Facial Expression Recognition
CMPE 258 – 02 Deep Learning

San Jose State University
Fall 2023
BUNPHENG CHHAY
SID: 014562782
Github: https://github.com/Bunphengchhay/GViT21k

Table of Content

1. Introduction

1.1 Objective

The primary objective of this project is to leverage the advanced capabilities of pre-trained models in facial expression recognition. By utilizing models from Hugging Face, a leading repository for state-of-the-art machine learning models, and Vision Transformer (ViT), this project aims to develop a robust system capable of accurately identifying and classifying a range of human facial expressions.

The focus is on fine-tuning these pre-existing, powerful models to adapt to the specific nuances of facial expression data. This approach is expected to enhance the model's performance in terms of accuracy, efficiency, and reliability in real-world applications.

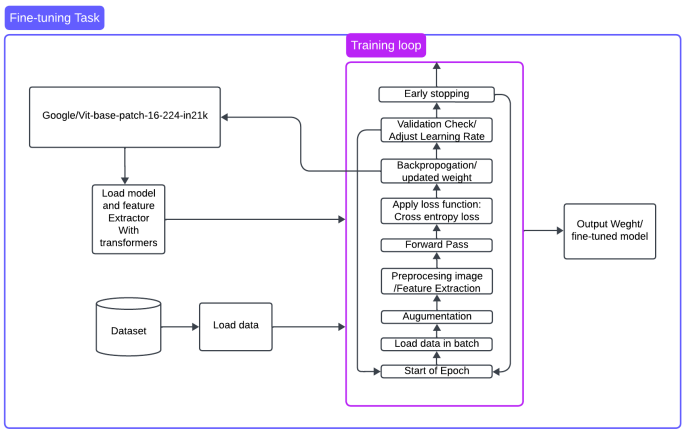
1.2 Motivation

The motivation behind this project stems from the increasing significance of automated facial expression recognition in various fields such as psychology, security, marketing, and human-computer interaction. Accurate recognition of facial expressions plays a crucial role in understanding human emotions, intentions, and behaviors.

Traditional methods often fall short in capturing the subtleties and complexities of human expressions. The advancement in deep learning, particularly in the field of computer vision, presents an opportunity to overcome these limitations. Pre-trained models, such as those offered by Hugging Face and Google's ViT, have shown remarkable success in image recognition tasks.

By fine-tuning these models for facial expression recognition, this project aims to harness their power to provide a more nuanced, accurate, and efficient solution for understanding human expressions. This not only contributes to the academic and research field but also has potential applications in enhancing user experience and engagement across various industries.

1.3 Proof of concept solution diagram



The fine-tuning process of a Vision Transformer (ViT) model, as depicted in the diagram, commences with the **Start of Epoch**, indicating the initiation of a training cycle. During each epoch, the **Load Data in Batch** step sequentially feeds batches of data into the model. This data undergoes **Augmentation**, a crucial step for enhancing model generalization by introducing

variability into the training dataset. Subsequently, **Preprocessing Image/Feature Extraction** is applied, standardizing the data format and extracting relevant features for model input.

As the preprocessed data moves through the **Forward Pass**, the ViT model processes it to generate predictions. These predictions are then evaluated against the true labels during the **Apply Loss Function** step, specifically utilizing Cross-Entropy Loss to quantify the prediction error. Following this, the **Backpropagation/Updated Weight** phase uses the calculated loss to adjust the model's weights through gradient descent, optimizing the model's parameters for improved accuracy.

Post-backpropagation, the model undergoes a **Validation Check** which doubles as an opportunity to **Adjust Learning Rate** based on the model's performance on the validation set. This serves as a mechanism for fine-tuning the learning process, possibly involving learning rate scheduling based on validation metrics. The **Early Stopping** criterion is then evaluated, providing a conditional halt to the training process if the model ceases to show improvement, thus preventing overfitting and resource waste.

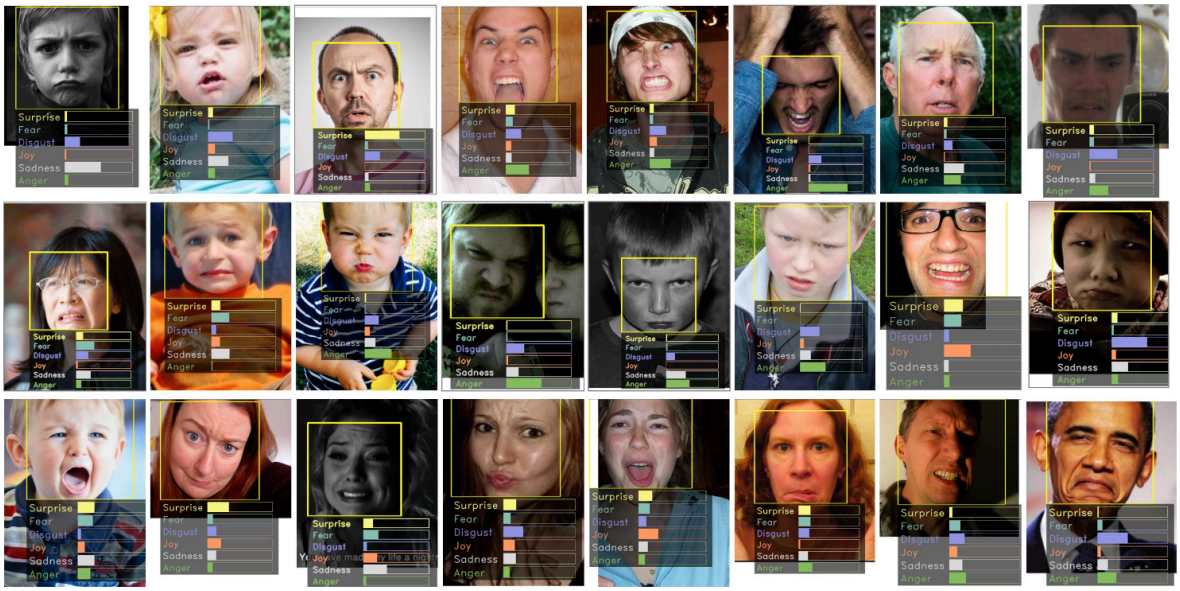
Finally, the culmination of these iterative and interconnected steps yields the **Output Weight/Fine-tuned Model**. This output represents the fine-tuned ViT model, optimized across several epochs, ready for deployment or further evaluation. Each step in this workflow is crucial and contributes to the overall efficacy of the model training, ensuring that the fine-tuned model is robust, accurate, and capable of generalizing well to new data.

2. Data collection

2.1 Data Source

The Real-world Affective Faces Database (RAF-DB) is a vast collection of facial expressions, boasting approximately 30,000 diverse facial images sourced from the internet. These images have been meticulously annotated by a group of around 40 individuals through crowdsourcing. The database exhibits significant variability in terms of subjects' age, gender, ethnicity, head orientations, lighting conditions, obstructions (such as eyeglasses, facial hair, or self-obscuring elements), as well as post-processing alterations (including various filters and special effects) (RAF-DB, 2023). RAF-DB offers a wealth of diversity, quantity, and comprehensive annotations, including (RAF-DB, 2023):

- A total of 29,672 real-world images.
- For each image, a 7-dimensional expression distribution vector.
- Two distinct subsets: a single-label subset, comprising 7 categories of basic emotions, and a two-tab subset, containing 12 categories of compound emotions.
- Additional annotations for each image, such as 5 precise landmark locations, 37 automated landmark positions, bounding box information, race, age range, and gender attributes.
- Baseline classifier outputs for both basic emotions and compound emotions.



To facilitate an objective evaluation of performance for subsequent users, the database has been partitioned into a training set and a test set. The training set is five times larger than the test set, and the distribution of expressions in both sets closely resembles each other.

2.2 Emotion Labels

The Emotion_Labels dictionary is a data structure that associates numerical labels with specific human emotional states. This mapping allows for a more structured and standardized representation of emotions in various applications, such as psychology, human-computer interaction, and machine learning. Each numerical key in the dictionary corresponds to a particular emotion, and the associated value provides the name or label for that emotion. Here's a breakdown of the emotions represented in the Emotion_Labels dictionary:

1. **Surprise:** This emotion is typically characterized by a sudden and unexpected reaction to a surprising or extraordinary event. It often involves widened eyes and raised eyebrows.
2. **Fear:** Fear is the emotional response to a perceived threat or danger. It can trigger a "fight or flight" reaction and is associated with feelings of anxiety and unease.
3. **Disgust**: Disgust is a strong aversion or revulsion towards something unpleasant or offensive. It often manifests as a facial expression of wrinkled nose and a turned-up mouth.
4. **Happiness:** Joy represents a state of happiness, delight, or contentment. It is associated with positive feelings and often includes smiling and laughter.
5. **Sadness:** Sadness is the emotional response to loss, disappointment, or sorrow. It can manifest with facial expressions such as a downturned mouth and teary eyes.
6. **Anger:** Anger is a strong emotional reaction to a perceived injustice, frustration, or annoyance. It often includes facial expressions like clenched fists and a furrowed brow.
7. **Neutral:** The "Neutral" category represents a lack of strong emotional expression. It is used when a person's facial expression doesn't clearly convey any of the other specific emotions.

In various applications, including facial recognition technology, emotion detection, and sentiment analysis, this Emotion_Labels dictionary can be used as a reference guide to interpret and classify human emotional states based on numerical labels, making it a valuable tool for understanding and responding to human emotions in a standardized way.

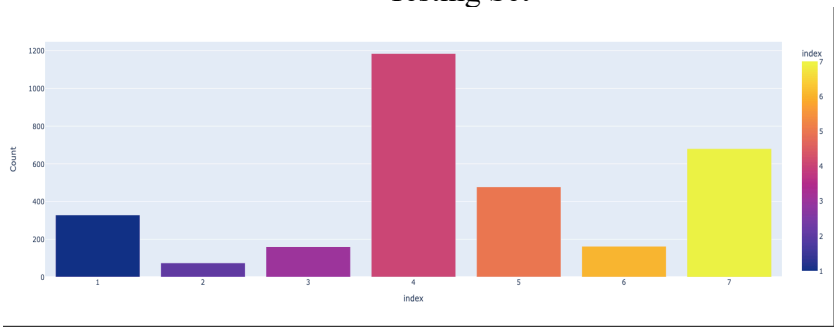
2.3 Dataset

In the context of our research, the dataset has been meticulously structured and organized within the "RAFDB/DATASET" directory. Within this primary directory, we have categorized our data into various subfolders, each representing a distinct subset of the dataset. Notably, we have segregated the data into "test" and "train" folders, following a common practice in machine learning to facilitate model evaluation. Additionally, for each of these subsets, we have included corresponding "tes.csv" and "train.csv" files.

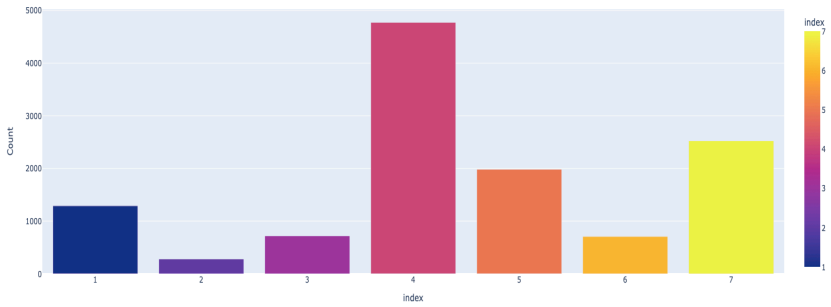
These label files are instrumental in annotating and providing crucial information about the images in their respective sets. Furthermore, to introduce class-specific distinctions, we have used subfolders labeled from 1 to 7 within both the "test" and "train" directories. This hierarchical organization is pivotal in effectively managing our dataset, enabling a clear division between training and testing data, and ensuring that our machine learning models can make informed predictions while maintaining data integrity.

```
RAFDB/
|
└── DATASET/
    |
    ├── Train/
    |   ├── 1/
    |   ├── 2/
    |   ├── ...
    |   └── 7/
    |
    ├── Test/
    |   ├── 1/
    |   ├── 2/
    |   ├── ...
    |   └── 7/
    |
    ├── train.csv
    └── test.csv
```

Testing Set



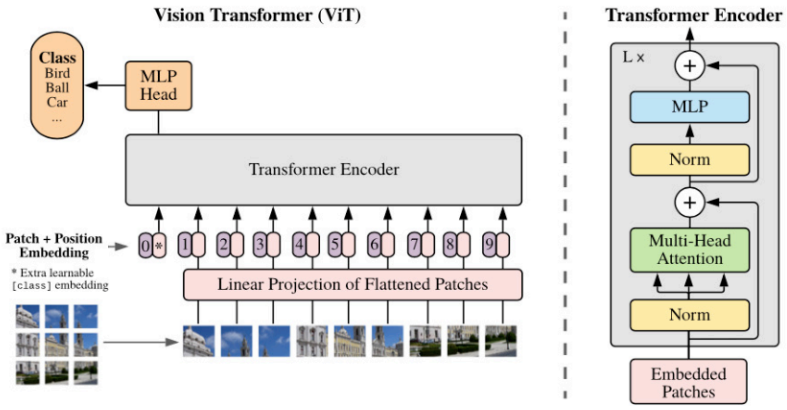
Training Set



3. Model Architecture

3.1 Base Model

In a similar fashion to how transformer-based models brought about a paradigm shift in the field of Natural Language Processing (NLP), we are witnessing a proliferation of research papers applying these models across various domains. One of the most groundbreaking developments in this regard is the introduction of the Vision Transformer (ViT) in June 2021 by a team of researchers at Google Brain. Below is the ViT architecture (Huggingface, 2023).



The ViT paper delves into a fascinating concept: the transformation of images into a format akin to sentences, thus enabling their utilization with transformer models during the training process. This idea, while seemingly straightforward, entails the following steps (Huggingface, 2023):

1. Dividing an image into a grid of sub-image patches.
2. Employing linear projection to embed each of these patches.
3. Treating each embedded patch as a token, thus creating a sequence of embedded patches that can be input to the model.

In essence, this innovative approach extends the applicability of transformer models beyond text, ushering in a new era in computer vision and image processing.

3.2 Model google/vit-base-patch16-224-in21k

The Vision Transformer (ViT) model has undergone pretraining on the extensive ImageNet-21k dataset, encompassing a remarkable 14 million images categorized into 21,843 unique classes. This model was originally introduced in the research paper "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale" authored by Dosovitskiy et al., and it initially became available within their repository. Notably, the conversion of the model's weights to PyTorch was accomplished by Ross Wightman, who had previously taken on the task of transitioning them from JAX to PyTorch (Huggingface, 2023) It is imperative to acknowledge his significant contributions in this transformation process.

The Vision Transformer (ViT) stands as a transformer model pretrained on a substantial image dataset, ImageNet-21k, at a resolution of 224x224 pixels. In its approach, ViT views images as sequences composed of fixed-size patches, each measuring 16x16 pixels. It incorporates a distinct [CLS] token tailored for classification tasks. While ViT does not include fine-tuned heads, it provides a pretrained pooler designed for downstream applications like image classification (Huggingface, 2023). Through pretraining, ViT acquires the ability to learn representations of images, which can be applied to diverse tasks by simply adding a linear layer atop the [CLS] token, representing the entire image (Huggingface, 2023).

4. Training

Fine-tuning a pre-trained Vision Transformer (ViT) model involves a series of systematic steps designed to adapt the model to a specific task. This process is essential to leverage the learned representations from large datasets and transfer that knowledge to a new domain, often with a smaller dataset.

4.1 Start of Training Epochs

The training loop commences with an epoch, a complete pass through the entire training dataset. Each epoch consists of several iterations, where an iteration is one forward and backward pass with a subset of the data, known as a batch.

4.2 Batch Loading and Data Augmentation

For each iteration, data is loaded in batches to maintain a balance between computational efficiency and memory constraints. Data augmentation is applied to this batched data, which involves random transformations like rotations, flips, or color jittering. These augmentations prevent overfitting by artificially expanding the dataset with varied examples, encouraging the model to learn more robust features.

Data transformation plays a pivotal role in preparing images for effective model input and improving the robustness of the Vision Transformer (ViT) model. We have employed specific data transformations for both training and testing datasets:

- **Training Data Transformation:** The training data undergo a series of augmentations to enhance the model's ability to generalize and recognize patterns effectively. These transformations include:
    - **Random Horizontal Flip:** Randomly flipping images horizontally, which diversifies the training data and helps the model become invariant to horizontal orientation variations.
    - **Random Rotation:** Applying random rotations (up to 10 degrees) to the images, introducing variations in the dataset to make the model more resilient to rotated objects.
    - **Random Resized Crop:** Randomly cropping and resizing images to a standard size of 224x224 pixels. This operation introduces spatial variability while maintaining a consistent input size.
    - **Color Jittering:** Injecting slight randomness in image color attributes, including brightness, contrast, saturation, and hue. This augmentation technique adds robustness to color variations in real-world images.
- **Testing Data Transformation:** In contrast to the training data, the testing data undergo minimal transformations to ensure that the model's predictions align with real-world scenarios. The testing transformations include:
    - **Resize:** Resizing the images to a uniform size of 224x224 pixels. This standardization ensures consistency in model input.
    - **To Tensor:** Converting the processed images into PyTorch tensors, making them suitable for feeding into the ViT model.
    - **Normalization:** Applying normalization to the tensor values using mean and standard deviation values. This step ensures that the input data adheres to the model's expected data distribution.

These carefully designed data transformations not only facilitate effective model training but also enable the Vision Transformer to make accurate predictions on real-world images during inference. The combination of training augmentations and testing standardization helps the ViT model handle diverse and complex image data, making it a valuable tool in computer vision applications.

4.3 **Preprocessing and Feature Extraction**

Following augmentation, the data is preprocessed, which aligns the input to the model's expected format. This typically includes resizing the images, converting them to tensors, and normalizing them against a standard set of statistics (mean and standard deviation). Feature extraction then occurs implicitly during the forward pass through the pre-trained layers of the ViT model, where the model leverages its existing weights to generate predictive features from the input data.

- **dataset Directory Setup**

First, define the directory paths where your training and testing datasets are located. In your code, you've specified these paths as **TRAIN_DIR** and **TEST_DIR**. These

directories should contain the images organized into subfolders, with each subfolder representing a specific class or label.

- **Custom Dataset Class**

To efficiently handle the dataset, created a custom dataset class named **CustomDataset**. This class inherits from PyTorch's **Dataset** class and is responsible for loading and processing the image data along with their corresponding labels.

- In the class constructor (**__init__** method), it provides the path to the dataset (**images_folder**), the labels file (**labels_file**), and an optional data transformation (**transform**).
- load the image labels from the CSV file specified in **labels_file**. These labels are a pair of image filenames and their associated class labels.
- The **__len__** method returns the total number of images in the dataset.
- The **__getitem__** method loads an image and its label given an index (**idx**). It constructs the image path, opens the image using PIL (Python Imaging Library), and applies the specified transformation (if provided).

- **Data Loaders:** Defined data loaders for both training and testing datasets using **DataLoader** from PyTorch. Data loaders allow the function to efficiently load and iterate through batches of data during training and evaluation. Created data loaders using the **CustomDataset** instances, specifying the batch size (**BATCH_SIZE**) and whether to shuffle the data.
- **Feature Extraction**

Feature extraction is primarily performed using the Hugging Face Transformers library, which provides the necessary tools to work with the ViT model.

- **Model Loading**

Load the ViT model using Hugging Face Transformers. Utilize the ViTFeatureExtractor and ViTForImageClassification classes.

- ViTFeatureExtractor: This class is responsible for preprocessing images before they are fed into the model. Loading with ViTFeatureExtractor.from_pretrained(). It handles tasks like resizing, normalizing, and converting images to suitable inputs for the model.
- ViTForImageClassification: This class represents the ViT model architecture pretrained for image classification. Loading it using ViTForImageClassification.from_pretrained(). It includes the model's encoder, which is used for feature extraction.

- **Applying Feature Extraction**

Once the ViT model and feature extractor are loaded, feature extraction is applied to input images as follows:

- Open and load an image using PIL.
- The image is preprocessed using the **ViTFeatureExtractor**, which includes resizing, normalization, and converting to a format suitable for the model input.
- The preprocessed image is then passed through the ViT model's encoder, resulting in feature embeddings.

The feature embeddings obtained from the model can be used for various downstream tasks, such as image classification or feature extraction for further analysis.

4.4 Forward Pass and Loss Computation

The forward pass involves feeding the preprocessed data through the model to obtain logits, the raw prediction outputs. These logits are then compared to the true labels using a loss function, with cross-entropy loss being the standard for classification tasks. The loss function quantifies the error between the predictions and the actual labels, providing a scalar value that the training loop aims to minimize.

4.5 Backpropagation and Weight Update

In the subsequent backpropagation step, the gradients of the loss function with respect to the model parameters are computed. This gradient information is used to update the model's weights. The update is performed by an optimizer, such as Adam, which adjusts the weights in the direction that reduces the loss, with the learning rate dictating the size of the step taken in this weight space.

4.6 Learning Rate Adjustment

The learning rate is a hyperparameter that controls the extent to which the weights are updated during training. An optimal learning rate is crucial as it ensures convergence to a good set of weights without overshooting or getting stuck in local minima. Learning rate schedules or policies may be employed to adjust the learning rate throughout training. For instance, a common approach is to reduce the learning rate by a factor (e.g., by 10%) after a certain number of epochs have elapsed without improvement in validation loss.

4.7 Validation and Early Stopping

At the end of each epoch, a validation check is performed using a separate dataset not seen during training. The validation performance gives an unbiased estimate of model accuracy. To prevent overfitting, early stopping is employed, which monitors the validation loss and stops training if there hasn't been an improvement for a number of epochs, defined by the 'patience' parameter. For example, setting **patience=5** would mean training is halted if there is no decrease in validation loss after five consecutive epochs.

4.8 Output of the Fine-Tuned Model

Upon completion of the training loop or triggering of early stopping, the fine-tuned model weights are output. These weights have been optimized to perform well on both the

training data and the unseen validation data, ensuring that the model not only learns the training data patterns but also generalizes to new, unseen examples.

The fine-tuning process for a ViT model involves careful orchestration of data handling, augmentation, feature extraction, optimization, and regularization strategies. The training loop is the crucible where the model's predictive capabilities are refined and honed for the specific task at hand, resulting in a fine-tuned model ready for evaluation and deployment.

5. Testing and Evaluation

In the realm of machine learning, the classification of emotions represents a sophisticated task that bridges the gap between computational algorithms and human psychology. The performance of such a classification model is crucial, as it dictates the accuracy and reliability of emotion recognition in various applications, from customer feedback analysis to mental health assessment. This report evaluates the performance of an emotion classification model, dissecting its precision, recall, f1-score, and support across seven distinct emotions.

	precision	recall	f1-score	support
Surprise	0.87	0.86	0.86	329
Fear	0.66	0.54	0.59	74
Disgust	0.57	0.61	0.59	160
Happiness	0.95	0.93	0.94	1185
Sadness	0.80	0.87	0.83	478
Anger	0.80	0.78	0.79	162
Neutral	0.82	0.81	0.82	680
accuracy			0.85	3068
macro avg	0.78	0.77	0.78	3068
weighted avg	0.85	0.85	0.85	3068

Model Performance Overview

Upon inspection of the classification report, the model demonstrates a commendable overall accuracy of 85%, which indicates its effectiveness in emotion recognition tasks. However, a granular analysis reveals nuanced insights into its performance across different emotions. Precision, recall, and f1-score serve as the key metrics, offering a three-dimensional view of the model's predictive capabilities.

High Precision and Recall

The model exhibits superior performance in identifying 'Happiness' with precision, recall, and f1-score metrics all surpassing the 0.90 mark, suggesting an excellent rate of correct predictions and minimal false negatives. Similarly, 'Neutral' emotions are detected with high reliability, with metrics consistently above 0.80. Such robustness in detecting 'Happiness' and 'Neutral' emotions underscores the model's strengths and its potential utility in environments where these emotions are particularly pertinent.

Challenges in Emotion Recognition

Conversely, the emotions 'Fear' and 'Disgust' display lower f1-scores of 0.59, signaling a relative difficulty for the model to discern these emotions accurately. This could be attributed to the nuanced and subtle expressions of 'Fear' and 'Disgust' or an imbalance in the training dataset. The report points to a critical need for model refinement in these areas, possibly by introducing a more diverse and representative dataset or by fine-tuning the model's parameters to enhance its sensitivity to these specific emotions.

Balanced versus Weighted Metrics

The macro average score, which treats each class equally, stands at 0.78, while the weighted average, which accounts for class imbalance by considering the support, is noted at 0.85. The close proximity of these scores is indicative of a well-rounded model performance, yet it subtly hints at the disparity caused by unequal class distributions. It stresses the importance of a balanced dataset for training to achieve unbiased and uniform model performance across all emotions.

Quantitative Insights

The 'support' metric quantifies the number of instances for each emotion within the dataset, which reveals a skewed distribution. 'Happiness' dominates with 1185 instances, potentially contributing to the higher metric scores due to more substantial training data. In contrast, 'Fear' has the least representation with merely 74 instances, which may lead to the observed underperformance for this class.

Technical Considerations and Environment Setup

The report also surfaces technical warnings related to the infrastructure setup, such as conflicts in registering CUDA components and the absence of TensorRT. While these do not directly impact the classification metrics, they may influence the model's operational efficiency. Optimizing the computational environment, including the resolution of these warnings, is essential for real-time applications and large-scale deployments.

6. Conclusion

The fine-tuning of the Vision Transformer (ViT) model for the emotion recognition task demonstrates the efficacy of transfer learning in leveraging pre-trained models for domain-specific applications. The fine-tuning process, detailed in our methodology, carefully adjusted the pre-existing weights of a ViT model to align with the idiosyncrasies of our emotion-labeled dataset.

The evaluation results underscore the success of this fine-tuning task. The model achieved a commendable overall accuracy of 85%, a testament to its ability to generalize well to new data. Notably, the model performed exceptionally well in recognizing 'Happiness' with a precision of 0.95, recall of 0.93, and an F1-score of 0.94, likely due to the higher representation of this emotion in the dataset, as indicated by the support values.

The precision and recall values across different emotions such as 'Surprise', 'Sadness', and 'Neutral' further illustrate the model's robustness, with scores consistently above 0.80. However, certain categories like 'Fear' and 'Disgust' present opportunities for improvement, as reflected by their relatively lower scores in precision and recall. These discrepancies could be attributed to the underrepresentation of these classes in the training data or the inherent difficulty in distinguishing these complex emotions.

The macro and weighted averages provide a holistic view of the model's performance across classes, accounting for the imbalance in the dataset. The macro average of 0.78 for precision, recall, and F1-score indicates that the model, on average, performs uniformly across all classes. In contrast, the weighted average considers the support for each class, highlighting the model's effectiveness on the more frequently represented emotions.

In conclusion, the fine-tuned ViT model presents a compelling case for the application of advanced deep learning architectures in emotion recognition tasks. The results obtained lay a solid foundation for future work, which could explore strategies such as oversampling minority classes, experimenting with more sophisticated data augmentation techniques, or incorporating additional contextual information to further enhance the model's performance, particularly in less represented emotion categories.

The journey from a generalist model pre-trained on large-scale datasets to a specialist in emotion recognition encapsulates the transformative power of fine-tuning, enabling the model to distill vast, pre-learned visual knowledge into nuanced, task-specific insights.

Bibliography

Huggingface. (2023, December 9). *fine-tune-vit*. Retrieved from https://huggingface.co: https://huggingface.co/blog/fine-tune-vit

RAF-DB. (2023, Dec 9). *model1*. Retrieved from http://www.whdeng.cn/raf/: http://www.whdeng.cn/raf/model1.html