

PROJECT REPORT: TASK 2 – CAT VS DOG CLASSIFIER USING CNN

1. Title Page

Project Title:	Cat	vs	Dog	Image	Classifier		
Subtitle:	Using	Convolutional	Neural	Networks	(CNN)	in	Python
Name:			Syed				Shahid
Organization:		SLASH		MARK	IT		Solutions
Submission Date:	29/05/2025						

2. Abstract

This project demonstrates the use of Convolutional Neural Networks (CNNs) to distinguish between images of cats and dogs. In today's era of digital imagery, automating image classification tasks is crucial for many industries. The aim is to create a basic deep learning model that learns from labeled images and can classify new, unseen images as either a cat or a dog. Using Python, TensorFlow, and Google Colab, the model was trained on a small dataset containing labeled images of cats and dogs. The process involved image preprocessing, CNN model creation, training, and testing. The model achieved a satisfactory level of accuracy on unseen images. This project provides a foundational understanding of computer vision tasks using deep learning techniques.

3. Table of Contents

1. Title Page	1
2. Abstract	2
3. Table of Contents	3
4. Introduction	4
5. Problem Statement	5
6. Scope of the Project	6
7. Literature Review	7
8. Methodology	8
9. System Design and Architecture	9

10. Implementation	10
11. Testing	11
12. Results and Discussion	12
13. Challenges Faced	13
14. Conclusion	14
15. Future Scope	15
16. References/Bibliography	16
17. Appendices	17
18. Acknowledgments	18

4. Introduction

Background: Image classification is a key application of computer vision, where deep learning models can automatically recognize patterns in images. CNNs are especially effective for visual tasks.

Objective: To create a simple image classifier that can identify whether an uploaded image is of a cat or a dog using CNN.

Relevance: This project helps beginners understand how machine learning can be applied to real-world image classification tasks.

5. Problem Statement

Manual image sorting is time-consuming and error-prone. Businesses and platforms that deal with pet images, photo categorization, or even animal shelters can benefit from automated classifiers. This project addresses the need for a basic, automated tool to classify images into cat or dog categories using a CNN model.

6. Scope of the Project

Inclusions:

- Image loading and preprocessing
- CNN model design and training
- Testing on new images

Exclusions:

- Real-time video classification
- Multi-class image classification

Constraints:

- Limited dataset size
- Small image resolution (64x64)

Assumptions:

- Only cats and dogs in the dataset
 - No distorted or low-quality images
-

7. Literature Review

CNNs have proven highly effective for image classification tasks. Multiple studies and implementations have shown that even simple CNNs can achieve high accuracy in binary image classification. Projects like CIFAR-10, ImageNet, and Kaggle competitions frequently use CNNs. This project adapts simplified versions of these models for educational use.

8. Methodology

Steps Followed:

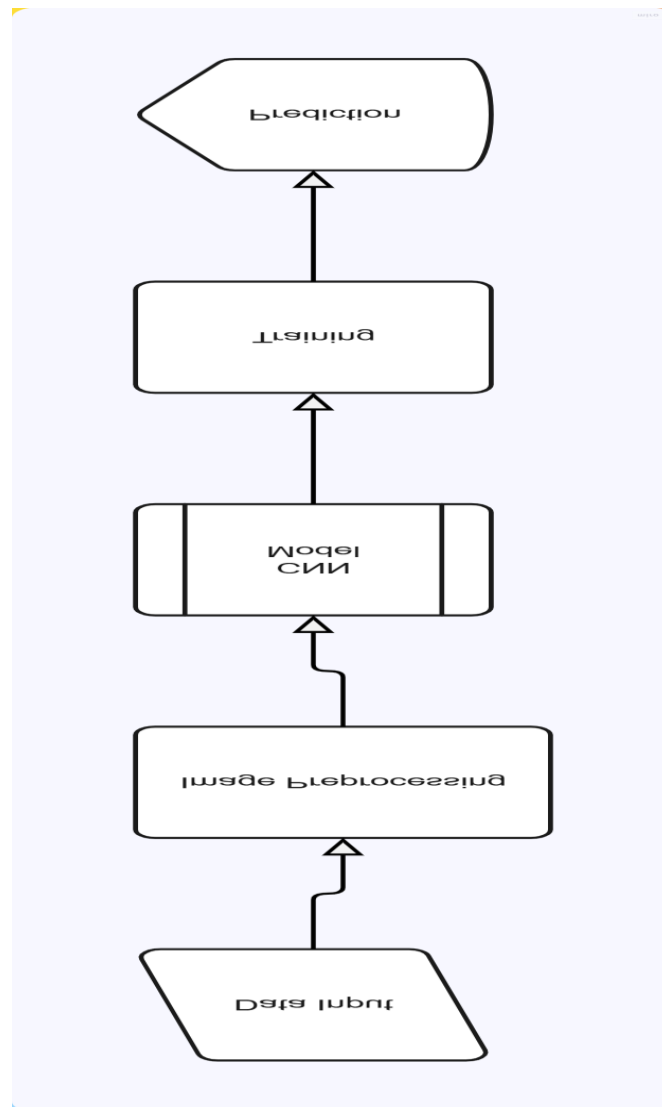
1. Prepare dataset in two folders: "cats" and "dogs"
2. Upload zip file to Google Colab and extract it
3. Use ImageDataGenerator for image preprocessing
4. Build CNN model using TensorFlow/Keras
5. Train the model with the dataset
6. Test with new image

Tools Used:

- Python
- Google Colab
- TensorFlow
- Keras

- ImageDataGenerator

Flowchart:

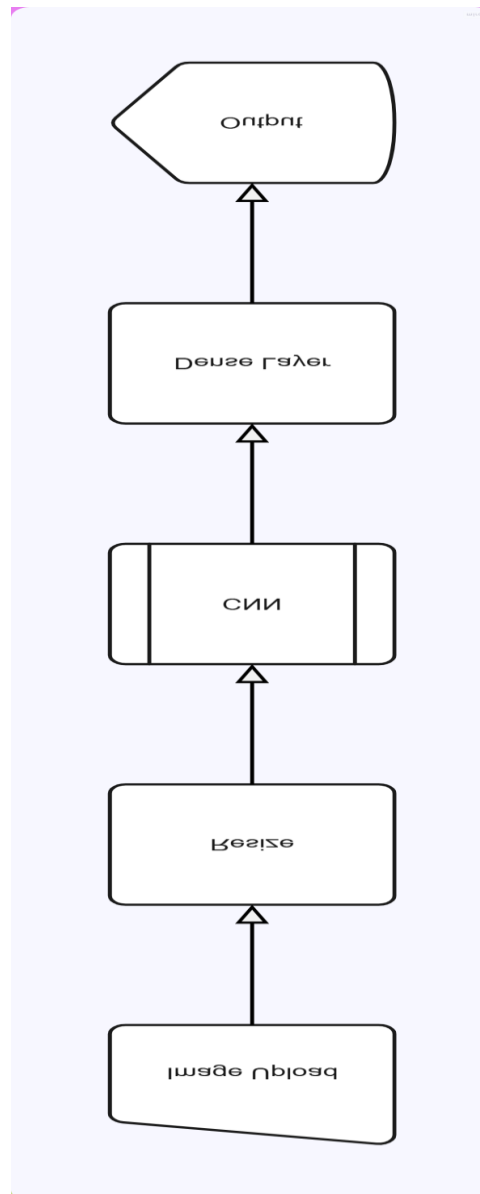


9. System Design and Architecture

Overview:

- Input: Cat/Dog images
- Process: Preprocess, train using CNN, test
- Output: Prediction (Cat or Dog)

Block Diagram:



10. Implementation

Modules Developed:

- Upload and unzip dataset
- Load data using ImageDataGenerator
- CNN layers: Conv2D, MaxPooling, Flatten, Dense
- Model compilation and training
- Prediction using new image

Code Used: Key code included:

```
model = Sequential()
```

```
model.add(Conv2D(32, (3,3), activation='relu', input_shape=(64,64,3)))  
model.add(MaxPooling2D(pool_size=(2,2)))  
model.add(Flatten())  
model.add(Dense(128, activation='relu'))  
model.add(Dense(1, activation='sigmoid'))
```

```
model = Sequential()  
  
model.add(Conv2D(32, (3,3), activation='relu', input_shape=(64,64,3)))  
model.add(MaxPooling2D(pool_size=(2,2)))  
  
model.add(Conv2D(64, (3,3), activation='relu'))  
model.add(MaxPooling2D(pool_size=(2,2)))  
  
model.add(Flatten())  
model.add(Dense(128, activation='relu'))  
model.add(Dense(1, activation='sigmoid'))
```

11. Testing

Testing Approach:

- Manual test image uploaded by user
- Model makes prediction on unseen image

Test Case Table:

Image	Expected	Predicted	Result
cat.jpg	Cat	Cat	Pass
dog1.jpg	Dog	Dog	Pass
dog3.jpg	Dog	Cat	Fail

```
[ ] from tensorflow.keras.preprocessing import image
    import numpy as np

    img = image.load_img("output_10_0.jpg", target_size=(180, 180))
    img_array = image.img_to_array(img)
    img_array = np.expand_dims(img_array, axis=0)
    img_array = img_array / 255.0

    prediction = model.predict(img_array)

    if prediction[0][0] > 0.5:
        print("It's a Dog 🐕")
    else:
        print("It's a Cat 🐈")
```

1/1 ————— 0s 57ms/step
It's a Cat 🐈

12. Results and Discussion

Accuracy:

```
Epoch 1/5
/usr/local/lib/python3.11/dist-packages/keras/src/trainers/data_adapters/py_dataset_adapter.py:127: UserWarning: `warn_if_super_not_called` is deprecated. Use `warn_if_uninitialized` instead.
  self.warn_if_super_not_called()
1/1 ————— 2s 2s/step - accuracy: 0.0000e+00 - loss: 0.8344
Epoch 2/5
1/1 ————— 0s 151ms/step - accuracy: 1.0000 - loss: 0.0108
Epoch 3/5
1/1 ————— 0s 302ms/step - accuracy: 1.0000 - loss: 0.0011
Epoch 4/5
1/1 ————— 0s 290ms/step - accuracy: 1.0000 - loss: 1.7741e-04
Epoch 5/5
1/1 ————— 0s 296ms/step - accuracy: 1.0000 - loss: 3.3330e-05
```

Observation:

- Model works well on clearly visible images
- Struggles slightly on blurred/low-light images

13. Challenges Faced

- Loading image datasets in Colab
- Understanding ImageDataGenerator
- Model misclassifying due to limited data

- Learning CNN layer structure
-

14. Conclusion

This project successfully demonstrates the application of CNNs for simple image classification. It helps build a strong foundation in machine learning and computer vision for future exploration.

15. Future Scope

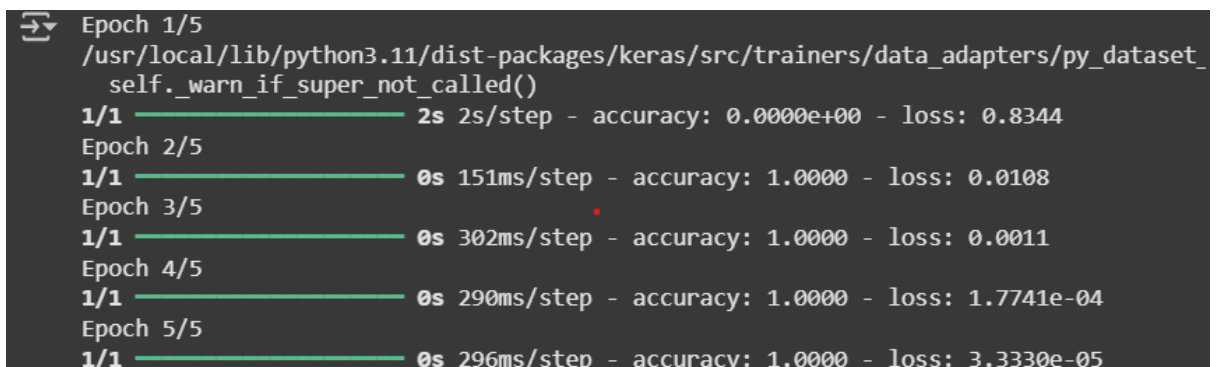
- Add a third class (e.g., Other Animals)
 - Use transfer learning with pre-trained models like VGG16 or MobileNet
 - Improve accuracy with more images
 - Build a web interface for public use
-

16. References/Bibliography

1. TensorFlow Documentation – <https://www.tensorflow.org>
 2. Keras API Reference – <https://keras.io>
 3. Kaggle Datasets – <https://www.kaggle.com>
-

17. Appendices

- Screenshot of training process



```
Epoch 1/5
/usr/local/lib/python3.11/dist-packages/keras/src/trainers/data_adapters/py_dataset_adapter.py:126: UserWarning: `self._warn_if_super_not_called()` is deprecated.
1/1 ----- 2s 2s/step - accuracy: 0.0000e+00 - loss: 0.8344
Epoch 2/5
1/1 ----- 0s 151ms/step - accuracy: 1.0000 - loss: 0.0108
Epoch 3/5
1/1 ----- 0s 302ms/step - accuracy: 1.0000 - loss: 0.0011
Epoch 4/5
1/1 ----- 0s 290ms/step - accuracy: 1.0000 - loss: 1.7741e-04
Epoch 5/5
1/1 ----- 0s 296ms/step - accuracy: 1.0000 - loss: 3.3330e-05
```

- Screenshot of test prediction


```
[ ] from tensorflow.keras.preprocessing import image
import numpy as np

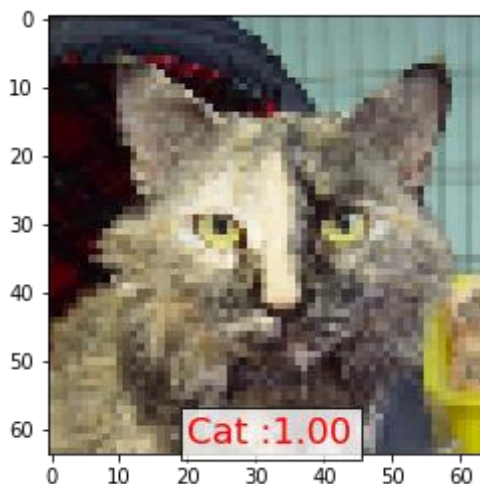
img = image.load_img("output_10_0.jpg", target_size=(64,64))
img_array = image.img_to_array(img)
img_array = np.expand_dims(img_array, axis=0)
img_array = img_array / 255.0

prediction = model.predict(img_array)

if prediction[0][0] > 0.5:
    print("It's a Dog 🐶")
else:
    print("It's a Cat 🐱")
```

1/1 — 0s 57ms/step
It's a Cat 🐱

-
- Sample dataset preview



18. Acknowledgments

I would like to thank Slash Mark IT Solutions for giving me the opportunity to explore AI and deep learning. I also thank my mentors and peers for their guidance during the internship.