Cat vs Dog Challenge Report

Muhammad Uzair Khattak*, Talal Abdullah Fadhl Algumaei†, Muhammad Hamza Sharif‡
Department of Science, Computer Vision
MBZUAI University, Masdar City, Abudhabi
Email: *21010225@mbzuai.ac.ae, †21010220@mbzuai.ac.ae, ‡21010214@mbzuai.ac.ae

Abstract—**This report illustrates the methods and processes used to process the cat/dog images and build the AI classifier for automatic identification of cats and dogs.**

I. Introduction

The main objective of this challenge is to build a classifier that would automatically recognize cat and dog images accurately.

The dataset was obtained from the kaggle [1] website, which includes **25000 labelled** images for training and **12500 unlabeled** images for testing of the AI model. The training dataset was highly balanced. Each **RGB** image in training set had a different resolution. This necessitated that each team member use a different strategy for pre-processing raw images, a different family of pre-trained deep learning architectures, and a different deep-learning framework to conduct experiments.

The basic strategy followed by each team member was:

- Transform the raw images into processed images by changing resolution, cropping from center, flipping images either horizontally or vertically, rotating to certain degree, and normalizing the pixels values.
- Split labeled dataset by applying **80/20** or **90/10** train-test split rule.
- Train the layer of pre-trained deep learning architectures with the train labeled dataset and validate its performance on test labeled data-set.
- Use the trained classifier, for classification of unseen cat and dog images.

Various experiments were conducted, and the performance of each classifier was ranked based on training and testing accuracy. In the following section, we will go over the specifics of each classifier and their accuracy results.

II. Experiment

In this section, we will explain the various experiments that were performed for classification of cat and dog images using different pre-trained deep learning architectures. Due to limitations and unavailability of GPU machines, experiments were performed in the ***google Colab and Kaggle*** environments.

A. Deep Learning Architectures

Experiments were carried out using the transfer learning approach, with the final layer being trained with our provided parameters. Before feeding dataset to the network, different augmentation techniques were applied which are explained in Table I.

The following is a list of pre-trained deep learning architectures and their variants that have been implemented to execute environment tasks.

- Densely Connected Convolutional Network (DenseNet)
- Residual Network (ResNet)
- VGG Neural Network
- Efficient Net

The labelled dataset was used to train these architectures. Multiple iterations were performed to determine the optimal model parameters. The performance of each model with optimized parameters is shown in Table II.

III. Results

Each classifier's performance accuracy on the validation set is significantly greater than 95%. However, we choose the weights of the trained Efficient B2 model because it outperforms others in cat and dog classification. Figure a depicts the results of the model's accurate prediction on an unlabeled dataset.

(a) Results of EfficientNet B2

References

[1] "Dogs vs. cats." [Online]. Available: https://www.kaggle.com/c/dogs-vs-cats/data

Table I: **Augmentation on Labeled Data**

Experiment	Framework	Model type	Resolution Size	Augmentation
Experiment_1	Pytorch	DenseNet169	224 x 224	Rotation by 20°, Horizontal Flipping, Normalization by [0.485,0.456,0.406],[0.229,0.224,0.225]
Experiment_2	Pytorch	DenseNet201	224 x 224	Rotation by 20°, Horizontal Flipping, Normalization by [0.485,0.456,0.406],[0.229,0.224,0.225]
Experiment_3	Pytorch	Resnet50	224 x 224	Rotation by 20°, Horizontal Flipping, Normalization by [0.485,0.456,0.406],[0.229,0.224,0.225]
Experiment_4	Pytorch	Resnet121	256 x 256	Cropping from Center(224), Normalization by [0.5],[0.5]
Experiment_5	FastAI	Resnet121	256 x 256	max_lighting=0.1, max_zoom=1.05, max_warp=0.,max_rotate=15, p_affine=0.75, p_lighting=0.75
Experiment_6	FastAI	EfficientNet B2	256 x 256	max_lighting=0.1, max_zoom=1.05, max_warp=0.,max_rotate=15, p_affine=0.75, p_lighting=0.75
Experiment_7	FastAI	EfficientNet B3	256 x 256	max_lighting=0.1, max_zoom=1.05, max_warp=0.,max_rotate=15, p_affine=0.75, p_lighting=0.75

Table II: **Model Performance**

Experiment	Framework	Model type	train_test split ratio	Learning Rate	Epoch	Classifier Layer	Training Time	Validation_Accuracy
Experiment_1	Pytorch	DenseNet169	80/20	0.003	10	Linear(1664,4096), Linear(4096 ,512), Linear(512,2), ReLU() ,Dropout(0.5), LogSoftMax	≈ 50 min.	99.1%
Experiment_2	Pytorch	DenseNet201	80/20	0.0003	10	Linear(1920,4096), Linear(4096, 512), Linear(512,2), ReLU() ,Dropout(0.5), LogSoftMax	≈ 48 min.	99.04%
Experiment_3	Pytorch	Resnet50	80/20	0.0001	10	Linear(2048,4096), Linear(4096, 512), Linear(512,2), ReLU() ,Dropout(0.5), LogSoftMax	≈ 60 min.	98.6%
Experiment_4	Pytorch	Resnet121	80/20	0.003	10	Linear(in_features= 2048, out_features=2, bias=True)	≈ 60 min.	98.94%
Experiment_5	FastAI	Resnet121	90/10	slice(3e-03)	13	N/A	≈ 120 min.	99.32%
Experiment_6	FastAI	EfficientNet B2	90/10	Dynamic LR [slice(5e-05), slice((5e-05)/10)]	29	N/A	≈ 180 min.	99.48%
Experiment_7	FastAI	EfficientNet B3	90/10	Dynamic LR [slice(5e-05), slice((5e-05)/10)]	22	N/A	≈ 120 min.	99.56%