**Flower Classification Model: Report & Documentation**

**1. Data Preprocessing and Feature Engineering**

**Dataset:**

* The dataset consists of five classes: **daisy, dandelion, rose, sunflower, tulip**.
* Images are loaded from a directory structure where each folder corresponds to a class.

**Preprocessing Steps:**

1. **Load Dataset:** 
   * Used tf.keras.preprocessing.image\_dataset\_from\_directory() to load images.
   * Applied **training (70%), validation (15%), and test (15%)** split.
   * Ensured class labels are assigned correctly.
2. **Normalization:**
   * Rescaled pixel values from **[0, 255] to [0, 1]** using tf.keras.layers.Rescaling(1./255).
3. **Data Augmentation:**
   * Applied RandomFlip, RandomRotation, and RandomZoom to improve model generalization.
   * Only applied augmentation to the training dataset.
4. **Caching and Prefetching:**
   * Used .cache() and .prefetch(buffer\_size=AUTOTUNE) to optimize dataset loading.

**2. Model Selection and Optimization**

**Model Architecture:**

* **Pretrained CNN:** MobileNetV2 (pretrained on ImageNet, with include\_top=False).
* **Additional Layers:**
  + GlobalAveragePooling2D() to reduce dimensionality.
  + Dense(256, activation='relu') for feature extraction.
  + Dropout(0.3) to prevent overfitting.
  + Dense(5, activation='softmax') as the final classification layer.

**Compilation:**

* **Optimizer:** Adam (learning\_rate=0.001)
* **Loss Function:** Sparse Categorical Crossentropy
* **Metrics:** Accuracy

**Training Optimization:**

* **Early Stopping:** Monitored val\_loss, stopped if it didn't improve for 3 epochs.
* **Learning Rate Scheduling:** Reduced learning rate by a factor of 0.5 after 2 epochs of no improvement.

**3. Deployment Strategy and API Usage Guide**

**Deployment Approach:**

* **Framework:** FastAPI for model inference.
* **Authentication:** Basic authentication (admin/password) to secure the API.
* **Containerization:** Planned Docker deployment for production.

**API Endpoints:**

**1. Home Page (GET /)**

* Returns an HTML page with a file upload form.

**2. Prediction Endpoint (POST /predict/)**

* **Inputs:** Image file (JPEG, PNG, etc.).
* **Authentication:** Requires username/password.
* **Process:**
  1. Reads and preprocesses the image.
  2. Uses the trained CNN model to classify the flower.
  3. Returns a JSON response with the predicted class and confidence score.
* **Response Format:**

{

"prediction": "sunflower",

"confidence": 0.98

}

**Running the API:**

uvicorn app:app --host 0.0.0.0 --port 8000

**Future Enhancements:**

* Convert the model to TensorFlow Lite for mobile deployment.
* Deploy the API using **Docker + AWS/GCP**.
* Implement **JWT authentication** for better security.

**4. Code Availability**

* The complete code is available on [GitHub](https://chatgpt.com/c/67d30281-2d78-8009-97c1-1a4b3bd3a091) (provide repo link).
* Includes Jupyter Notebook for training and a FastAPI script for deployment.

This report provides an overview of the **Flower Classification Model** and its deployment strategy. Feel free to reach out for improvements! 🚀