Human Emotion Recognition

Advanced Programming and Deep Learning for AI

Source

Georgii Kutivadze

Milan, 2025

Outline

1. Introduction
   1. What is the goal?
   2. What are the tasks?
2. Data & Preprocessing
   1. What Datasets are used?
   2. What is the Size & Image Format?
   3. What is the Class Distribution?
   4. Which Role each Dataset plays?
3. Models & Training
   1. What is the Model Architecture
   2. How many epochs? What is the Learning Rate?
   3. What are the Train and Validation Loss?
   4. What are the Train and Validation Accuracy?
4. Evaluation & Comparison
   1. Which Model is the Best?
   2. What is the Best Model's Performance on Test Set?
   3. Is the Score Satisfactory?
5. Generalization Check
   1. What if we apply zero-shot of our Best Model on a New Dataset?
   2. What if we Fine-Tune our Best Model on it and Repeat Evaluation?
   3. What is the Generalization Capability of our Best Model?
6. Conclusion
   1. What are the key findings?
   2. What are the most promising directions for future work?

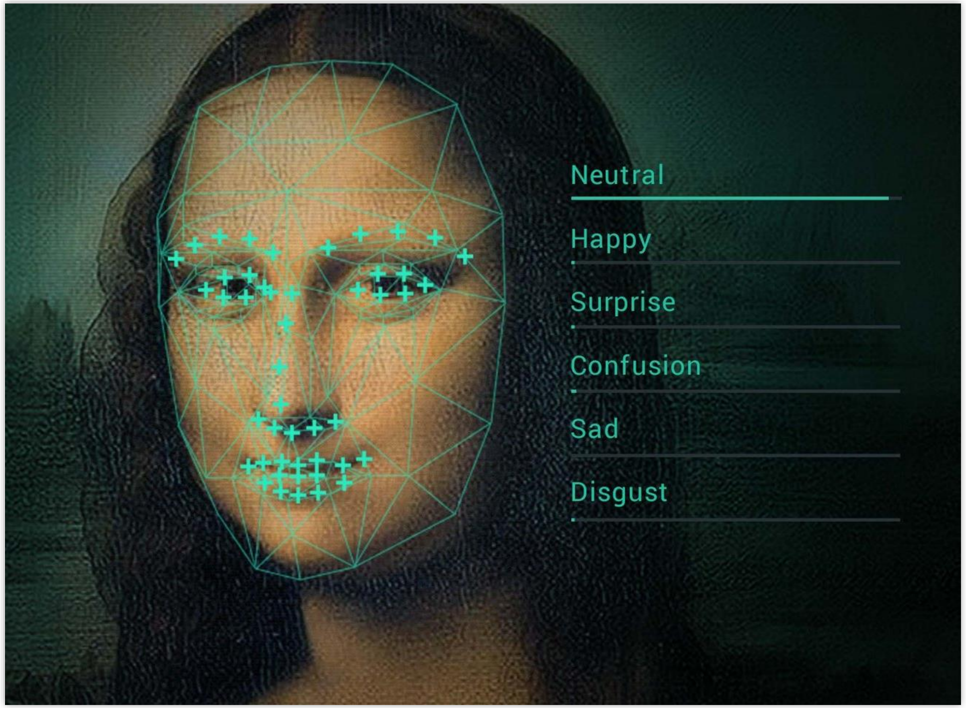
Can We Recognize Human Emotions from a Face Image?

GOAL

Systematically compare baseline and advanced deep-learning models for **facial expression recognition** on FER-2013, and study how well the best model transfers to RAF-DB

SUBTASKS

1. Build and train **baseline models** on FER-2013
2. Design and train stronger **CNN architectures** on FER-2013
3. Compare all models on the same **FER-2013** train/val/test splits and select the **best one**
4. Evaluate **transfer** of the best FER model **to RAF-DB** (before and after fine-tuning)
5. Analyze class distributions and **errors** on both datasets to understand **remaining limitations**



Source

Datasets: FER-2013 for Main Training, RAF-DB for Fine-Tuning & Generalization

SIZE

- **FER-2013:** larger (~35k images)
- **RAF-DB:** smaller (~15k images)

IMAGE FORMAT

- **FER-2013:** low-resolution, grayscale, 48×48
- **RAF-DB:** higher-quality RGB images, resized and converted to grayscale in our pipeline

LABEL SPACE

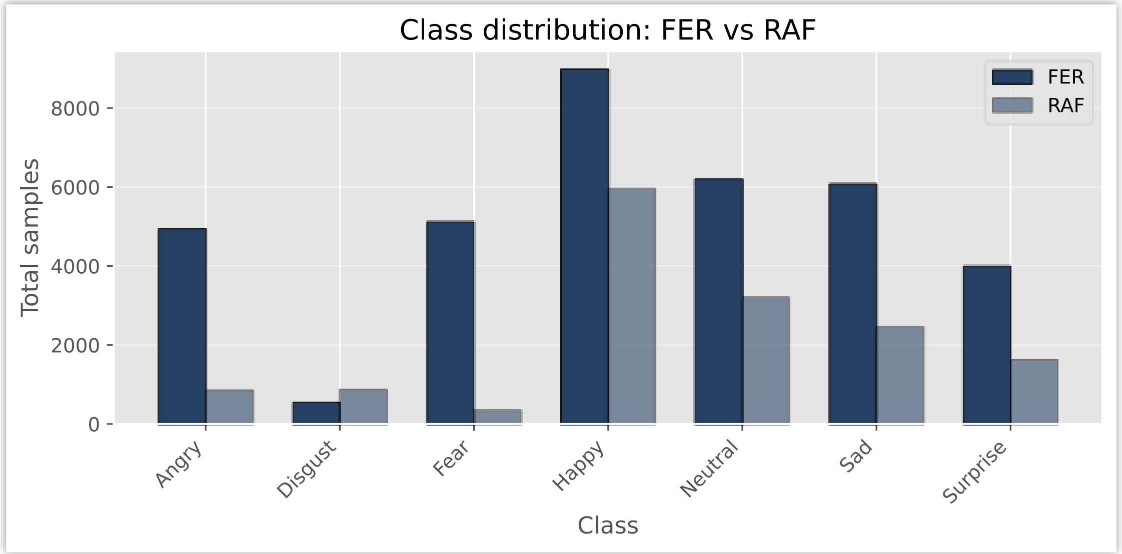
- Both use the same **7 basic emotions**: Angry, Disgust, Fear, Happy, Sad, Surprise, Neutral
- For RAF-DB, class indices are **remapped** to match FER-2013 labels

CLASS DISTRIBUTION

- Both are **heavily skewed** towards Happy, Neutral, and Sad
- Minority classes (Disgust, Fear) are especially **under-represented** in RAF-DB

ROLE IN THIS PROJECT

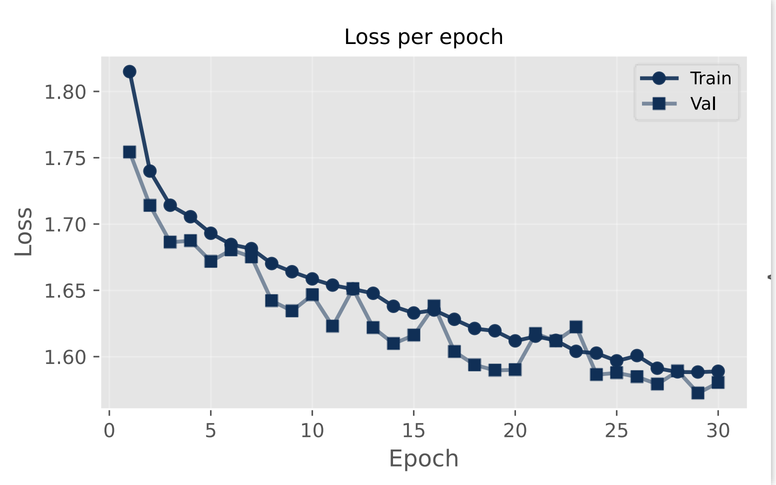
- **FER-2013:** main dataset for training and model comparison
- **RAF-DB:** used for **transfer learning** and checking cross-dataset generalization

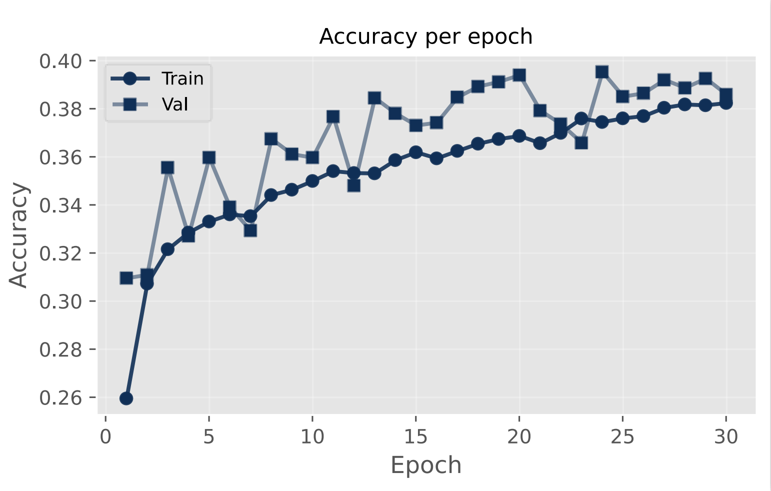


Model №1: Simple MLP

ARCHITECTURE

```
class SimpleMLP(nn.Module):
    def __init__(self, num_classes=7):
        super().__init__()
        self.flatten = nn.Flatten()
        self.net = nn.Sequential(
            nn.Linear(1 * 48 * 48, 256),
            nn.ReLU(),
            nn.Linear(256, num_classes)
        )

    def forward(self, x):
        x = self.flatten(x)
        x = self.net(x)
        return x
```





TRAIN LOSS
1.5884

TRAIN ACCURACY
38.24%

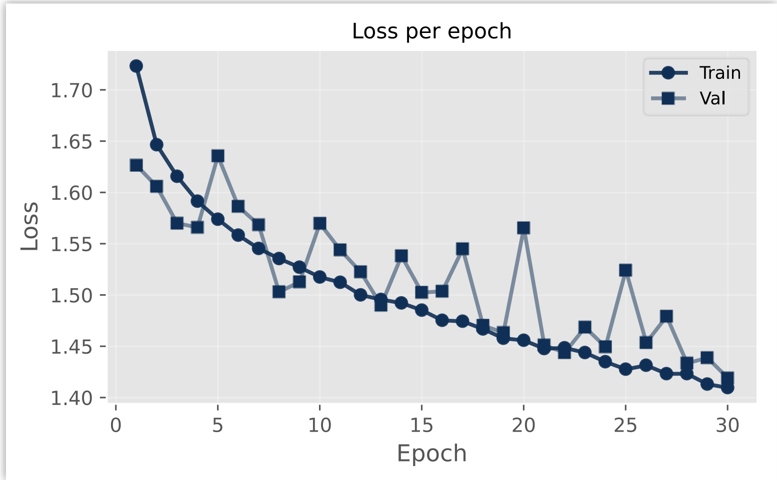
VAL LOSS
1.5726

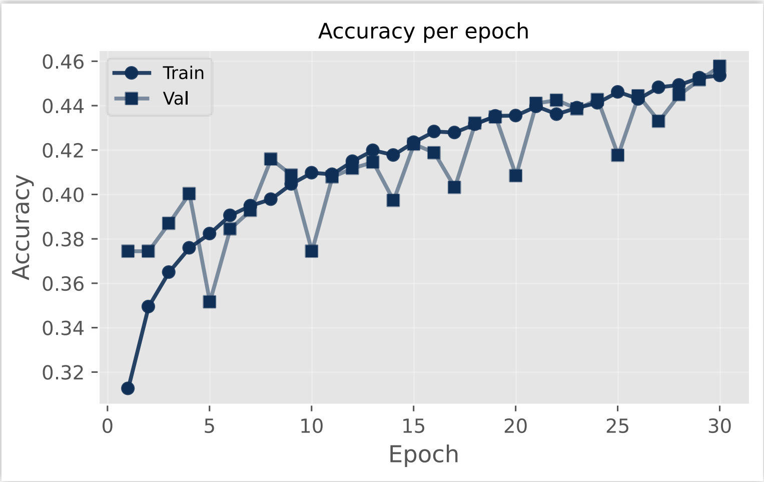
VAL ACCURACY
39.54%

Model №2: Improved MLP (BatchNorm + Dropout)

ARCHITECTURE

```
class ImprovedMLP(nn.Module):
    def __init__(self, num_classes=7):
        super().__init__()
        self.flatten = nn.Flatten()
        self.net = nn.Sequential(
            nn.Linear(1 * 48 * 48, 512),
            nn.BatchNorm1d(512),
            nn.ReLU(),
            nn.Dropout(0.3),

            nn.Linear(512, 256),
            nn.BatchNorm1d(256),
            nn.ReLU(),
            nn.Dropout(0.3),

            nn.Linear(256, 128),
            nn.BatchNorm1d(128),
            nn.ReLU(),

            nn.Linear(128, num_classes)        )

def forward(self, x):
        x = self.flatten(x) x = self.net(x) # [B, 1, 48, 48] -> [B, 2304]#
[B, 7]
        return x
```





TRAIN LOSS

1.4097

TRAIN ACCURACY

45.36%

VAL LOSS

1.4192

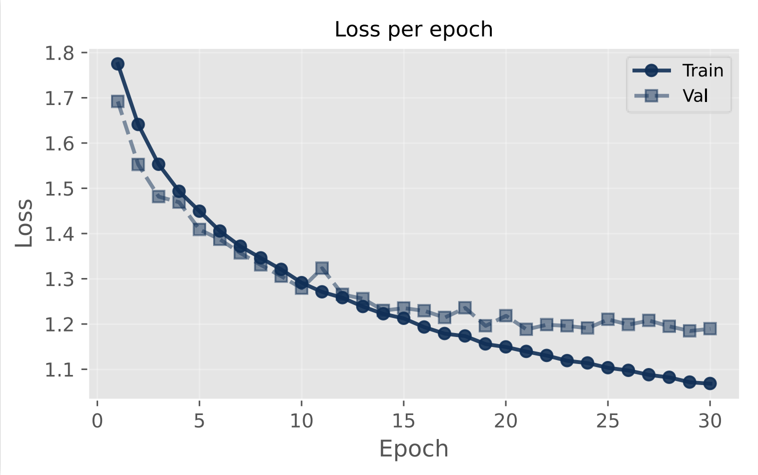
VAL ACCURACY

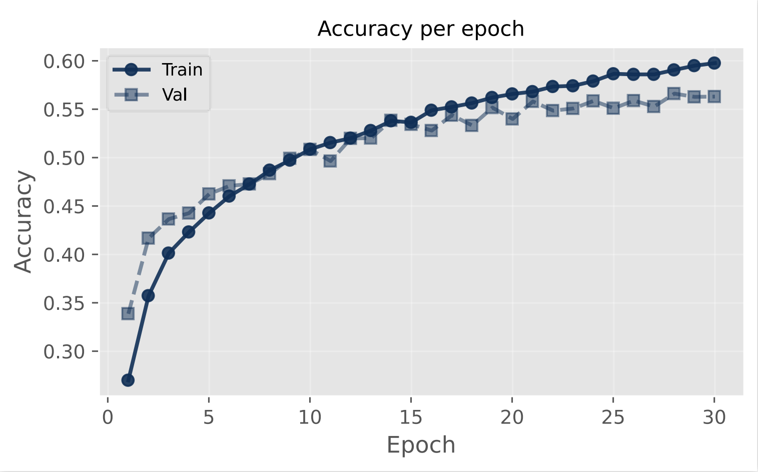
45.78%

Model №3: Simple CNN

ARCHITECTURE

```
class SimpleCNN(nn.Module):
    def __init__(self, num_classes=7):
        super().__init__()
        self.features = nn.Sequential(
            # Вход: (B, 1, 48, 48)
            nn.Conv2d(1, 16, kernel_size=3, padding=1),
            nn.ReLU(),
            nn.MaxPool2d(2, 2),

            nn.Conv2d(16, 32, kernel_size=3, padding=1),
            nn.ReLU(),
            nn.MaxPool2d(2, 2),

            nn.Conv2d(32, 64, kernel_size=3, padding=1),
            nn.ReLU(),
            nn.MaxPool2d(2, 2),
        )

        self.classifier = nn.Sequential(
            nn.Flatten(),
            nn.Linear(64 * 6 * 6, 128),
            nn.ReLU(),
            nn.Linear(128, num_classes)
        )
...
```





TRAIN LOSS

1.0686

TRAIN ACCURACY

59.77%

VAL LOSS

1.1850

VAL ACCURACY

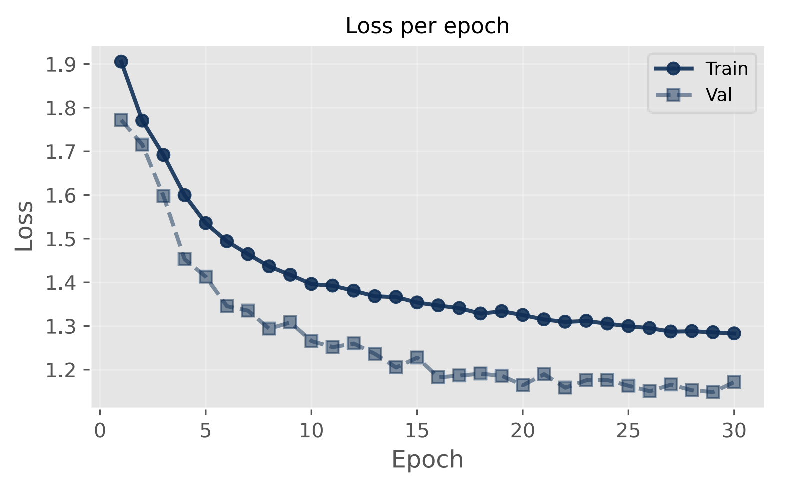
56.62%

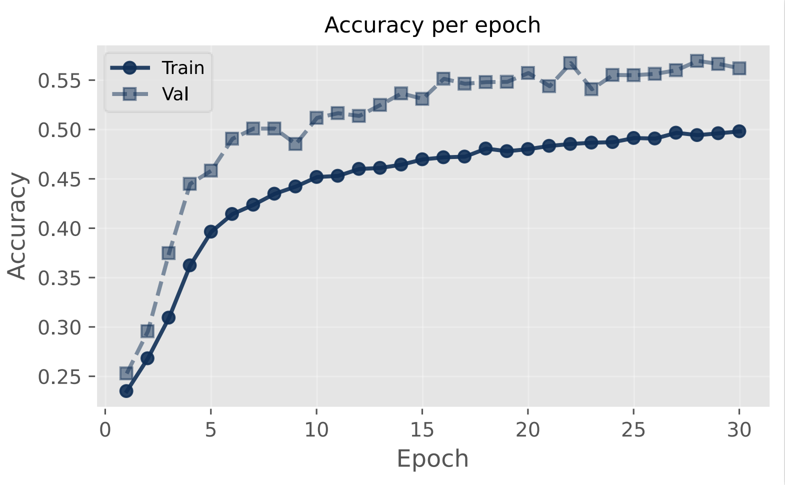
Model №4: Improved CNN

ARCHITECTURE

```
# Block 1: (B, 1, 48, 48) -> (B, 32, 24, 24)
nn.Conv2d(1, 32, kernel_size=3, padding=1),
nn.BatchNorm2d(32),
nn.ReLU(),
nn.Conv2d(32, 32, kernel_size=3, padding=1),
nn.BatchNorm2d(32),
nn.ReLU(),
nn.MaxPool2d(2, 2),              # 48 -> 24

# Block 2: (B, 32, 24, 24) -> (B, 64, 12, 12)
nn.Conv2d(32, 64, kernel_size=3, padding=1),
nn.BatchNorm2d(64),
nn.ReLU(),
nn.Conv2d(64, 64, kernel_size=3, padding=1),
nn.BatchNorm2d(64),
nn.ReLU(),
nn.MaxPool2d(2, 2),              # 24 -> 12

# Block 3: (B, 64, 12, 12) -> (B, 128, 6, 6)
nn.Conv2d(64, 128, kernel_size=3, padding=1),
nn.BatchNorm2d(128),
nn.ReLU(),
nn.MaxPool2d(2, 2),              # 12 -> 6
```

```
self.classifier = nn.Sequential(
    nn.Flatten(),                        # (B, 128*6*6)
    nn.Dropout(0.5),
    nn.Linear(128 * 6 * 6, 256),
    nn.ReLU(),
    nn.Dropout(0.5),
    nn.Linear(256, num_classes)
)
```





TRAIN LOSS

1.2832

TRAIN ACCURACY

49.82%

VAL LOSS

1.1493

VAL ACCURACY

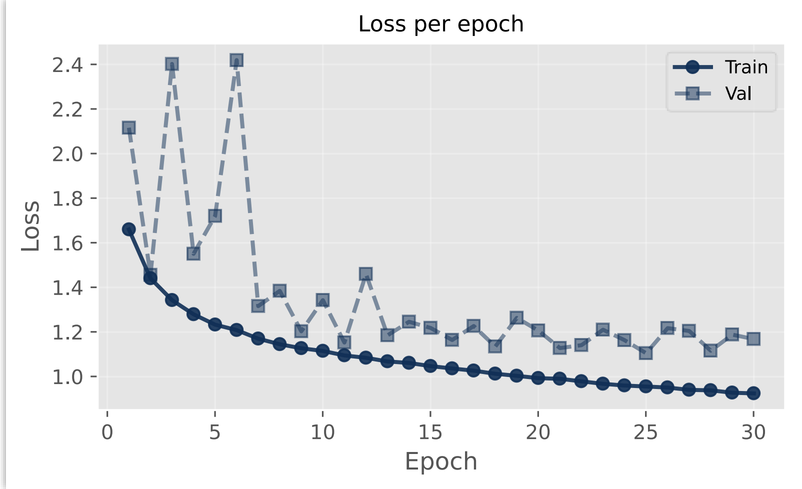
56.95%

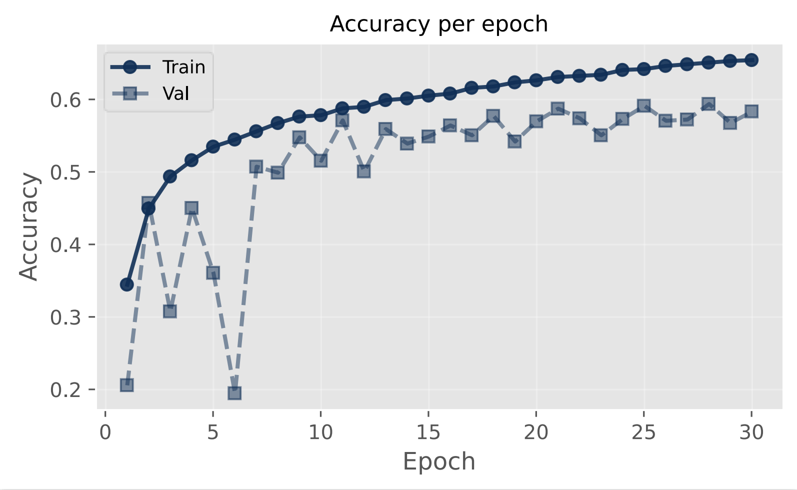
Model №5: GAP CNN

ARCHITECTURE

```
# Block 1: 1x48x48 -> 32x24x24
nn.Conv2d(1, 32, kernel_size=3, padding=1),
nn.BatchNorm2d(32),
nn.ReLU(),
nn.MaxPool2d(2, 2),

# Block 2: 32x24x24 -> 64x12x12
nn.Conv2d(32, 64, kernel_size=3, padding=1),
nn.BatchNorm2d(64),
nn.ReLU(),
nn.MaxPool2d(2, 2),

# Block 3: 64x12x12 -> 128x6x6
nn.Conv2d(64, 128, kernel_size=3, padding=1),
nn.BatchNorm2d(128),
nn.ReLU(),
nn.MaxPool2d(2, 2),

# Block 4: 128x6x6 -> 256x6x6
nn.Conv2d(128, 256, kernel_size=3, padding=1),
nn.BatchNorm2d(256),
nn.ReLU(),
```

```
self.gap = nn.AdaptiveAvgPool2d(1)    # -> (B, 256, 1, 1)
self.classifier = nn.Linear(256, num_classes)
```





TRAIN LOSS

0.9249

TRAIN ACCURACY

65.43%

VAL LOSS

1.1051

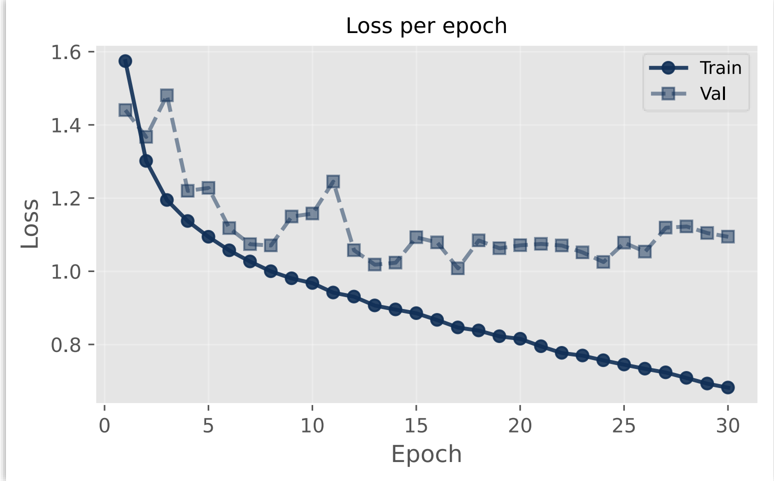
VAL ACCURACY

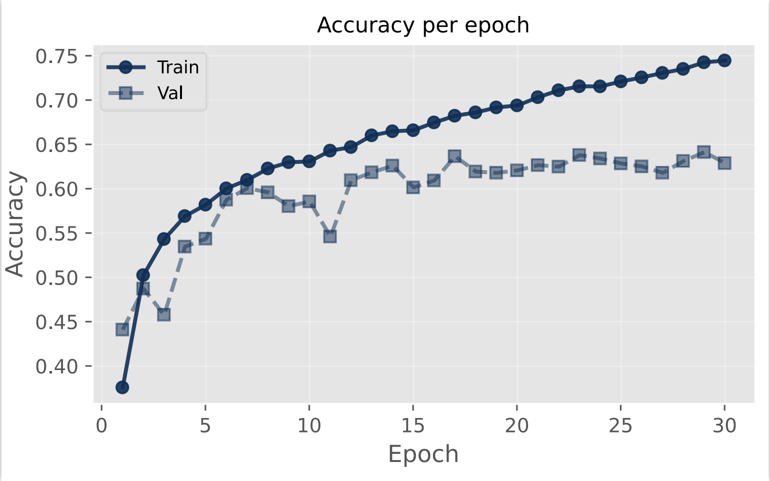
59.38%

Model №6: MiniResNet

ARCHITECTURE

- Input: **1×48×48**
- **Stem:** 3×3 Conv + BN + ReLU → 32 channels
- **3 residual stages (BasicBlock):**
- Stage 1: 32 ch, 48×48 → 24×24
- Stage 2: 64 ch, 24×24 → 12×12
- Stage 3: 128 ch, 12×12 → 6×6
- **Skip connections** with downsampling
- **Global Average Pooling**
- **Fully connected layer -> 7 classes**





TRAIN LOSS

0.6823

TRAIN ACCURACY

74.47%

VAL LOSS

1.0082

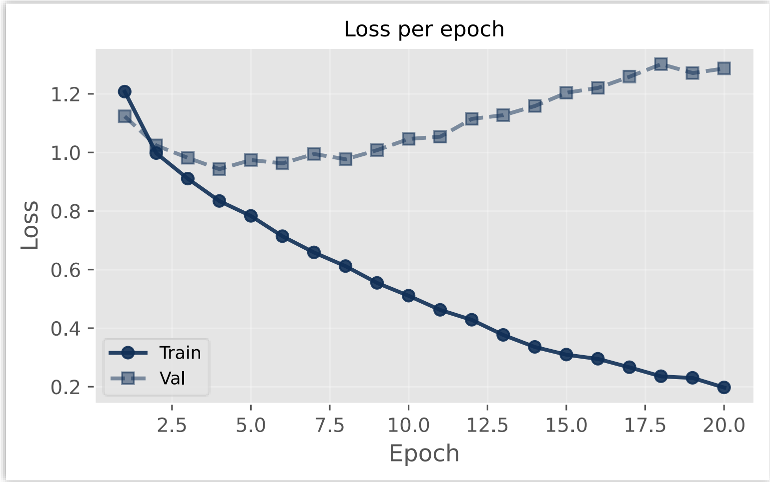
VAL ACCURACY

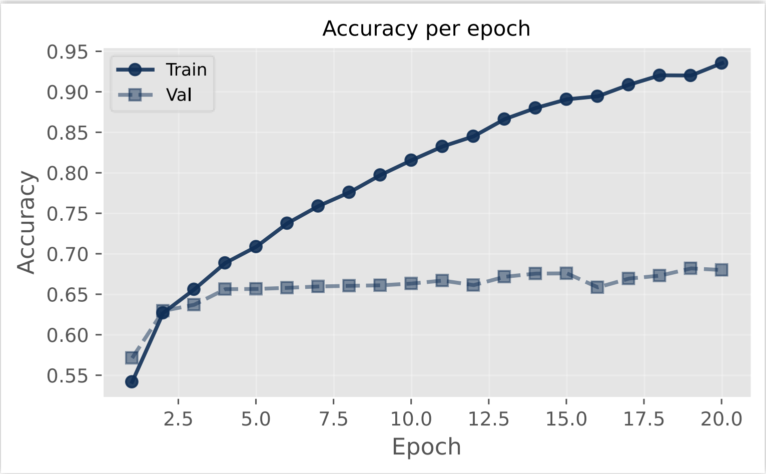
64.14%

Model №7: ResNet-18 (Transfer Learning)

ARCHITECTURE

- **FERResNet18 (Transfer Learning with ResNet-18)**
- Backbone: **ResNet-18** (ImageNet-pretrained, optional)
- Head: replace **final FC -> 7-class classifier**
- Input pipeline (FER 48×48 grayscale):
- **repeat 1 -> 3 channels**
- **resize 48×48 -> 224×224** (bilinear)
- **ImageNet normalization** (mean/std)
- Training strategy:
- **Stage 1:** freeze backbone, train **classifier head only** (Adam, lr=1e-3)
- **Stage 2:** unfreeze all layers, **fine-tune full network** (Adam, lr=1e-4)





TRAIN LOSS
0.1976

TRAIN ACCURACY
93.55%

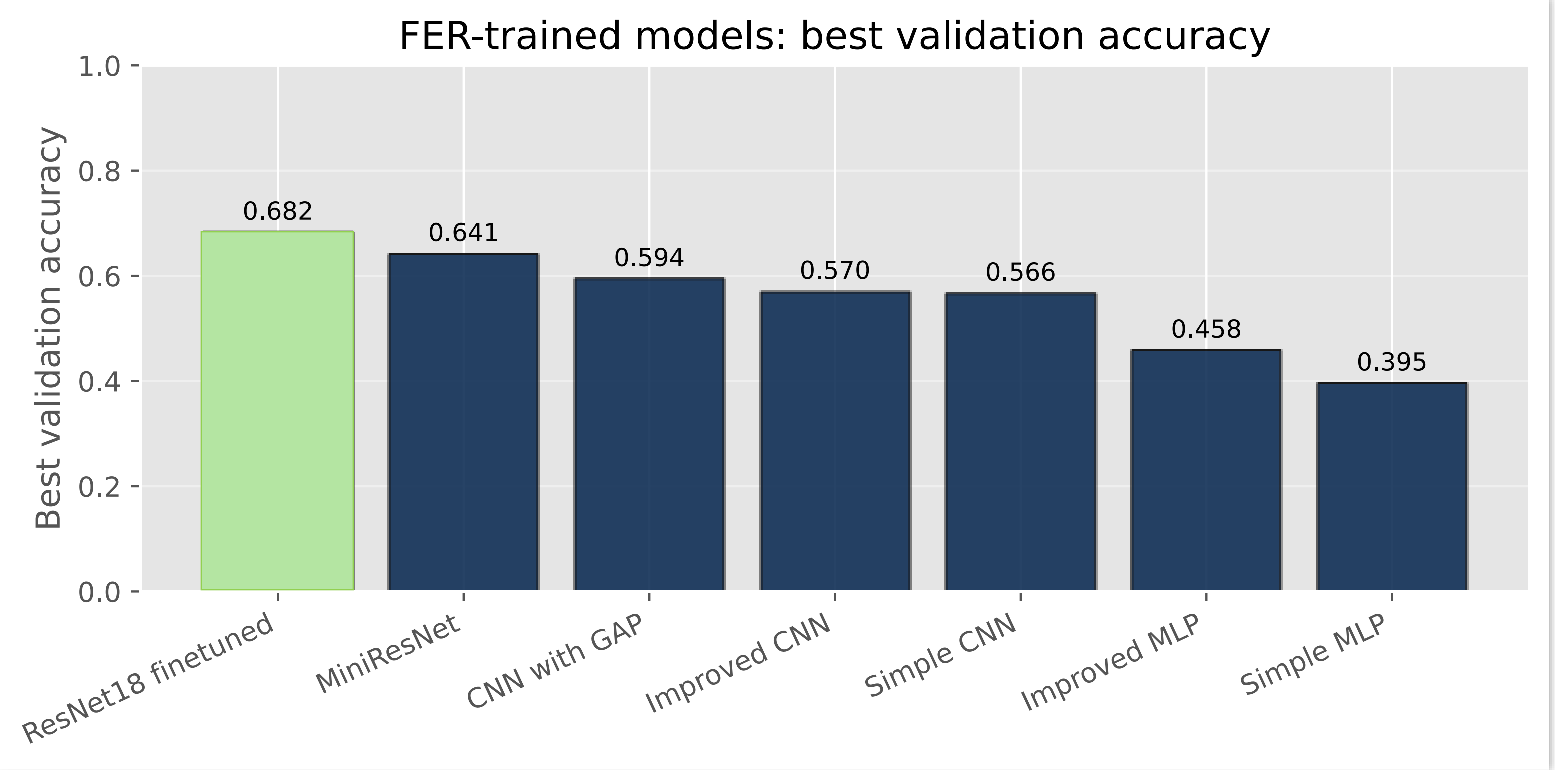
VAL LOSS
0.9433

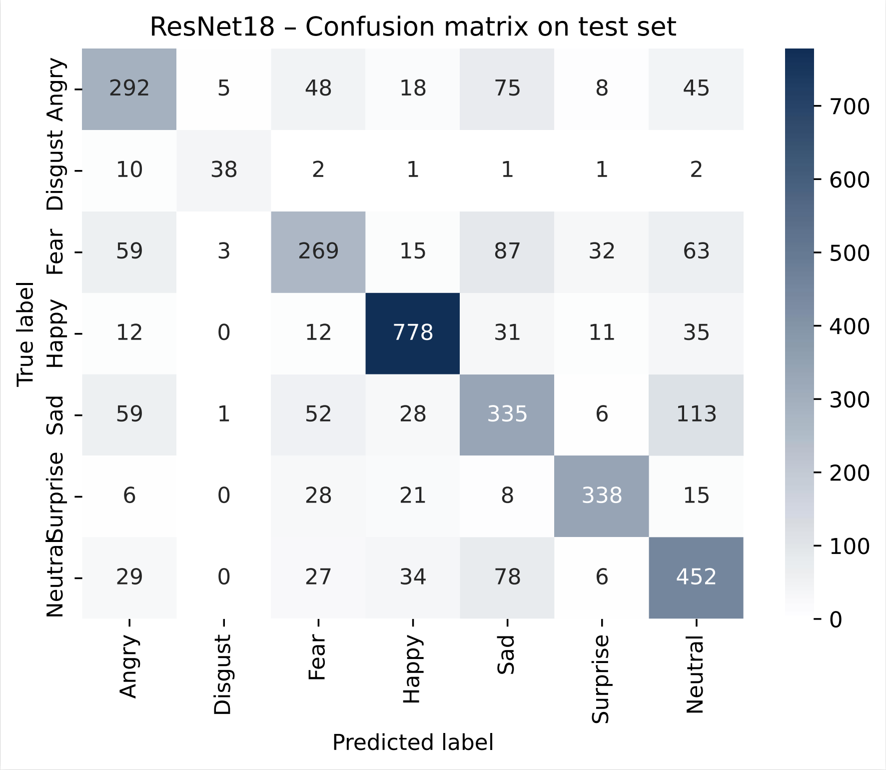
VAL ACCURACY
68.21%

Evaluation & Comparison

Fine-tuned ResNet18 achieves the **best validation accuracy**, clearly outperforming simpler CNN and MLP-based architectures, highlighting the advantage of transfer learning for FER tasks

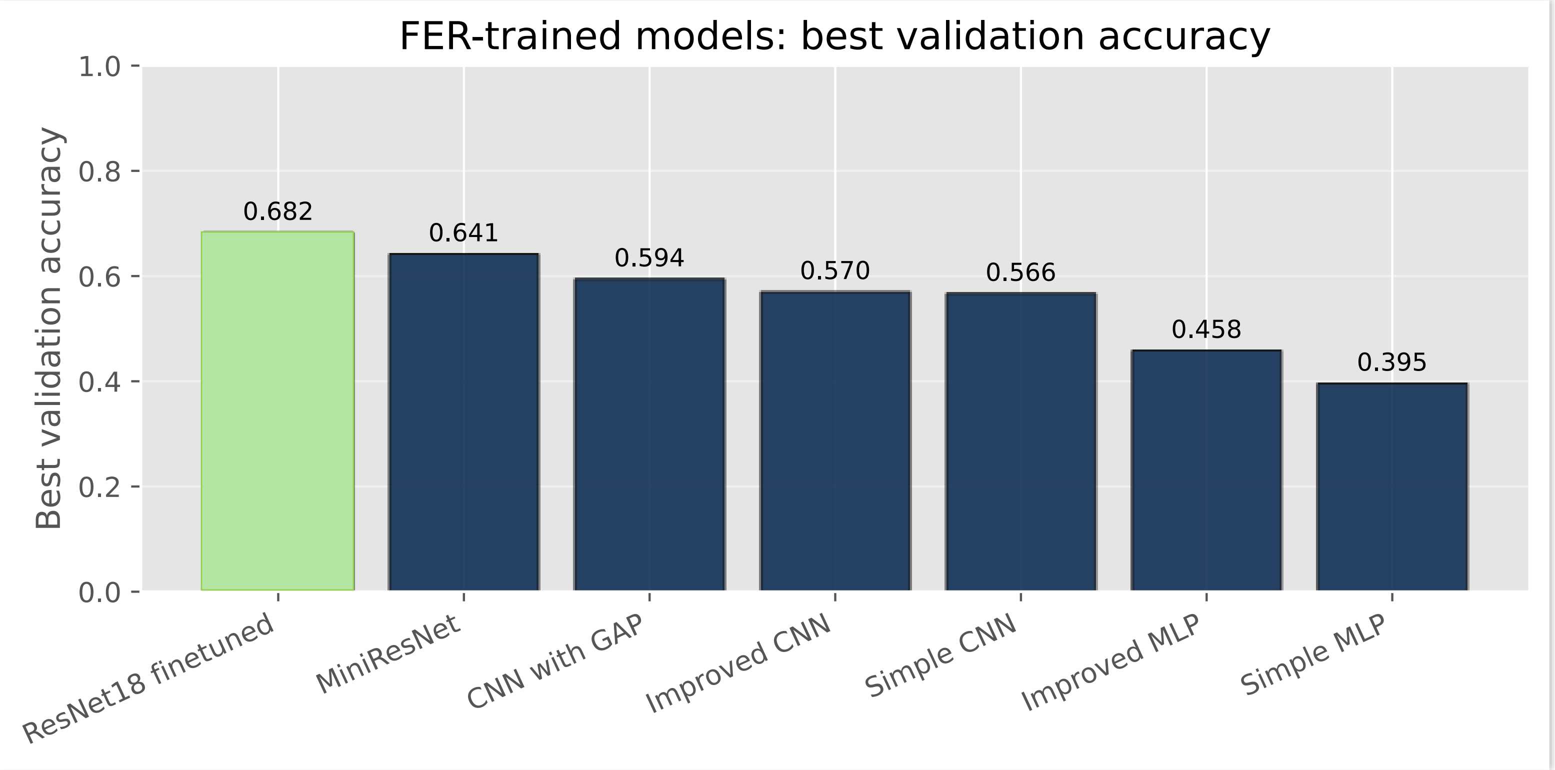
Strong performance for **Happy & Neutral** classes, while notable confusion **between similar emotions**, indicating **class overlap** in facial expressions





Evaluation & Comparison

Fine-tuned ResNet18 achieves the **best validation accuracy**, clearly outperforming simpler CNN and MLP-based architectures, highlighting the advantage of transfer learning for FER tasks

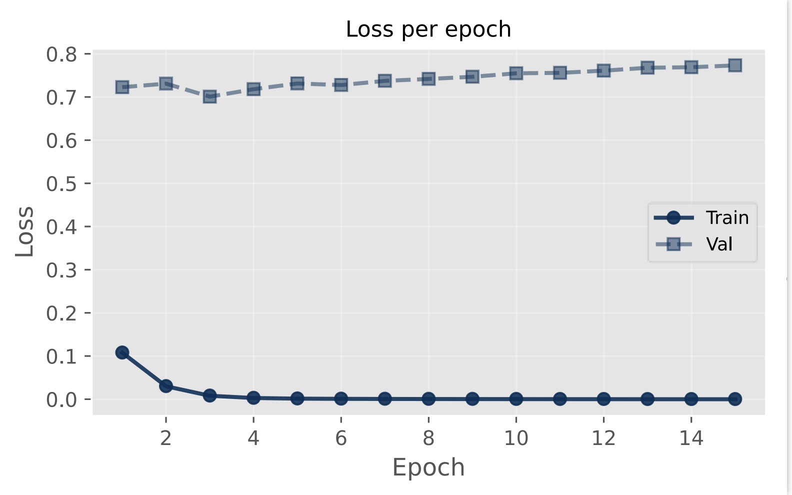


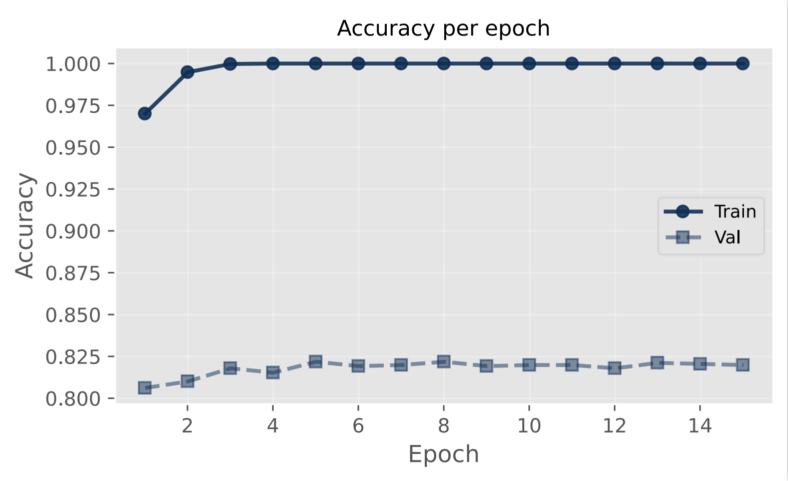
Fine-tuned ResNet18 achieves the solid test accuracy however, there is **still room for improvement**

FER TEST LOSS	FER TEST ACC.
1.1669	69.71%

- Good convergence
- Satisfactory results
- But not perfect

Best Model Generalization Check: RAF-DB Dataset





RAF TRAIN LOSS
0.0004

RAF TRAIN ACC.
100.0%

RAF VAL LOSS
0.7009

RAF VAL ACC.
82.19%

RAF TEST LOSS
0.6347

RAF TEST ACC.
82.21%

FER TEST LOSS
2.4567

FER TEST ACC.
54.04%

The performance gap reveals **poor generalization capability. Why?**

Best Model Generalization Check: RAF-DB Dataset

FAR TEST LOSS	FAR TEST ACC.
0.6347	82.21%
2.4567	54.04%

The performance gap reveals **poor generalization capability. Why?**

SOURCES OF THE ISSUES

- Domain Shift
- Overfitting
- Class Imbalance
- Catastrophic Forgetting
- Conflicting Optimal Solutions

RECOMMENDATIONS

- Joint Training
- Freeze Lower Layers During Fine-Tuning
- Hyperparameter Tuning, Regularization, and Early Stopping
- Improve Data Alignment

Key Conclusions

1. CNNs significantly outperform MLPs for facial expression recognition
2. Architectural improvements and residual learning provide consistent gains
3. Pretraining is the most impactful factor for overall performance
4. Class imbalance and dataset bias remain major error sources
5. Cross-dataset generalization is limited and requires fine-tuning

Human Emotions Recognition