Abstract

Introduction and Literature Review

Convolutional Neural Networks (CNNs) have demonstrated strong performance on structured datasets, but real-world classification tasks pose additional challenges such as background clutter, lighting variations, and occlusions. Building on the Dog Classification project from last semester, we now apply our previous architectures, ResNet50 and VGG16, to the more complex "10 Big Cats of the Wild" dataset. This dataset’s natural environment images introduce greater intra-class diversity and inter-class similarity, making classification more difficult. Our project addresses two questions: Can CNN models trained on dog images generalize to big cat species, and what adjustments are needed for optimal performance? We leverage transfer learning strategies and practical training methods inspired by Nguyen *et al.* [1], Simonyan and Zisserman [2], He *et al.* [3], Kornblith *et al.* [4], and Sabottke and Spieler [5] to guide our approach.

Nguyen *et al.* [1] demonstrated CNN effectiveness in wildlife recognition, showing generalization potential, while our work focuses specifically on transfer between related categories. Simonyan and Zisserman [2] highlighted the performance advantages of deeper networks, directly informing our use of VGG16. He *et al.* [3] introduced residual learning in ResNet, improving deep network optimization, which we test for transferability to a new domain. Kornblith *et al.* [4] emphasized the importance of hyperparameter tuning for successful transfer learning, influencing our adjustments in learning rate, batch size, and dropout. Although Sabottke and Spieler [5] focused on medical imaging, their insights on input resolution trade-offs guide our preprocessing choices. Together, these studies provide the theoretical and methodological foundation for evaluating and adapting CNNs for big cat species classification.

Method

Data

Initially we wanted to use one dataset for our experiments. However, since the aim is to observe the impact of the data set of classification models we decided to use three different datasets. The first being the “Big Cats Images Dataset” which contains ten breeds of big cats with each class ranging between 88 and 100. The second dataset we used was the “**wild\_cats” with** five different wild cats, and each class containing over 560 images. For the third data set we used the “**Big Cats Images Dataset” with** 10 breeds of big cats and around 230 images per class. **\*\*\*Insert Bar graph of each data set\*\*\***

Data Preparation and Preprocessing

The datasets used did not require much preparation as compared to the Stanford dog data set used in the dog breed classification. All the images were cropped up already and we only needed to performed data augmentation techniques and split the data. We used multiple data splits which included 60/30/10, 70/20/10, 80/20. Refer to table x below for the data augmentation techniques used.

\*\*\*insert table 1 here\*\*\*

Deep Learning Structures

For this project, we will utilize transfer learning by starting with the pretrained VGG16 and ResNet50 models that have been trained on ImageNet. We will remove their original fully connected layers and replace them with a new classification head. The ResNet50 head will include a Global Average Pooling layer, one or more dense layers with ReLU activations, a dropout layer with a dropout rate of 0.5 for regularization, and a final softmax output layer designed for 10-class classification. This can be seen in figure x below. We changed the output layer from 120 to 10, to match the classes in the data set, this is not present in figure x yet, as we plan to redraw the architecture. Currently the last 30 layers in addition to the custom layers of the ResNet50 model are trainable.

\*\*\*Insert resnet50 diagram here\*\*\*

Training will be conducted in two phases. In the first phase, known as feature extraction, all convolutional layers of the pretrained models will be frozen, allowing only the new classification head to be trained. This process enables the model to quickly adapt high-level learned features to the new task without interfering with the generalized lower-level features. In the second phase, which is called fine-tuning, we will selectively unfreeze the last few convolutional blocks and retrain the model using a much lower learning rate. This approach fine-tunes the feature representations to better match the characteristics of the wild cat images while preserving the general knowledge obtained from ImageNet.

        Hyperparameters will be set based on prior experience. The learning rate will begin at 0.001 during the feature extraction phase and will be reduced to 1e-5 during fine-tuning. Batch sizes will range from 32 to 64, depending on the available GPU memory. We will start with the Adam optimizer, but we will also evaluate the performance of SGD with momentum and RMSprop to assess optimizer sensitivity. Early stopping will be implemented to avoid overfitting as well as to save resources, monitoring validation loss and halting training if no improvement is observed over a predetermined number of epochs. Additionally, a learning rate scheduler will be employed to automatically reduce the learning rate when the validation performance plateaus.

        To enhance generalization and prevent overfitting, we will implement regularization techniques such as dropout, data augmentation, early stopping, and, optionally, L2 weight decay. Data augmentation is particularly crucial due to the relatively small size of the dataset and the variability found in real-world images.

We also aim to perform the same experiments using our best VGG16 model from last semester. Which can be seen in figure x below. We added a customer layer which contains a flatten, dropout, dense, batch normalization, dropout, dense, batch normalization and finally a soft max dense layer for an output probability of X classes, where X is the number of classes in each data set.

\*\*\* insert vgg16 diagram here \*\*\*

Evaluation Metrics

\*\*\* insert formula for accuracy, precision and f1 score, why we used these as our evaluation metrics. Use of loss curve to check for generalization and overfitting. Use of confusion matrix to check the performance of each class.

Experiments and Results

Acknowledgement