COSC-4117EL: Assignment 3 Report

Group Number: 2
Group Member:

NAME	STUDENT#	EMAIL	CONTRIBUTION
Haoliang Sheng	0441916	hsheng@laurentian.ca	60%
Zihao Zhou	0429993	zzhou3@laurentian.ca	20%
Jiazhou Ye	0426609	Jye1@laurentian.ca	20%

1. Introduction and Problem Statement

In the realm of computer vision, **facial expression recognition** stands as a critical component, serving a plethora of applications ranging from psychological research to enhancing user experiences in technology interfaces. The capacity to accurately discern human emotions through facial expressions has substantial implications in fields such as **affective computing, security, and human-computer interaction**.

This project is dedicated to developing a **real-time facial expression detection system** that classifies facial expressions into three distinct categories: **happy, neutral, and surprise**. This system aims to employ a webcam to capture and process visual data on the fly, thus enabling the dynamic assessment of human emotions. The significance of this system is multifaceted; it not only provides a non-invasive method for gauging emotional states but also offers a foundation for advanced applications such as **mood-based service customization or emotional state monitoring for mental health assessments**.

Video Demonstration: https://youtu.be/hOzWkHtAuMA

2. Dataset Description

The FER-2013 dataset, sourced from Kaggle, is a comprehensive collection tailored for facial expression recognition tasks. Originally, this dataset encompasses seven distinct emotion categories: **Angry, Disgust, Fear, Happy, Sad, Surprise, and Neutral**. However, for the scope of this project, we focus on three primary expressions: **Happy, Neutral, and Surprise**, due to their pronounced and distinct nature, making them more suitable for real-time recognition.

The dataset has a total of 35,886 images, of which 28,708 are used for training and 3,589 are used for testing. Each image in the dataset is a **grayscale picture of 48x48 pixels**, providing a uniform standard for processing and analysis. We choose **1000 images for each class** to **train** the model, and **1000 images for each class** to **evaluate** the model. The relatively low resolution of these images, compared to the high-resolution input from webcams, presents a significant challenge in ensuring the model's effectiveness in real-world scenarios.

To mitigate this disparity and enhance the model's robustness, we have employed grayscale images in our training process. This approach serves two purposes: firstly, it reduces the complexity of the model by focusing on **structural features rather than color information**, which is less relevant for facial expression recognition. Secondly, it mimics the conditions of **real-time webcam input**, where lighting conditions and background variability can significantly alter color information. This grayscale approach, coupled with our other data augmentation techniques **(random flips, rotations, brightness, contrast adjustments, and affine transformations)**, ensures a robust model capable of accurately classifying facial expressions in diverse and dynamic real-world environments.

Memory Limitation in Data Loading:

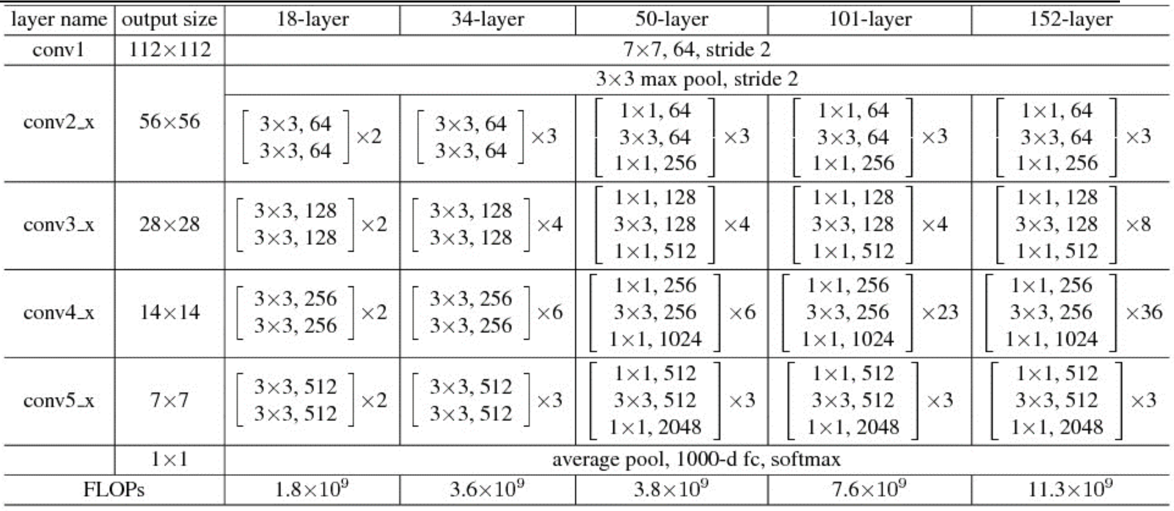
- **Challenge:** While loading the FER-2013 dataset, the large data size led to insufficient memory when trying to load all the data into memory at once.
- **Solution:** We employed **`FER2013Dataset`** and **`DataLoader`** for data handling. FER2013Dataset allowed for efficient management of dataset loading, and **`DataLoader`** provided a way to iteratively load a small batch of data into memory, reducing memory usage and improving data handling efficiency.

3. Methodology

Transfer Learning Approach

In this project, we have employed **transfer learning** as a strategic approach to leverage the sophisticated feature extraction capabilities of pre-trained networks. We chose the **ResNet18, ResNet34, and ResNet50** model, three versions of the Residual Network, to find out the balance of depth and computational efficiency, finding an ideal model for real-time applications.

Architecture of models



layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		[3×3, 64 3×3, 64] ×2	[3×3, 64 3×3, 64] ×3	[1×1, 64 3×3, 64 1×1, 256] ×3	[1×1, 64 3×3, 64 1×1, 256] ×3	[1×1, 64 3×3, 64 1×1, 256] ×3
conv3_x	28×28	[3×3, 128 3×3, 128] ×2	[3×3, 128 3×3, 128] ×4	[1×1, 128 3×3, 128 1×1, 512] ×4	[1×1, 128 3×3, 128 1×1, 512] ×4	[1×1, 128 3×3, 128 1×1, 512] ×8
conv4_x	14×14	[3×3, 256 3×3, 256] ×2	[3×3, 256 3×3, 256] ×6	[1×1, 256 3×3, 256 1×1, 1024] ×6	[1×1, 256 3×3, 256 1×1, 1024] ×23	[1×1, 256 3×3, 256 1×1, 1024] ×36
conv5_x	7×7	[3×3, 512 3×3, 512] ×2	[3×3, 512 3×3, 512] ×3	[1×1, 512 3×3, 512 1×1, 2048] ×3	[1×1, 512 3×3, 512 1×1, 2048] ×3	[1×1, 512 3×3, 512 1×1, 2048] ×3
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

The image above is a table outlining the different configurations of ResNet models for various depths (18-layer, 34-layer, 50-layer, 101-layer, and 152-layer).

Modifications for Facial Expression Classification:

- We adapted the **final fully connected layer** of the models to output three classes corresponding to our focus: Happy, Neutral, and Surprise.
- Given the distinct nature of our task and the peculiarities of the FER-2013 dataset, the model was fine-tuned on this specific dataset to ensure that the higher-level feature representations are well-suited for facial expression recognition.
- The network was trained with **fine-tuning**, where the initial layers were frozen to retain the learned features from ImageNet, and only the latter layers were trained to adapt to our specific classification task.
- This approach, leveraging a pre-trained model and tailoring it to our specific needs, offered a significant advantage in terms of training time and resources, while still ensuring high accuracy and efficiency in real-time classification.

GPU Acceleration Issue:

- **Challenge:** Despite having a GPU, **\`torch.cuda.is_available\`** initially returned False, indicating the inability to utilize GPU acceleration. This was due to outdated GPU drivers and CUDA version.
- **Solution:** We updated the GPU drivers and CUDA to version 12.3. This process involved reallocating disk space due to insufficient capacity on the C drive. After upgrading, PyTorch also needed to be updated. Due to complex dependencies in the conda environment, we opted to reinstall Anaconda and set up a new environment, successfully upgrading PyTorch to version 12.1. This change allowed **\`torch.cuda.is_available\`** to return True and increased the training speed nearly tenfold.

Differentiating Data Processing for Training and Evaluation:

- **Challenge:** Initially, the same data transformations were used for training, evaluation, and live prediction, including data augmentation steps, leading to poor prediction performance and instability in live prediction.

- **Solution:** We separated the data processing steps for training and evaluation/prediction by defining **`get_train_transforms`** and **`get_eval_transforms`**. This ensured that data augmentation was not used during evaluation and live prediction, improving the stability and accuracy of the model.

Real-Time Processing

The real-time processing of facial expressions is a cornerstone of this project, enabling the system to analyze and classify expressions as they occur. This section outlines the steps taken to capture and process images from a webcam for classification by our CNN.

Webcam Image Capture and Processing:

- **Image Capture:** We employ a webcam to capture live video feeds. Each frame of the video is treated as an individual image for processing and classification.
- **Preprocessing:** The captured frames are subjected to preprocessing to match the input requirements of the ResNet18 model. This involves resizing images to 224x224 pixels and converting them to grayscale, aligning with our training data's format.

Custom Modifications for ResNet Model:

- **Real-Time Adaptation:** The **ResNet34** model is adapted to work efficiently with real-time data. This includes optimizing the model for lower latency and ensuring it can process images quickly without significant lag.
- **Streamlined Processing:** Given the real-time nature of the application, the image processing pipeline is optimized to minimize delay, ensuring that the classification occurs almost instantaneously as the facial expressions are captured by the webcam.

Why ResNet34?

- **ResNet34** is part of the ResNet family, known for its ability to overcome the vanishing gradient problem, allowing deeper networks to be trained.
- It consists of **34 layers**, which is relatively lightweight compared to deeper versions like ResNet50 or ResNet101, hence suitable for our real-time processing requirements.
- Despite its fewer layers, ResNet34 has demonstrated remarkable performance in various vision tasks, providing a solid foundation for our facial expression classification task.

OpenCV Version Issue:

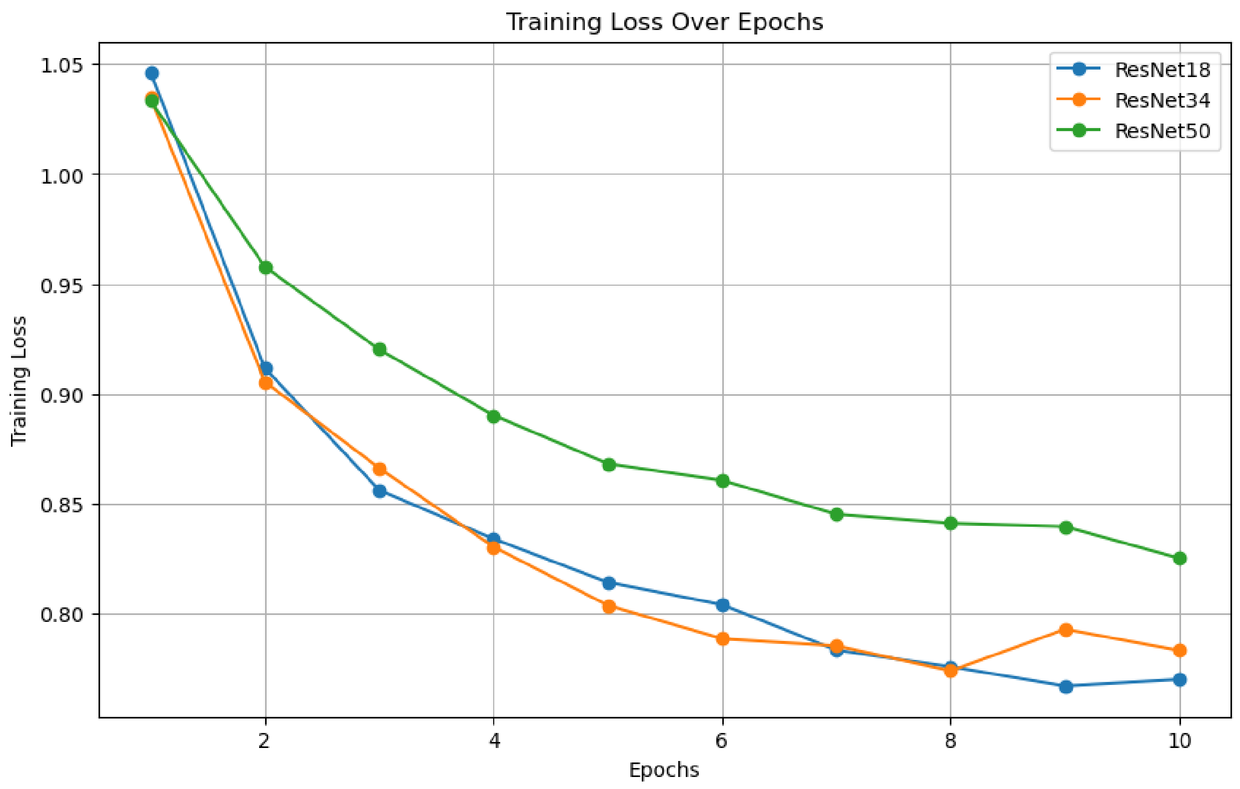
- **Challenge:** The initial version of OpenCV installed via **conda** (version 4.6.0) was outdated and lacked certain components, such as **`cv2.imshow`**.
- **Solution:** We installed the latest version of OpenCV using **`pip install opencv - python`** (version 4.8.1), resolving the compatibility issues.

By carefully crafting the image processing pipeline and adapting the CNN, we have developed a system capable of efficiently and accurately performing real-time facial expression classification.

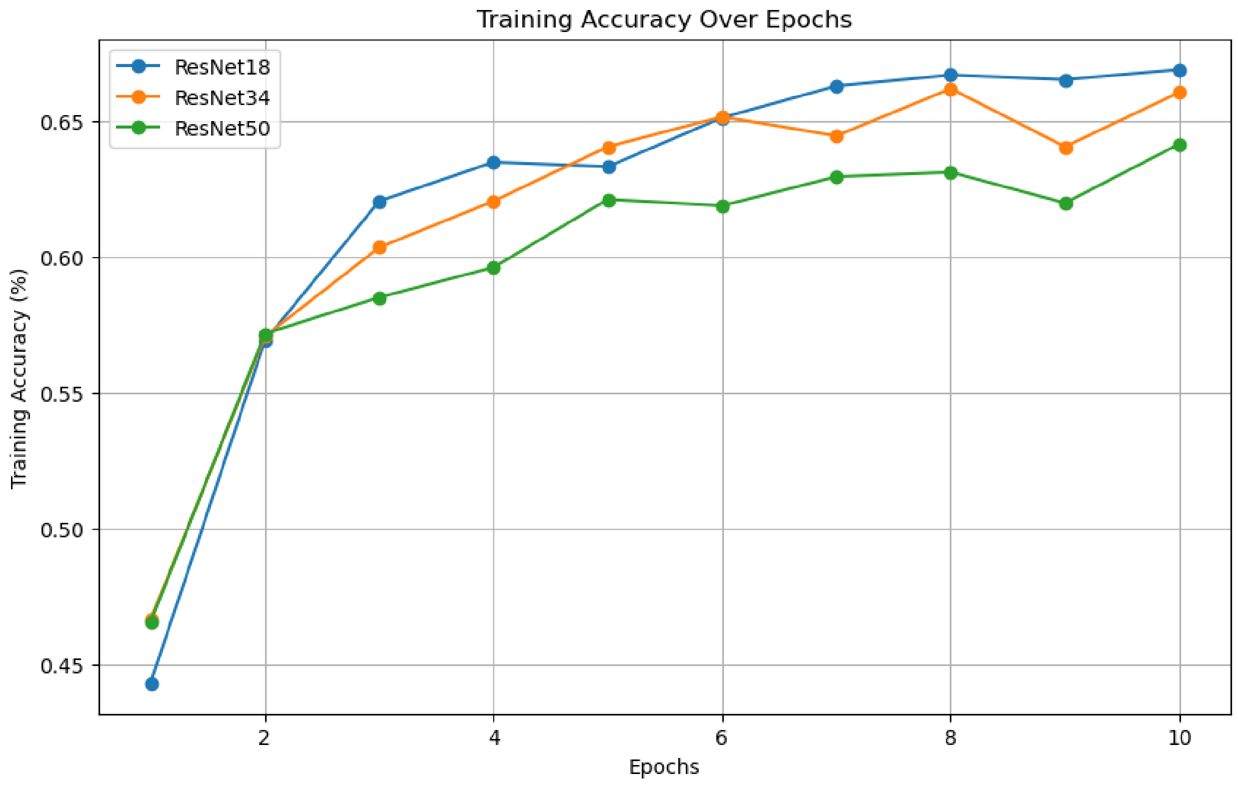
4. Experimental Setup and Results

This section details the comprehensive approach we took to evaluate our model, both during training and testing phases, and the metrics used to gauge its performance between different models.

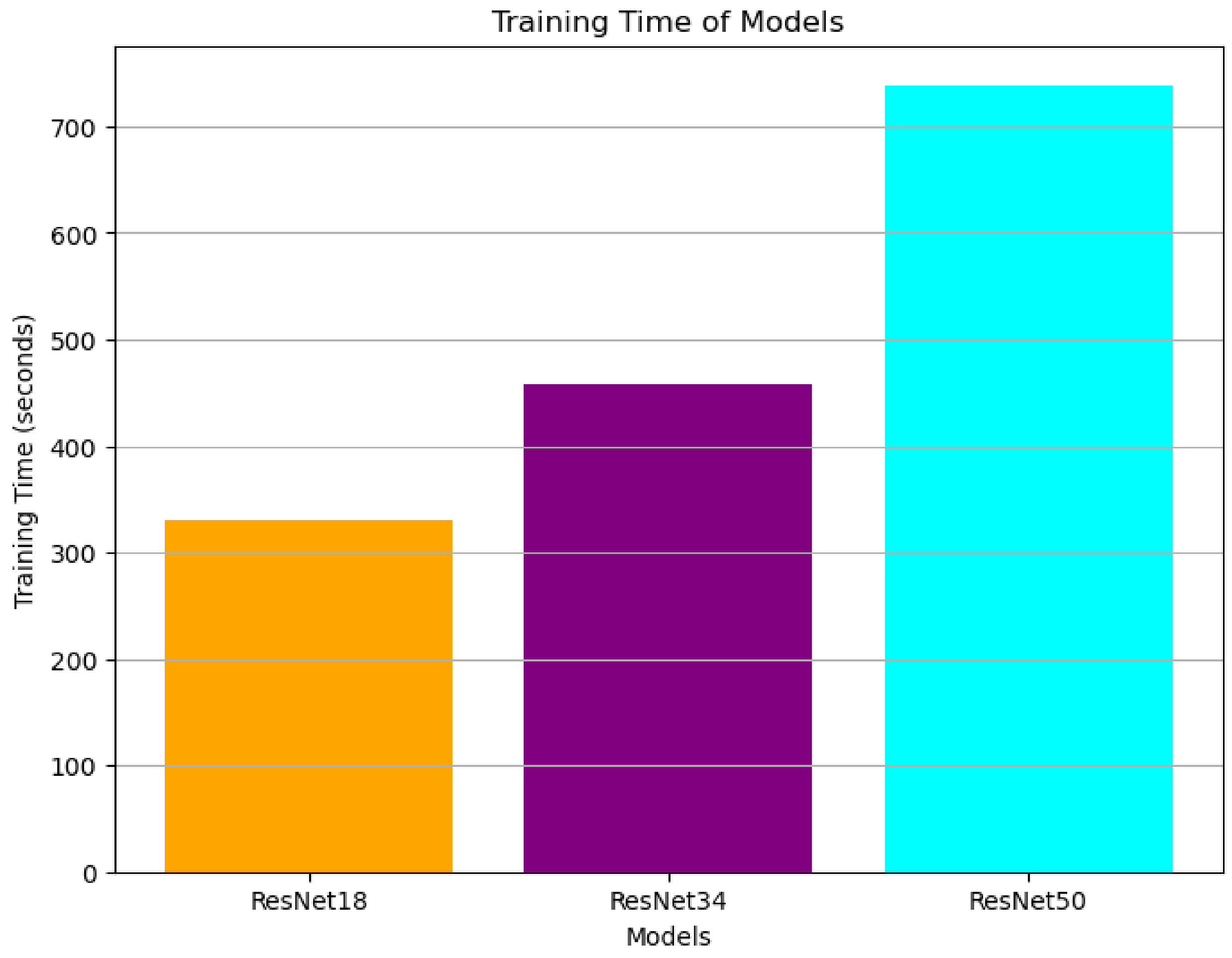
Training Performance:



The chart above displays the **training loss over epochs for each of the ResNet models** (ResNet18, ResNet34, and ResNet50). From this visualization, we can observe how the training loss evolves over the course of training for each model, providing valuable insights into their learning efficiency and behavior.



The chart above illustrates the **training accuracy over epochs** for the ResNet18, ResNet34, and ResNet50 models. It clearly shows how the accuracy of each model changes and improves as the training progresses, which is crucial for understanding the effectiveness of the training process for each model.



The bar chart above illustrates the **total training time for each of the ResNet models** (ResNet18, ResNet34, and ResNet50). This visualization is useful for comparing the computational efficiency and time required for training each model, which is a significant factor in real-time applications and overall system design considerations.

Testing Performance:

Following the training, the model was evaluated on a separate testing dataset to assess its generalization capabilities.

Evaluation Metrics:

To thoroughly assess our model's performance, we employed several key metrics, each offering unique insights into different aspects of the model's accuracy and efficiency.

1. Accuracy:

- **Formula:** Accuracy = (TP + FP) / (TP+FP+TN+FN)
- **Function:** This metric provides an overall measure of the model's ability to correctly classify images. It is a straightforward indicator of the model’s general effectiveness but doesn’t provide detailed insights into class-specific performance.

2. Precision:

- **Formula:** Precision = TP / (TP+FP)
- **Function:** Precision indicates the proportion of positive identifications that were actually correct. It is crucial for scenarios where the cost of false positives is high. High precision means that an algorithm returned substantially more relevant results than irrelevant ones.

3. Recall (Sensitivity):

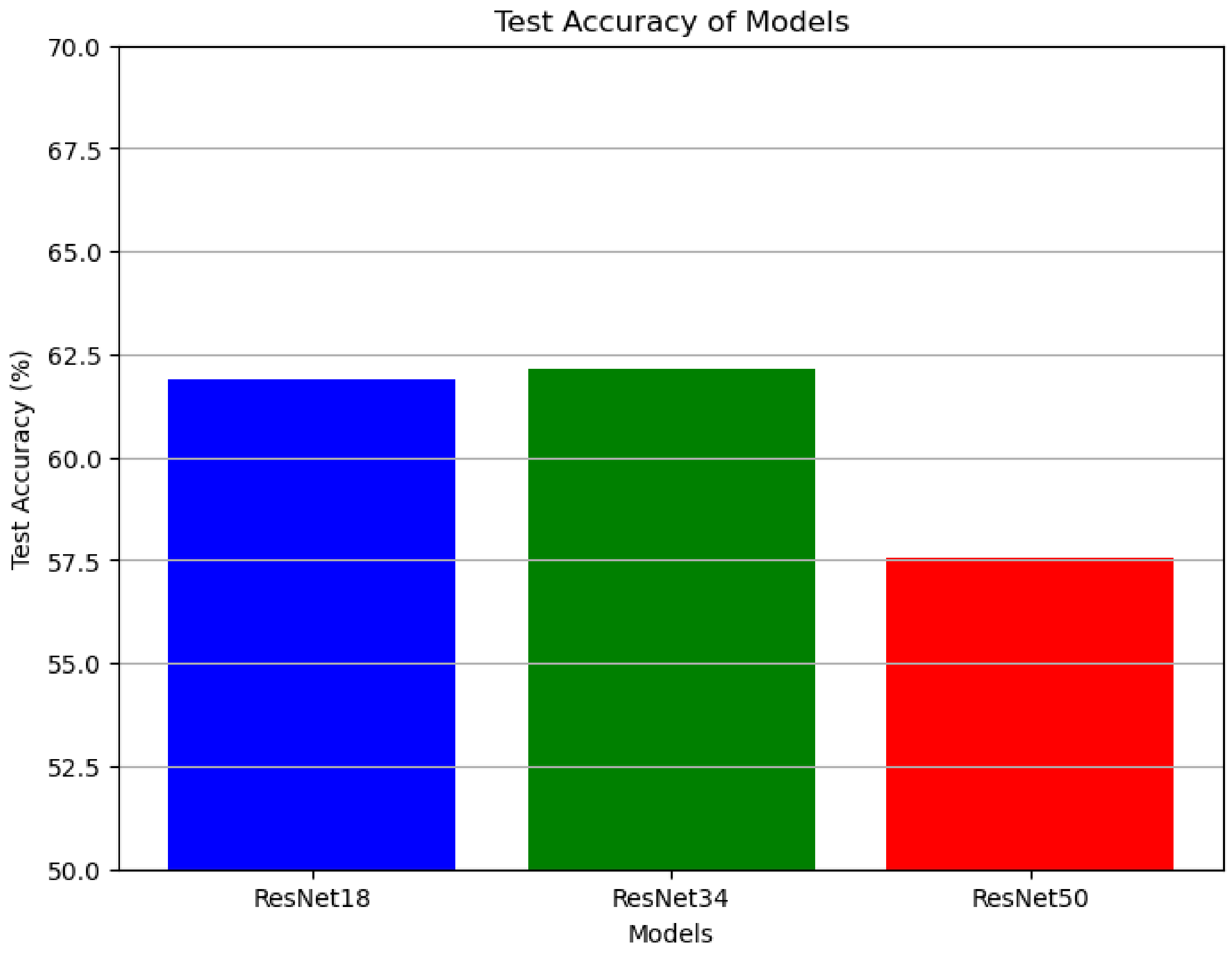
- **Formula:** Recall = TP / (TP+FN)
- **Function:** Recall assesses the model's ability to find all relevant cases within a dataset. It is particularly important in situations where missing a positive (such as failing to identify a correct facial expression) is more problematic than falsely identifying one.

4. F1 Score:

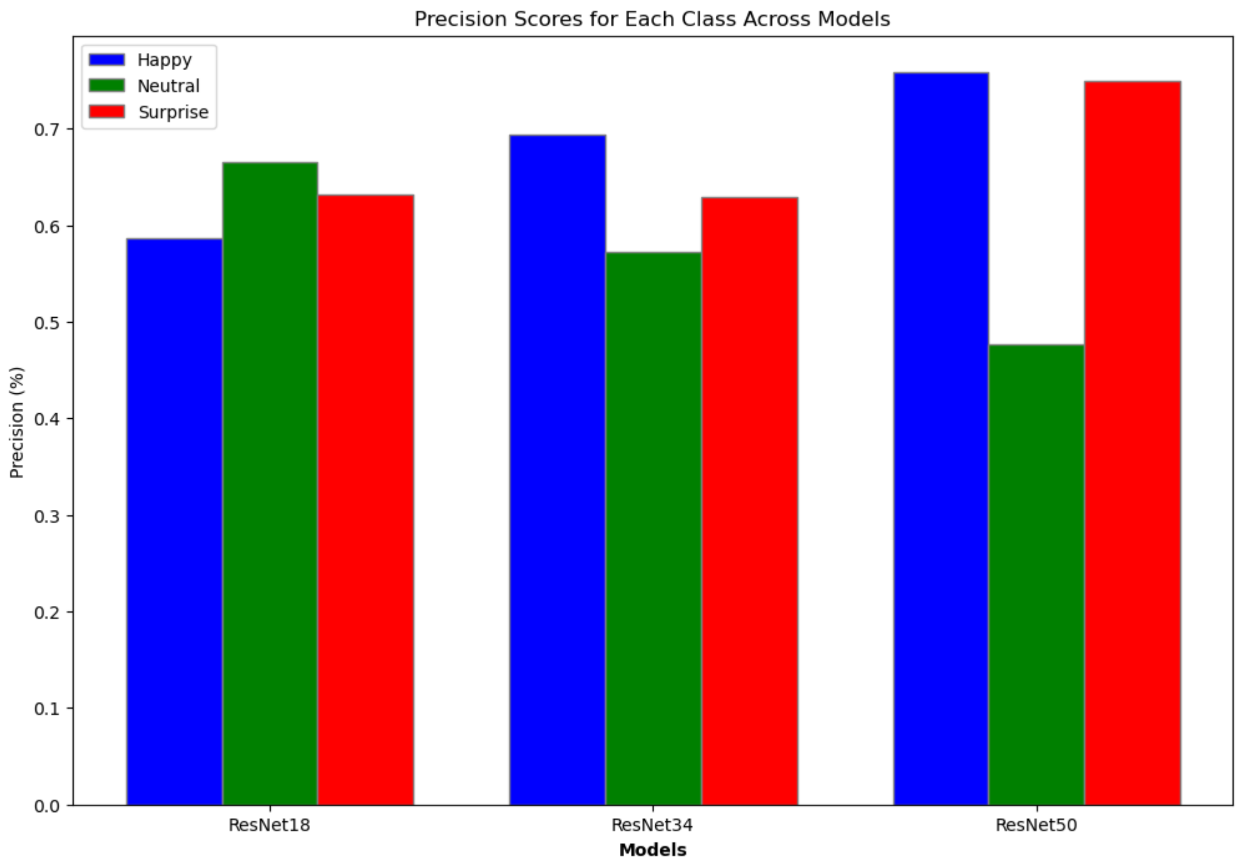
- **Formula:** F1 = 2 * (Precision*Recall) / (Precision+Recall)
- **Function:** The F1 Score is the harmonic mean of precision and recall, providing a balance between the two metrics. It is useful when you need to take both false positives and false negatives into account.

5. AUC Score:

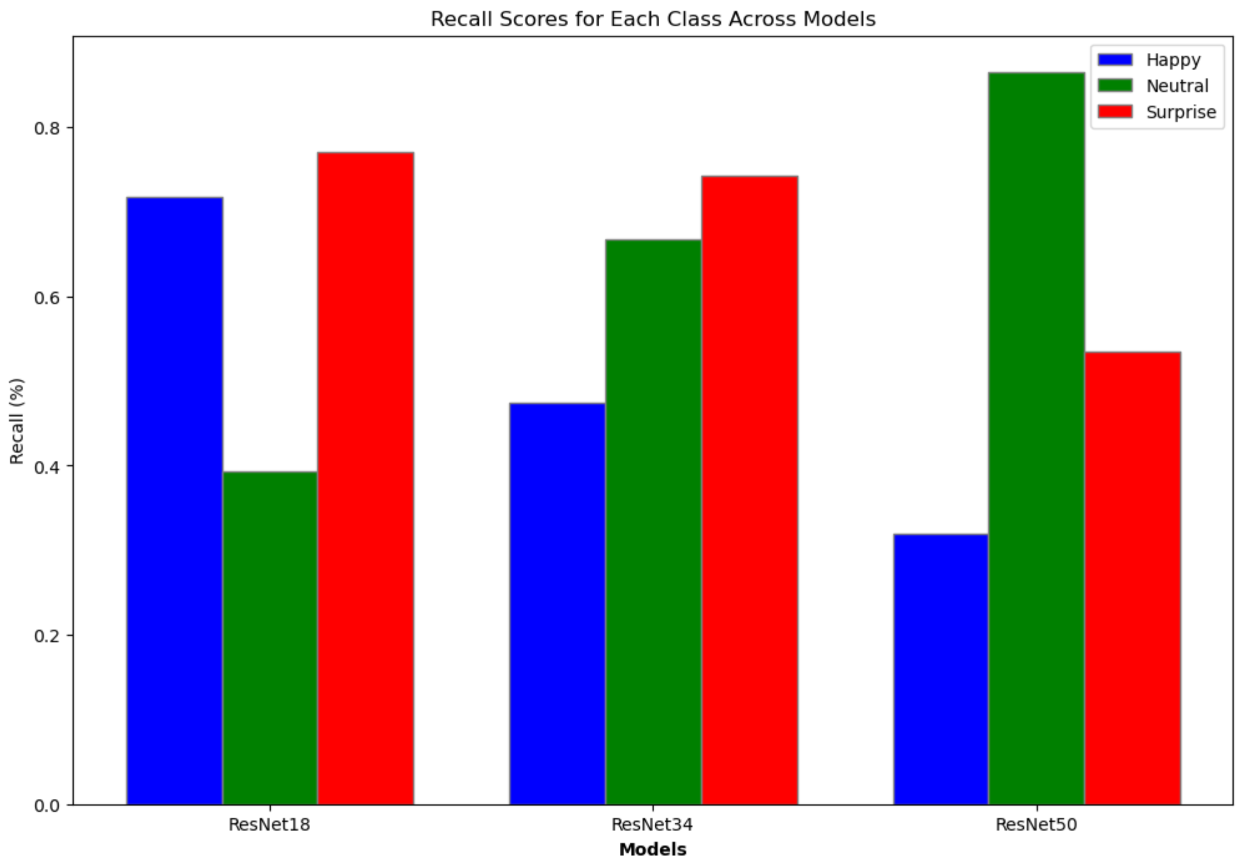
- **Formula:** TPR = TP / (TP+FN); FPR = FP / (FP+TN)
- **Function:** The AUC score provides a measure of a model's ability to distinguish between the positive and negative classes. A higher AUC score means that the model is better at correctly classifying positive and negative examples. In the context of multi-class classification, AUC is often calculated for each class against all others and then averaged to provide a single score.



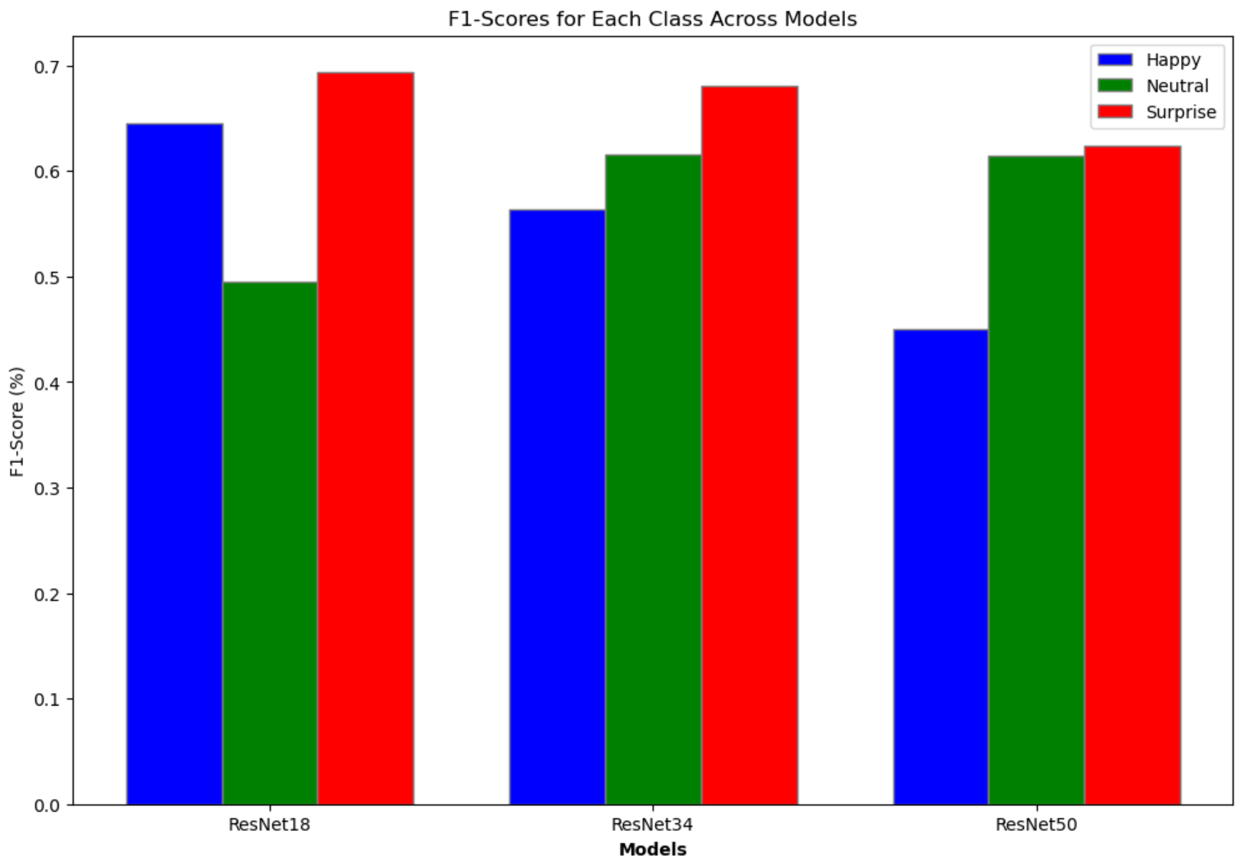
The bar chart above displays the **test accuracy for each of the ResNet models** (ResNet18, ResNet34, and ResNet50). This visualization offers a clear comparison of how each model performs on the test dataset, which is crucial for evaluating their overall effectiveness in facial expression detection.



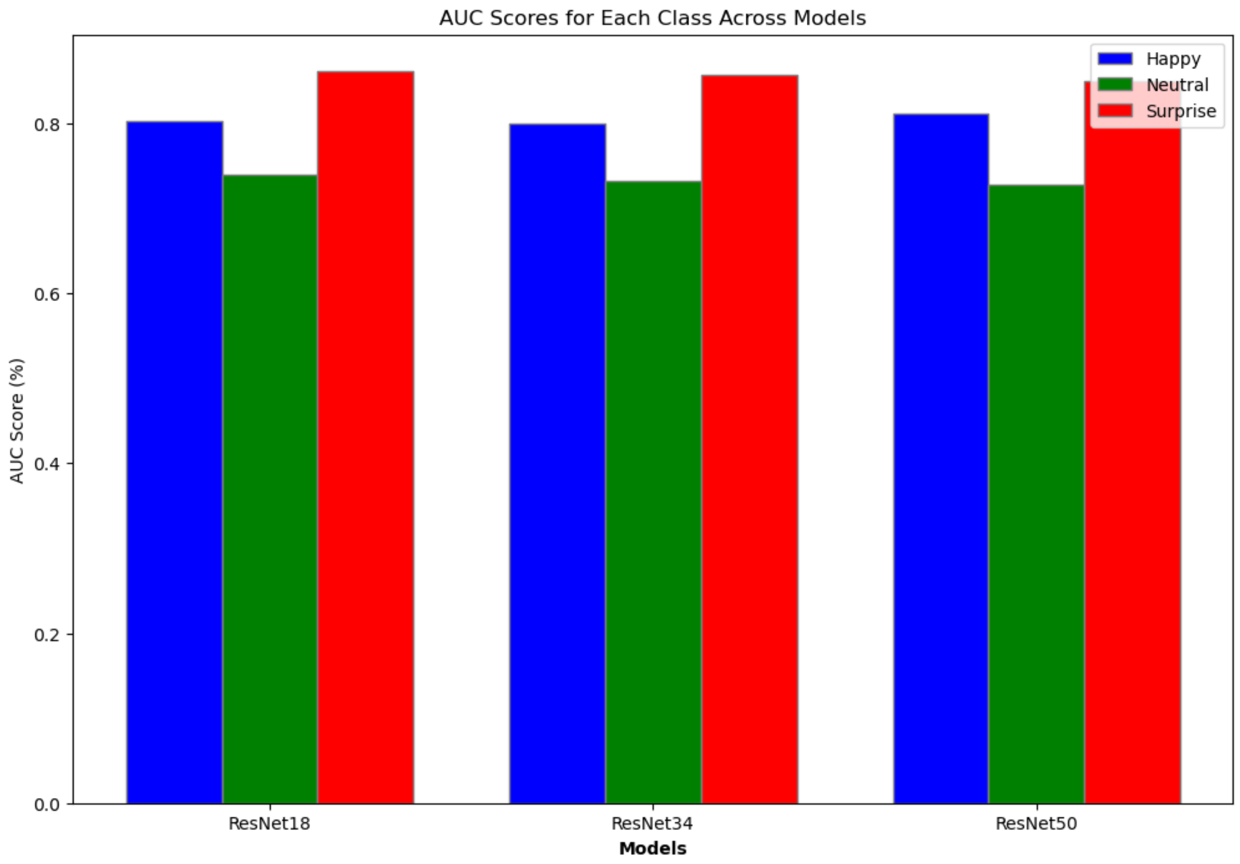
The chart above displays the **precision scores for each class** (Happy, Neutral, Surprise) **across the three ResNet models** (ResNet18, ResNet34, and ResNet50). This visualization offers a clear view of the precision, i.e., how accurately the models are identifying each class without falsely labeling other classes as that particular class.



The chart above shows the **recall scores for each class** (Happy, Neutral, Surprise) **across the ResNet models** (ResNet18, ResNet34, and ResNet50). This visualization allows for an easy comparison of how well each model is able to correctly identify each class.



The chart above displays the **F1-scores for each class** (Happy, Neutral, Surprise) **across the three ResNet models** (ResNet18, ResNet34, and ResNet50). The F1-score is a useful metric as it combines both precision and recall, providing a more holistic view of each model's performance for each class.



The chart above shows the **AUC (Area Under the Curve) scores for each class** (Happy, Neutral, Surprise) **across the three ResNet models** (ResNet18, ResNet34, and ResNet50). AUC is a key metric for evaluating the performance of a classification model, especially in cases with imbalanced classes. Higher AUC values indicate better model performance.

These results indicate that while the model shows a strong ability to recognize happy expressions, it experiences some challenges in identifying neutral expressions with the same level of accuracy. The performance in recognizing surprise expressions is also notable.

5. Discussion and Conclusion

In this project, we evaluated the performance of three different ResNet models (ResNet18, ResNet34, ResNet50) for real-time facial expression detection, specifically focusing on three expressions: happy, neutral, and surprise. Our comprehensive evaluation encompassed various aspects: training loss and accuracy, training time, testing accuracy, and detailed testing metrics including precision, recall, F1-score, and AUC score for each expression. Below is a discussion of our findings.

Training Loss and Accuracy

- **ResNet18 and ResNet34** showed a consistent decrease in training loss over epochs, indicating effective learning. **ResNet50** also demonstrated a decrease in loss, albeit with a slightly higher final loss value.
- In terms of training accuracy, **ResNet34 and ResNet18** showed comparable performance, with both models achieving over 66% accuracy. **ResNet50**, while slightly lower in final accuracy, still showed a significant improvement over epochs.

Training Time

- **ResNet50** took the longest training time, which is expected due to its deeper architecture. **ResNet34 and ResNet18** were more efficient, with **ResNet18** being the fastest. This highlights a trade-off between complexity and speed.

Testing Accuracy

- **ResNet34** achieved the highest test accuracy, closely followed by **ResNet18**. **ResNet50**, despite its complexity, did not outperform the other models in test accuracy. This suggests that additional complexity does not always translate to better performance, especially in a constrained dataset.

Precision, Recall, and F1-Score

- The precision, recall, and F1-scores varied across models and expressions.
- **ResNet34** showed balanced performance across all expressions, making it a reliable choice for diverse facial expression detection.
- **ResNet18** was particularly effective in detecting the 'Surprise' expression with high recall.
- **ResNet50** excelled in precision for 'Happy' and 'Neutral' but had lower recall scores, indicating a tendency to be more selective but less sensitive in classifying expressions.

AUC Score

- All models performed well in terms of AUC scores, with **ResNet50** slightly leading in the 'Happy' expression. High AUC scores across all models indicate good separability between classes.

Conclusion

In conclusion, each model has its strengths and areas of improvement:

- **ResNet34** stands out as the most balanced model, offering a good compromise between accuracy, precision, recall, and computational efficiency. It is suitable for real-time applications where diverse expression detection with balanced precision and recall is essential.
- **ResNet18**, with its faster training time and high recall for 'Surprise', is a good choice for applications prioritizing speed and specific expression detection.
- **ResNet50**, despite its complexity, did not show a proportional improvement in overall performance. However, its high precision in certain classes might make it suitable for scenarios where false positives are particularly costly.

Ultimately, the choice of the model should be guided by the specific requirements of the application, including the importance of accuracy versus speed, and the need for balanced performance across different expressions. In **Real-Time Processing**, we choose **ResNet34**.

6. Appendix

Metrics.json

```
{
    "resnet18": {
        "train_loss": [
            1.0464306029867618,
            0.9116407391872812,
            0.856094148564846,
            0.8339289908713483,
            0.8141320043421806,
            0.8039337130303078,
            0.7832099478295509,
            0.7756666997645764,
            0.7669280285530902,
            0.7699356269329152
        ],
        "train_accuracy": [
            0.44333333333333336,
            0.569,
            0.6203333333333333,
            0.6346666666666667,
            0.633,
            0.651,
            0.6626666666666666,
            0.6666666666666666,
            0.665,
            0.6686666666666666
        ],
        "train_time": 330.2731719017029,
        "test_accuracy": 61.88625927234193,
        "test_report": {
            "happy": {
                "precision": 0.5861224489795919,
                "recall": 0.718,
                "f1-score": 0.6453932584269664,
                "support": 1000
            },
            "neutral": {
                "precision": 0.6655405405405406,
                "recall": 0.394,
                "f1-score": 0.49497487437185933,
                "support": 1000
            },
            "surprise": {
                "precision": 0.631163708086785,
                "recall": 0.7701564380264742,
                "f1-score": 0.6937669376693767,
                "support": 831
            },
            "accuracy": 0.6188625927234193,
            "macro avg": {
                "precision": 0.6276088992023059,
                "recall": 0.6273854793421582,
                "f1-score": 0.6113783568227341,
                "support": 2831
            },
            "weighted avg": {
                "precision": 0.6273966905475983,
                "recall": 0.6188625927234193,
                "f1-score": 0.6064600699406845,
                "support": 2831
            }
        },
        "auc": {
            "happy": 0.8031630256690332,
            "neutral": 0.7395357728017478,
            "surprise": 0.861642900120337
        }
    },
    "resnet34": {
        "train_loss": [
            1.0350383634262896,
            0.9053734287302545,
            0.8661571817195162,
            0.830270344906665,
            0.8036319233001546,
            0.7884414043832333,
            0.7852023472177222,
            0.7737959876973578,
            0.7927450233317436,
            0.7830896960928085
        ],
        "train_accuracy": [
            0.4666666666666667,
            0.5706666666666667,
            0.6033333333333334,
            0.6203333333333333,
            0.6403333333333333,
        0.6513333333333333,
        0.6443333333333333,
        0.6616666666666666,
        0.6403333333333333,
        0.6603333333333333
    ],
    "train_time": 458.5292718410492,
    "test_accuracy": 62.13352172377252,
    "test_report": {
        "happy": {
            "precision": 0.6934306569343066,
            "recall": 0.475,
            "f1-score": 0.5637982195845698,
            "support": 1000
        },
        "neutral": {
            "precision": 0.5725321888412017,
            "recall": 0.667,
            "f1-score": 0.6161662817551963,
            "support": 1000
        },
        "surprise": {
            "precision": 0.6289500509683996,
            "recall": 0.7424789410348978,
            "f1-score": 0.6810154525386314,
            "support": 831
        },
        "accuracy": 0.6213352172377252,
        "macro avg": {
            "precision": 0.6316376322479693,
            "recall": 0.6281596470116325,
            "f1-score": 0.6203266512927992,
            "support": 2831
        },
        "weighted avg": {
            "precision": 0.6317980706924227,
            "recall": 0.6213352172377252,
            "f1-score": 0.6167037592367957,
            "support": 2831
        }
    },
    "auc": {
        "happy": 0.7994279082468596,
        "neutral": 0.73200327689787,
            "surprise": 0.8580255716004814
        }
    },
    "resnet50": {
        "train_loss": [
            1.0333963835493047,
            0.95777557504938,
            0.9201579436342767,
            0.8901786880290254,
            0.8681502392951478,
            0.8605391370489243,
            0.845118819399083,
            0.8410445464418289,
            0.8395177747340913,
            0.825047381380771
        ],
        "train_accuracy": [
            0.4656666666666667,
            0.5716666666666667,
            0.585,
            0.596,
            0.621,
            0.6186666666666667,
            0.6293333333333333,
            0.631,
            0.6196666666666667,
            0.6413333333333333
        ],
        "train_time": 738.843836069107,
        "test_accuracy": 57.576827975980216,
        "test_report": {
            "happy": {
                "precision": 0.7582938388625592,
                "recall": 0.32,
                "f1-score": 0.450070323488045,
                "support": 1000
            },
            "neutral": {
                "precision": 0.4765840220385675,
                "recall": 0.865,
                "f1-score": 0.6145648312611013,
                "support": 1000
            },
            "surprise": {
                "precision": 0.7491582491582491,
                "recall": 0.5354993983152828,
                "f1-score": 0.624561403508772,
                "support": 831
            },
            "accuracy": 0.5757682797598022,
            "macro avg": {
                "precision": 0.6613453700197919,
                "recall": 0.5734997994384275,
                "f1-score": 0.5630655194193062,
                "support": 2831
            },
            "weighted avg": {
                "precision": 0.6561032730313076,
                "recall": 0.5757682797598022,
                "f1-score": 0.5593944475679745,
                "support": 2831
            }
        },
        "auc": {
            "happy": 0.8117479519388312,
            "neutral": 0.7283315128345167,
            "surprise": 0.8494545728038507
        }
    }
}
```