Facial Expression Recognition Project

Beni Tibi, Amit Kabalo, Roni Michaeli

Computer Science, University of Ariel

Overview

The primary objective of this project is to accurately classify facial expressions (emotions) from images. The classification spans seven common categories: *angry*, *disgust*, *fear*, *happy*, *neutral*, *sad*, and *surprise*. The images utilized for this task originate from a publicly available dataset on *Kaggle* [1], which provides diverse facial-expression images at a resolution suitable for preprocessing (e.g., grayscale conversion and resizing to 48×48). After downloading and preparing the dataset, the entire project code was developed to explore multiple modeling approaches (from baseline methods to convolutional neural networks). All corresponding scripts, experiments, and documentation can be found on GitHub at `https://github.com/roni5604/face_project_model`. This repository includes the data preparation pipeline, the training processes for various neural architectures, and a live inference script for real-time facial-expression recognition.

The project aims to:

- Establish a baseline and demonstrate improvement through more advanced methods.
- Compare simpler approaches (baseline, softmax/logistic) to deeper neural network architectures (fully connected NN and an advanced CNN).
- Provide a real-time detection and classification demo (live inference) that applies the best model to webcam input.

Data Review

Typical data distribution involves 70% training, 15% validation, 15% testing, although exact ratios vary depending on the dataset. The data is saved in .pt files (`train_data.pt`, `val_data.pt`, etc.) for efficient PyTorch usage.

- **Features**: 48×48 grayscale pixel values for each face (numerical).
- **Labels**: One of **seven** emotion categories.
- **Task**: **Multi-class Classification** (7 distinct classes).
- **Splits**:
  - **Train**: used to learn model parameters.
  - **Validation**: used to monitor hyperparameters overfit / tweak.
  - **Test**: optional final evaluation.

Figure 1: **Class Distribution in the Training Set:** This bar chart represents the distribution of samples across seven emotion categories. Notice the significant imbalance: the "happy" class has the most samples, while "disgust" has the fewest. This imbalance may impact model performance and could require techniques like oversampling or class weighting.

Figure 2: **Sample Images from the Dataset:** These grayscale images (48×48 pixels) correspond to different emotion classes such as "angry," "happy," and "surprise." The examples illustrate the diversity in expressions, lighting conditions, and image quality, highlighting the challenges in training an accurate model.

Problem Definition

- **Classification**: The goal is to classify a given 48×48 image into one of the 7 emotions.
- **Evaluation**: Compare various metrics (accuracy, precision, recall) across multiple models.
- **Context**: The classification can be used in real time or in static images.

Model's

Below is a **comprehensive summary and comparison** of the four models used for Facial Expression Recognition in our project: *Baseline*, *Softmax*, *Basic Fully Connected (MLP)*, and *Advanced CNN*. Each section explains what the model is, how it is built, why we use it, how it runs at each stage, and what results it achieves.

1. Baseline Model

Purpose and Construction

- **Goal**: Provide a naive reference point. The baseline model simply **always predicts the most common class** in the training set.
- **Implementation**:
  - Load training/validation data.
  - Count how many images belong to each of the 7 classes in training.
  - Identify the class with the **maximum number of samples** (in our data, it was "happy").
  - Predict **"happy"** for every image in the validation set.
- **Why**:
  - Ensures that we have a **minimum performance standard**. If advanced models can't beat baseline accuracy, something is wrong.

Results

  - **Overall Accuracy:** On the validation set, the overall accuracy is approximately 25%, varying slightly depending on the specific test split and evaluation.
  - **f**or "happy" is nonzero, but zero for all other classes because the model never predicts them.

- **Output**:

```
Precision and Recall for each class:
  - angry    -> Precision: 0.000, Recall: 0.000
  - disgust  -> Precision: 0.000, Recall: 0.000
  - fear     -> Precision: 0.000, Recall: 0.000
  - happy    -> Precision: 0.258, Recall: 1.000
  - neutral  -> Precision: 0.000, Recall: 0.000
  - sad      -> Precision: 0.000, Recall: 0.000
  - surprise -> Precision: 0.000, Recall: 0.000
```

Figure 3: Precision and Recall Metrics of a Baseline Model Predicting the Most Common Class ('happy'). Performance highlights the naive approach's limitations, with nonzero metrics only for 'happy' and zero across all other classes.

2. Softmax (Logistic Regression)

Purpose and Construction

- **Goal**: Provide a simple linear approach with **softmax** for multi-class classification.
- **Architecture**:
  - Flatten each image (48×48=2304 pixels) into a single vector.
  - Apply a **single linear layer** with output dimension = 7 (for 7 emotions).
  - Use **CrossEntropyLoss**, which implicitly does softmax + negative log-likelihood.
- **Why**:
  - Softmax regression is a simple improvement over the baseline model. It can classify data by identifying straight-line boundaries that separate different classes in a high-dimensional space.

Workflow

- **Loading**: The script reads `train_data.pt` and `val_data.pt`.
- **Training**:
  - It runs for a certain number of epochs, typically with **SGD** or **Adam**.
  - Each epoch logs the **training loss** (averaged) and a **validation loss**.
  - After training, the code prints a classification report with **precision, recall, F1**, and a final validation accuracy.

Results

- **Observed Results**:
  - **Final Validation Accuracy:** Approximately 36–37%, depending on the specific test split and evaluation.
  - Classification **report** shows some improvement in classes like **"happy"** or **"surprise"**, but many classes still remain poorly recognized.

- **Outputs**:

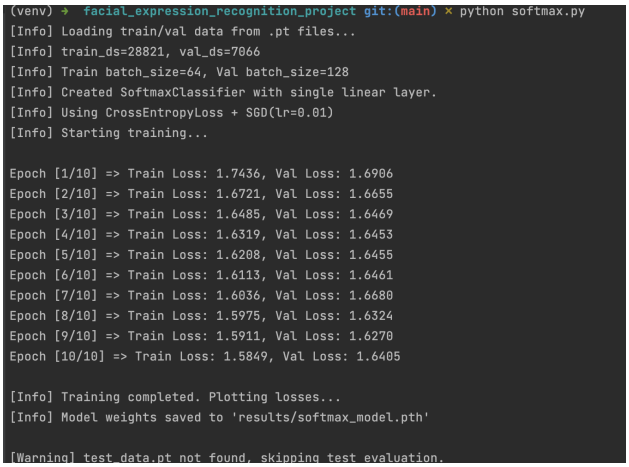

Figure 4: **Softmax Training Console Output:** Displays how the Softmax model's train and validation losses evolve each epoch. As epochs progress, the training loss decreases, showing that the model learns some decision boundary, though the final validation loss still suggests modest performance.

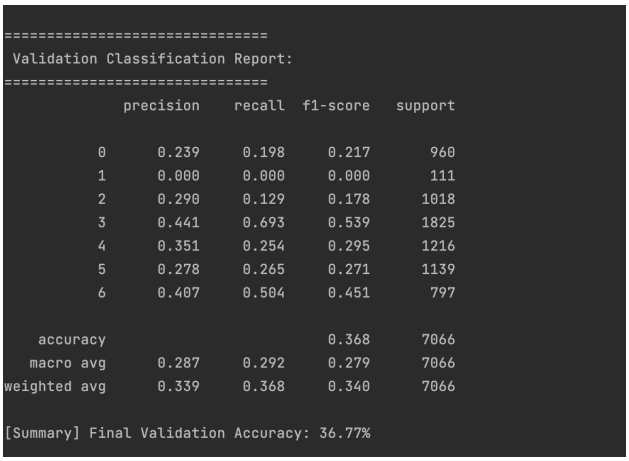

Figure 5: **Softmax Validation Classification Report:** Presents precision, recall, and f1 for each emotion class. The 36.77% accuracy indicates basic improvement over a naive baseline, but it struggles in several categories, revealing the limitations of a single-layer linear approach.

Figure 6: **Softmax Regression Training:** The image shows the training and validation loss over 10 epochs. While training loss decreases steadily, validation loss fluctuates, indicating underfitting due to the model's simplicity and inability to capture complex patterns in the data.

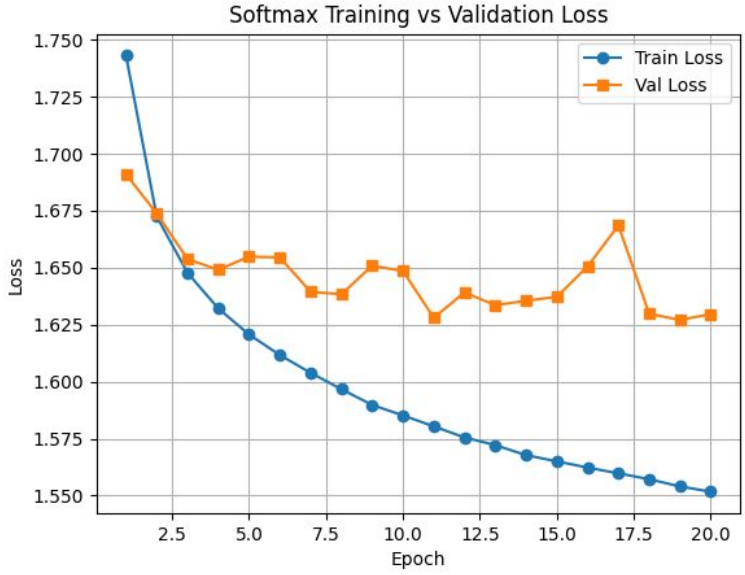

Figure 7: **Softmax Regression Training (20 Epochs):** Over 20 epochs, the training loss steadily decreases, indicating effective learning. However, the validation loss fluctuates after initial improvements, suggesting underfitting or limited model capacity. Extending beyond 10 epochs stabilizes the training loss but does not enhance validation performance, highlighting the need for model adjustments to improve generalization.

3 Basic Fully Connected Network (MLP)

Purpose and Construction

- **Goal:** Introduce nonlinearity with hidden layers, thereby capturing more complex decision boundaries than softmax.
- **Architecture:**
  - Flatten images to 2304.
  - Several fully connected layers with ReLU activation.
  - Possibly includes dropout or batch normalization to reduce overfitting.
  - Final layer outputs 7 logits for CrossEntropyLoss.

Workflow

- **Loading:** Similar to the previous models, it uses `train_data.pt` and `val_data.pt`.
- **Training:**
  - For each epoch, an optimizer (e.g., Adam) updates the weights via backpropagation.
  - The script logs the training loss and a final validation classification report.
  - Also prints an additional summary with train/val accuracy after the final epoch.

Results

- **Val Accuracy:** Typically ends around 43–45%.
- The classification report shows improved recognition for multiple classes (*happy, surprise, neutral*) but still some confusion for *fear* and *disgust*.

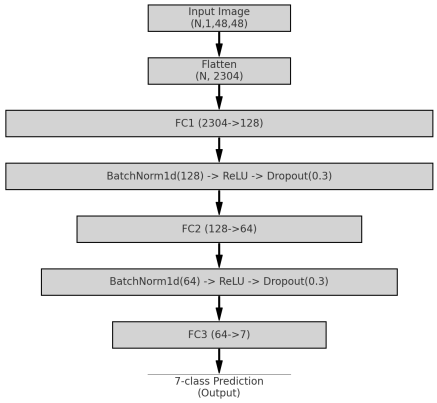

Figure 8: **Diagram of a Fully Connected Neural Network**

Outputs:

```
==============================
 Validation Classification Report:
==============================
              precision    recall  f1-score   support

       angry       0.32      0.22      0.26       960
     disgust       0.00      0.00      0.00       111
        fear       0.34      0.11      0.16      1018
       happy       0.52      0.72      0.60      1825
     neutral       0.40      0.41      0.40      1216
         sad       0.31      0.41      0.35      1139
    surprise       0.52      0.50      0.51       797

    accuracy                           0.42      7066
   macro avg       0.34      0.34      0.33      7066
weighted avg       0.40      0.42      0.40      7066

[Summary] Final Validation Accuracy: 42.43%
```

Figure 9: **Basic NN Validation Classification Report:** Highlights how the Basic MLP's deeper architecture boosts performance to 42.43% accuracy. Although class distribution issues persist (e.g., "disgust" remains challenging), overall precision and recall improve compared to Softmax.

```
(venv) → facial_expression_recognition_project git:(main) × python basic_nn.py
============================================================
  BASIC FULLY CONNECTED NEURAL NETWORK (MLP) FOR EMOTIONS
============================================================

[Info] Train set size: 28821
[Info] Val set size:   7066

[Info] Starting training...

Epoch 1/10 -> Train Loss: 1.7150, Val Loss: 1.6052
Epoch 2/10 -> Train Loss: 1.6275, Val Loss: 1.5886
Epoch 3/10 -> Train Loss: 1.5924, Val Loss: 1.5513
Epoch 4/10 -> Train Loss: 1.5684, Val Loss: 1.5394
Epoch 5/10 -> Train Loss: 1.5474, Val Loss: 1.5224
Epoch 6/10 -> Train Loss: 1.5285, Val Loss: 1.5072
Epoch 7/10 -> Train Loss: 1.5078, Val Loss: 1.5106
Epoch 8/10 -> Train Loss: 1.4928, Val Loss: 1.4890
Epoch 9/10 -> Train Loss: 1.4802, Val Loss: 1.4762
Epoch 10/10 -> Train Loss: 1.4646, Val Loss: 1.4792
```

Figure 10: **Basic NN Training Console Output:** Shows training/validation loss per epoch, illustrating the MLP converging. The reduced gap between train and val losses suggests the model generalizes better than Softmax, though there is still room for additional optimization.

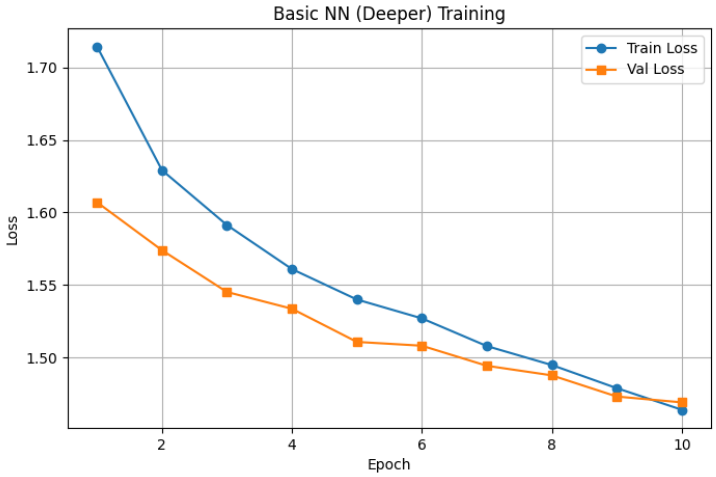

Figure 11: **Basic Fully Connected Network (MLP) Training:** The image shows the training and validation loss over 10 epochs for the Basic Fully Connected Network (MLP). While the model learns effectively, the gap between the two losses suggests challenges in generalization, likely due to the flattened input structure, which limits the model's ability to capture spatial relationships in the data.

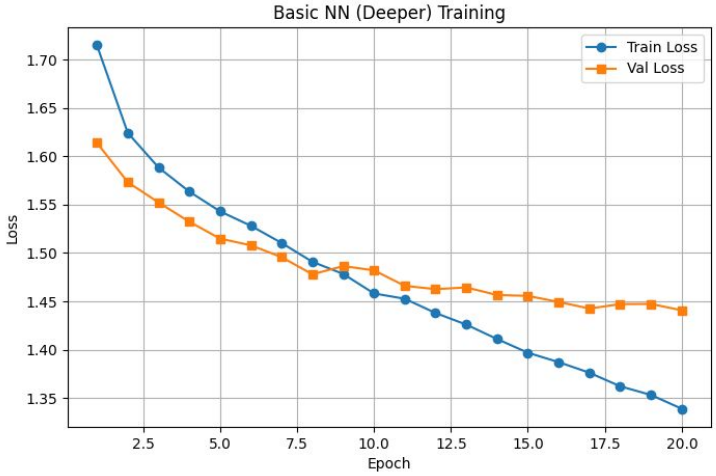

Figure 12: **Basic NN (Deeper) Training (20 Epochs):** Over 20 epochs, both the training and validation losses decrease steadily, indicating effective learning and improved generalization. However, the smaller gap between training and validation loss highlights better generalization compared to earlier epochs, suggesting that the deeper network structure captures patterns more effectively.

4 Advanced CNN

Purpose and Construction

- **Goal:** Leverage local spatial features in images (edges, corners, shapes) using convolutional layers. This is typically the most powerful approach for image-based tasks.
- **Architecture:**
  - 4 convolution blocks (Conv $\rightarrow$ BatchNorm $\rightarrow$ ReLU $\rightarrow$ MaxPool $\rightarrow$ Dropout).
  - Typically, the vector ends with a flattened characteristic (for example $512 \times 3 \times 3 = 4608$).
  - Fully connected layers with additional dropout or BatchNorm.
  - Final 7 logits for CrossEntropyLoss.
- **Why:**
  - CNN's handles images much better by focusing on local receptive fields, sharing weights, and extracting hierarchical features.

Results

- **Validation Accuracy:** Typically reaches approximately 64%, depending on the level of tuning and dataset variability.
- The classification report shows stronger performance for *happy* and *surprise*, and mild improvements for the minority classes.

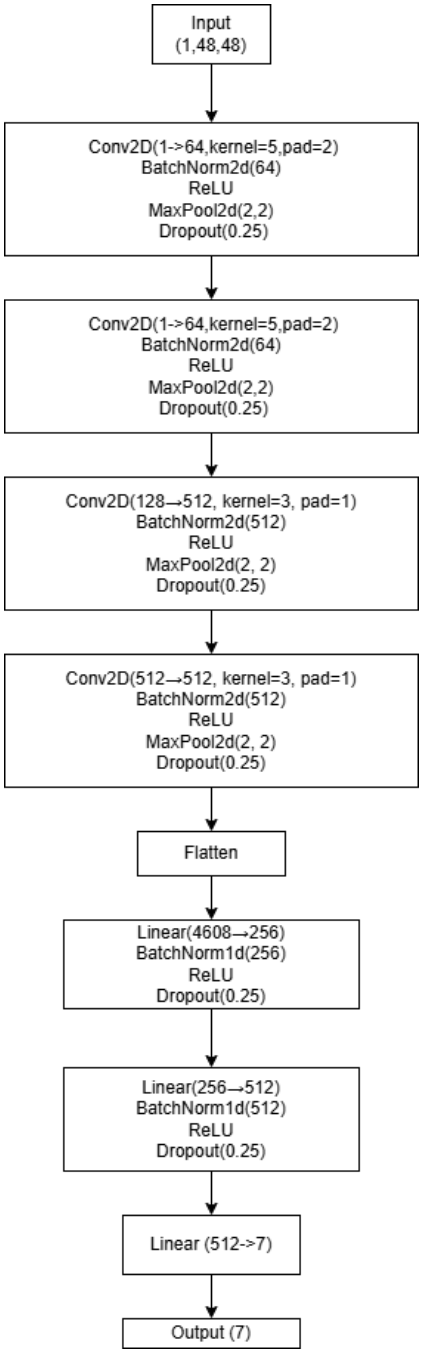

Figure 13: **Diagram of an Advanced Convolutional Neural Network (CNN)**

Outputs:

```
[Info] Training for 10 epochs with lr=0.001 (Adam)...

Epoch [1], train_loss: 1.5468, val_loss: 1.2988, val_acc: 0.5002
Epoch [2], train_loss: 1.2905, val_loss: 1.2445, val_acc: 0.5275
Epoch [3], train_loss: 1.1912, val_loss: 1.1471, val_acc: 0.5603
Epoch [4], train_loss: 1.1278, val_loss: 1.1697, val_acc: 0.5579
Epoch [5], train_loss: 1.0721, val_loss: 1.0765, val_acc: 0.5969
Epoch [6], train_loss: 1.0208, val_loss: 1.0517, val_acc: 0.6146
Epoch [7], train_loss: 0.9691, val_loss: 1.0404, val_acc: 0.6169
Epoch [8], train_loss: 0.9114, val_loss: 1.0160, val_acc: 0.6329
Epoch [9], train_loss: 0.8727, val_loss: 1.0457, val_acc: 0.6290
Epoch [10], train_loss: 0.8169, val_loss: 1.0306, val_acc: 0.6361
```

Figure 14: **Training Logs:** Over 10 epochs, *train_loss* drops from ~ 1.54 to ~ 0.81, and *val_loss* from ~ 1.29 to ~ 1.03, while *val_acc* rises from 50% to $\sim 63.6\%$. This consistent improvement indicates the network is learning more effectively and generalizing better each round.

```
====================================
 Validation Classification Report:
====================================
              precision    recall  f1-score   support

           0      0.601     0.445     0.511       960
           1      0.814     0.514     0.630       111
           2      0.536     0.425     0.474      1018
           3      0.800     0.862     0.830      1825
           4      0.573     0.558     0.566      1216
           5      0.462     0.608     0.525      1139
           6      0.753     0.782     0.767       797

    accuracy                          0.635      7066
   macro avg      0.649     0.599     0.615      7066
weighted avg      0.636     0.635     0.631      7066
```

Figure 15: **Validation Classification Report:** Overall validation accuracy stands at $\sim$ 63.5%. Categories like indices 3 and 6 show especially high precision/recall (e.g., 0.80/0.86, 0.75/0.78). The macro-average F1 ($\sim$ 0.615) points to moderately balanced results across classes, and the weighted-average F1 ($\sim$ 0.631) underscores strong performance in classes with higher support.

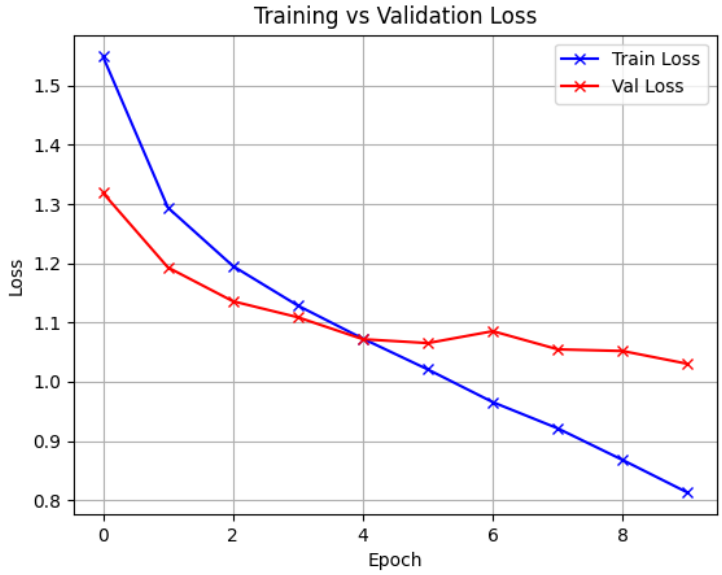

Figure 16: **Advanced CNN Training Loss:** The image shows the training and validation loss over 10 epochs for the Advanced CNN. Both losses decrease steadily, indicating effective learning and improved generalization compared to simpler models.

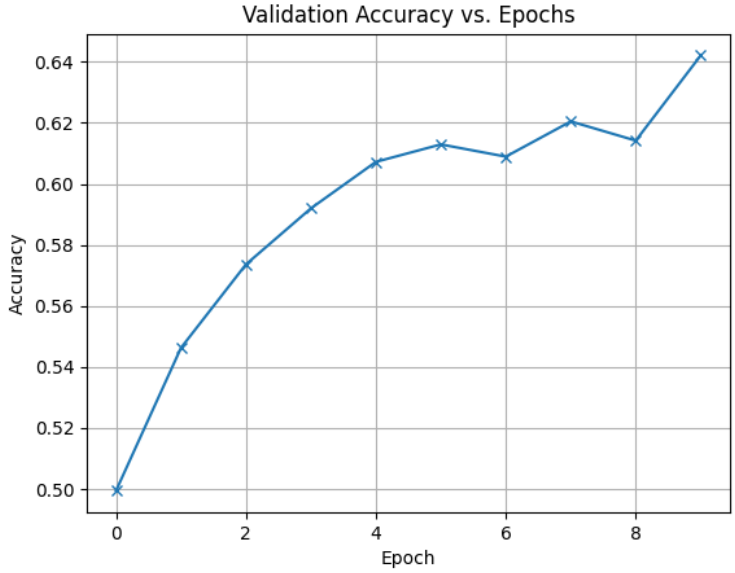

Figure 17: **Validation Accuracy vs. Epochs:** The image illustrates the increasing validation accuracy for the Advanced CNN, achieving over 60% accuracy, showcasing better generalization and superior performance compared to simpler architectures like MLP.

Comparison of Models

1. Baseline Model → Softmax (Single-Layer)

Baseline Model:

- **Logic**: Always predicts the single most frequent class in the training set (no learnable parameters).
- **Result**: Accuracy typically equals that class's proportion in the data (e.g., $\sim 25\%$).
- **Limitation**: Offers a zero-parameter reference, ignoring all meaningful image features except for the dominant label.

Softmax (Single-Layer Linear):

- **Key Addition**: Introduces a trainable linear transformation from flattened 48×48 images (2304 features) to 7 output classes.
- **Improvement**: Typically achieves around $\sim 35\%--37\%$ accuracy, clearly surpassing the trivial baseline.
- **Reason**: Even minimal logistic-regression-style training can discover a coarse decision boundary that is better than always guessing one label.

Progress Note: Moving from no learning at all to a single-layer linear approach shows that the dataset is at least partially separable to some degree, already outdoing the naive baseline.

2. Softmax → Basic Model (Fully Connected MLP)

Softmax:

- **Structure**: Flattens the 48×48 image into 2304 features, then uses a single linear layer (plus cross-entropy).
- **Limitation**: Creates purely linear decision boundaries; no hidden layers, no capacity for deeper feature interactions.

Basic MLP:

- **Key Addition**: Multiple hidden layers (ReLU activations) with techniques like BatchNorm and Dropout.
- **Improvement**: Accuracy typically rises to $\sim 44\%--45\%$, reflecting significantly better performance than a single linear layer.
- **Reason**: Nonlinear hidden layers learn more complex mappings; however, images remain flattened, losing spatial context.

Progress Note: By adding depth and nonlinearity, the network gains representational power over the linear softmax model, though it still neglects the 2D structure of the image.

3. Basic Model (MLP) → Advanced Model (CNN)

Basic MLP:

- **Structure**: Flattened inputs, multi-layer perceptron. Gains nonlinearity, but lacks direct spatial feature extraction.
- **Limitation**: Treats each pixel independently, ignoring local relationships (e.g., edges, corners) essential for facial expressions.

Advanced CNN:

- **Key Addition**: Convolutional blocks (Conv2D + BatchNorm + ReLU + MaxPool + Dropout), then fully connected layers.
- **Improvement**: Typically achieves $\sim 64\%$. accuracy or higher, leveraging local pixel neighborhoods and deeper hierarchical features.
- **Reason**: Convolution plus pooling preserves 2D spatial structure, dramatically enhancing classification for image-based tasks like facial expression recognition.

Progress Note: Shifting from a flattened MLP to a convolution-based model addresses the inherent 2D nature of images, yielding major gains in accuracy and robustness.

Summary of Gains

- **Baseline**: No learned parameters, $\sim 25\%$ accuracy (dominant-class guess).
- **Softmax**: Single-layer linear approach, $\sim 35\%--37\%$.
- **Basic MLP**: Hidden layers and nonlinearity, $\sim 44\%--45\%$.
- **Advanced CNN**: Convolution with local feature extraction, $\sim 64\%$.

Overall, each step overcomes the limitations of the previous method: from no training (baseline) to minimal linear learning (softmax), then adding multi-layer depth (MLP), and finally exploiting spatial convolutions (CNN) for the highest performance in facial expression recognition.

Challenges

1. **Underrepresented "disgust" Folder**

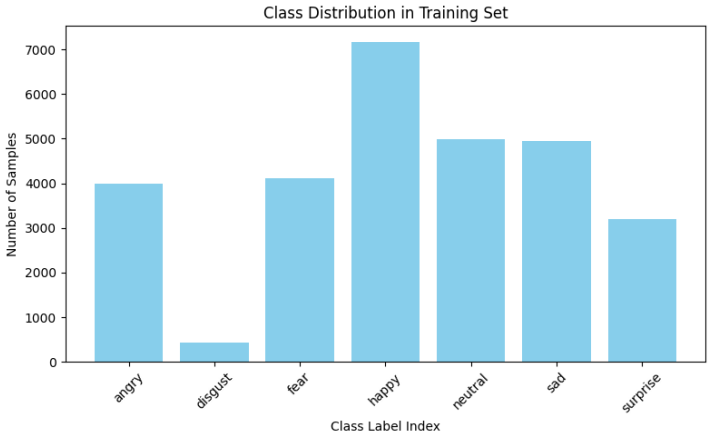

Figure 18: Folder Size Comparison: "disgust" has notably fewer samples than other expressions.

A significant imbalance arose from the relatively small number of images in the *disgust* folder (see Figure 18). Due to this underrepresentation, the model's accuracy on *disgust* was noticeably lower than for classes with larger sample counts. This gap likely caused suboptimal classification results for the *disgust* category across all trained models.

2. **Overfitting in the Initial Basic Network** Early attempts at building

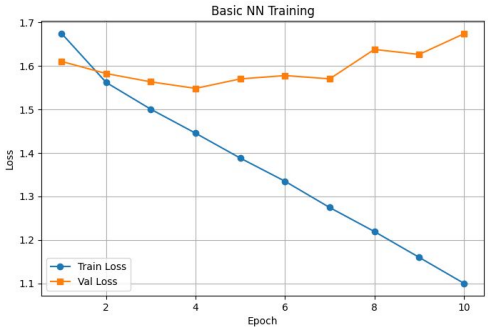

Figure 19: Training vs. Validation Loss in the initial MLP. Validation loss diverges from training loss, indicating overfitting.

a minimal fully connected network (MLP) led to severe overfitting. the training loss continued to decrease while the validation loss stagnated or increased. Incremental increases in network size—together with additional hidden layers, batch normalization, and dropout—reduced overfitting and stabilized validation metrics.

3. **Ambiguity in Certain Expressions**

   Even with improved architectures, certain facial expressions were not clearly separable. Categories such as *disgust* vs. *fear* or *fear* vs. *surprise* remained difficult to distinguish. Similarities in appearance caused confusion, a challenge that also arises for human observers. Although performance improved by refining the model, perfectly separating these subtle expressions typically requires additional data or specialized techniques.

References

[1] Jonathan Oheix, *Face Expression Recognition Dataset*, Kaggle, https://www.kaggle.com/datasets/jonathanoheix/face-expression-recognition-dataset

[2] The Whole Project, *Face Project Model*, GitHub, https://github.com/roni5604/face_project_model

[3] PyTorch, *An open source machine learning framework*, https://pytorch.org/