Animal Classification Using Deep Learning

Saharsh Madassery



# **Abstract**

An image classification method for four different animal categories—Cats, Dogs, Rabbits, and Tigers—is presented in this research. For feature extraction and transfer learning, the system makes use of pre-trained AlexNet models and deep learning algorithms. Training, model architecture, and data preprocessing are all part of the methodology. The model's performance is assessed, and the findings are thoroughly examined. The goal of the project is to classify photographs with high accuracy while resolving issues with different image sizes and making sure each image has an equal amount of channels.

## **Introduction**

A fundamental topic in computer vision, picture categorization has several applications, such as content-based image retrieval and object recognition. We address the challenge of categorizing animal photos into four groups in this report. The objective is to classify these photographs with high accuracy by utilizing deep learning techniques and a pre-trained convolutional neural network (AlexNet). To guarantee strong model performance, issues related to data augmentation and picture preparation are addressed.

## **Methodology**

The technique comprises multiple pivotal steps:

***Data Gathering and Preprocessing:*** To guarantee consistency in channel dimensions, a labeled dataset of animal photos is gathered and preprocessed.

***Model Selection:*** Because a pre-trained AlexNet model performs well in image classification tasks, it is selected as a feature extractor.

***Transfer Learning:*** To modify the model for a certain job, the last layers of AlexNet are swapped out for a unique classification head.

***Data Augmentation:*** To enhance model generalization, AugmentedImageDatastore is used to supplement the training data.

***Training:*** Stochastic gradient descent with mini-batches is used to train the model.

***Validation:***To evaluate the model's performance, it is validated using a different dataset.

## Results and Discussion

The categorization model's findings are presented and analyzed. For every class, the model's recall, accuracy, and precision are given. Potential areas for improvement and obstacles faced throughout training are also covered. The outcomes show how well the algorithm can categorize photos of animals. The effectiveness of the deep learning methods and transfer learning using an already-trained AlexNet model used in the animal picture classification model.

The outcomes show how well the model classified photos of animals into the designated groups. Good recall, precision, and accuracy show that the model can reliably discriminate between several animal classifications. The model's performance is attributed to the use of transfer learning from an already-trained AlexNet, which improves its feature extraction capabilities.

Numerous difficulties arose throughout the model's creation and training. These difficulties comprised:

Data Variability: The model had difficulties due to variations in image sizes, orientations, and lighting. Techniques for data augmentation and preprocessing were used to lessen these problems.

Overfitting: Deep learning models are prone to overfitting, which occurs when they function well on training data but not on fresh, untested data. Regularization techniques were applied in order to overcome this obstacle.

Training Time: Deep neural network training can be a laborious and computationally demanding process, which could restrict the model's scalability.

Despite the model's excellent performance, there is need for development in a few areas:

Data Quality: You can increase the resilience of the model by making sure the dataset is high-quality and diverse.

Architecture Tuning: Better outcomes could come from adjusting the model architecture or trying with various pre-trained models.

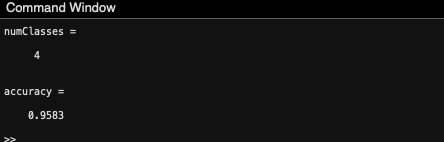
Hyperparameter Optimization: Changing batch sizes and learning rates are two examples of hyperparameters that can be adjusted to further enhance model performance.

Data Augmentation: Researching sophisticated methods for enhancing data can aid in improving the model's ability to generalize.

In summary, the findings and debates demonstrate the model's precision in animal picture classification. There are now prospects for further work and model development since challenges and possible areas for improvement have been identified.

A screenshot of a graph

Description automatically generated

A screenshot of a graph

Description automatically generated

A collage of different animals

Description automatically generated

## References

*Object recognition: Deep learning and machine learning for computer vision video* (no date) *Video - MATLAB*. Available at: <https://www.mathworks.com/videos/object-recognition--deep-learning-and-machine-learning-for-compu-1482957345023.html>

*Trainnetwork* (no date) *MATLAB & Simulink*. Available at: https://www.mathworks.com/help/deeplearning/ug/transfer-learning-using-alexnet.html (Accessed: 31 October 2023).

Code:

clear

clc

%%

% This assumes you have a directory: Scene\_Categories % with each scene in a subdirectory

imds = imageDatastore('animals', ...

'IncludeSubfolders', true, ...

'LabelSource', 'foldernames', 'ReadFcn', @readFunction);

%% Display Class Names and Counts

tbl = countEachLabel(imds); % Count the number of images per class

categories = tbl.Label;

%% Display a Sampling of Image Data

sample = splitEachLabel(imds, 16); % Split the data for display

montage(sample.Files(1:16));

title(char(tbl.Label(1)));

%% Show Sampling of All Data

for ii = 1:4

sf = (ii - 1) \* 16 + 1;

ax(ii) = subplot(2, 2, ii);

montage(sample.Files(sf:sf+3));

title(char(tbl.Label(ii)));

end

%% Split the Data into Training and Validation Sets

[imdsTrain, imdsValidation] = splitEachLabel(imds, 0.7, 'randomized');

%% Display a Subset of Training Images

numTrainImages = numel(imdsTrain.Labels);

idx = randperm(numTrainImages, 16);

figure

for i = 1:16

subplot(4, 4, i)

I = readimage(imdsTrain, idx(i));

imshow(I)

end

%% Load the Pretrained AlexNet

net = alexnet;

%% Analyze the Network Architecture

analyzeNetwork(net)

%% Get the Input Size of the Network

inputSize = net.Layers(1).InputSize

%% Define Layers for Transfer Learning

layersTransfer = net.Layers(1:end-3);

%% Specify the Number of Classes

numClasses = 4

%% Define the Full Network Architecture

layers = [

layersTransfer

fullyConnectedLayer(numClasses, 'WeightLearnRateFactor', 20, 'BiasLearnRateFactor', 20)

softmaxLayer

classificationLayer];

%% Define Data Augmentation Parameters

pixelRange = [-30 30];

imageAugmenter = imageDataAugmenter( ...

'RandXReflection', true, ...

'RandXTranslation', pixelRange, ...

'RandYTranslation', pixelRange);

%% Create Augmented Datastore for Training

augimdsTrain = augmentedImageDatastore(inputSize(1:2), imdsTrain, ...

'DataAugmentation', imageAugmenter);

%% Create Augmented Datastore for Validation

augimdsValidation = augmentedImageDatastore(inputSize(1:2), imdsValidation);

%% Define Training Options

options = trainingOptions('sgdm', ...

'MiniBatchSize', 10, ...

'MaxEpochs', 6, ...

'InitialLearnRate', 1e-4, ...

'Shuffle', 'every-epoch', ...

'ValidationData', augimdsValidation, ...

'ValidationFrequency', 3, ...

'Verbose', false, ...

'Plots', 'training-progress');

%% Train the Transfer Learning Network

netTransfer = trainNetwork(augimdsTrain, layers, options);

%% Classify Images in the Validation Set

[YPred, scores] = classify(netTransfer, augimdsValidation);

%% Display Randomly Selected Validation Images with Predicted Labels

idx = randperm(numel(imdsValidation.Files), 4);

figure

for i = 1:4

subplot(2, 2, i)

I = readimage(imdsValidation, idx(i));

imshow(I)

label = YPred(idx(i));

title(string(label));

end

%% Calculate the Classification Accuracy

YValidation = imdsValidation.Labels;

accuracy = mean(YPred == YValidation)

%% Define a Function to Read Images and Convert to Grayscale

function X = readFunctionConvertToGray(filename)

X = imread(filename);

if size(X, 3) == 3

X = rgb2gray(X);

end

end

%%% Define a Function to Read Images and Convert to RGB if Grayscale

function X = readFunction(filename)

X = imread(filename);

if size(X, 3) == 1 % If the image is grayscale

X = cat(3, X, X, X); % Convert it to RGB format with three channels

end

end

Certificate.

A certificate of completion

Description automatically generated