Project description: The goal is to create an AI project that will identify between 10 animal species based on pictures.

image Resources:Kaggle(Animals-10 (kaggle.com))

1. Installing Required Packages:

Before running the code, you need to make sure you have the required packages installed. You can install them using the following commands:

```
pip install keras
pip install tensorflow(Only works with python3.9,3.10,3.11 and latest pip package)
pip install matplotlib
pip install numpy
```

2. Data Preparation:

2.1. Directory Structure:

Each subdirectory should contain images of a specific species.

2.2. Data Augmentation:

The `ImageDataGenerator` is used for data augmentation. It rescales pixel values to the range [0, 1] and applies various transformations to increase the diversity of the training dataset. Here are some of the transformations:

- `rotation_range`: Degree range for random rotations.
- `width_shift_range` and `height_shift_range`: Ranges for random horizontal and vertical shifts.
- `shear_range`: Shear intensity (shear angle in degrees).
- `zoom_range`: Range for random zoom.
- `horizontal_flip` and `vertical_flip`: Randomly flip images horizontally and vertically.
- `fill_mode`: Strategy for filling in newly created pixels.

2.3. Data Generators:

The `flow_from_directory` function is used to create data generators for training and validation. It loads images from the specified directory, resizes them to the target size (224x224 in this case), and generates batches of augmented images.

- `target_size`: The dimensions to which all images will be resized.
- `batch_size`: The number of samples in each batch.
- `class_mode`: Specifies the type of label arrays returned. 'sparse' means integer labels.
- `subset`: Specifies if the generator is for training or validation.

3. Convolutional Neural Network (CNN) Architecture:

The CNN model is created using Keras Sequential API. Here's a breakdown of the architecture:

- Convolutional Layers:
    - The first convolutional layer (`Conv2D`) with 32 filters, each of size (3, 3), and ReLU activation.
    - Max-pooling layer (`MaxPooling2D`) with pool size (2, 2).
    - Batch normalization (`BatchNormalization`) to normalize the activations.
    - The second convolutional layer with 64 filters and ReLU activation.
    - Another max-pooling layer.
    - Batch normalization.
    - The third convolutional layer with 128 filters and ReLU activation.
    - Another max-pooling layer.
    - Batch normalization.
- Flatten Layer:
    - Flattens the output to a one-dimensional array before fully connected layers.
- Fully Connected Layers:
    - Dense layer with 256 neurons and ReLU activation.
    - Dropout layer with a dropout rate of 0.5 for regularization.
    - Output layer with units equal to the number of classes and softmax activation for multi-class classification.

4. Model Compilation:

The model is compiled with the Adam optimizer, sparse categorical crossentropy loss, and accuracy as the metric.

5. Learning Rate Schedule and Early Stopping:

- Learning Rate Schedule:
    - The learning rate is scheduled to decrease to 10% of its value after 10 epochs.
- Early Stopping:
    - Monitors validation loss and stops training if there's no improvement after 5 epochs. Restores the best weights.

6. Model Training:

The `fit_generator` function is used to train the model with the specified generators, steps per epoch, validation data, validation steps, and callbacks.

7. Visualizing Images with Predictions:

A function `visualize_images` is defined to visualize a few images along with their true and predicted labels using Matplotlib.

8. Model Evaluation:

The model is evaluated on the validation data using the `evaluate_generator` function, and the loss and accuracy are printed.

9. Making Predictions:

The model is used to make predictions on the validation data using the `predict_generator` function, and the predicted species names are printed.

Additional Notes:

- Image Size:
    - The target size for images is set to (224, 224). Adjust this size based on your specific requirements and available computational resources.
- Batch Size:
    - The batch size is set to 32, but you can modify it depending on your hardware specifications.
- Number of Epochs:
    - The model is trained for 30 epochs. You can adjust this based on the convergence behavior of your model.
- Data Exploration:

- It's recommended to explore the data before training to understand the distribution of classes and ensure the dataset is balanced.

By following these steps, you can install the required packages, set up the data directory, and train a CNN for image classification using the provided code.