DLE305 Deep Learning
Assessment 1 - Image Classification

Barry Rerecich
Media Design School
barry.rerecich@gmail.com

HuggingFace-Hosted Demo Classifier:
https://huggingface.co/spaces/barrymoana/CNN-Cat-Dog-Classifier

GitHub Repo: https://github.com/barrymoana/CNN_CatDog

Abstract

This report presents an in-depth study on image classification of cats and dogs using Convolutional Neural Networks (CNNs). The project aims to investigate the performance of various CNN architectures and configurations, with a particular focus on the impact of regularization techniques such as dropout. The models are trained and evaluated on a dataset sourced from Kaggle. The key findings reveal that complex CNN architectures yield superior performance and that dropout as a regularization technique may not always be effective. Recommendations for future work and alternative approaches are also discussed.

1. Introduction

In the realm of computer vision, the task of image classification has garnered significant attention. The project at hand aims to build an image classifier to distinguish between images of cats and dogs. Utilizing Convolutional Neural Networks (CNNs), a type of deep learning architecture particularly effective for image recognition tasks, various models have been designed and trained to achieve this classification. The key objective was to compare the performance of different CNN architectures and configurations, both with and without regularization techniques like dropout. This report elucidates the simulation environment, implementation details, dataset attributes, and key insights gained throughout the project.

2. Simulation Environment

The project was implemented using Python, leveraging multiple libraries to provide a comprehensive solution. TensorFlow was used for building and training the neural networks. For image manipulation and resizing, the PIL library was employed. NumPy and pandas were used

for numerical operations and data handling, respectively. Matplotlib and Plotly were utilized for generating various plots and visualizations, and Scikit-learn's train_test_split function was used for data splitting.

The code follows Python naming conventions and is well-commented to enhance readability and maintainability. Various utilities and functions were modularized to promote reusability and to keep the main codebase clean and concise.

3. Dataset

https://www.kaggle.com/datasets/shaunthesheep/microsoft-catsvsdogs-dataset
The dataset serving as the foundation for this project consists of cat and dog images, which were obtained from a publicly accessible Kaggle dataset. The images in this dataset presented a range of dimensions and quality levels. To address this variability and ensure data consistency—crucial for effective model training—the images were resized to a uniform 128x128 pixels. Further, to facilitate a comprehensive evaluation of the models, the dataset was divided into separate training and validation sets. This partitioning aligns with deep learning best practices and enables the assessment of the models' generalization capabilities on data they have not previously encountered.

4. Implementation

The image classifier was systematically constructed, adhering to a series of key steps influenced by deep learning principles. These are as follows:

Data Loading and Pre-processing:
In the initial phase of the project, the objective was to prepare the dataset for effective model training. To achieve this, images of cats and dogs were sourced from Kaggle and resized to a uniform dimension of 128x128 pixels. This step ensured data consistency, a crucial factor for the successful training of neural network models. Beyond resizing, image normalization and augmentation techniques were employed using TensorFlow's ImageDataGenerator. These pre-processing steps not only expanded the dataset but also enhanced the model's ability to generalize to new, unseen data, aligning with core deep learning principles.

Model Architectures (Simple, Two-Layer, and Bigger CNNs):
The next pivotal aspect of the project aimed to assess the impact of architectural complexity on model performance. To this end, three distinct Convolutional Neural Network (CNN) architectures were designed and evaluated. The Simple CNN served as the baseline model, comprising just one convolutional layer followed by a max-pooling layer. A more advanced Two-Layer CNN was also developed, featuring two convolutional layers, each succeeded by a max-pooling layer. For a more in-depth examination, a Bigger CNN was constructed with three convolutional layers and corresponding max-pooling layers. All models underwent training for 10 epochs, utilizing the Adam optimizer and binary cross-entropy as the loss function. This multi-architectural approach was rooted in deep learning principles, enabling the models to

automatically learn spatial features from the data. The increased complexity in the architectures allowed for the learning of more nuanced and hierarchical features, further enhancing the model's performance.

Regularization Techniques:
To address the common challenge of overfitting in neural network models, regularization techniques were employed. Specifically, dropout was integrated into the network architectures as a method of regularization. Variants of all the previously discussed models—Simple CNN, Two-Layer CNN, and Bigger CNN—were trained with and without the inclusion of dropout layers. In line with deep learning principles, dropout serves to deactivate a random subset of neurons during each training iteration. This prevents any single neuron from becoming overly specialized in the data, thereby enhancing the model's capacity to generalize well to new, unseen data.

Model Evaluation and Comparisons:
The final cornerstone of the project was the evaluation and comparison of the trained models to ascertain their performance capabilities. Various metrics, including training and validation loss and accuracy, were meticulously tracked throughout the training process. These metrics not only gauge the effectiveness of the models but also their capacity to generalize to new, unseen data. To derive meaningful conclusions, several comparative analyses were conducted, pitching the Simple CNN against the Two-Layer and Bigger CNNs. Additionally, evaluations were made to distinguish the performance of models with and without data augmentation and regularization techniques. These evaluations are firmly rooted in deep learning principles, providing invaluable insights into the models' learning efficacy and their adaptability to new data.

5. Key Insights and Recommendations

Training and Validation Accuracy:
The Bigger CNN model showed marked superiority in performance, reaching a peak validation accuracy of 85.36%. This contrasts with simpler architectures, where the Two-Layer CNN reached around 80.46% validation accuracy. The close gap between training and validation accuracies—less than 3% in the case of the Bigger CNN—suggests that overfitting was not a significant issue. These findings affirm that more complex architectures can better capture data intricacies and that training for more epochs could potentially push the validation accuracy even higher.

Effect of Dropout:
Incorporating dropout as a regularization technique led to constrained performance. For instance, the Bigger CNN model without dropout reached an 81.10% training accuracy, while the same architecture with dropout lagged behind at 72.93%. These findings suggest that dropout might not be universally beneficial, prompting the exploration of alternative regularization techniques like L1 or L2, which might offer a more balanced approach without compromising learning capabilities.

Data Augmentation:
The lack of a significant performance boost with data augmentation was unexpected. Despite the prevailing wisdom in deep learning, our models' performance did not improve by more than 1-2% with data augmentation. This raises questions about the dataset's inherent diversity and the actual necessity of augmentation in this specific case.

Recommendations
Based on these insights, future work could focus on extending the training duration, especially given the absence of overfitting and a consistent uptick in validation accuracy beyond the 10-epoch mark. Reevaluating the role of dropout, particularly for more complex models, could lead to improved performance. Lastly, given the Bigger CNN model's effectiveness, further experimentation with even more intricate architectures could potentially push validation accuracy past the current 85.36% peak.

6. Conclusion

The project successfully deployed Convolutional Neural Networks (CNNs) for classifying images of cats and dogs. Key findings highlighted the importance of architectural complexity, as the Bigger CNN model outperformed simpler architectures. However, commonly used techniques like dropout and data augmentation did not enhance performance as expected. These observations prompt a reevaluation of traditional deep learning practices and suggest avenues for future research, such as longer training durations and alternative regularization methods. Overall, the study provides valuable insights into the optimization of CNNs for image classification tasks.

References:
Github.com. (n.d.). *A tutorial on convolutional neural networks with TensorFlow eager API*. https://github.com/StrikingLoo/Cats-and-dogs-classifier-tensorflow-CNN

Kaggle.com. (n.d.). D*ogs vs. cats: Create an algorithm to distinguish dogs from cats.* https://www.kaggle.com/c/dogs-vs-cats