AUBURN WAVES

REPORT

Prepared By :
Utkarsh Gupta
Dhruv Pani Chandra

DATA PREPROCESSING

DATA AUGMENTATION

The data augmentation process is defined using a Sequential model, which applies a series of transformations to the input images to artificially increase the size and variability of the training dataset. This helps in improving the robustness and generalization of the model by introducing variations.

- **Random Flip**: Horizontally flips the images randomly. This helps the model learn that the object can appear on either side.
- **Random Rotation**: Rotates the images randomly by a factor of 0.1 radians. This makes the model invariant to small rotations.
- **Random Zoom**: Zooms into the images randomly by a factor of 0.1. This allows the model to focus on different parts of the image.

RESCALING

Rescaling the pixel values of the images from the range [0, 255] to [0, 1]. This is done to normalize the input data, which helps in faster convergence during training.

MODEL ARCHITECTURE

The model is defined as a Sequential model, which stacks layers in a linear manner. Here's a breakdown of the layers used:

Data Augmentation Layer: Applies the data augmentation transformations to the input images.
Rescaling Layer: Normalizes the pixel values to the range [0, 1].
Convolutional and Pooling Layers: These layers are used to extract features from the input images.
Conv2D(16, 3, padding='same', activation='relu'): Applies 16 filters of size 3x3 to the input images with ReLU activation. Padding is set to 'same' to ensure the output size matches the input size.
MaxPooling2D(): Downsamples the spatial dimensions (width and height) of the input.
This process is repeated with an increasing number of filters (32, 64, 128) to learn more complex features at each stage.
Dropout Layer: Regularizes the model by randomly setting 20% of the input units to 0 during training. This helps in preventing overfitting.
Flatten Layer: Flattens the input, converting the 2D feature maps into a 1D vector.
Dense Layers: Fully connected layers used for classification.
Dense(128, activation='relu'): A dense layer with 128 units and ReLU activation.
Dense(8): The output layer with 8 units (assuming 8 classes), without activation function, as the loss function SparseCategoricalCrossentropy expects logits.

MODEL COMPILATION

The model is compiled with the following configurations:

Optimizer: Adam optimizer, which is an adaptive learning rate optimization algorithm that is widely used for training deep learning models.
Loss Function: Sparse Categorical Crossentropy, which is suitable for multi-class classification problems when labels are provided as integers.
Metrics: Accuracy, to monitor the training and validation accuracy during training.

MODEL TRAINING

The model is trained using the fit method, which takes the training dataset (train_ds), the number of epochs (100 in this case), and the validation dataset (val_ds).

SUMMARY

This approach uses data augmentation to increase the diversity of the training dataset, convolutional layers to extract features, and dense layers for classification. The model is regularized using dropout to prevent overfitting, and it is trained using the Adam optimizer and Sparse Categorical Crossentropy loss function. This setup is commonly used for image classification tasks and should work well if the data is sufficient and diverse.

CONTRIBUTION

Teammate 1: Utkarsh

- Data Preprocessing and Augmentation: Utkarsh designed and implemented the data augmentation pipeline. He identified the need for enhancing the training dataset's variability and robustness, leading to the integration of random horizontal flips, rotations, and zooms. This significantly improved the model's ability to generalize to unseen data.
- Model Architecture Design: Utkarsh contributed to the overall architecture of the Convolutional Neural Network (CNN). He proposed using multiple convolutional layers with increasing filter sizes and MaxPooling layers to effectively capture hierarchical features. He also integrated the Dropout layer to mitigate overfitting.
- Code Implementation: Utkarsh wrote the initial code for setting up the Sequential model, data augmentation, and the core layers of the CNN. He ensured the model followed best practices in deep learning and was structured efficiently.

Teammate 2:Dhruv

- Model Compilation and Optimization: Dhruv focused on the optimization aspects of the model. He selected the Adam optimizer for its adaptive learning rate properties and fine-tuned the learning rate for optimal performance. He also chose the Sparse Categorical Crossentropy loss function and ensured it was suitable for the multi-class classification problem.
- Model Training and Evaluation: Dhruv handled the training process, including setting the number of epochs and monitoring the model's performance. He tracked the training and validation accuracy, analyzed the results, and suggested adjustments to improve the model's accuracy and reduce overfitting.
- Documentation and Reporting: Dhruv documented the entire process, including the rationale behind each design choice, the hyperparameters used, and the results obtained. He prepared detailed reports and visualizations of the model's performance over epochs, which were crucial for understanding the model's behavior and communicating progress to stakeholders.