Deep Learning Group Project

Manipal Sidhu
School of Engineering
Technology and Applied Science
Centennial College
Toronto, Ontario, Canada
msidhu45@my.centennialcollege.ca

Mahpara Rafia Radmy
School of Engineering
Technology and Applied Science
Centennial College
Toronto, Ontario, Canada
mradmy@my.centennialcollege.ca

Ronald Saenz Huerta
School of Engineering
Technology and Applied Science
Centennial College
Toronto, Ontario, Canada
rsaenzhu@my.centennialcollege.ca

Mohammed Assem
School of Engineering
Technology and Applied Science
Centennial College
Toronto, Ontario, Canada
massem@my.centennialcollege.ca

I. Introduction

The dataset we used for this paper was the 2019 edition of iWildCam images from the sixth Fine-Grained Visual Categorization (FGVC6) workshop. It consists of photographs of animals in the wild that were taken by camera traps. A camera trap is a camera that takes a picture when it detects any movement and is thus used to monitor biodiversity in areas like forests. Our dataset contains photographs of animals taken by cameras in The American Southwest and the American Northwest.

We ran 3 different experiments on our dataset. The first experiment involved supervised learning. We built a Convolutional Neural Network and trained it to recognize and classify photographs of animals in our dataset. The second experiment involved unsupervised learning. Specifically, we used a pair of generative adversarial networks to generate fake images of different animals. The third experiment we ran involved using the state-of-the-art InceptionV3Inception v3 model. We built a Convolutional Neural Network and trained it to recognize and classify photographs of animals of our dataset using transfer learning from the pre-trained InceptionV3 model.

II. Methodology

A. *Dataset*

Dataset for training have retrieved from Kaggle dataset:

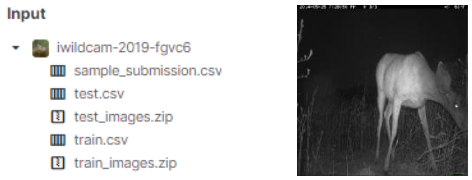
https://www.kaggle.com/c/iwildcam-2019-fgvc6

The dataset called "iWildCam 2019 - FGVC6" consists of images of animals with labels indicating the type of animal: deer, opossum, raccoon, fox, coyote, etc.

The original (training) dataset includes 196,299 images from 100 locations in Southern California.

The pixel value range is 0-255

Here are the input dataset and a sample image from the dataset:



License: CC0: Public Domain

B. *Data Preprocessing*

After performing data exploration, we learned a few facts about our data. For example, we learned that there were 23 classes of animals labeled, with one class being labeled 'empty' for photos without animals. We also noticed that 131,457 of the 196,299 images in our dataset were empty.

In other words, the dataset was unbalanced. So, we removed all the empty photographs. Another good reason to remove empty photographs is that they did not contain any relevant information.

Features with information that was not useful were also deleted from our data. The only features we kept were the images, labels, and their respective file names.

C. *Supervised Learning Experiment*

Parameters and Data Preparation:

Our experiment employed TensorFlow and Keras for building a CNN model for image classification. The dataset, sourced from a CSV file, comprised images of wildlife. We initially cleaned the data by dropping irrelevant columns such as 'date_captured', 'frame_num', 'id', etc. and examined the distribution of image dimensions and categories.

Labeling and Data Balancing:

We mapped numeric category IDs to wildlife species names for interpretability. To address class imbalance, we employed

down sampling and ensuring each category was equally represented. The dataset was stratified split into training, validation and testing sets with careful attention to maintaining class proportionality in each split. The split ratios were 80%, 10%, and 10% respectively.

Image Preprocessing:

Images were resized to a uniform dimension and normalized for model input. Categorical encoding was applied to the labels for compatibility with our CNN model. Images were compressed to (32, 32, 3). We scale the images' pixel value into 0-1 by diving all pixel value with 255. For our labels, we one-hot encoded them.

Model Architecture:

The CNN model consists of a sequence of layers designed for efficient feature extraction and classification from wildlife images. The initial layer of the model is a Convolutional 2D (Conv2D) layer with 64 filters, a kernel size of 3x3, and 'same' padding, which helps in capturing the spatial features from the images. This layer is followed by a dropout layer with a rate of 35% to reduce the risk of overfitting. Subsequently, the model includes another Conv2D layer, this time with 128 filters, maintaining the same kernel size and padding. This layer is designed to further refine the features extracted by the previous layer. Following this layer, another dropout layer is introduced, with a higher rate of 45%, to further mitigate overfitting. Each of these Conv2D layers is followed by a MaxPooling2D layer with a pool size of 2x2. These pooling layers serve to down sample the feature maps, reducing their dimensions and the overall computational complexity. After the convolutional and pooling layers, the model includes a Flatten layer, which converts the pooled feature maps into a one-dimensional vector. This is crucial for transitioning from convolutional layers to dense layers. Next, there is a Dense layer with 1024 neurons, which acts as a fully connected layer in the network. This layer is accompanied by a significant drop out of 75%, aiming to prevent overfitting while maintaining substantial model complexity. The final layer of the model is another Dense layer, which corresponds to the number of categories in the wildlife dataset. This layer uses a SoftMax activation function, making it suitable for multi-class classification tasks.

Training and Evaluation:

The model is trained on wildlife images for 150 epochs with a batch size of 64, employing Early Stopping to prevent overfitting. Post-training, its performance was evaluated on separate test and validation sets to assess generalization capabilities. Accuracy metrics were particularly emphasized and the best model configurations were saved for reproducibility and future reference.

Hyperparameter Tuning and Model Predictions:

We utilized Keras Tuner for hyperparameter optimization, fine-tuning aspects like units, activation functions, dropout rates and saving the best parameters for consistency. The model's efficacy was demonstrated through predictions on test images, highlighting true vs. predicted labels, with accuracy metrics and detailed results recorded in CSV files for comprehensive analysis.

*D. Unsupervised Learning Experiment - **GAN***

For the unsupervised learning experiment, a pair of generative adversarial networks were used to generate images of animals. More specifically, the generator model was used to generate images. We experimented with a few different types of sequential models before settling on the architecture we chose. We also tried a few different architectures for the discriminator model before choosing its architecture.

Parameters and Data Preparation:

Our experiment employed TensorFlow and Keras for building a generator and discriminator model for image generation. The dataset, sourced from a CSV file, comprised images of wildlife. We initially cleaned the data by dropping irrelevant columns such as 'date_captured', 'frame_num', 'id', etc. and examined the distribution of image dimensions and categories.

We split the dataset into the animal's categories like deer, opossum, coyote, etc., to try to generate new animal images for each category.

Image Preprocessing:

Images were resized to a uniform dimension and normalized for model input. Categorical encoding was applied to the labels for compatibility with our CNN model. Images were compressed to (64, 64, 3). We scale the images' pixel value into -1 +1 by diving all pixel value with 127.5. It was necessary to use data augmentation to generate more training images. We apply random cropping techniques. We prepared 25000 images in total for our generative model.

Model Architecture:

The GAN model consists of two parts: generator and discriminator. We utilized TensorFlow and Keras to build those models using CNN.

The generator model for 64x64 images in a GAN starts with a dense layer, transforming a 100-dimensional noise vector into a 4x4 feature map with 1024 channels. Batch normalization and Leaky ReLU activation ensure stable training. A reshape layer prepares for upsampling, using Conv2DTranspose layers to progressively increase spatial dimensions from 4x4 to 64x64. The final Conv2DTranspose layer outputs RGB images with a tanh activation. This architecture enables the generator to produce realistic high-resolution images through the GAN training process.

The discriminator model for 64x64 images in a GAN is designed to distinguish between real and generated images. It follows a series of convolutional layers to downsample the input images and identify key features. It initiates with a Conv2D layer with 64 filters, a 5x5 kernel size, and a stride of 2, processing the 64x64 RGB images. The Leaky ReLU activation introduces non-linearity, while dropout layers help prevent overfitting. The model outputs a single neuron (1D vector) representing the probability of the input being a real image. The absence of an activation function in the output layer is intentional, as the loss function will handle the necessary activation. This architecture enables the discriminator to

effectively discern between real and generated images during the GAN training process.

We used an Adam optimizer with a learning rate of 0.0002 and a beta-1 value equal to 0.5 for both the generator and the discriminator.

Training and Evaluation:

The model is trained on wildlife images for 100 epochs with a batch size of 64, and a latent dimension of 100. Post-training, its performance was evaluated on separate test and validation sets to assess generalization capabilities. Loss metrics were particularly emphasized, and the model configurations were saved for reproducibility and future reference.

E. *Transfer Learning from State-of-the-Art Model Experiment – InceptionV3*

Parameters and Data Preparation:

We initialized our experiment by setting up the necessary parameters for image processing. This included image resizing to fit the InceptionV3 model's expected input dimensions (75x75 pixels) and preprocessing to match the model's training conditions.

Label Encoding and Stratification:

Each image was labeled with a corresponding category from a predefined set of wildlife classes. Labels were encoded as integers and then converted to one-hot vectors to facilitate the classification process. To ensure the model's ability to generalize, we employed stratified sampling to maintain an even distribution of classes within the training, validation, and test datasets.

Transfer Learning Framework:

We utilized the InceptionV3 model, which has been pre-trained on ImageNet, as a feature extractor. The base model was loaded without its top layer to allow for the integration of custom dense layers designed for our classification task. A Global Average Pooling layer was appended to condense the feature maps followed by a dense layer with 1024 neurons and ReLU activation. The final layer consisted of a SoftMax activation function with a neuron for each wildlife category, providing a probability distribution over the classes.

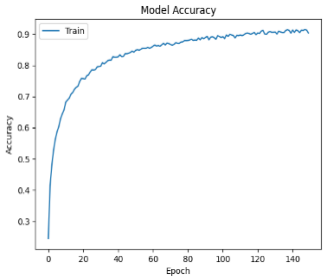
Model Training and Validation:

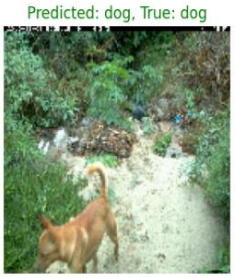
The composite model is compiled with an Adam optimizer and categorical cross-entropy loss function. We trained the model for 60 epochs with a batch size of 64. An EarlyStopping callback was utilized to halt training if the validation accuracy did not improve, mitigating overfitting. The training progress was visualized by plotting the accuracy and loss for each epoch.

III. Results

A. *Supervised Learning Experiment*

The model demonstrated a high training accuracy of 99% with a loss converging close to zero which indicates a strong fit to the training data. It achieved 76.7% on validation and 81.2% on the test set, reflecting effective generalization despite some signs of overfitting. The higher test accuracy suggests the model's potential robustness in real-world applications.

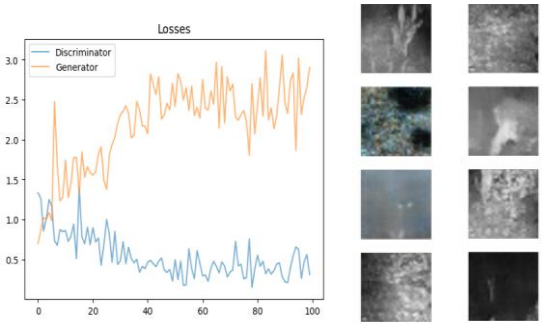




B. *Unsupervised Learning Experiment - GAN*

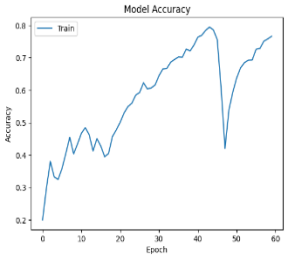
The generator model did not generate very good images of deer. We suspect that the images would have gotten clearer with more training epochs and perhaps more computational power.

We tried to generate new images using the animal Opossum, but the results were not good enough because most of the photos were taken at night. The grayscale generates some problems in unsupervised models. Then, we tried with animal Deer, and the results were better.



C. *Transfer Learning from State-of-the-Art Model Experiment – InceptionV3*

The InceptionV3-based transfer learning model achieved a peak training accuracy of 76%, with the validation and test accuracies reaching approximately 63% and 64% respectively. The accuracy and loss plots indicate learning with potential volatility, suggesting further model tuning might be beneficial to enhance stability and performance.



D. *Performance of all 3 experiments*

Curiously, the convolutional neural network built during the supervised learning experiment performed worse with transfer learning than without. Meanwhile, the unsupervised learning experiment produced the worst results.

IV. Contribution

A. Manipal Sidhu

- Supervised Learning Model
- State-of-the-art Model
- Report and Presentation

B. Mahpara Rafia Radmy

- State-of-the-art Model
- Report and Presentation

C. Ronald Saenz Huerta

- Unsupervised Learning Model
- Data Processing

D. Mohamed Assem

- Data Exploration
- Report and Presentation

V. References

[1] K. Team, "Keras Documentation: Inceptionv3," Keras, https://keras.io/api/applications/inceptionv3/ (accessed Dec. 12, 2023).

[2] CVPR2019, https://cvpr2019.thecvf.com/ (accessed Dec. 12, 2023).

[3] "The IWILDCAM 2019 challenge dataset - arxiv.org," kaggle.com, https://arxiv.org/pdf/1907.07617v1 (accessed Dec. 12, 2023).