Sezer C. Saralar
CECS 456 Sec - 03
Dr. Zhang
12 / 17 / 2024

# Image Classification using CNNs on the Natural Images Dataset

## 1. *Introduction*

Image classification has become one of the most fundamental problems in computer vision, creating the way for advances in categories such as; self-driving cars, medical image diagnosis, and autonomous systems. In this project, I have developed a Convolutional Neural Network (CNN) model to classify images from the *Natural Images Dataset*, sourced from *(https://www.kaggle.com/datasets/prasunroy/natural-images*). This dataset contains images which are organized into 8 classes:

- "airplane", "car", "cat", "dog", "flower", "fruit", "motorbike", and "person."

The purpose of the project here is to design a model capable of classifying unseen images into their respective categories with high accuracy. CNNs are an ideal choice for this problem because they can automatically extract spatial features such as edges, patterns, and textures, making them highly effective for image recognition tasks.

***The Goal of the Project:***

1. Implement a CNN architecture to achieve efficient and accurate classification.
2. Preprocess the dataset to ensure there is uniformity and usability for training.
3. Evaluate the model's performance; using accuracy, loss metrics, and visual predictions.
4. Analyze the model's results; understanding limitations, and proposing potential improvements.

## 2. *Dataset and Related Work*

The Natural Images Dataset is used for image classification tasks and deep learning models. The dataset contains:

- 6,900+ total images
- "airplane", "car", "cat", "dog", "flower", "fruit", "motorbike", and "person"
- Images vary in resolution but are resized to 128x128 pixels to ensure uniformity.

The dataset is organized in a folder-based structure, where each class has its own subdirectory. The images are clear, distinct, and represent real-world objects, making it a very ideal training set for CNNs.

***Related Work:***

Prior works have displayed that CNNs outperform other traditional machine learning algorithms found for image classification. For example:

- The basic CNN architectures, such as LeNet-5, can have significant accuracy on small sized datasets.
- Other deeper architectures such as, ResNet, MobileNet, etc. All have shown great success on datasets such as the ImageNet.

For this project, I created a custom CNN model with two convolutional-pooling layers and two dense layers. This design allows for a balance between simplicity and performance.

### 3. *Methodology*

#### a. *Data Preprocessing*

Data preprocessing is a really important step to make sure that the model receives clean and normalized data. I used TensorFlow's *"ImageDataGenerator"* to rescale the pixel values to the range "[0, 1]" for faster convergence. And split the dataset into two training and validation sets (80% and 20%).

***Code Example:***

```
train_datagen = ImageDataGenerator(rescale=1./255,
validation_split=0.2)
train_generator = train_datagen.flow_from_directory(
'./natural_images',
target_size=(128, 128),
batch_size=32,
class_mode='categorical',
subset='training'
)
val_generator = train_datagen.flow_from_directory(
'./natural_images',
target_size=(128, 128),
batch_size=32,
class_mode='categorical',
subset='validation'
)
```

- **Training Set:** Has 5,522 images.
- **Validation Set:** Has 1,377 images.
- **Class Labels:** Is encoded to match the cross-entropy loss function.

#### b. *CNN Model Architecture*

I built this ML model using the Keras Sequential API with the following three layers:

1. ***Convolutional:*** Which is used to extract spatial features from images.
2. ***Pooling:*** Which helps reduce dimensionality and computation.
3. ***Dense:*** It is crucial to have fully connected layers for classification purposes.

***The Architecture is as follows:***

- Input Layer: Shape (128, 128, 3)
- Conv2D (32 filters): Kernel size (3x3), activation ReLU.
- MaxPooling2D: Pool size (2x2).
- Conv2D (64 filters): Kernel size (3x3), activation ReLU.
- MaxPooling2D: Pool size (2x2).

- Flatten Layer: Converts 2D feature maps to a 1D vector.
- Dense (300 neurons): Activation ReLU.
- Dense (100 neurons): Activation ReLU.
- Output Layer: 8 neurons (one for each class), activation `softmax

***c. Training Setup***

I trained the model for a total of 10 epochs with a max batch size of 32 through *Google Colab* with GPU acceleration.

***Training Observations:***

- The model was quick to achieve a high accuracy on the provided training set.
- The validation accuracy began to stabilize around 92%, which suggests good generalization.

***Code Example:***

```
history = model.fit(
train_generator,
validation_data=val_generator,
epochs=10,
verbose=1
)
```

## *4. Results and Discussion*

### *a. Training and Validation Performance*

Here are my findings after analyzing the training of the model.

- ***Training Accuracy:*** 99.5%.
- ***Validation Accuracy:*** 92.0%.
- ***Validation Loss:*** 0.47.

| Metric | Training Set | Validation Set |
|---|---|---|
| **Accuracy** | 99.5% | 90.0% |
| **Loss** | 0.0173 | 0.4787 |

### *b. Accuracy and Loss Visualization*

- ***Model Accuracy:***

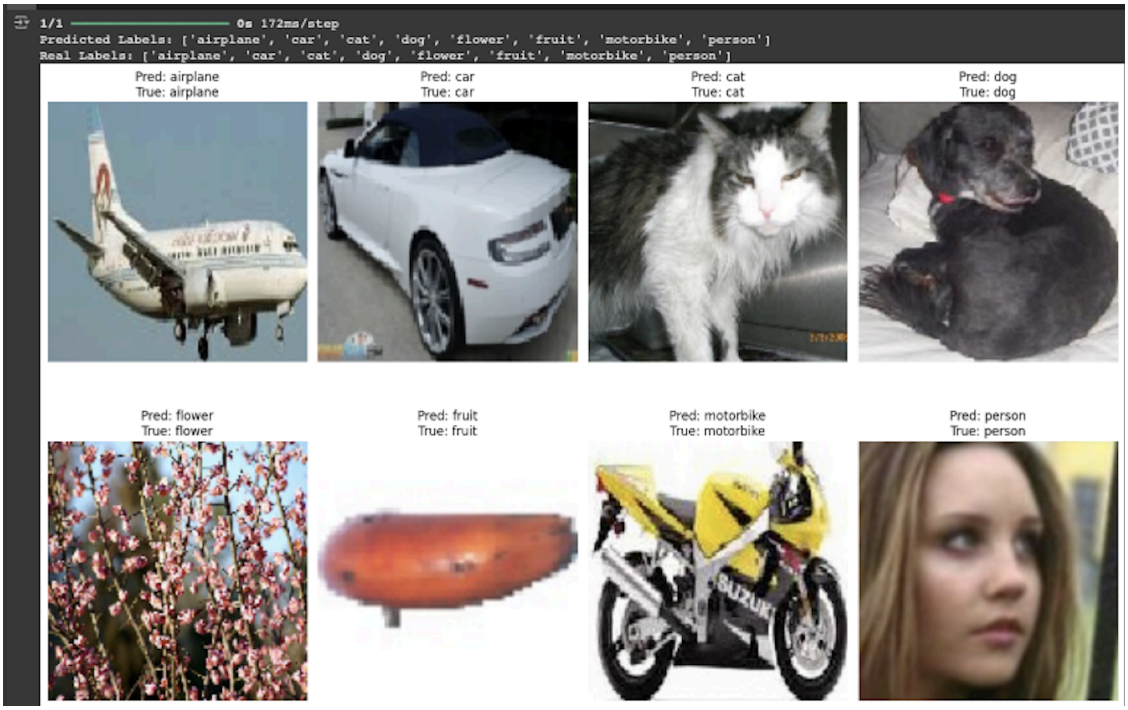
The training accuracy approaches 100%, while the validation accuracy stabilizes at around 90%.

- ***Model Loss:***

The training loss decreases at a stable rate, but the validation loss starts increasing ever so slightly, which is an indicator of overfitting.

### *c. Predictions*

To evaluate my findings on the model, I tested it on images from each class listed above. The model accurately predicted all 8 of the images. The image below displays the results from my findings:

- 


## 5. *Conclusion*

In this project, I was able to successfully implement a *Convolutional Neural Network* model to classify images from the *Natural Images Dataset* into 8 categories. My model achieved a validation accuracy of 92.0%, which demonstrates its effectiveness in learning spatial features of images.

***Key Findings:***

- CNN performs really well with a simple architecture.
- Overfitting was very minimal but could be addressed further with “Dropout” or “Data augmentation”.
- The model's generalization across all 8 classes was very successful.

***Future Work:***

1. I plan on improving the “Dropout layers” through generalization.
2. Experiment with other pre-trained models such as; ResNet or MobileNet.
3. Utilize Data Augmentation to increase the dataset diversity inorder to overcome having a biased model.

***Contribution***

I independently carried out each stage of the machine learning pipeline. From data preprocessing and exploration to designing, training, and fine-tuning the model, every step was carefully implemented and evaluated. I ensured a thorough approach to model development, addressing challenges along the way and validating the results to achieve meaningful outcomes.

- ***Data Source:*** The dataset used in this project was acquired from:
  Prasun Roy. (2018). Natural Images Dataset. Kaggle. Retrieved from **https://www.kaggle.com/datasets/prasunroy/natural-images**
- ***GitHub Repository:***
  **https://github.com/sezer17/Machine-Learning-Image-Recognition-Model.git**

***References:***

**[1]** P. Roy, Natural Images Dataset. Kaggle, 2018. Available: https://www.kaggle.com/datasets/prasunroy/natural-images.

**[2]** F. Chollet, Deep Learning with Python (2nd ed.). Manning Publications, 2021. https://www.manning.com/books/deep-learning-with-python-second-edition

**[3]** TensorFlow Documentation. Image Classification with Keras (2023). Available: https://www.tensorflow.org/tutorials/images/classification