International Journal of Leading Research Publication (IJLRP)

E-ISSN: 2582-8010 ● Website: www.ijlrp.com ● Email: editor@ijlrp.com

Deployment of a Dog Breed Classifier Using CNN and ResNet-50 on AWS EC2

Sadhana Paladugu

Senior Software Engineer

Abstract:
This paper presents the development and deployment of a machine learning model for dog breed classification. The model leverages Haar Cascade classifiers for initial detection and Convolutional Neural Networks (CNN) with ResNet-50 for classification. The deployment was carried out on AWS EC2, encountering challenges related to maintaining a live production link with continuous updates. The study discusses model performance, deployment considerations, and solutions to challenges encountered in production.

1. Introduction

Dog breed classification is a challenging computer vision task with applications in pet identification, veterinary diagnostics, and animal conservation. Machine learning and deep learning techniques provide an effective means for automating this process with high accuracy. This paper describes the implementation of a classifier using deep learning techniques and its deployment in a cloud environment, aiming to provide insights into both model architecture and deployment considerations.

2. Methodology

2.1 Dataset

A dataset comprising multiple dog breeds was utilized, featuring labeled images for training and validation. The dataset was preprocessed to ensure uniformity in resolution and format. Data augmentation techniques, including rotation, scaling, and flipping, were applied to enhance model generalization and reduce overfitting.

2.2 Model Architecture

- **Haar Cascade Classifier**: Used for initial face detection to localize dog faces in images before classification. This helped reduce computational load by focusing on relevant image regions.
- **CNN**: A deep learning-based feature extractor for breed classification, employing multiple convolutional and pooling layers to extract hierarchical features from images.
- **ResNet-50**: A residual learning framework that improved accuracy over traditional CNNs by mitigating the vanishing gradient problem through skip connections.

2.3 Training and Evaluation

- The models were trained on a GPU-enabled environment to expedite processing and handle complex computations efficiently.
- Categorical cross-entropy loss was used as the objective function, and the Adam optimizer was employed for adaptive learning rate adjustments.

- Accuracy was used as the primary evaluation metric, with additional assessments including precision, recall, and F1-score to ensure robustness.
- Hyperparameter tuning was conducted to optimize learning rate, batch size, and network depth, aiming to achieve the best trade-off between computational efficiency and accuracy.

3. Deployment on AWS EC2

- AWS EC2 was used to host the trained model, providing a scalable and flexible cloud solution. The server was configured to handle incoming API requests and return classification results in real time.
- The model was exposed via a REST API to allow real-time predictions, utilizing Flask as the backend framework.
- Key deployment challenges included configuring EC2 instances, handling dependencies, ensuring load balancing, and securing API endpoints against unauthorized access.
- Solutions involved automating instance setup using shell scripts, optimizing storage by compressing model files, and implementing logging mechanisms to monitor system performance and diagnose failures.

4. Challenges and Solutions

- **First-time AWS deployment**: As this was the first deployment experience, significant effort was required to understand EC2 instance selection, security groups, and access management. Step-by-step debugging and AWS documentation were utilized to overcome initial hurdles.
- **Maintaining a live production link**: Updating the model without causing downtime was a challenge. Implementing a rolling update strategy helped in minimizing service disruptions. Load balancing and blue-green deployment strategies were explored to enhance reliability.
- **Scalability and Latency Issues**: Initial deployment faced latency issues due to inefficient model loading. Optimizing inference by converting the model to TensorFlow Lite and utilizing GPU acceleration on EC2 helped improve response times.

5. Results and Discussion

- A comparative analysis of model performance was conducted, highlighting the accuracy differences between Haar Cascade, CNN, and ResNet-50.
- ResNet-50 outperformed traditional CNN architectures due to its residual learning approach, achieving higher accuracy on test data.
- Deployment performance was evaluated based on response time, API uptime, and system scalability under increased traffic.
- Considerations for future improvements include implementing auto-scaling with AWS Lambda, incorporating a CI/CD pipeline for seamless updates, and integrating model monitoring tools for real-time performance tracking.

6. Conclusion

This study demonstrated an end-to-end pipeline for dog breed classification using deep learning and cloud deployment. Despite challenges in deploying to AWS EC2, strategic solutions ensured stable model performance in production. Future work includes optimizing inference speed, enhancing automation in updates, and expanding the model to support additional breeds with multilingual API accessibility.

References

1. Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. Advances in Neural Information Processing Systems, 25.
2. He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 770-778.
3. Viola, P., & Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 511-518.
4. Howard, A. G., et al. (2017). MobileNets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861.
5. AWS Documentation - Amazon EC2. Retrieved from https://docs.aws.amazon.com/ec2/index.html
6. Chollet, F. (2017). Deep Learning with Python. Manning Publications.
7. Brownlee, J. (2018). Deep Learning for Computer Vision. Machine Learning Mastery.