Facial Expression Classification Using PyTorch

Farid Vakili

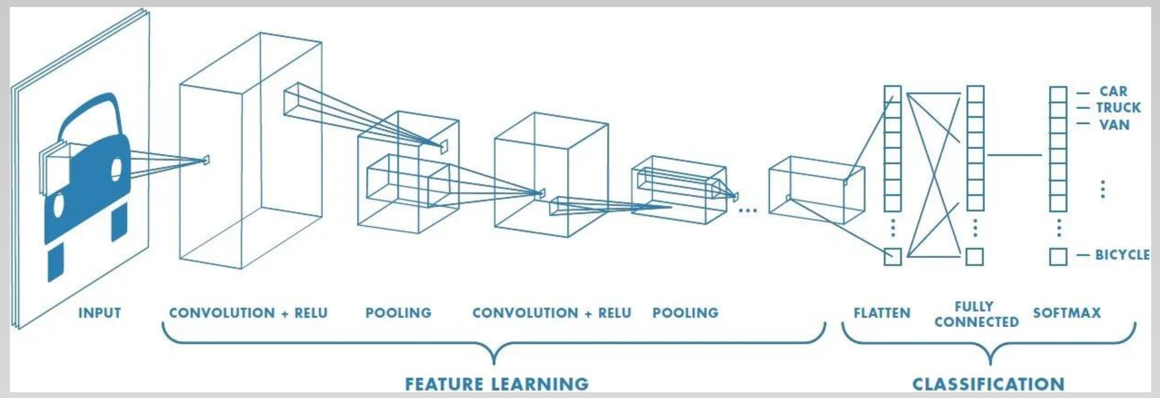
California State Polytechnic University Pomona

ABSTRACT

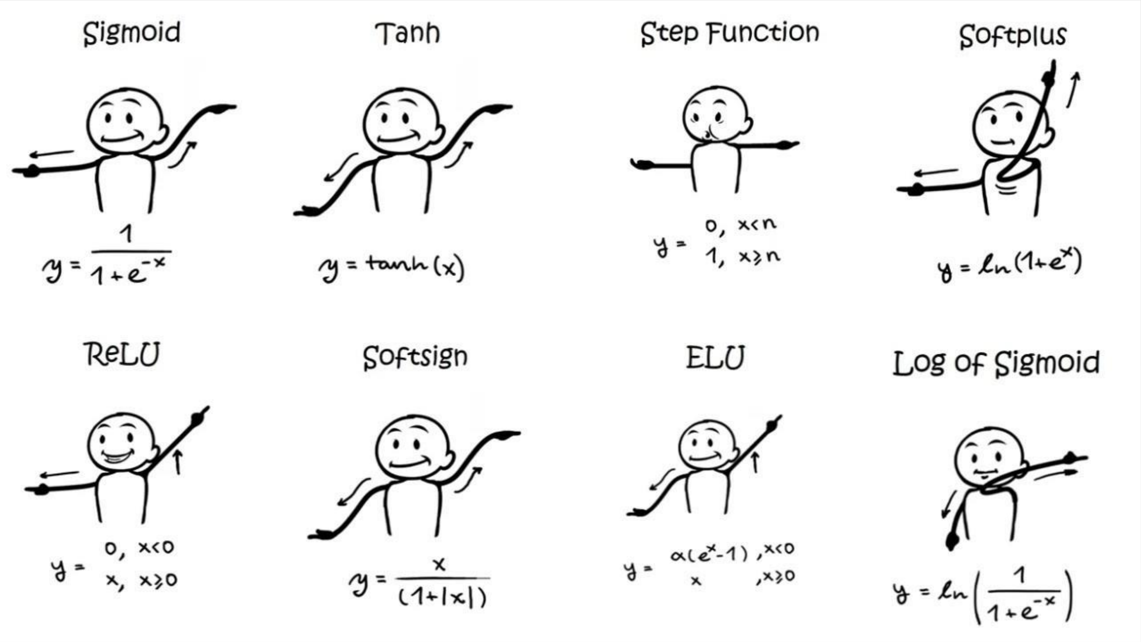
Facial expression recognition is a crucial task in the field of computer vision with applications ranging from human-computer interaction to psychological studies. This project utilizes convolution neural network (CNN) to classify facial expressions using PyTorch (an open-source machine learning library)

OBJECTIVES

Our objective was to implement a CNN for classifying facial expressions into three categories: Angry, Happy, and Neutral (we were also given a base line accuracy score to beat). The model was trained and evaluated on a publicly available dataset from a previous Kaggle competition



https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53



https://www.linkedin.com/pulse/activation-functions-101-sigmoid-tanh-relu-softmax-ben-hammouda

MATERIALS & METHODS

Dataset

The dataset from the "Challenges in Representation Learning: Facial Expression Recognition Challenge" on Kaggle https://www.kaggle.com/c/challenges-in-representation-learning-facial-expression-recognition-challenge/data

- **Training Set:** 16,175 grayscale images of size 48x48 pixels
- **Test Set:** 3,965 images of the same dimensions
- **Classes:** Angry (0), Happy (1), Neutral (2)

Data Preprocessing

- **Normalization:** Pixel values scaled to the range [0, 1]
- **Reshaping:** Images reshaped to (48, 48, 1) to represent single-channel grayscale images

Data Augmentation

To enhance model generalization, data augmentation techniques are applied:

- **Random Horizontal Flip:** Simulates mirror images
- **Random Rotation:** Images are rotated within a range of ±10 degrees

Model Architecture

The CNN model:

- **Convolutional Layers:** Three blocks with increasing filter sizes (32, 64, 128) and kernel size 3x3
  – Each block includes a convolutional layer, ReLU activation, batch normalization, and max pooling
- **Fully Connected Layers:**
  – A dense layer with 256 neurons and ReLU activation
  – Dropout layer to prevent overfitting
  – Output layer with 3 neurons (each class)

Training Settings

- **Loss Function:** Cross-Entropy Loss
- **Optimizer:** Adam optimizer (initial learning rate 0.001)
- **Batch Size:** 64
- **Number of Epochs:** 20
- **Device:** Training uses GPU (if available)

Validation Strategy

- **Train-Validation Split:** Split the training data to monitor validation performance
- **Early Stopping:** The model saves weights when the validation accuracy improves

RESULTS

Training and Validation Performance

Over 20 epochs, the model showed steady improvement in both training and validation metrics

•**Final Training Loss:** Decreased significantly (indicating good learning capacity)

•**Validation Accuracy:** Achieved approximately 80% accuracy

```
Epoch [1/20], Train Loss: 1.0169, Val Loss: 0.8143, Val Acc: 0.6357
Model saved!
Epoch [2/20], Train Loss: 0.8081, Val Loss: 0.6818, Val Acc: 0.7008
Model saved!
Epoch [3/20], Train Loss: 0.7150, Val Loss: 0.6412, Val Acc: 0.7237
Model saved!
Epoch [4/20], Train Loss: 0.6641, Val Loss: 0.6075, Val Acc: 0.7365
Model saved!
Epoch [5/20], Train Loss: 0.6248, Val Loss: 0.5874, Val Acc: 0.7480
Model saved!
Epoch [6/20], Train Loss: 0.5958, Val Loss: 0.5709, Val Acc: 0.7694
Model saved!
Epoch [7/20], Train Loss: 0.5794, Val Loss: 0.5605, Val Acc: 0.7655
Epoch [8/20], Train Loss: 0.5588, Val Loss: 0.5688, Val Acc: 0.7715
Model saved!
Epoch [9/20], Train Loss: 0.5375, Val Loss: 0.5531, Val Acc: 0.7678
Epoch [10/20], Train Loss: 0.5253, Val Loss: 0.5586, Val Acc: 0.7814
Model saved!
Epoch [11/20], Train Loss: 0.5128, Val Loss: 0.5386, Val Acc: 0.7762
Epoch [12/20], Train Loss: 0.4946, Val Loss: 0.5352, Val Acc: 0.7826
Model saved!
Epoch [13/20], Train Loss: 0.4794, Val Loss: 0.5429, Val Acc: 0.7797
Epoch [14/20], Train Loss: 0.4591, Val Loss: 0.5416, Val Acc: 0.7826
Epoch [15/20], Train Loss: 0.4530, Val Loss: 0.5495, Val Acc: 0.7822
Epoch [16/20], Train Loss: 0.4356, Val Loss: 0.5664, Val Acc: 0.7797
...
Epoch [18/20], Train Loss: 0.4109, Val Loss: 0.5426, Val Acc: 0.7923
Model saved!
Epoch [19/20], Train Loss: 0.4037, Val Loss: 0.5512, Val Acc: 0.7915
Epoch [20/20], Train Loss: 0.3924, Val Loss: 0.5979, Val Acc: 0.7902
```

Techniques to Improve Performance

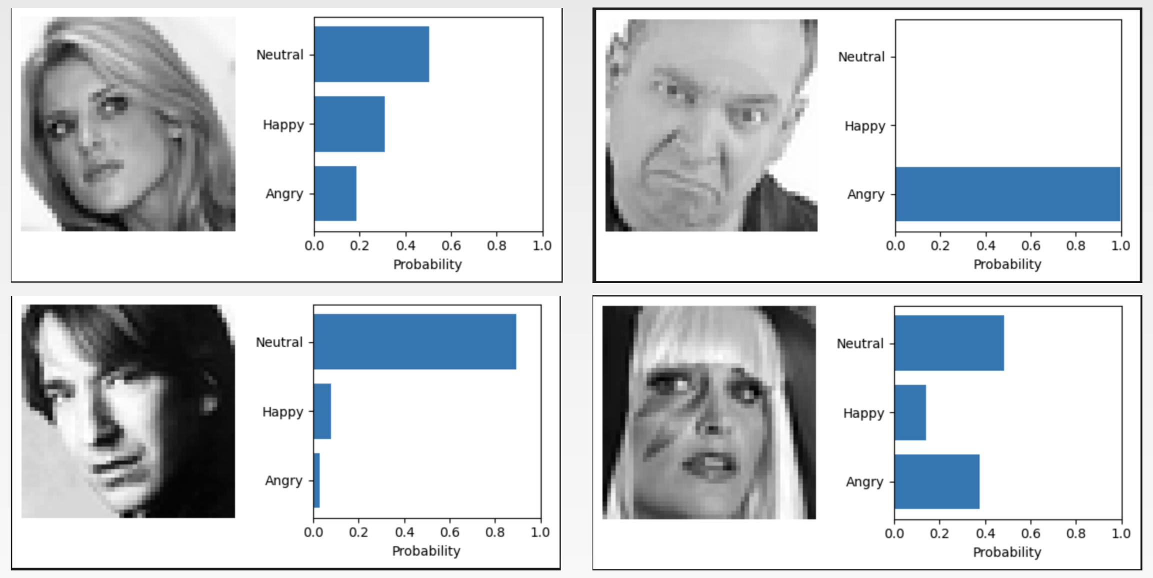
•**Data Augmentation:** Helped reduce overfitting by providing more varied training examples

•**Batch Normalization:** Improved training stability and convergence speed

•**Dropout Layer:** Prevented overfitting by randomly deactivating neurons during training

•**Hyperparameter Tuning:** Adjusted learning rates and batch sizes

-Image Classification Probabilities-



CONCLUSION

This assignment demonstrates the capability of CNNs in classifying facial expressions. Following the steps of the machine learning development process, our model generalizes pretty well to new data. Future research could explore deeper architectures, or incorporate attention mechanisms to further enhance performance

REFERENCES & ACKNOWLEDGMENTS

1. Li, S., & Deng, W. (2020). "Deep Facial Expression Recognition: A Survey." *IEEE Transactions on Affective Computing*, 1-1

2. Alizadeh, Shima, and Azar Fazel. "Convolutional Neural Networks for Facial Expression Recognition" *https://Arxiv.Org/Abs/1704.06756*, 22 Apr. 2017, vision.stanford.edu/teaching/cs231n/reports/2016/pdfs/005_Report.pdf

1. Andrew Ng – Stanford Deeplearning.AI https://www.coursera.org/specializations/machine-learning-introduction

2. Parth Dhameliya – Machine Learning Instructor Coursera https://www.coursera.org/projects/facial-expression-recognition-with-pytorch (demonstrated the function for image classification probability outputs)

Dr. Andrew Ng

Dr. Hao Ji (our professor!)