predict images correctly, hence our CNN model to predict cats and dogs in images is working fine. For better performance we can use Transfer Learning and perform hyperparameter tuning. Once we understand the concept of Image Classification now we can try different classification like [Lung Cancer Detection using CNN](https://www.geeksforgeeks.org/lung-cancer-detection-using-convolutional-neural-network-cnn/).