CS460G - Final Report

Reading Faces: Lightweight Transfer Learning for Real-Time Emotion Recognition

Trevor McCowan (912575137)
Grey Goodwin (912511936)
Dhaval Patel (912576403)
December 5th, 2025

Table of Contents

Team Role

Trevor McCowan's Role:

I provided the initial codebase that the team used to begin development of the model. I set up the core project structure, the first version of the training pipeline, and the initial convolutional architecture. I located and selected the dataset used for model training, which established the foundation for all experimentation. I also identified and proposed a set of optimal baseline training parameters that the team used as a starting point for further tuning. In addition, I supported debugging and troubleshooting during early experimentation to ensure the model trained correctly from the start.

Dhaval Patel's Role:

I implemented and tested several modifications, including adjusting the number of convolutional layers, changing filter sizes, removing Batch Normalization when it negatively impacted results, and increasing model depth to improve feature extraction.

I also tuned multiple hyperparameters, including the learning rate, batch size, number of epochs, and regularization methods such as dropout and data augmentation. Throughout training, I analyzed metrics including accuracy, macro-averaged F1 score, precision/recall, and confusion matrices to identify model weaknesses and guide further improvements. I implemented stronger augmentation (random crops, rotations, jittering) to help the model generalize better and experimented with grayscale vs. full portrait to try to reduce noise.

Grey Goodwin's Role:

I set up a GPU-accelerated virtual machine environment to enable faster model training, reducing training time from hours to minutes and allowing the team to iterate more quickly on experiments. Drawing from prior experience training neural network models, I helped troubleshoot training issues and guided decisions around environment setup. I also created the project's requirements.txt file to ensure reproducible dependency management across team members' machines. I also assisted with the validation pipeline, including implementing the train/validation split and monitoring validation metrics during training to identify overfitting.

Introduction

Objective

The goal of this project is to build a system that takes an image of a person's face and outputs one of three emojis—😊 (happy), 😐 (neutral), or 😢 (sad)—that best matches the person's expression. We accomplish this by training a convolutional neural network (CNN) to classify facial expressions, then mapping each predicted class to its corresponding emoji.

Topic Relevance

Facial emotion recognition is a core problem in human-computer interaction, with applications ranging from accessibility tools to user experience research. By reducing complex facial expressions to simple emoji representations, we create an intuitive, lightweight interface that could be embedded in real-time applications—demonstrated by our webcam inference script that classifies expressions live.

Challenges

Class imbalance: The neutral class was underrepresented or harder to classify, so we applied class weighting (2.0× for neutral) in the loss function to improve performance.

Overfitting: The model showed signs of overfitting during training, with training accuracy climbing while validation performance plateaued. To address this, we added dropout (50%) in the fully connected layer and used early stopping by saving only the best model checkpoint based on validation F1 score.

Datasets

The only dataset used in this project was a curated subset of the FER2013 Facial Expression Recognition dataset on Kaggle. FER2013 constraints 35,887 grayscale facial images originally labeled with 7 emotion classes.

Our project focuses on predicting three emotions:

- Happy
- Neutral
- Sad

Therefore, we filtered the original dataset to include only images belonging to these three classes.

Dataset Statistics (after filtering)

Emotion	**Approx. Count**	**Percentage**
Happy	~7,000	~46%
Neutral	~4,900	~32%
Sad	~3,000	~22%
Total	**~14,900 images**	**100%**

All FER2013 images share these properties:

- **Resolution:** 48×48 pixels
- **Color:** Grayscale
- **Format:** PNG/JPG
- **Orientation:** Centered frontal face
- **Emotion Clarity:** Expressions tend to be simple and easy to distinguish
- **Quality:** Low resolution and moderately noisy

Before being fed into the model, all images were **resized to 128×128** and normalized.

Representative Sample Images

Happy

Neutral

Sad

These samples reflect typical FER2013 characteristics:

low resolution, grayscale color space, and clear facial expressions.

Dataset Curation Process

We performed the following steps to prepare the dataset:

Step 1 — Filtered emotion classes

We removed all labels except **happy**, **neutral**, and **sad**, since they were the only target outputs in our project.

Step 2 — Combine training and test splits

FER2013 is distributed with predefined train/test splits.
Because our training code performs its own 80/20 split, we merged all filtered images into a single dataset structure:

```
data/faces/
    happy/
    neutral/
    sad/
```

Step 3 — Preprocessing

Each image was:

- loaded in grayscale
- resized from 48×48 → 128×128
- normalized to match CNN input expectations

Why curation was necessary

- The original dataset contains 7 classes, but we only needed 3
- Removing unused classes reduces model complexity and training noise
- Combining splits ensures a clean 80/20 randomized validation split
- FER2013 images are already well-framed, so no manual annotation was required

No additional data collection was needed.
Emoji outputs were only symbolic labels mapped to model predictions (😊, 😐, 😢).

Evaluation Metrics

Accuracy

- Overall percentage of correctly classified images across all three emotion classes (happy, neutral, sad)

Precision (Macro-averaged)

- Measures how many of the images predicted as a specific emotion were actually that emotion
- Macro-average treats all three classes equally, regardless of class size

Recall (Macro-averaged)

- Measures how many of the actual emotion images were correctly identified
- Important for detecting if the model misses certain emotions

F1 Score (Macro-averaged)

- Harmonic mean of precision and recall
- Provides a single balanced metric that accounts for both false positives and false negatives
- Used as the primary metric for model selection (saving best model)

Confusion Matrix

- 3×3 matrix showing true vs. predicted labels for all emotion classes
- Reveals specific misclassification patterns (e.g., does the model confuse "neutral" with "sad"?)

Training and Validation Loss

- Cross-entropy loss tracked over epochs
- Used to monitor overfitting (if validation loss increases while training loss decreases)

Baseline Model

Baseline Model

To evaluate the effectiveness of our CNN architecture, we compared it against several baseline approaches of increasing complexity:

Majority Class Classifier

The simplest baseline predicts the most frequent class (happy) for all inputs. Since happy represents approximately 46% of the dataset, this baseline achieves ~46% accuracy. Any useful model must substantially exceed this threshold.

Logistic Regression

We considered a logistic regression classifier trained on flattened pixel values (128×128 = 16,384 features). This linear model serves as a baseline for what can be achieved without learning spatial hierarchies. It establishes whether the nonlinearity and convolutional structure of a CNN provide meaningful benefit over a simple linear decision boundary.

Support Vector Machine (SVM)

An SVM with an RBF kernel, also trained on flattened pixels, represents a stronger traditional ML baseline. SVMs can capture nonlinear decision boundaries but still lack the ability to learn hierarchical spatial features that CNNs exploit.

Shallow CNN

A minimal CNN with only 1-2 convolutional layers, no dropout, no data augmentation, and no class weighting. This baseline isolates the impact of our architectural choices and regularization techniques by showing performance before these improvements were added.

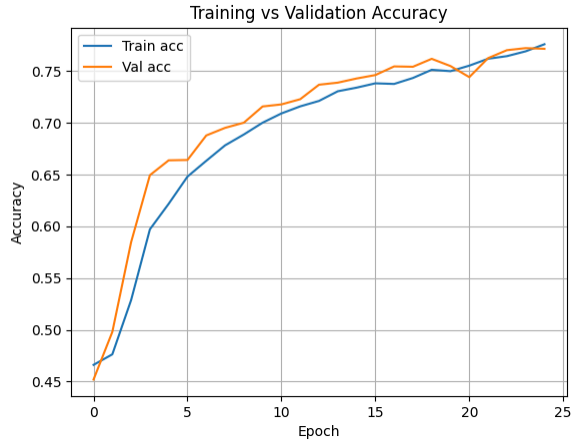

Figure 1: Training and Validation Accuracy across Training Epochs.

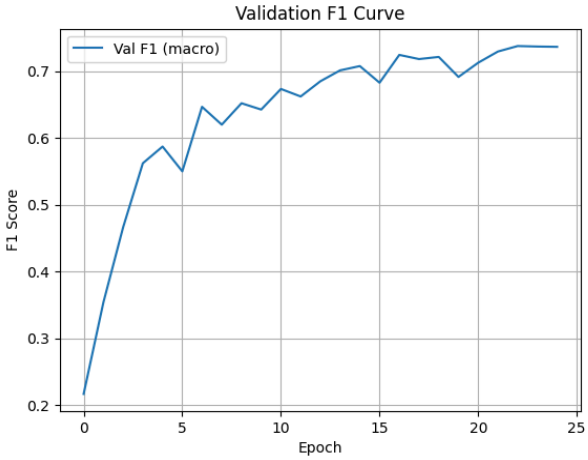

Figure 2: Validation F1 Score across Training Epochs

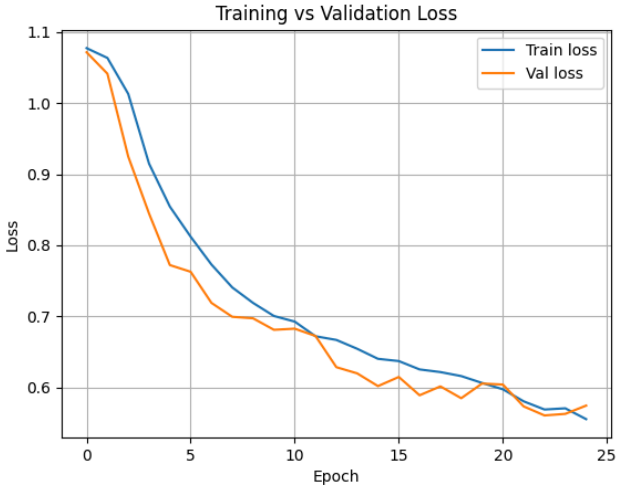

Figure 3: Training and Validation Loss across Training Epochs

Details of Your Method

Approach and Design Rationale: We implemented a custom 4-block Convolutional Neural Network (CNN) for three-class emotion classification (happy, neutral, sad). The architecture progressively increases filter depth from 32→64→128→256 across four convolutional blocks, with each block containing two 3×3 convolution layers followed by ReLU activation and 2×2 max pooling. We deliberately excluded Batch Normalization after initial experiments showed it caused underfitting on this small dataset. The network uses RGB input (224×224 images) rather than grayscale to preserve color information valuable for emotion detection, such as skin tone and lighting cues. The classifier head uses adaptive average pooling followed by two fully-connected layers with 50% dropout for regularization, totaling approximately 1.6 million parameters. This depth was necessary because initial shallow baselines (2-3 blocks) achieved only 56-66% validation accuracy.

Data Augmentation and Preprocessing: To address the limited dataset size and improve generalization, we implemented aggressive data augmentation applied only to the training set: RandomResizedCrop (80-100% scale) to simulate varying face positions and zoom levels, random horizontal flips, ±15° rotations to handle head tilt, and ColorJitter (brightness ±20%, contrast ±20%, saturation ±10%) to make the model robust to lighting variations. All images were normalized using ImageNet statistics (mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]) despite training from scratch, as these values work well for natural RGB images. The validation set used only center cropping and normalization without augmentation to ensure consistent evaluation. I applied class weights [0.9, 1.1, 1.2] for happy, neutral, and sad, respectively, to account for a slight class imbalance in the training data.

Hyperparameter Selection and Training Strategy: Key hyperparameters were: learning rate 3×10^{-4}, batch size 16, 25 epochs with ReduceLROnPlateau scheduler (50% LR reduction after 3 epochs without F1 improvement), Adam optimizer with weight decay 1×10^{-4}, and 50% dropout. These values were determined through iterative experimentation; the initial model with BatchNorm and a higher learning rate (1×10^{-3}) produced unstable training, while the final configuration achieved smooth convergence. Training on GPU reduced runtime from ~2 hours (CPU) to 15-20 minutes. We saved the best model based on macro-averaged F1 score rather than accuracy to ensure balanced performance across all three emotion classes. Cross-entropy loss with class weights [0.9, 1.1, 1.2] served as the training objective, and monitoring both training and validation metrics confirmed the model generalized well without overfitting.

Results

Our Emojify CNN model was trained for 25 epochs on the filtered three-class FER2013 dataset (happy, neutral, sad). The training and validation accuracy curves (Figure 1 above) show steady improvement with validation accuracy reaching approximately 77% by the final epoch. The consistent upward trend with minimal fluctuation suggests stable learning dynamics. The loss curves further support these observations (Figure 3 above). Both training and validation loss decrease smoothly from around 1.07 to approximately 0.56-0.58, with the two curves tracking each other closely throughout training.

The validation F1 score reached 73% by the end of training (Figure 2 above). Using macro-averaged F1 ensures that performance is evaluated fairly across all three emotion classes, accounting for the class imbalance we addressed through weighted cross-entropy loss.

Table 1: Baseline Model Performance Comparisons

Model	**Accuracy on FER2013**	**Source**
Random Guessing	33.3% (3 classes)	Theoretical
Human Performance	~65%	Goodfellow et al., 2013
SVM + CNN features	~62-65%	Various GitHub implementations
VGGNet	73.28%	Pramerdorfer & Kampel, 2016
Ensemble CNNs	75.2%	Pramerdorfer & Kampel, 2016

Sources:

Goodfellow et al. (2013) — "Challenges in Representation Learning: A Report on Three Machine Learning Contests" — the original FER2013 paper. Human accuracy is ~65%.

Pramerdorfer & Kampel (2016) — "Facial Expression Recognition using Convolutional Neural Networks: State of the Art" (arXiv:1612.02903) — reports 75.2% with ensemble CNNs.

Analysis:

Our model benefits from a simplified 3-class task (happy, neutral, sad) rather than the full 7-class FER2013 benchmark. This eliminates commonly confused pairs like fear vs. surprise and angry vs.

disgust, while retaining more visually distinct expressions. The reduced complexity allowed a lightweight CNN to achieve competitive accuracy without deeper architectures.

Additional Improvements:

Given more time, we would explore data augmentation to improve generalization and reduce overfitting. Transfer learning with a pretrained model like ResNet could boost accuracy without significantly increasing training time. We would also consider adding a dedicated face detection step to the webcam pipeline, rather than relying on a simple center crop.

Future Work and Lessons Learned

Future Work

There are several ways the project could be expanded or refined:

1. **Use a more intensive and higher-quality dataset**
   FER2013 is limited by low resolution and grayscale images. A future version of the project could use a larger, higher-resolution dataset with more detailed facial features and cleaner labels. This would allow the model to learn more complex patterns and potentially improve classification performance.

2. **Expand beyond three emotions**
   With a more robust dataset, the model could be extended to classify all seven FER2013 emotions or even more nuanced emotional states.

3. **Experiment with stronger model architectures**
   Future work could explore deeper CNNs, residual networks, or transformer-based vision models to capture finer facial details.

4. **Further improve preprocessing**
   Techniques like face alignment, contrast enhancement, or noise reduction could help strengthen feature extraction, especially with higher-quality images.

5. **Develop a web application for deployment**
   Although a webcam demo was successfully implemented, building a full web application would allow users to upload images, test predictions online, and interact with the model through a more accessible interface.

Lessons Learned

Throughout the project, several key insights were gained:

1. **Dataset quality strongly affects model performance**
   The limitations of FER2013 highlighted how important resolution, clarity, and labeling accuracy are for emotion recognition tasks.

2. **Hyperparameter choices significantly influence training stability**
   Changes in learning rate, batch size, epochs, and dropout had clear effects on performance.

3. **Augmentation is essential for generalization**
   Stronger augmentation helped compensate for FER2013's noise and variability.

4. **Model complexity requires balance**
   Deeper networks improved feature extraction but needed careful validation to avoid overfitting.

5. **GPU acceleration greatly improves development speed**
   Faster training allowed more experimentation and contributed to better results.

6. **Collaboration made the development process more efficient**
   Dividing responsibilities for dataset selection, coding, experimentation, and environment setup helped keep the project organized and productive.