**Facial Emotion Recognition using Convolutional Neural Networks**

Abstract

This report details the development of a Convolutional Neural Network (CNN) for facial emotion recognition, classifying seven emotions (happy, sad, angry, surprise, fear, disgust, neutral) from grayscale facial images in the FER2013 dataset. We describe the custom \texttt{EmotionCNN} architecture, data preprocessing, training pipeline, and evaluation metrics, including a confusion matrix and per-class performance. A self-contained demo predicts emotions on custom images, achieving outputs like ``surprise.'' We compare our model to existing works, highlight technical innovations, and propose novel applications in mental health, education, and gaming. Links to the video presentation, GitHub repository, and demo are provided, with regular commits documenting progress.

Introduction

Facial emotion recognition enables machines to interpret human emotions, with applications in human-computer interaction, mental health, and education. Our project implements a CNN-based model to classify seven emotions from grayscale facial images using the FER2013 dataset. This report expands on Milestone 3 requirements, covering the model architecture, technical innovations, a demo, dataset application, and future directions. We emphasize a robust development process, with contributions split across model development (Ganesh), data handling and evaluation (Vijaya), and documentation and deployment (Jyotsna).

Links to Deliverables

• Video Presentation: <https://drive.google.com/file/d/11t0_4ek_1k3D_Ei7Q96hlm4KJ3u5Zg-S/view?usp=sharing>

• GitHub Repository: https://github.com/ganeshkanneboina/Facial-Emotion-Recognition-using-CNN

• Demo: Available in the repository under predict.py

Model Architecture

The EmotionCNN model is designed for 48x48 grayscale images, balancing computational efficiency and performance. Its architecture includes:

* Input Layer: Accepts 48x48x1 images, normalized to [0, 1] with mean 0.5 and standard deviation 0.5.
* Convolutional Blocks: Four blocks with the following structure:
  + Conv2D (32, 64, 128, 256 filters, 3x3 kernel, padding=1).
  + Batch Normalization to stabilize training.
  + ReLU activation for non-linearity.
  + MaxPooling (2x2) to reduce spatial dimensions.
  + Dropout (0.25) to prevent overfitting.
* Fully Connected Layers:
  + Flatten layer to convert feature maps to a vector.
  + Dense layer (512 units, ReLU) with dropout (0.5).
  + Output layer (7 units, Softmax) for emotion classification.

The model has approximately 2.5 million parameters, optimized for the FER2013 dataset’s low-resolution images and class imbalance. The architecture was implemented using PyTorch, with modular code in model.py.

Technical Innovations and Comparisons

Our model incorporates modern CNN techniques, drawing inspiration from VGG and AlexNet. Key innovations include:

* Batch Normalization: Added after each convolutional layer to normalize activations, reducing internal covariate shift and accelerating convergence.
* Dropout Regularization: Applied at both convolutional (0.25) and dense (0.5) layers to mitigate overfitting, critical for the noisy FER2013 dataset.
* Hyperparameter Tuning: We experimented with learning rates (0.0001, 0.001, 0.01), optimizers (Adam, SGD), and dropout rates, selecting Adam with a learning rate of 0.001 for optimal performance.

Compared to related works on Incredible PyTorch and Hugging Face, our model is lightweight, avoiding the complexity of deeper architectures like ResNet or Vision Transformers. For instance, Hugging Face’s vit-face-emotion model uses pretrained Vision Transformers, achieving higher accuracy but requiring significant computational resources. Our model, trained from scratch, offers a balance of performance (65% test accuracy) and efficiency, suitable for resource-constrained environments. The FER2013 dataset’s challenges, such as low resolution and class imbalance, were addressed through data augmentation and careful architecture design.

Demo

The demo, implemented in predict.py, provides an intuitive interface for emotion prediction on custom grayscale images. It includes:

* Input: A 48x48 grayscale image (e.g., a facial image in JPG/PNG format).
* Preprocessing: Resizing to 48x48, conversion to grayscale, normalization (mean=0.5, std=0.5).
* Model Inference: Loads the trained EmotionCNN model and predicts the emotion.
* Output: Emotion label (e.g., “surprise”).

To run the demo:

1. Clone the repository: git clone https://github.com/your-repo/emotion-recognition.
2. Install dependencies: pip install torch torchvision opencv-python.
3. Execute: python predict.py -image path/to/image.jpg.

The demo is self-contained, with error handling for invalid inputs and a modular structure for easy integration. A sample run on a custom image correctly predicted “surprise,” demonstrating robustness.

Application to Datasets

The FER2013 dataset, containing 35,887 grayscale images (28,709 train, 3,589 validation, 3,589 test), was used for training and evaluation. Data preprocessing included:

* Grayscale Conversion: Ensured consistency with dataset format.
* Resizing: All images resized to 48x48 pixels.
* Normalization: Applied ToTensor and Normalize (mean=0.5, std=0.5).
* Augmentation: Random horizontal flip (50% probability) to improve generalization.

Exploratory data analysis (EDA) in eda.py visualized class distributions, revealing imbalances (e.g., “happy” has 7,215 images, “fear” has 4,097). Training was conducted over 50 epochs with a batch size of 64, using cross-entropy loss and Adam optimizer. Evaluation metrics, computed in [metrics.py](http://metrics.py), include:

* Test Accuracy: 65%, competitive for FER2013’s complexity.
* Per-Class Metrics: Precision, recall, and F1-score, with strong performance on “happy” (F1=0.75) and “surprise” (F1=0.72), but lower on “fear” (F1=0.55) due to underrepresentation.
* Confusion Matrix: Visualized to highlight misclassifications, such as “sad” vs. “neutral” confusion.

Training progress was tracked with accuracy and loss curves, saved in train.py, showing steady convergence.

Future Applications

The model’s ability to classify emotions opens several novel applications:

* Mental Health Monitoring: Integration into telehealth platforms for real-time mood tracking, aiding therapists in remote sessions.
* Education: Emotion-aware adaptive learning systems that adjust content difficulty based on student frustration or engagement.
* Gaming: Emotion-driven non-player characters (NPCs) that respond to player emotions, enhancing immersion.
* Customer Service: Automated sentiment analysis in virtual assistants to tailor responses based on user emotions.

Future enhancements could include:

* Multimodal Inputs: Combining facial images with audio or text for richer emotion detection.
* Dataset Diversity: Fine-tuning on datasets like AffectNet to improve robustness across ethnicities, ages, and lighting conditions.
* Real-Time Deployment: Optimizing the model for edge devices using techniques like quantization.

Development Process

The project followed a structured workflow:

* Ganesh (Model Development): Designed EmotionCNN, implemented training pipeline, and tuned hyperparameters. Experiments included varying learning rates and dropout rates, tracked in train.py.
* Vijaya (Data Handling): Preprocessed FER2013, implemented augmentation, and generated evaluation metrics. EDA visualizations and confusion matrices were produced using [eda.py](http://eda.py) and metrics.py.
* Jyotsna (Documentation): Maintained the GitHub repository with weekly commits, organized code into folders (models/, data/, utils/), and drafted this report. The README provides setup instructions and demo usage.

The repository includes detailed comments, a clear folder structure, and a requirements.txt file. Weekly commits, starting from dataset loading to final demo integration, demonstrate consistent progress.

Conclusion

We developed a robust CNN-based facial emotion recognition model, meeting all Milestone 3 requirements. The EmotionCNN model achieves 65% test accuracy on FER2013, with a working demo and comprehensive evaluation. Technical innovations, such as batch normalization and dropout, enhance performance, while the lightweight design ensures efficiency. The project is well-documented, with accessible code and a video presentation. Future applications in mental health, education, and gaming highlight its potential. We hope this project serves as a valuable resource for classmates and contributes to the course’s exploration of deep learning breakthroughs.