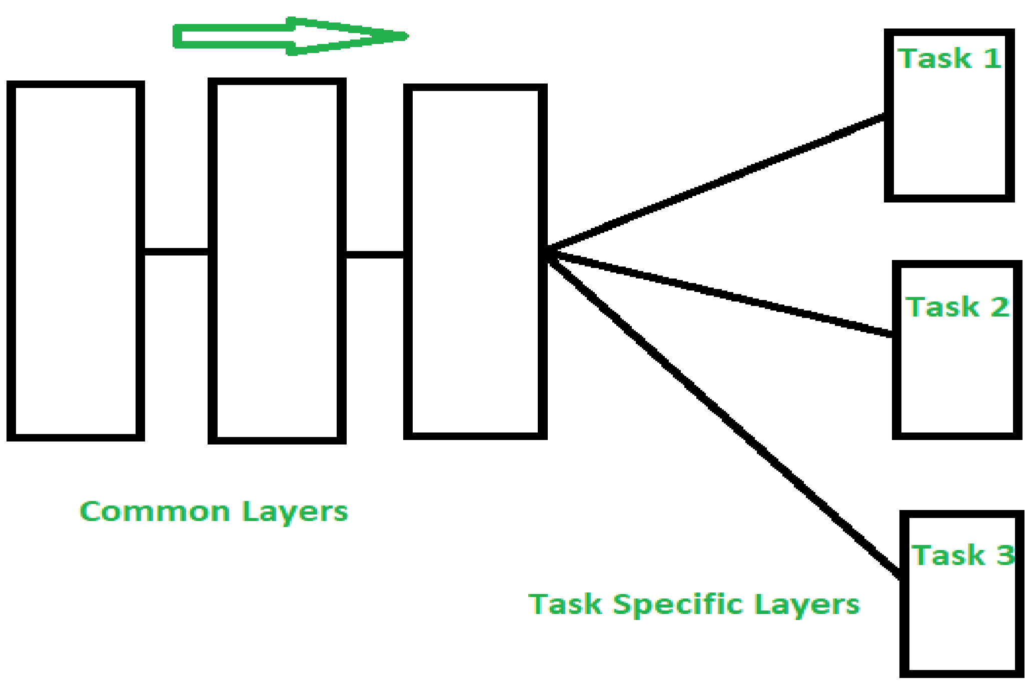
Faculty of Computers and Data Science

Intelligent Systems Department

Deep Learning Project

Custom Image Classification with CNN using OpenCV on Python



Team Members

Name	***ID***
Abdelrhman Mohamed Abdelhady Hodib	***2022513643***
Ahmed Mostafa AbdelRahman	***20221372883***
Mazen Gaber Mahmoud	***20221372110***

Contents

1 Introduction

1.1 Project Tittle

Custom animal type(cat/dog) and color(white/black) Classification with CNN using OpenCV on Python

1.2 Project Goals

Develop an accurate CNN model capable of classifying images into cat or dog categories and identifying their colors white or black with high precision and recall.

2 Data Creation

We have made two files (cats and dogs) inside each file we made two files (white and black) and here is the number of images:

Cats file

white file has 400 images of cats with white color downloaded from google images

black file has 400 images of cats with black color downloaded from google images

dogs file

white file has 400 images of dogs with white color downloaded from google images

black file has 400 images of dogs with black color downloaded from google images

3 Data Preprocessing

3.1 Data Loading and Structuring

- Images from categories "cats" and "dogs" are loaded.
- Each image is associated with labels indicating whether it's a cat or a dog, as well as its color (black or white).
- Dog/Cat label (0 for cats, 1 for dogs) and Color label (0 for black, 1 for white)

3.2 Data resizing, normalization, augmentation

Techniques	Explanation	Display
		

Resizing	Resize the image to dimensions (150x150 pixels) to match the model's input size.	
Normalization	Image pixel values are normalized to a scale between 0 and 1 which helps in standardizing the data, making it easier for the model to learn patterns effectively.	
Rotation	Rotate the image by a certain angle (20).	

Width Shift and Height Shift	Perform shifting of the image horizontally and vertically.	Width Shifted Image  Height Shifted Image 

Horizontal Flip	Flip the image horizontally	

- Original images and their augmented versions are combined into a single dataset ,This combined dataset contains a mix of both the original images and their augmented versions to let model train on more data which improves accuracy and makes it less prone to overfitting

```
The size of data before the augmentation : 1600

The size of data after the augmentation : 19200
```

3.3 Train-Validation-Test Split

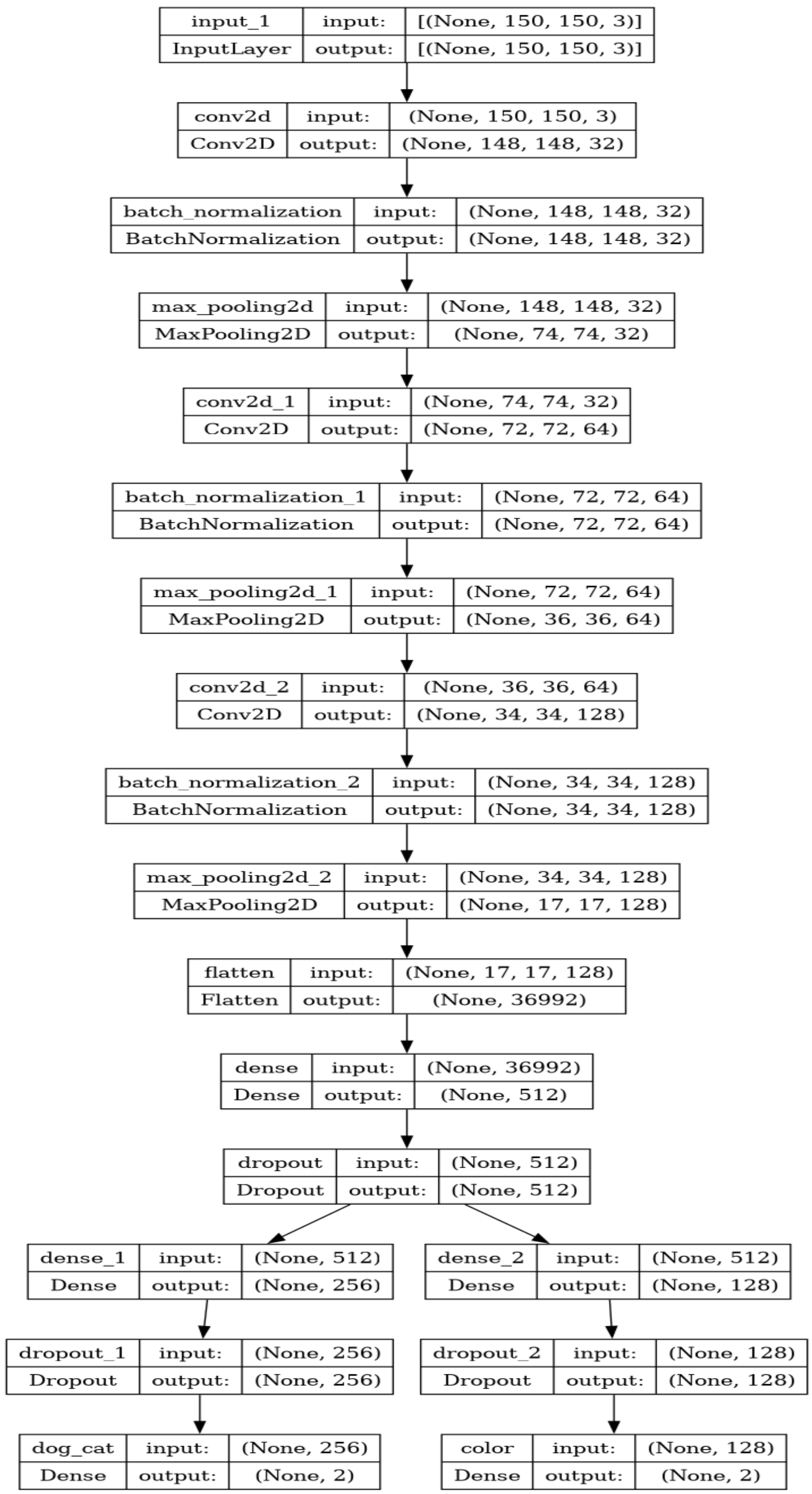
The combined dataset is divided into three subsets: training, validation, and test sets.

- Training set: Used to train the CNN model.
- Validation set: Used to fine-tune model parameters and avoid overfitting.
- Test set: Used to evaluate the model's performance on unseen data.

3.4 One-Hot Encoding

- Categorical labels for both dog/cat and color classifications are transformed into binary vectors, This transformation allows the model to understand and work with categorical data effectively during training.

3 CNN Model


input_1 | InputLayer | input: [(None, 150, 150, 3)] | output: [(None, 150, 150, 3)]
conv2d | Conv2D | input: (None, 150, 150, 3) | output: (None, 148, 148, 32)
batch_normalization | BatchNormalization | input: (None, 148, 148, 32) | output: (None, 148, 148, 32)
max_pooling2d | MaxPooling2D | input: (None, 148, 148, 32) | output: (None, 74, 74, 32)
conv2d_1 | Conv2D | input: (None, 74, 74, 32) | output: (None, 72, 72, 64)
batch_normalization_1 | BatchNormalization | input: (None, 72, 72, 64) | output: (None, 72, 72, 64)
max_pooling2d_1 | MaxPooling2D | input: (None, 72, 72, 64) | output: (None, 36, 36, 64)
conv2d_2 | Conv2D | input: (None, 36, 36, 64) | output: (None, 34, 34, 128)
batch_normalization_2 | BatchNormalization | input: (None, 34, 34, 128) | output: (None, 34, 34, 128)
max_pooling2d_2 | MaxPooling2D | input: (None, 34, 34, 128) | output: (None, 17, 17, 128)
flatten | Flatten | input: (None, 17, 17, 128) | output: (None, 36992)
dense | Dense | input: (None, 36992) | output: (None, 512)
dropout | Dropout | input: (None, 512) | output: (None, 512)
dense_1 | Dense | input: (None, 512) | output: (None, 256)
dense_2 | Dense | input: (None, 512) | output: (None, 128)
dropout_1 | Dropout | input: (None, 256) | output: (None, 256)
dropout_2 | Dropout | input: (None, 128) | output: (None, 128)
dog_cat | Dense | input: (None, 256) | output: (None, 2)
color | Dense | input: (None, 128) | output: (None, 2)

The CNN model is structured for image classification tasks with two parallel branches, enabling the model to perform two distinct tasks: dog/cat classification and color classification for the identified class. Here is the explanation of each section of the CNN model:

4.1 Input Layer

Input Shape (150, 150, 3): Represents input images with a height and width of 150 pixels and three channels (RGB).

4.2 Convolutional Layers

Three convolutional layers are used:

- 32 filters with a size of (3, 3).
- 64 filters with a size of (3, 3).
- 128 filters with a size of (3, 3).

Each convolutional layer applies a ReLU to introduce non-linearity.

These layers are like filters that analyze different parts of the images. They search for specific patterns or features using different techniques for understanding shapes and colors.

4.3 Batch Normalization

Applied after each convolutional layer to normalize and stabilize activations.

This step adjusts the data to ensure a stable and balanced understanding of the patterns detected in the images.

4.4 Max Pooling Layers

Used after each pair of convolutional layers to downsample spatial dimensions by a factor of (2, 2).

It Helps reduce the size of the information while keeping the essential details. It simplifies the images by picking the most important information from certain areas.

4.5 Flatten Layer

Flattens the output from convolutional layers into a one-dimensional array for dense layers.

It turns the complex processed information into a simpler format so that the system can work with it more efficiently.

4.6 Dense Layers

- A dense layer with 512 units and ReLU activation follows the flattening layer.
- Dropout with a rate of 0.5 is made to prevent overfitting by randomly dropping 50% of the units during training.

These layers process the simplified information. They try to understand the features discovered earlier.

Task 1: Dog/Cat Classification

- **Dense Layer (Dog/Cat):**
    - Consists of a dense layer with 256 units and ReLU activation.
    - Dropout with a rate of 0.5 to prevent overfitting.
    - Output layer with 2 units and softmax activation for dog/cat classification.

Task 2: Color (white/black) Classification

- **Dense Layer (Color):**
    - Another dense layer with 128 units and ReLU activation.
    - Dropout with a rate of 0.5 to prevent overfitting.

- Output layer with 2 units and softmax activation for color classification of the identified class.

These layers focus on specific jobs. Some determine if the images contain dogs or cats, while others decide if the color is black or white. They use what they've learned to make these decisions and give us the results.

5 Model training

The model training process involves several steps aimed at optimizing the model's performance. Here is an explanation of the process:

1. Optimizer Initialization

- **Learning Rate :** The learning rate is set to control the step size taken during optimization. A higher learning rate can lead to faster convergence, but it may overshoot the optimal solution. At contrast, a lower learning rate might take longer to converge but might find a more accurate solution.
- **Exponential Decay Schedule:** The learning rate schedule, based on Exponential Decay, gradually reduces the learning rate over time. This schedule helps in fine-tuning the optimization process by decreasing the learning rate as the training progresses. This helps in converging to a more accurate and stable model.

2. Model Compilation

- **Optimizer Assignment:** The model is compiled using the RMSprop optimizer initialized with the previously defined learning rate schedule. RMSprop is an adaptive learning rate optimization algorithm that helps improve convergence during training.

- **Loss Functions and Metrics:** Different loss functions are specified for each task the model performs (dog/cat and color classifications). These loss functions measure the difference between predicted and actual values for each task during training. Additionally, accuracy is chosen as the evaluation metric to assess the model's performance.

3. Callback Setup

- **Early Stopping:** A callback named EarlyStopping is configured to monitor the validation loss. This callback helps prevent overfitting by stopping the training process if the validation loss does not improve for a specified number of epochs (patience).
- **Model Checkpointing:** Another callback, ModelCheckpoint, is implemented to save the best-performing model based on validation loss during training.

4. Model Training

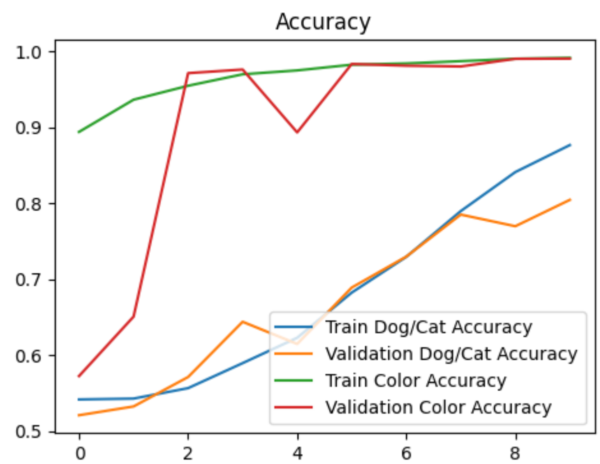
The model is trained using the training data for a fixed number of epochs and a specific batch size (64 samples per batch).

During training, the model learns to make predictions by adjusting its parameters to minimize the specified loss functions.

The combination of these steps ensures that the model learns from the training data, optimizes its parameters using the defined optimizer, evaluates its performance using specified metrics, and prevents overfitting through early stopping while saving the best model achieved during training.

6 Results

After training the model here is a plot showing the loss and accuracy of the training and validation



Training Results:

- Accuracy: The accuracy for both dog/cat classification and color classification tasks improves gradually over epochs. Initially, the accuracy for both tasks starts around 54%-55% and progressively increases.
- Loss: The overall loss (including both dog/cat and color losses) decreases significantly from an initial value of around 6.97 to a much lower value, indicating improvement in the model's performance.

Validation Results:

- Validation loss and accuracy follow similar patterns to the training set, indicating the model's consistency in learning and generalizing.
- Dog/Cat Accuracy on Validation: varies around 52% to 80%

- Color Accuracy on Validation: Stays consistently high.

Testing Results:

```
Test Accuracy for dog/cat classification: 79.22%
Test Accuracy for color classification: 98.85%
```

The test results indicate a considerable accuracy level in both dog/cat classification and color classification tasks.

Specifically, the model achieved an accuracy of 79.22% in identifying whether an image contains a dog or a cat, and an accuracy of 98.85% in classifying the color of the identified animal.

These results indicate that the model performed well in distinguishing between dogs and cats while also accurately predicting the colors of the animals in the test dataset.

Overall, it showcases a strong performance in both classification tasks based on the test data.

classification Reports:

```
Classification Report for dog/cat classification:
              precision    recall  f1-score   support

           0       0.90      0.67      0.77       988
           1       0.72      0.92      0.81       932

    accuracy                           0.79      1920
   macro avg       0.81      0.80      0.79      1920
weighted avg       0.82      0.79      0.79      1920
```

For the dog/cat classification:

- **Precision and Recall:** The precision values for identifying class 0 (dog) and class 1 (cat) are 0.90 and 0.72, respectively. This demonstrates that when the model predicts an image as a dog, it is correct around 90% of the time, while for cats, it is correct about 72% of the time. The recall values (sensitivity) indicate that the model correctly identified 67% of actual dogs and 92% of actual cats in the dataset.
- **F1-Score:** It's a measure of a test's accuracy and ranges from 0 to 1, where a higher score indicates better performance. classes (dog and cat) have an F1-score of 0.77 and 0.81, respectively. reflecting a somehow balanced performance between precision and recall for both categories.
- **Accuracy:** The overall accuracy of 79 % indicates the proportion of correctly predicted dog/cat classifications out of the total test set. It suggests that the model correctly identified approximately 79 % of the images as either dogs or cats.

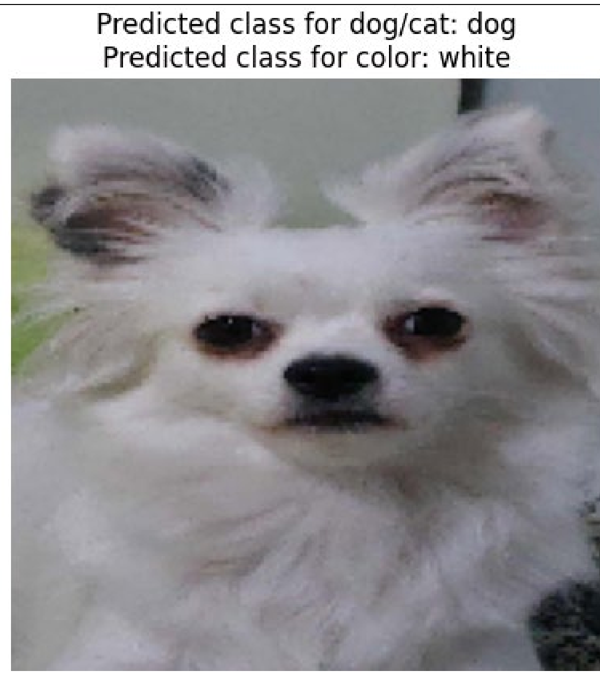
Classification Report for color classification:

	precision	recall	f1-score	support
0	0.98	0.99	0.99	959
1	0.99	0.98	0.99	961
accuracy			0.99	1920
macro avg	0.99	0.99	0.99	1920
weighted avg	0.99	0.99	0.99	1920

For the color classification:

- **Precision and Recall:** For both colors (class 0 - black and class 1 - white), the precision and recall values are very high, black : 0.98 – white 0.99. These scores imply that when the model predicts an image as black or white, it is accurate around 99% of the time. Moreover, it correctly identifies about 99% of the actual black and white images in the dataset.
- **F1-Score:** The F1-score, is also 0.99 for both classes. This score suggests a high level of accuracy in identifying both black and white colors, reflecting a balance between precision and recall.
- **Accuracy:** The overall accuracy of 99% indicates that the model correctly classified 99% of the images with respect to their colors, whether black or white.

7 Testing


Predicted class for dog/cat: dog
Predicted class for color: white


Dog/Cat: dog, Color: white