Big Cat Classification

Tucker R. Surdock

Washington State University ID: 11644744

CPT_S 434 Neural Network Design And Application

Instructor: *Yan Yan*

04/23/2023

Abstract

I have changed the challenge I will be working on from the original one I presented in the proposal because the competition is no longer available, and the webpage has been removed. With that being said, I have decided to work on an image classification dataset called “10 Big Cats of the Wild”[1]. I have chosen this dataset because it involves a computer vision project that uses neural networks (NN's) to classify images. I will compare the test accuracies of different NN's against each other using the test dataset provided by Kaggle.

Introduction

The Big Cat dataset is intended for use in image classification, which is one of the most common applications of neural networks in computer vision. A neural network can be trained to classify images into several categories, in this case, different types of big cats. I will compare the test accuracies of different neural networks. Test accuracy refers to the performance of a neural

network on a set of unseen data. Neural networks and other machine learning models are trained on a labeled dataset of images and then tested on an unlabeled dataset. For this project, Kaggle provides both the training and test sets from which I will train and test several neural networks.

Literature Review

One Kaggle user, Om Bharat, posted a notebook related to this dataset in which he used the VGG16 convolutional neural network (CNN) architecture originally trained on the ImageNet dataset for image classification. Bharat implemented transfer learning by using the pre-trained CNN as a starting point for classification of the Big Cat image set. This is useful because VGG16 has learned lower-level features, such as edges and shapes, that can be applied to many image classification tasks. To do this, Bharat froze the earlier convolutional layers and fine-tuned the dense layers, which are meant to perform classification tasks by taking the features learned by previous layers and applying them to a final classification [2].

The ResNet is a well-known pre-trained CNN used for image classification that consists of numerous convolutional layers, each having 3x3 masks (except the first layer, which has 7x7 masks). ResNet-152 is a variant of the model that won the ILSVRC in 2015 and has 152 layers. Despite its large size, ResNet has lower complexity compared to VGGNet, another popular CNN that VGG16 is derived from. This is achieved by using small kernels in the layers and a single fully connected layer with 1000 neurons instead of two with 4096 neurons, thereby reducing the number of parameters. This reduction results in improved performance while maintaining “computational efficiency” [3].

Technical Plan

Data Preprocessing:

The image data needs to be prepared before training and testing the neural networks. This includes resizing the images to a uniform size and normalization.

Transfer Learning using VGG16:

Use the pre-trained VGG16 model as a starting point for classification of the Big Cat image set. Freeze the earlier convolutional layers and fine-tune the dense layers to perform classification tasks.

Transfer Learning using ResNet-152:

Use the pre-trained ResNet-152 model for image classification. Fine-tune the model by training the last layer with the Big Cat dataset.

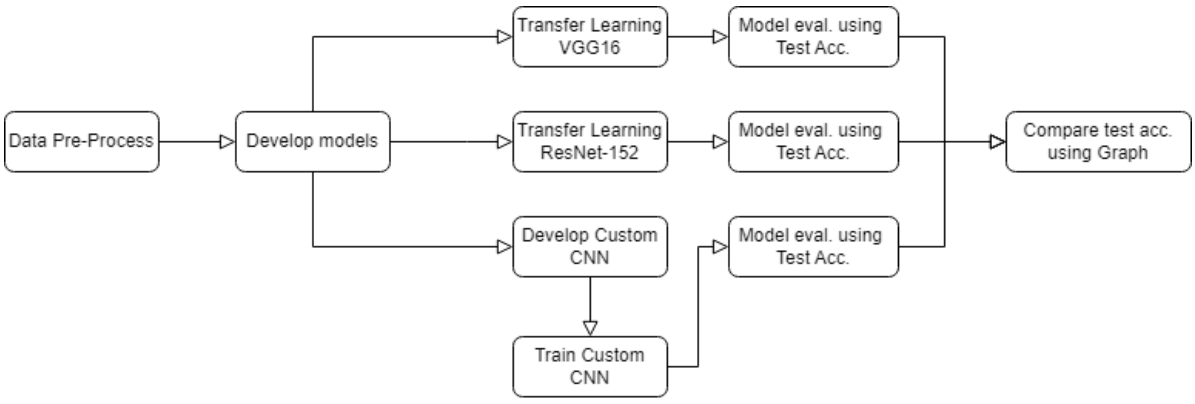
Train a Custom CNN:

Train a custom CNN architecture from scratch using the Big Cat dataset. This includes designing the architecture, setting hyperparameters, and training the model.

Model Evaluation:

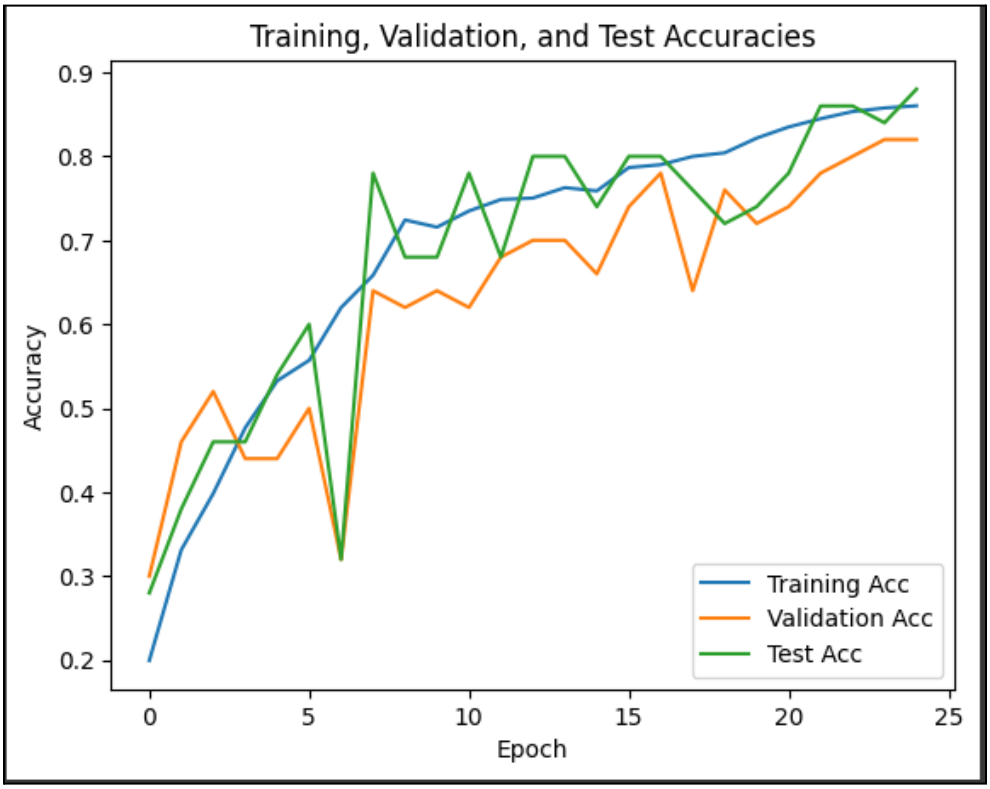
Evaluate the performance of the different neural network models using the test dataset provided by Kaggle. Compare the test accuracies of the different neural networks against each other.

Flow Chart:



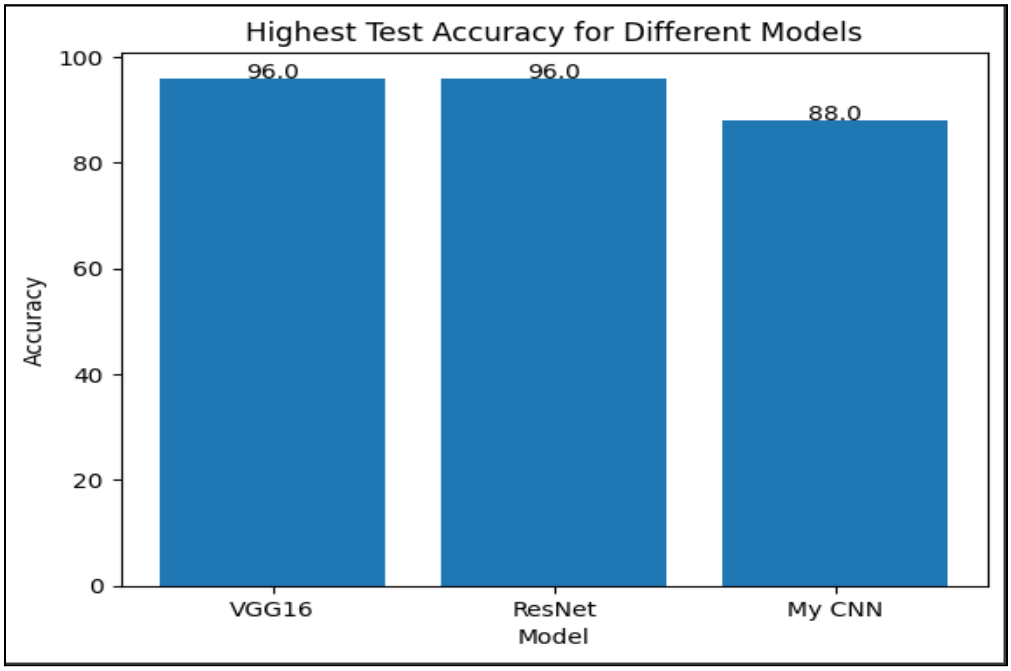
Complete Results

I have concluded my experiment and implemented a few changes to my custom CNN model. Firstly, I incorporated deeper convolutional layers and switched the ReLU activation function to Silu, potentially enabling the model to capture more intricate features in the images and enhance its accuracy. Secondly, I replaced the previous optimizer with SGD to ensure a fair comparison between models. Thirdly, I standardized the loss function for all models to cross-entropy, widely recognized for classification tasks. These modifications contributed to a noticeable boost in the performance of my custom CNN model, achieving an accuracy of approximately 90% compared to the previous 86%.



This outcome suggests that thoughtful consideration of architecture, optimization, and loss function can significantly enhance the CNN model's performance for image classification tasks.

Overall, this result indicates the importance of optimizing the model's architecture, optimization algorithm, and loss function to achieve superior performance. While my custom CNN model performed well on the training and validation sets, it did not generalize as best as it could to the new test set utilized to compare all three models.



Therefore, I plan to continue experimenting to improve the model's generalization capabilities. Regarding the runtimes, VGG16 exhibited the best optimization, followed by my custom CNN, and then ResNet152, which was expected due to VGG16's simpler architecture and fewer parameters than ResNet152.

Future Work

One potential area of future work for this project is to improve the generalization capabilities of the custom CNN model. While the model performed well on the training and validation sets, it did not generalize as well to the new test set. One approach to improving generalization could involve data augmentation techniques such as rotation, scaling, and flipping to create more

diverse image data. Another approach could involve using more regularization techniques such as dropout or L2 regularization to reduce overfitting of the model to the training data.

Another area of future work could involve exploring different architectures or variants of existing architectures for image classification tasks. For example, there are newer architectures such as EfficientNet [4], which have achieved state-of-the-art performance on various computer vision tasks, including image classification. It could be interesting to explore the use of these newer architectures or variants of existing architectures to improve the accuracy of the model on the Big Cat dataset.

Overall, the project highlights the importance of optimizing the model's architecture, optimization algorithm, and loss function to achieve superior performance in image classification tasks. These lessons can be applied to future work in exploring different architectures or improving the generalization capabilities of the model.

References

[1] Gerry. (2023, February 28). 10 big cats of the wild - image classification. Kaggle. Retrieved March 26, 2023, from https://www.kaggle.com/datasets/gpiosenka/cats-in-the-wild-image-classification

[2] Ombharat. (2023, March 2). Wild cats: 96%: Transfer learning VGG16. Kaggle. Retrieved March 26, 2023, from https://www.kaggle.com/code/ombharat/wild-cats-96-transfer-learning-vgg16/notebook

[3] Mrgrhn. (2021, February 3). Resnet with tensorflow (transfer learning). Medium. Retrieved March 26, 2023, from https://medium.com/swlh/resnet-with-tensorflow-transfer-learning-13ff0773cf0c

[4] Tan, M., & Le, Q. V. (2020, September 11). EfficientNet: Rethinking model scaling for Convolutional Neural Networks. arXiv.org. Retrieved April 23, 2023, from https://arxiv.org/abs/1905.11946v5