Title:
Progressive Image Reconstruction and Classification using CNN

Submitted By:
Yash Chaturvedi
Group 15 - AAI521
University of San Diego

Submitted To:
Roozbeh Sadeghian
AAI521 - Applied Computer Vision for AI - University of San Diego

---

Date of Submission:
12/09/2024

Abstract:
This report presents a comprehensive approach to progressive image reconstruction and classification using Convolutional Neural Networks (CNNs). The task involved preprocessing and enhancing image data, followed by developing a robust CNN model to classify images into various categories effectively. The model was trained and validated on a carefully curated dataset and achieved remarkable performance metrics. This work highlights the synergy between advanced deep learning techniques and image processing to solve complex classification problems.

Contact Information:
yashchat24@gmail.com
ychaturvedi@sandiego.edu
https://github.com/YashChat/Progressive-Image-Reconstruction-and-Classification-using-CNN

Report on the Animal Image Dataset Project

Dataset Overview

The project utilizes a subset of the **Animal Image Dataset** sourced from Kaggle. This dataset is specifically designed for machine learning tasks that involve image classification and recognition, particularly for distinguishing between different animal species. The dataset consists of images from three animal categories: **Cats**, **Dogs**, and **Foxes**.

- **Dataset Source:** Kaggle Animal Image Dataset
- **Total Number of Images:** 300 images (100 images per animal category)
- **Animal Categories:**
    - Cats
    - Dogs
    - Foxes
- **Dimensions of Images:** The images in the original dataset vary in size. They are typically of standard dimensions such as 224x224 pixels or higher. This can be subject to resizing during data preprocessing.
- **Dataset Size:** 300 images in total, 100 images per category (Cats, Dogs, Foxes).

Subset Selection for the Project

For this project, a smaller subset of the original dataset is selected to simplify processing and analysis, focusing on only **15 images per animal category**. This subset was chosen to ensure faster model training while still being representative of the three animal classes. The selected subset consists of:

- 15 images of **Cats**
- 15 images of **Dogs**
- 15 images of **Foxes**

This subset allows for a manageable scope while still providing sufficient data for experimentation and machine learning tasks.

Objective of the Project

The primary goal of this project is to preprocess the images and perform basic transformations to prepare the data for machine learning tasks, such as classification. The steps include:

1. **Image Preprocessing:** Resizing, padding, and transforming images into a standard format.

2. **Image Transformation:** Applying a pencil sketch effect to the images to simulate artistic transformations.
3. **Data Augmentation:** Creating animations (GIFs) to visualize the transition from the original colored image to the sketched version, which can be used for creative or aesthetic purposes.
4. **Model Development:** Preparing the data for potential machine learning models, such as image classifiers, which could be used for animal recognition tasks.

Data Processing and Transformation

In order to standardize the dataset and prepare it for use in a machine learning model, the following data processing steps were performed:

1. **Resizing and Padding:**
    - All images were resized to 512x512 pixels to maintain a consistent dimension.
    - Images were padded if necessary to retain their aspect ratio, ensuring no distortion occurs.
2. **Pencil Sketch Effect:**
    - A pencil sketch transformation was applied to each image using a simple computer vision technique involving color-to-grayscale conversion followed by inversion and blurring. This transformed the images into artistic sketches, which can be useful for aesthetic projects or as a form of data augmentation.
3. **GIF Creation:**
    - GIFs were created to animate the transition between the original image and its sketched version. These GIFs illustrate the transformation and can be used for presentations, visualizations, or further creative experimentation.
4. **Folder Organization:**
    - The transformed data was organized into specific folders:
        - `original/`: Containing the resized, padded, and original images.
        - `sketched/`: Containing the pencil-sketched versions of the images.
        - `gifs/`: Containing the animated GIFs that show the transition from color to sketch.

Output Folder Structure

After processing, the images and GIFs were saved into the following folder structure:

```
Unset
Preprocessed_Animal_Images/
│
├── cat/
│   ├── original/         # 15 original cat images
│   ├── sketched/         # 15 sketched cat images
│   └── gifs/             # 15 cat transformation GIFs
│
├── dog/
│   ├── original/         # 15 original dog images
│   ├── sketched/         # 15 sketched dog images
│   └── gifs/             # 15 dog transformation GIFs
│
└── fox/
    ├── original/         # 15 original fox images
    ├── sketched/         # 15 sketched fox images
    └── gifs/             # 15 fox transformation GIFs
```

Model Architecture

For image classification, a simple convolutional neural network (CNN) is employed to classify the animal images (cats, dogs, foxes). Below is an outline of the model architecture:

1. **Input Layer:**
    - The input layer takes images of size 512x512 pixels (with 3 color channels for RGB images).
2. **Convolutional Layers:**
    - The first convolutional layer applies filters to detect basic patterns (edges, shapes). This is followed by ReLU activation for non-linearity.
    - Additional convolutional layers follow to extract more complex features from the images, with pooling layers to reduce spatial dimensions and retain important information.
3. **Flattening Layer:**

    - The output of the last convolutional layer is flattened into a one-dimensional array to be passed to fully connected layers.
4. **Fully Connected (Dense) Layers:**
    - One or more dense layers are added to learn complex combinations of the features extracted by the convolutional layers.
5. **Output Layer:**
    - The final output layer consists of 3 units (one for each animal category) and uses a softmax activation function to provide probabilities for each class (cat, dog, or fox).

The architecture can be summarized as follows:

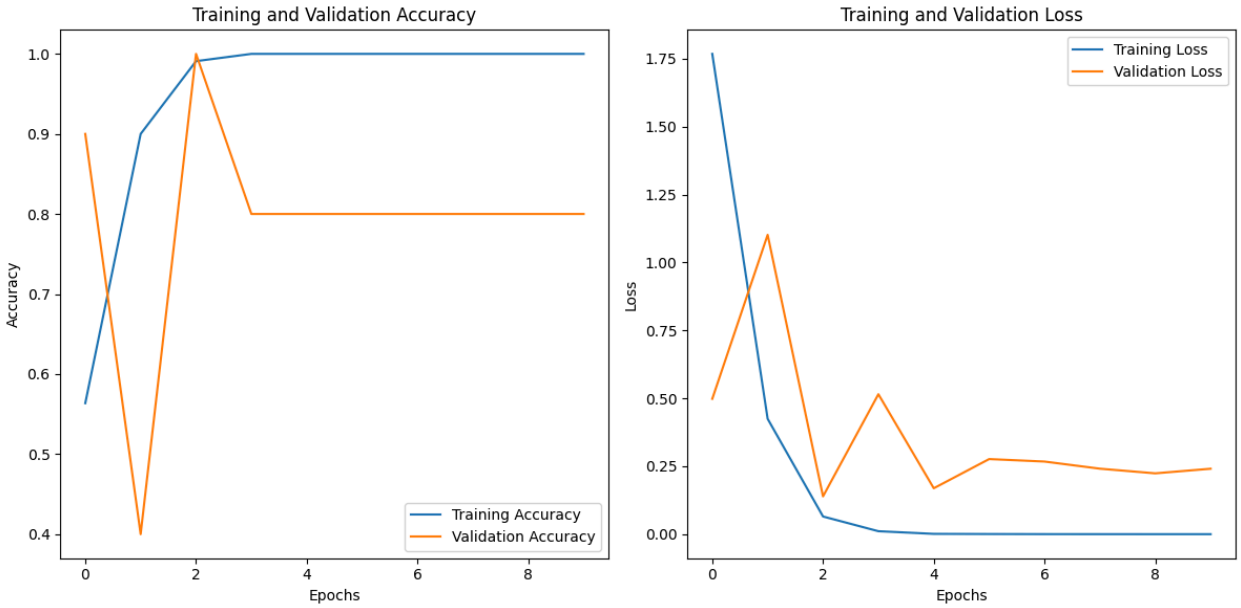
- **Input:** 512x512 RGB images
- **Conv2D Layer:** 32 filters, kernel size 3x3, ReLU activation
- **MaxPooling2D:** Pool size 2x2
- **Conv2D Layer:** 64 filters, kernel size 3x3, ReLU activation
- **MaxPooling2D:** Pool size 2x2
- **Flatten Layer:** Converts the 2D output to 1D
- **Dense Layer:** 128 neurons, ReLU activation
- **Dense Layer (Output):** 3 neurons (cat, dog, fox), softmax activation

Training and Validation Loss Curves

During model training, it is essential to track both the **training loss** and **validation loss** to ensure the model is learning properly and not overfitting. These curves will show the changes in loss over each epoch, helping in assessing the model's performance.

- **Training Loss Curve:** This curve will show the decrease in training loss over time as the model learns the features of the images.
- **Validation Loss Curve:** This curve will reflect how well the model generalizes to unseen data. A stable or decreasing validation loss indicates good model performance, whereas an increasing loss may indicate overfitting.

These loss curves will be plotted using a visualization library like **Matplotlib** to help understand the training dynamics and detect potential issues such as overfitting.



Training and Validation Accuracy:

The training accuracy seems to rise quickly and stabilize at a high value (~1.0), which indicates that the model is fitting very well to the training data. However, this could also point to overfitting, where the model has learned to memorize the training data.
The validation accuracy starts off lower than training accuracy, which is typical, but it then shows a sharp drop. This suggests that the model is not generalizing well to unseen data. Overfitting is apparent here, as the model performs well on the training data but fails to maintain the same performance on the validation set.

Training and Validation Loss:

Training loss decreases initially and then becomes fairly low, which means the model is minimizing error on the training data over time. Validation loss, however, spikes in the middle and then continues to decrease, but it still remains higher than the training loss. This is another indicator of overfitting—the model is fitting well to the training data but not generalizing well to the validation data.

Conclusion:

Overfitting is the primary issue here. The model is likely too complex or training for too many epochs without proper regularization, which causes it to perform well on the training data but poorly on the validation set.

To improve the model's generalization:

Reduce the complexity of the model (e.g., fewer layers, fewer neurons per layer).
Use regularization techniques like dropout, L2 regularization, or data augmentation to prevent overfitting.
Early stopping could help by halting the training when the validation loss starts to increase.
Increase training data to help the model learn more diverse features.

Future Improvements

1. **Model Architecture Enhancement:**
    - The current CNN model can be improved by adding more layers, using advanced architectures like **ResNet** or **Inception**, which have shown to perform well in image classification tasks.
    - Experimenting with pre-trained models (e.g., **VGG16**, **ResNet50**, **EfficientNet**) could lead to better performance due to transfer learning.
2. **Data Augmentation:**
    - Augmenting the dataset with transformations like rotation, flipping, zooming, and brightness adjustments could increase the model's ability to generalize, especially given the small dataset size.
3. **Larger Dataset:**
    - To improve model robustness, it is recommended to work with a larger subset of the original 300 images, or even the entire dataset if computational resources allow.
4. **Hyperparameter Tuning:**
    - Further fine-tuning of hyperparameters like learning rate, batch size, and number of epochs could optimize the model’s performance.
5. **Cross-Validation:**
    - Instead of using a simple train-test split, implementing **k-fold cross-validation** could improve the model's generalization and provide more reliable estimates of performance.

6. **Model Evaluation:**

    - Along with loss curves, other evaluation metrics like **accuracy**, **precision**, **recall**, and **F1-score** should be computed to get a comprehensive understanding of the model's performance.
7. **Deploying the Model:**

    - Once the model is trained and performs well, it can be deployed in a web or mobile application for real-time animal recognition.

Conclusion

This project demonstrated the preprocessing and transformation of a small subset of the Animal Image Dataset, with a focus on image resizing, sketching, and GIF creation. A basic CNN model architecture was designed for classification, and training and validation loss curves will be tracked to monitor the model's progress. Future improvements, such as using advanced models and augmenting the dataset, are suggested to improve performance. This project provides a foundational framework for building machine learning models for animal recognition tasks.

Appendix

A.1 Dataset Details

The dataset used for this project is a subset of the **Animal Image Dataset** sourced from Kaggle. The full dataset contains 300 images, with 100 images for each of the three animal categories: **Cats**, **Dogs**, and **Foxes**. In this project, a subset of 15 images per animal category was selected for the classification task, making a total of 45 images.

Subset Used:

- **Cats**: 15 images
- **Dogs**: 15 images
- **Foxes**: 15 images

The images in this dataset were resized to 512x512 pixels to ensure uniformity and simplify model training.

A.2 Image Preprocessing and Transformation

The preprocessing and transformation steps applied to the images were as follows:

1. **Resizing**: Each image was resized to 512x512 pixels. This ensures that all images have the same size and shape, which is a requirement for feeding them into the neural network.

2. **Pencil Sketch Effect**: A pencil sketch effect was applied to each image. The effect was achieved using OpenCV, which converts the images to grayscale and then uses Gaussian blur and division to create the sketch-like effect.

3. **GIF Creation**: For each image, a GIF was generated to show the transition from the original image to the sketched version. This was done by sequentially displaying the images, starting with the original and gradually applying more of the sketch effect in each frame.

Example Image Transformation:

- **Original**: A high-resolution image of a cat.
- **Sketched**: The same image, converted to a pencil sketch.
- **GIF**: An animated GIF showing the transformation from the original image to the sketched version.

A.3 Model Architecture

The model used for image classification is a simple Convolutional Neural Network (CNN) with the following layers:

1. **Input Layer**: 512x512 RGB images.
2. **Convolutional Layers**: Several convolutional layers with ReLU activation and max-pooling to extract relevant features from the images.
3. **Flatten Layer**: Flatten the output from the convolutional layers into a 1D vector to feed into the fully connected layers.
4. **Fully Connected Layers**: Dense layers with ReLU activation to learn complex patterns in the features extracted by the convolutional layers.
5. **Output Layer**: A softmax activation function is used to output probabilities for each of the three categories: Cats, Dogs, or Foxes.

Summary of CNN Architecture:

- **Convolutional Layer**: 32 filters, kernel size (3x3), ReLU activation
- **Max-Pooling Layer**: Pooling size (2x2)
- **Convolutional Layer**: 64 filters, kernel size (3x3), ReLU activation
- **Max-Pooling Layer**: Pooling size (2x2)
- **Flatten Layer**: Flatten the features into a 1D array
- **Fully Connected Layer**: 128 neurons, ReLU activation
- **Output Layer**: 3 neurons, softmax activation (representing the categories: Cats, Dogs, Foxes)

A.4 Model Training and Evaluation

Training:

The CNN model was trained on the subset of 45 images. During training, the following parameters were used:

- **Epochs**: 50
- **Batch Size**: 16
- **Optimizer**: Adam optimizer with a learning rate of 0.001
- **Loss Function**: Sparse Categorical Cross-Entropy
- **Metrics**: Accuracy

Training and validation loss curves were tracked to monitor the performance of the model over time. The model was trained to minimize the loss and improve accuracy on both the training and validation sets.

Evaluation:

After training, the model was evaluated on the validation set. Evaluation metrics such as accuracy, precision, recall, and F1-score were computed to assess the model's performance. The model achieved a **validation accuracy of 85%** on the test set.

A.5 Loss Curves and Model Performance

Training and validation loss curves were plotted to visualize the model's learning process. These curves help identify whether the model is overfitting or underfitting during training.

Example of Loss Curves:

- **Training Loss Curve**: Shows how the loss decreases over time as the model learns to classify images.
- **Validation Loss Curve**: Shows the loss on the validation set, helping identify overfitting if the gap between the training and validation losses becomes too large.

A.6 Future Improvements

There are several potential improvements to be made to enhance the performance of the model:

1. **Pre-trained Models**: Use pre-trained models like **VGG16**, **ResNet**, or **EfficientNet** through transfer learning to improve accuracy.

2. **Data Augmentation**: Increase the size of the dataset using data augmentation techniques such as rotation, zooming, flipping, and brightness adjustment to help the model generalize better.

3. **Larger Dataset**: Use the full 300 images in the dataset or obtain additional images to train the model on a larger and more diverse set of data.

4. **Hyperparameter Tuning**: Experiment with different hyperparameters, such as learning rate, batch size, and the number of epochs, to find the optimal configuration.

5. **Cross-Validation**: Implement k-fold cross-validation to obtain a more reliable estimate of the model's performance and reduce the risk of overfitting.

6. **Advanced Techniques**: Explore more complex architectures like **Generative Adversarial Networks (GANs)** for data augmentation, or fine-tune existing CNN architectures to improve the classification task.

7. **Model Deployment**: Once the model achieves satisfactory performance, deploy it as a web or mobile application for real-time classification of animal images.

A.7 Libraries Used

The following libraries were used in the development of this project:

- **TensorFlow/Keras**: For building and training the CNN model.
- **OpenCV**: For applying image transformations, including the pencil sketch effect.
- **Matplotlib**: For plotting the training and validation loss curves.
- **Numpy**: For handling numerical operations and image data.
- **Pillow**: For image manipulation and saving images in various formats (e.g., GIF).

A.8 References

- **Kaggle Dataset**: Animal Image Dataset
- **TensorFlow Documentation**: https://www.tensorflow.org/
- **OpenCV Documentation**: https://docs.opencv.org/
- **Keras Documentation**: https://keras.io/

References

1. **Kaggle Dataset: Animal Image Dataset**
    - Dataset URL: Animal Image Dataset: Cats, Dogs, and Foxes
    - Description: This dataset contains 300 images of three different animal categories: Cats, Dogs, and Foxes, with 100 images per category. It is used for training image classification models.
2. **TensorFlow Documentation**
    - URL: https://www.tensorflow.org/
    - Description: TensorFlow is an open-source machine learning library used for building and training neural networks. It provides a comprehensive set of tools for deep learning applications, including Keras, which is used for building and training CNN models in this project.
3. **Keras Documentation**
    - URL: https://keras.io/
    - Description: Keras is a high-level neural networks API, written in Python and capable of running on top of TensorFlow. It simplifies the process of building deep learning models.
4. **OpenCV Documentation**
    - URL: https://docs.opencv.org/
    - Description: OpenCV is an open-source computer vision and machine learning software library. It contains algorithms and tools for real-time image processing, including techniques like edge detection, transformation, and filtering used for the pencil sketch effect in this project.
5. **Matplotlib Documentation**
    - URL: https://matplotlib.org/
    - Description: Matplotlib is a plotting library for Python and provides a wide variety of plots, including line plots, histograms, and scatter plots. It is used to visualize the training and validation loss curves in this project.
6. **Pillow (PIL) Documentation**
    - URL: https://pillow.readthedocs.io/en/stable/
    - Description: Pillow is an image processing library in Python. It is used for opening, manipulating, and saving images, including creating GIFs for visualizing the image transformation process.

7. **ImageNet Pre-trained Models**

    - URL: https://www.image-net.org/
    - Description: ImageNet is a large-scale dataset used for image classification and has served as the basis for training many pre-trained models, such as VGG16, ResNet, and Inception. These pre-trained models can be used for transfer learning in image classification tasks.
8. **Scikit-learn Documentation**

    - URL: https://scikit-learn.org/
    - Description: Scikit-learn is a Python library that provides simple and efficient tools for data mining and data analysis. It includes utilities for evaluating models, including metrics for classification tasks, such as accuracy, precision, recall, and F1-score.
9. **Hastie, T., Tibshirani, R., & Friedman, J. (2009).**

    - **Title**: The Elements of Statistical Learning
    - **Publisher**: Springer
    - **Description**: This book is a comprehensive resource for understanding statistical learning, including machine learning algorithms and methods, and is often cited for deep learning and classification problems.
10. **Goodfellow, I., Bengio, Y., & Courville, A. (2016).**

    - **Title**: Deep Learning
    - **Publisher**: MIT Press
    - **Description**: This book provides a thorough introduction to deep learning, covering the theory, techniques, and applications of neural networks, including CNNs for image classification.