Project Documentation: Facial Expression Recognition Using ResNet18

**Project Overview**

**Project Name:** face-exp-resnet

**Objective:** The primary goal of this project is to develop a robust facial expression recognition system using deep learning techniques, specifically leveraging the ResNet18 architecture. The system is trained to classify facial expressions into seven distinct categories: anger, disgust, fear, happiness, sadness, surprise, and neutrality.

**Dataset:** We utilized the "Facial Expression Recognition" dataset from Kaggle. The dataset consists of images categorized into the seven aforementioned facial expression classes. The dataset link is [here](https://www.kaggle.com/manishshah120/facial-expression-recog-image-ver-of-fercdataset).

## Project Structure

1. **Data Preparation:** The dataset is organized into training and test sets, with separate folders for each facial expression category. Various transformations are applied to the training images to enhance the robustness of the model, including random horizontal flipping, random rotations, and color jittering.
2. **Model Architecture:** A pre-trained ResNet18 model is used as the backbone for the facial expression recognition task. The final fully connected layer of ResNet18 is modified to output seven classes, corresponding to the seven facial expressions.
3. **Training Setup:** PyTorch's DataLoader is used to load and preprocess the images. Training and validation data loaders are created with appropriate batch sizes. The code is configured to utilize GPU if available, otherwise default to CPU.
4. **Training and Validation:** A custom training loop with one-cycle learning rate scheduling is implemented. The training phase involves calculating loss and updating model weights using backpropagation. After each epoch, the model's performance is evaluated on the validation set to monitor the validation loss and accuracy.
5. **Evaluation Metrics:** The accuracy of the model predictions is calculated by comparing them with the true labels. Cross-entropy loss is used to quantify the difference between predicted and true labels.
6. **Results Visualization:** The training and validation loss over epochs is plotted to observe the model's learning progress and convergence. The accuracy over epochs is plotted to monitor improvements in the model's performance. The learning rate schedule over epochs is visualized to understand the effect of learning rate on training dynamics.

## Key Insights

1. **Data Augmentation:** Applying data augmentation techniques such as random horizontal flip, random rotations, and color jittering helps in improving the model's generalization by introducing variability in the training data.
2. **Model Selection:** Using a pre-trained ResNet18 model leverages transfer learning, allowing us to benefit from the knowledge gained from large-scale datasets. This improves the performance and speeds up the training process.
3. **One-Cycle Learning Rate Policy:** Employing a one-cycle learning rate policy helps in faster convergence and achieving better performance. It adjusts the learning rate dynamically, starting with a small value, increasing to a maximum, and then decreasing again.
4. **Gradient Clipping:** Clipping gradients during training prevents the issue of exploding gradients, ensuring stable training.
5. **Performance Monitoring:** Regular evaluation on the validation set helps in monitoring overfitting. By comparing training and validation metrics, we can make informed decisions on stopping training or tuning hyperparameters.

## Conclusion

The facial expression recognition project using ResNet18 demonstrates the effectiveness of deep learning techniques in image classification tasks. By carefully designing the data augmentation, model architecture, and training procedure, we achieved satisfactory performance in classifying facial expressions. Future work may involve exploring more advanced architectures or fine-tuning hyperparameters to further improve accuracy and robustness.

4o