Facial Expression Recognition (FER) with deep learning approaches

Group 14: Dang Kieu Trinh 20214933, Vu Duc Hung 20214902, Luyen Minh Khanh 20214905

Keywords:
Deep Learning, Facial Expression Recognition, ResNet, GoogleNet, Vision Transformers, Convolutional Neural Networks.

Abstract This report presents a study on facial expression recognition using deep learning methods. We leverage state-of-the-art Convolutional Neural Networks (CNNs) and Vision Transformers (ViT) to build our models. Specifically, we employ ResNet, GoogleNet and ViT, each offering unique architectural advantages. Our study provides a comparative analysis of these models, offering insights into their performance in the context of facial expression recognition.

© The Author(s) 2024. Submitted: 31 December 2023 as the Project report.

1. Introduction

Facial Expression Recognition (FER) is a rapidly growing field within the realm of computer vision and artificial intelligence. It involves the identification of human emotions through the analysis of facial features such as eyebrows, eyes, mouth, and other facial characteristics. The ability to accurately recognize facial expressions has numerous applications, including but not limited to, understanding human behavior, detecting mental disorders, and creating synthetic human expressions.

In the era of digital communication, the importance of FER has grown exponentially. It has found its place in various sectors such as healthcare, where it can be used to monitor patient responses to treatment; in entertainment, where it can be used to create more realistic and immersive experiences; and in security, where it can be used for identity verification and lie detection.

Despite the significant advancements in this field, achieving a high recognition rate remains a challenging task. The complexity of human emotions, variability in facial expressions across different individuals, and the subtlety of some expressions make this a non-trivial task. Furthermore, the performance of FER systems can be influenced by factors such as lighting conditions, face orientation, and occlusions, which add to the complexity of the problem.

Deep learning, a subset of machine learning, has shown great promise in tackling complex problems such as FER. Deep learning models, specifically Convolutional Neural Networks (CNNs) and Vision Transformers (ViT), have been successful in various computer vision tasks due to their ability to learn hierarchical features from raw data. This project aims to explore the application of these models, specifically ResNet, GoogleNet and ViT, in the task of facial expression recognition.

This report presents a comprehensive study on the application of these models in FER, providing a comparative analysis of their performance and offering insights into their effectiveness in the context of facial expression recognition.

1.1 Problem Statement

The automation of facial expression recognition (FER) poses a significant challenge in the field of computer vision, given the intricate nature of human emotions and the diverse range of expressions that can be exhibited. The task is further complicated by the inherent variability in facial expressions among different individuals, influenced by cultural, social, and personal factors. Additionally, external conditions such as inconsistent lighting, variations in face orientation, and potential occlusions can severely impact the accuracy of FER systems. These challenges necessitate the development of robust algorithms capable of interpreting the nuanced patterns of facial expressions with high precision and reliability.

1.2 Objective

The primary objective of this project is to conduct an in-depth exploration and evaluation of several advanced deep learning architectures for their efficacy in facial expression recognition. We aim to systematically assess the performance of ResNet, GoogleNet and Vision Transformers (ViT) in recognizing and classifying a range of facial expressions. Each of these models brings a unique set of architectural features and learning capabilities that have been proven successful in various image recognition tasks. Through rigorous experimentation and analysis, we intend to determine the strengths and limitations of each model when applied to the FER domain. The ultimate goal is to identify the most suitable model or combination of models that can consistently achieve high accuracy in FER, thereby pushing the boundaries of what is achievable with current technology and providing valuable insights for future research and applications.

2. Architecture and method

2.1 Convolutional neural network (CNN)

Convolutional Neural Networks (CNNs) have revolutionized the field of computer vision, providing state-of-the-art solutions for many image classification problems. CNNs are a class of deep learning models that are primarily used for analyzing visual imagery. They are designed to automatically and adaptively learn spatial hierarchies of features from the input data.

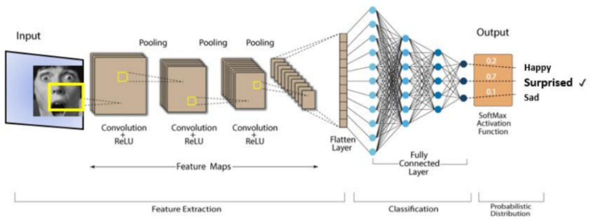


FIGURE 1. A basic CNN architecture illustrates the process of emotion classification

CNNs are composed of one or more convolutional layers, often followed by other components such as activation functions, pooling layers, fully connected layers, dropout layers, batch normalization, flatten layers, softmax layers, residual connections, global average pooling layer,.... Several model components are explained below and specify an architecture to learn multiple levels of abstraction and representations.

Convolutional layer The Convolutional layer stands as the foundational element within the architecture of Convolutional Neural Networks (CNNs), playing a pivotal role in intricate pattern recognition. When presented with an input tensor, this layer executes a convolution operation, utilizing a filter f_k with a specified kernel size of $n * m$. The convolutional process unfolds as an element-wise multiplication of the filter with the input tensor, as described by the equation below:

$$C(X_{U,V}) = \sum_{i=-n/2}^{n/2} \sum_{j=-n/2}^{n/2} f_k(i,j) x_{(u-i,v-j)}$$

Here:

- $C(X_{U,V})$ is the output of the convolution operation
- $f_k(i,j)$ represents the elements of the kernel or filter
- $x_{(u-i,v-j)}$ represents the elements of the input
- The double summation $\sum_{i=-n/2}^{n/2} \sum_{j=-n/2}^{n/2}$ indicates that the convolution operation is performed over the entire area of the filter.

Activation function The Rectified Linear Unit (ReLU) activation function serves as a cornerstone across the architectures of ResNet and GoogLeNet, imparting non-linearity to their computations. Specifically, ReLU calculates the output $(f(x))$ for a given input (x) according to the formula:

$$f(x) = \begin{cases} 0, & \text{for } x < 0 \\ 1, & \text{for } x \geq 0 \end{cases}$$

Pooling layer The Max Pooling layer significantly reduces the dimensionality of the input by employing the max function. Let m represent the size of the filter, and x_i denote the input. The output can be calculated as follows in the equation:

$$M(x_i) = \max\left(r_{i+k,i+l}\right) \quad \text{for } |k| \leq \frac{m}{2}, |l| < \frac{m}{2}, k,l \in \mathbb{N}$$

This process ensures that the output $(M(x_i))$ is determined by selecting the maximum value within the specified neighborhood around the input position x_i. The Max Pooling layer, through this approach, facilitates the extraction of key features, contributing to the downsampling of the input data and the creation of a more compact and abstract representation.

Fully connected layer A Fully Connected Layer, also known as a dense layer, is characterized by each neuron being connected to every neuron in the preceding layer, forming a fully connected network. The output of a fully connected layer is determined by a weighted sum of the input, followed by an activation function. Mathematically, the output y_i of the i-th neuron in the fully connected layer is calculated using the formula:

$$y_i = \sigma\left(\sum_{j=1}^{n} w_{i,j} x_j + b_i\right)$$

Here:

- σ is the activation function
- $w_{i,j}$ represents the weight of the connection between the j-th neuron in the preceding layer and the i-th neuron in the fully connected layer
- x_j is the input from the j-th neuron
- b_i is the bias term for the i-th neuron.

Dropout layers Dropout is a regularization technique in which randomly selected neurons are ignored (dropped out) during training. This means that the contribution of these neurons to the forward pass and backward pass is temporarily removed.

Softmax Softmax is a mathematical function that converts a vector of real numbers into a probability distribution. It takes an input vector and transforms it into a probability distribution over multiple classes.

Here's the mathematical definition of the softmax function: Given an input vector $\mathbf{z} = (z_1, z_2, ..., z_k)$, the softmax function is defined as:

$$\text{softmax}(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_j}}$$

Batch normalization Batch Normalization normalizes the input of each layer by subtracting the mini-batch mean and dividing by the mini-batch standard deviation. This helps stabilize and standardize the inputs, preventing large shifts in internal covariate distribution during training.

Given input x for a specific feature channel, the BatchNorm operation can be defined as:

$$\text{BN}(x) = \frac{x - \text{mean}(x)}{\sqrt{\text{Var}(x) + \epsilon}}$$

Flatten layers Flatten layer is commonly used in the transition from convolutional layers to fully connected layers. Its primary purpose is to convert a multi-dimensional tensor into a one-dimensional vector, which can then be fed into a dense (fully connected) layer for further processing or classification. The Flatten layer is essential when transitioning from the spatially organized data representation of convolutional layers to the flattened input required by dense layers.

Global average pooling layer Global Average Pooling (GAP) is a technique to reduce the spatial dimensions of each feature map to a single value by taking the average of all values. It is commonly used as an alternative to fully connected layers in the final stages of a network, especially when transitioning from convolutional layers to dense layers. Global Average Pooling helps reduce the number of parameters in the model and can contribute to better generalization.

In this project, we will focus on two CNN architectures: ResNet and GoogleNet.

2.1.1 GoogleNet

GoogLeNet, also known as Inception Net, is a convolutional neural network (CNN) developed by researchers at Google. The architecture was designed to keep computational efficiency in mind.

The GoogLeNet architecture is based on building a deeper model to achieve greater accuracy while keeping it computationally efficient. Neural networks with deeper architectures can capture complex patterns and extract hierarchical features from unseen data.

Key components of the GoogLeNet architecture include:

Inception module This module uses multiple filter sizes (5x5, 3x3, 1x1) to extract features at different scales and concatenates their output into a single output. This allows the network to choose between multiple convolutional filter sizes in each block.

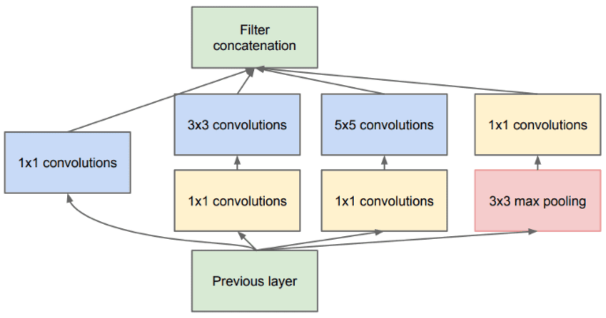


FIGURE 2. Inception module with dimension reductions

Global average pooling (GAP) This is used at the end of the GoogLeNet architecture before the fully connected layer to reduce the dimensionality of the grid, which helps reduce computational costs and overfitting.

Auxiliary classifiers These are used during training only and help in combating the gradient vanishing problem and also provide regularization. These branches consist of a 5x5 average pooling layer with a stride of 3, a 1x1 convolutions with 128 filters, two fully connected layers of 1024 outputs and 1000 outputs, and a softmax classification layer. The generated loss of these layers is added to the total loss with a weight of 0.3.

In the context of our dataset composed of 48x48 grayscale images, we made some necessary adjustments to the GoogleNet architecture while ensuring its fundamental structure remains intact. The input layer is tailored to accommodate the dimensions of our images, and modifications have been made to maintain the overall GoogleNet design. The inception modules, auxiliary classifiers, and global average pooling layers have been fine-tuned to align with the specifics of our grayscale image data. These adjustments are aimed at optimizing the network's performance for our dataset while preserving the essential characteristics of the original GoogleNet architecture.

2.1.2 ResNet50V2

The model architecture is based on a modified ResNet50V2, introduced by Kaiming He et al. in the paper "Deep Residual Learning for Image Recognition" (2015), is a deep neural network architecture that introduced the concept of residual learning. Residual networks are known for their ability to train very deep neural networks effectively. The "V2" in ResNet50V2 refers to the second version or iteration of improvements made to the original architecture. The modifications introduced in ResNet50V2 aim to address issues such as vanishing/exploding gradients and improve the overall performance and training of deep networks.

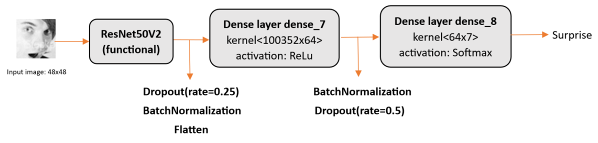


FIGURE 3. The architecture of the ResNet50v2 network

ResNet50V2 Base The ResNet50V2 component serves as a feature extractor, taking input images and producing a feature tensor with a shape of `(7, 7, 2048)`. This output shape is typical for the last convolutional layer of ResNet50V2.

Post-ResNet Processing After the ResNet50V2 base, there is a Dropout layer (dropout_9) and a BatchNormalization layer (batch_normalization_12). These layers are applied after the ResNet50V2 component for regularization and normalization purposes. Dropout (rate=0.25) is used for regularization, and Batch Normalization normalizes the activations, enhancing training stability.

Flatten The flatten layer transforms the 3D tensor into a 1D tensor, preparing it for the subsequent dense layers.

Fully Connected Layers The first dense layer, denoted as `dense_7`, is designed with 64 neurons. This layer serves the purpose of aggregating and combining features learned by the ResNet50V2 feature extractor. The high number of parameters (6,422,592) allows the layer to learn complex relationships within the feature space. The final dense layer, denoted as `dense_8`, has 7 neurons, aligning with the number of classes in the target task. This layer applies a Softmax activation function, transforming the network's raw output into class probabilities. It is responsible for making the final classification decisions. These dense layers constitute the classification head. Batch normalization and dropout are applied for regularization. The final dense layer, with 7 neurons, indicates a multi-class classification task with 7 classes.

2.2 Vision Transformer

Vision Transformers (ViT) represent a groundbreaking approach to computer vision tasks, departing from traditional convolutional neural networks (CNNs) by employing a transformer architecture. Transformers, initially designed for natural language processing tasks, have demonstrated remarkable success in various domains. ViT adapts this architecture to image processing, offering advantages such as increased scalability, flexibility, and enhanced performance on a wide range of tasks.

Transformers were introduced in 2017 with the paper Attention is all you need. They are deep learning models that adopt the self-attention mechanism, differentially weighting the significance of each part of the input data. An image is split into fixed-size patches, each of them are then linearly embedded, position embeddings are added, and the resulting sequence of vectors is fed to a standard Transformer encoder.

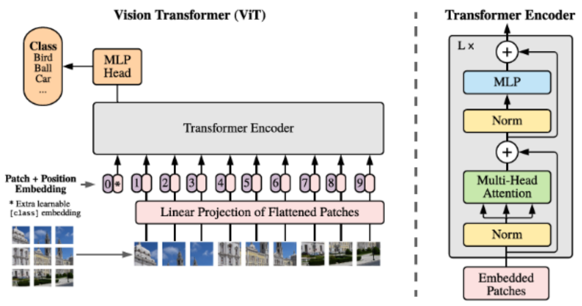


FIGURE 4. The architecture of the Vision Transformer

Image Patching The first step involves breaking down the input image into smaller, non-overlapping patches. Each patch captures a local region of the image and serves as a fundamental unit for processing. This patching strategy allows the Vision Transformer to handle images of varying resolutions without requiring significant changes to the model's architecture.

Linear Embedding After obtaining the patches, each one is linearly embedded into a fixed-size vector representation. This is achieved by applying a linear projection to each patch independently. Mathematically, this can be represented as follows:

$$LinearEmbedding(X) = XW_0 + b_0$$

Here,

- X represents the original patch, usually flattened into a vector.
- W_0 is the weight matrix for the linear transformation.
- b_0 is the bias term.

The linear embedding is essentially a linear transformation that maps the original pixel values of the image patch to a higher-dimensional space. This process allows the model to learn meaningful representations from the image data and prepares the input for further processing within the transformer architecture.

Positional Encodings As transformers do not inherently capture spatial information or the sequential order of input data, positional encodings are introduced to provide the model with information about the spatial arrangement of the patches. These positional encodings are added to the linearly embedded representations, ensuring that the model can distinguish between different patches and understand their relative positions in the image.

Adding a Classification Token In the context of ViT, after linearly embedding the image patches and incorporating positional encodings, an additional token known as the "classification token" is introduced. This token plays a pivotal role in transforming the ViT model into a robust image classifier.

The classification token is appended to the sequence of patch embeddings, effectively becoming the first token in the input sequence.

Transformer Encoder Within the transformer encoder, the multi-head self-attention mechanism shines as the tool for capturing global dependencies within the image. Simultaneously, the feedforward neural network focuses on local features, providing a comprehensive understanding of both fine-grained details and broader contextual information. This tandem operation is crucial for the model's ability to discern patterns and relationships within the input data.

Multi-Head Self-Attention In the multi-head self-attention mechanism, multiple attention heads operate in parallel, allowing the model to analyze diverse aspects and interactions within the image. This parallel processing capability enhances the model's ability to capture nuanced relationships and dependencies, contributing to its effectiveness in various computer vision tasks.

Feedforward Neural Network Following the self-attention mechanism, the feedforward neural network refines the information captured by emphasizing relevant features. This stage further refines the model's understanding of the visual input, enabling it to discern intricate patterns and representations essential for accurate and effective image processing.

Layer Normalization and Residual Connections To ensure the stability and efficient training of the Vision Transformer, layer normalization is applied before each sub-layer, and residual connections are introduced. These components contribute to the model's ability to handle the complexities of learning intricate features by facilitating the flow of gradients during the training process.

Fully Connected Layer (Classification Head) After the image patches go through the transformer blocks in a Vision Transformer (ViT), a fully connected layer (classification head) is employed. This layer takes the flattened output from the last transformer block as input. It maps the features to the number of output classes for the final classification.

Softmax Activation A softmax activation function is often applied to the output of the fully connected layer. This converts the raw output scores into probability distributions over the different classes.

Configuring and Fine-Tuning a ViT model However, Vision Transformers typically require large amounts of data and computational resources to train effectively. This is why we set up a Vision Transformer model for image classification specifically with 7 classes by configuring and fine-tuning a ViT using PyTorch and the Hugging Face Transformers library.

Model configuration We instantiate a class to create a ViT configuration object. The configuration is tailored for facial expressions classification with a total of seven output labels. Subsequently, a "ViTForImageClassification" model is initialized using the specified configuration and pre-trained weights from 'google/vit-base-patch16-224'.

Fine-tuning To adapt the pre-trained ViT model, the fine-tuning process involves modifying the model's architecture and freezing certain layers while allowing the training of others. The steps in this process are:

- Set the number of labels to 7
- Customizing classifier layer: A linear classifier is appended to the model to accommodate the specific number of output labels. This classifier is designed to operate on the features extracted by the Vision Transformer.
- Freezing layers: Parameters of certain layers are frozen to prevent further updates during fine-tuning
- Selective fine-tuning: For flexibility, only the final layer of the transformer encoder, layer normalization, and the classifier layer are designated for fine-tuning.

3. Application and experimentation

3.1 Dataset

In this project, we use the FER-2013 dataset provided by Kaggle. It was introduced for the "Challenges in Representation Learning: Facial Expression Recognition Challenge" held in conjunction with the International Conference on Machine Learning (ICML) in 2013.

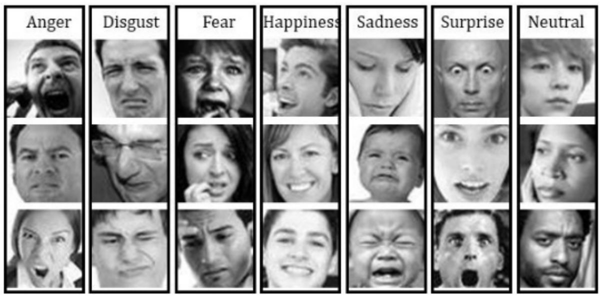


FIGURE 5. Example images from the FER22013 datasets

The FER-2013 consists of about 35,000 well-structured 48x48 pixel gray-scale images of faces. The images are processed in such a way that the faces are almost centered and each face occupies about the same amount of space in each image. Each image has to be categorized into one of the seven classes that express different facial emotions. These facial emotions are Angry, Disgust, Fear, Happy, Sad, Surprise and Neutral.

In addition to the image class number (a number between 0 and 6), the given images are already divided into two subset: train and test.

3.2 Preprocessing

3.2.1 Handling imbalanced data

Upon visualizing our dataset through bar and pie charts, we observed a significant class imbalance. Minority classes, such as 'disgust', constitute a mere 1.5% of the total images, while majority classes like 'happy' account for a substantial 25%. This disproportionate distribution of classes could potentially introduce bias into our model, as it may disproportionately favor the majority class during training. It is crucial to address this imbalance to ensure the model's robustness and its ability to generalize well across all classes.

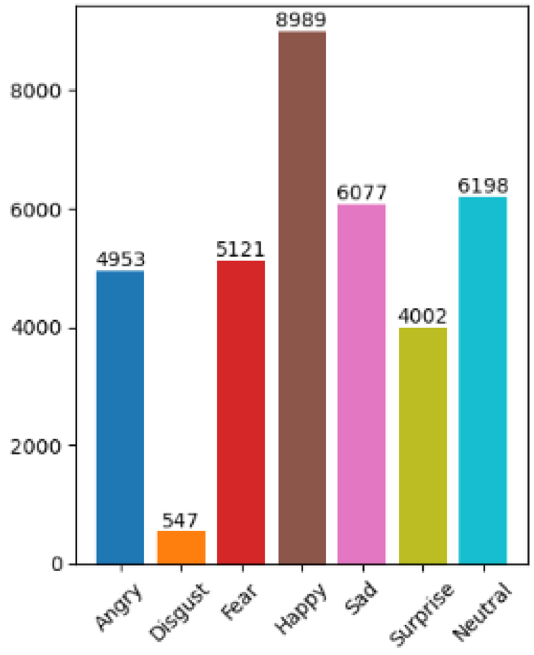


FIGURE 6. Distribution of different classes in the dataset

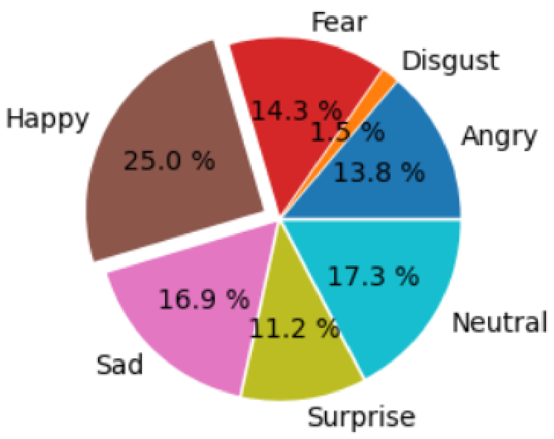


FIGURE 7. Percentage of different classes in the dataset

To address the issue of class imbalance in our dataset, we employed two strategies: class weighting and image augmentation.

Class weighting is a technique that assigns different weights to classes in the training process, ensuring that the model pays more attention to the minority classes. This is particularly useful in scenarios where certain classes are underrepresented, as it helps the model to learn the underlying patterns of these classes more effectively. The weights are typically assigned inversely proportional to class frequencies, which means that minority classes get higher weights. This approach helps to mitigate the risk of the model being biased towards the majority class, thereby improving its ability to classify the minority class

Image augmentation , on the other hand, is a process of artificially increasing the size of the training set by creating modified versions of existing images. This is achieved by applying various transformations such as flipping, resizing, cropping, and adjusting the brightness of the images. The augmented images resemble those already present in the original dataset but contain further variations, thereby increasing the diversity of the training data. This technique not only helps to balance the classes in the dataset but also acts as a form of regularization, reducing the risk of overfitting and improving the model's ability to generalize to unseen data.

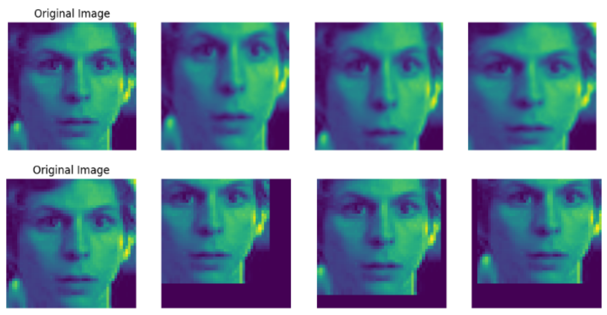


FIGURE 8. Illustration of image augmentation

3.2.2 Cropping, scaling and resizing

The detected faces in the images were cropped and scaled. This is a common preprocessing step to ensure that the model focuses on the relevant features (in this case, faces) and that all input images have the same dimensions. All images were resized to a uniform size of 48x48 pixels. This is necessary because neural networks require input data to have a consistent shape.

3.2.3 Color conversion

To ensure standardized input for the model, grayscale color channel was uniformly applied to all image in the dataset.

3.2.4 Normalization

The pixel values of the images were normalized by rescaling them to a range of 0-1 (by setting the rescale parameter to 1./255). This step is crucial for numerical stability during training, faster convergence, and consistent input ranges for the neural network.

3.2.5 One-hot Encoding of labels

In this project, the Keras library played a pivotal role in preprocessing both the training and testing images before their integration into the model. The choice of class_mode='categorical' in the data generators indicates that the model is designed for multi-class classification, where each image is associated with a single categorical label. The categorical class mode ensures that the labels are one-hot encoded, allowing the neural network to effectively learn and differentiate between the distinct emotion categories during training.

3.3 Spliting the dataset

Our dataset is initially partitioned into two subsets: a training set and a test set. However, to facilitate the model training process and to better evaluate the model's performance during training, we further divided the training set into two parts: a training subset and a validation subset. Approximately 20% of the original training data was allocated to the validation subset. This division allows us to monitor the model's performance on unseen data during the training process, providing a measure of the model's ability to generalize and helping to prevent overfitting.

3.4 Optimizer and loss function

Adam which stands for Adaptive Moment Estimation, is an optimization algorithm that can dynamically adjust the learning rate for each parameter. This optimizer combines the advantages of two other extensions of stochastic gradient descent, namely Adaptive Gradient Algorithm (AdaGrad) and Root Mean Square Propagation (RMSProp). During the training process of our machine learning model, we utilized the Adam optimizer with a learning rate of 0.001. By default, the learning rate we chose is a commonly used starting point, providing a balance between speed and stability in the convergence of the training process.

Categorical Cross-Entropy In conjunction with the Adam optimizer, we employed Categorical Cross-Entropy as our loss function. This choice is particularly suitable for multi-class classification problems, where the output is expected to be a probability distribution across multiple classes. Categorical Crossentropy measures the difference between the predicted probability distribution generated by the model and the true distribution of the labels. It penalizes the probability divergence, encouraging the model to steer its predictions towards the actual labels.

3.5 Performance evaluation

After training the three models - GoogleNet, ResNet, and ViT, we proceeded to evaluate their performance using the test set. The results are summarized in the table below:

Model	Accuracy	Training time	Epoch
GoogleNet	55%	2h	60
ResNet50V2-based	67%	3h	30
ViT	69%	25h	30

TABLE 1. Model Performance Comparison

The three models, GoogleNet, ResNet50V2-based, and ViT, each have their unique strengths and weaknesses in terms of performance, accuracy, and training time.

GoogleNet This model achieved an accuracy of 55% with a training time of 2 hours over 60 epochs. It has the lowest accuracy among the three models, suggesting that it may not be the best choice for this particular task or dataset[3][9].

ResNet50V2-based : This model achieved an accuracy of 67% with a training time of 3 hours over 30 epochs. It has a higher accuracy than GoogleNet, suggesting that it is better able to learn from the given dataset. However, it also takes longer time to train compared to GoogleNet.

ViT (Vision Transformer) This model achieved the highest accuracy of 69%, but it also had the longest training time of 25 hours over 30 epochs. In this case, the ViT model was able to achieve the highest accuracy, suggesting that it was able to learn more effectively from the dataset than the other two models. However, the trade-off is that it took significantly longer to train.

In summary, while the ViT model achieved the highest accuracy, it also required the longest training time. The ResNet50V2 model provided a balance between accuracy and training time, while the GoogleNet model was the fastest to train but achieved the lowest accuracy. The choice between these models would depend on the specific requirements of the task, such as the importance of accuracy versus training time.

These initial accuracy ratings lay the foundation for a more in-depth analysis, where subsequent sections will delve into the nuanced details of each model's performance. Detailed examinations, including training and validation loss curves, training and validation accuracy curves, confusion matrices will unravel specific strengths and areas for improvement within each model, guiding the path toward refined and reliable facial emotion recognition systems.

3.5.1 GoogleNet

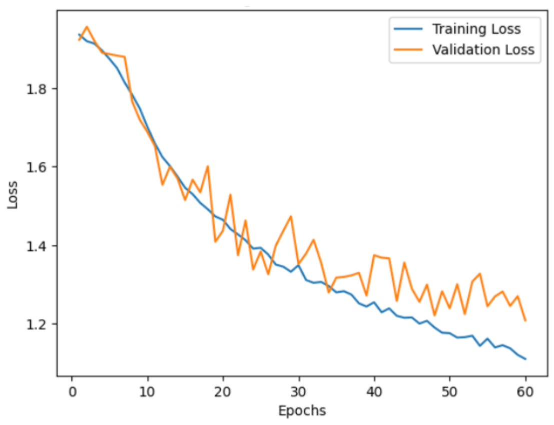


FIGURE 9. Training and validation loss of GoogleNet model during training process

The blue line represents the training loss, while the orange line represents the validation loss. Both lines show a downward trend, indicating that the model is learning and improving its performance over time without overfiting

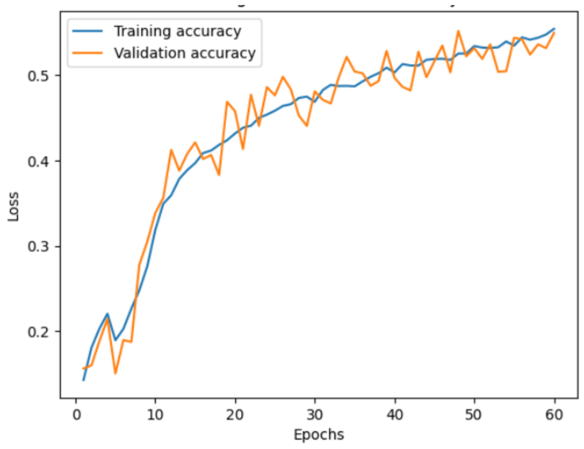
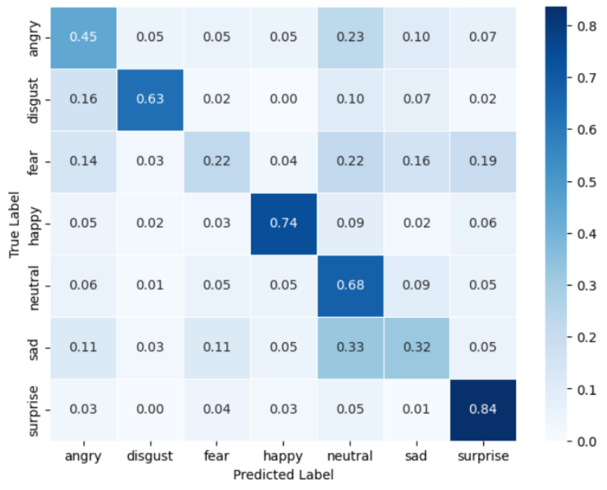


FIGURE 10. Training and validation accuracy of GoogleNet model during training process

The blue line represents the training accuracy, and the orange line represents the validation accuracy. Both lines show an upward trend, which suggests that the model's ability to correctly classify the training and validation data is improving as training progresses.

The fact that the two lines are close to each other and parallel indicates that the model is performing consistently on both the training and validation datasets. This is a positive sign that the model is generalizing well and not overfitting to the training data. Overfitting would be indicated by a high training accuracy but a significantly lower validation accuracy.

In summary, the model is learning effectively and is likely to perform well on unseen data, assuming that the test set is representative of the real-world scenarios the model is intended to work with.



True Label \ Predicted Label	angry	disgust	fear	happy	neutral	sad	surprise
angry	0.45	0.05	0.05	0.05	0.23	0.10	0.07
disgust	0.16	0.63	0.02	0.00	0.10	0.07	0.02
fear	0.14	0.03	0.22	0.04	0.22	0.16	0.19
happy	0.05	0.02	0.03	0.74	0.09	0.02	0.06
neutral	0.06	0.01	0.05	0.05	0.68	0.09	0.05
sad	0.11	0.03	0.11	0.05	0.33	0.32	0.05
surprise	0.03	0.00	0.04	0.03	0.05	0.01	0.84

FIGURE 11. The confusion matrix of GoogleNet model after testing with test data

Based on the confusion matrix provided, it is evident that the 'disgust' class, which was initially a minority class in the dataset, now exhibits a relatively high correct classification rate. This improvement suggests that the methods of class weighting and image augmentation have been effective. Class weighting has likely helped the model to pay more attention to the 'disgust' class despite its fewer instances, while image augmentation has increased the diversity of the 'disgust' class examples that the model was trained on. Consequently, these techniques have contributed to enhancing the model's ability to correctly classify instances of the 'disgust' class, thereby addressing the initial data imbalance and improving the overall performance of the model.

Beside 'disgust' class, the confusion matrix also reveals that the model works well on classes such as 'happy','neutral' and 'surprise'. The model performs most poorly on 'fear' class, and then followed by 'sad' class.

3.5.2 ResNet50V2-based

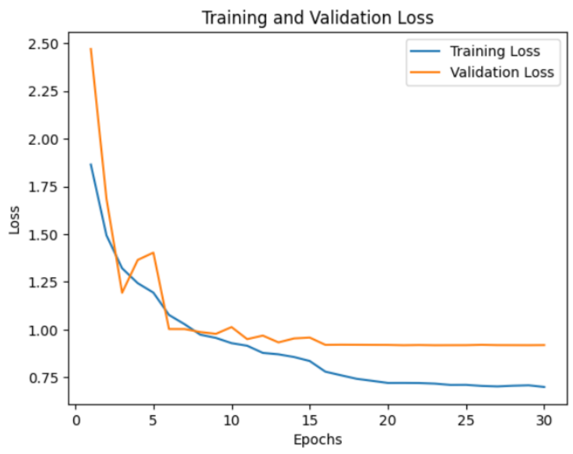


FIGURE 12. Training and validation loss of ResNet50V2 model during training process

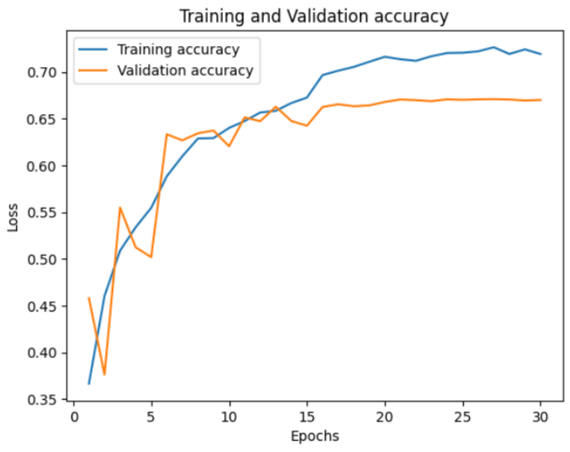
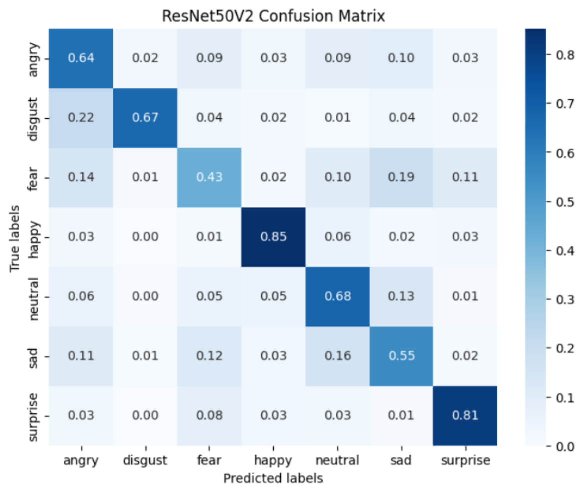


FIGURE 13. Training and validation accuracy of ResNet50V2 model during training process

The two graphs depicting the training and validation accuracy as well as the training and validation loss over epochs of the ResNet50V2 model. The key observations include:

- Training accuracy improves steadily, indicating the model learns well from the training dataset.
- Validation accuracy increases more slowly with fluctuations, suggesting decent generalization to new data but not as effective as on the training data.
- A gap between training and validation accuracy implies potential overfitting to the training data.
- Training loss decreases sharply initially and continues at a slower rate, typical of convergence.
- Validation loss plateaus earlier, suggesting the model may have learned as much as possible from the data.
- No significant increase in validation loss is a positive sign, but a slight gap between training and validation loss hints at possible overfitting.

While the model exhibits a decent performance on the test set with a 67% accuracy score, it's crucial to note that we have implemented optimization techniques such as Adam optimizer and ReduceLROnPlateau. However, there is room for improvement, and there are indications that the model might be overfitting. Further enhancements are needed to address potential overfitting issues and enhance overall performance enhancement.



	angry	disgust	fear	happy	neutral	sad	surprise
angry	0.64	0.02	0.09	0.03	0.09	0.10	0.03
disgust	0.22	0.67	0.04	0.02	0.01	0.04	0.02
fear	0.14	0.01	0.43	0.02	0.10	0.19	0.11
happy	0.03	0.00	0.01	0.85	0.06	0.02	0.03
neutral	0.06	0.00	0.05	0.05	0.68	0.13	0.01
sad	0.11	0.01	0.12	0.03	0.16	0.55	0.02
surprise	0.03	0.00	0.08	0.03	0.03	0.01	0.81

FIGURE 14. The confusion matrix of ResNet50V2 after testing with data

The model performs best at predicting "happy" (0.85) and "surprise" (0.81) emotions, as these have the highest values on the diagonal. The model struggles more with "fear" (0.43), which has the lowest value on the diagonal, indicating a lower correct prediction rate.

3.5.3 ViT

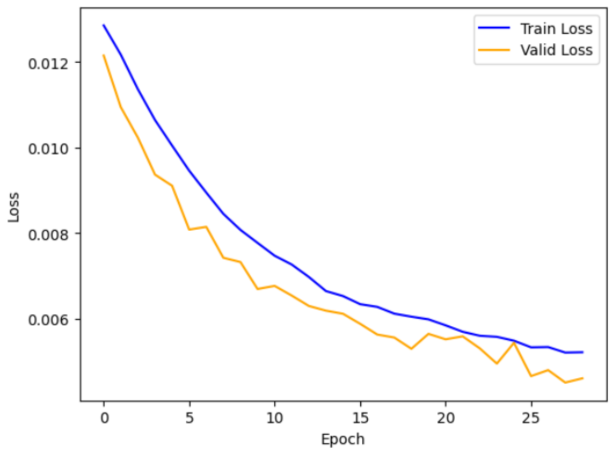


FIGURE 15. Training and validation loss of ViT model during training process

There is no significant increase in validation loss relative to the training loss, which suggests that the model is not overfitting.

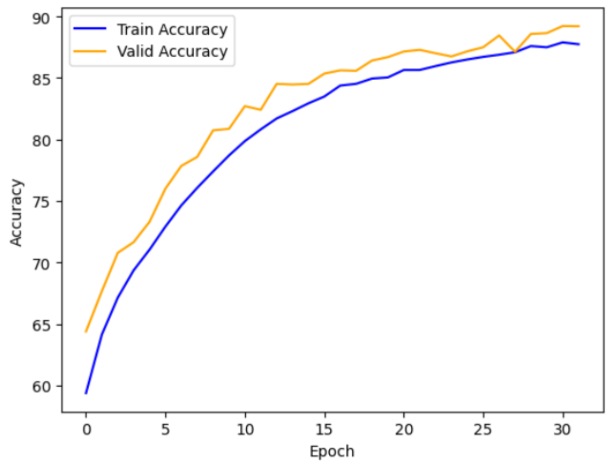


FIGURE 16. Training and validation accuracy of ViT model during training process

Both accuracies increase over time, indicating that the model's performance is improving with more training. The gap between training and validation accuracy is relatively small and stable, which suggests that the model generalizes well to unseen data.

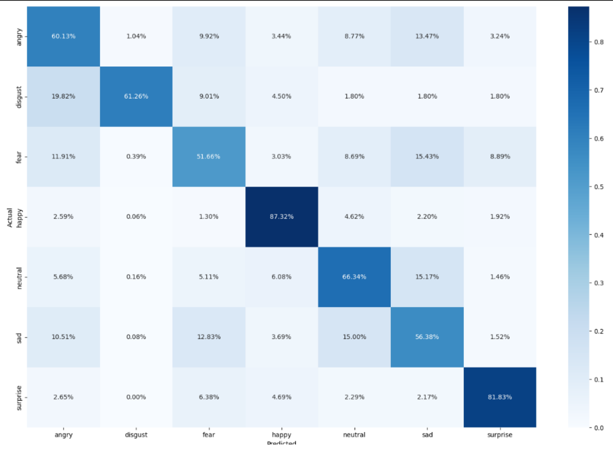


FIGURE 17. The confusion matrix of ViT model after testing with test data

The model performs best at predicting 'happy' and 'surprise' emotions, with accuracy of 87.32% and 81.83% respectively. The model seems to struggle more with 'fear' and 'sad', which have lower correct prediction rates of 51.66% and 56.38% respectively.

4. Conclusion and future work

4.1 Conclusion

In conclusion, our exploration into facial expression recognition utilizing diverse deep learning architectures has yielded promising results. The models, including GoogleNet, ResNet50V2-based, and Vision Transformer (ViT), have demonstrated varying degrees of accuracy in capturing and interpreting facial expressions. Notably, the ViT model outperformed the others with a commendable accuracy of 69%, showcasing its potential for intricate visual pattern recognition tasks.

The training times and epochs required for each model highlight the computational demands associated with achieving improved accuracy. The trade-off between accuracy and training time is a crucial consideration in real-world applications, where efficiency is often paramount. The ResNet50V2-based model strikes a balance between accuracy and training time, offering a competitive performance within a reasonable time frame.

However, it is essential to acknowledge the limitations of the current models. The achieved accuracies, while promising, leave room for improvement, especially in recognizing subtle facial expressions. Additionally, the computational cost of the Vision Transformer raises concerns about its practicality in resource-constrained environments.

4.2 Future work

In the future, there are several promising directions to explore in the field of Facial Expression Recognition (FER) using deep learning. Firstly, the exploration of other deep learning models could yield significant advancements. While Googlenet, ResNet, and ViT have been utilized, there are other models that could potentially improve FER performance. Secondly, the combination of multiple models, also known as ensemble learning, could be a potential area of exploration. This approach involves integrating the predictions from multiple contributing models to improve the overall performance. Techniques such as bagging, boosting, and concatenation, as well as the development of hybrid models that combine two or more different machine learning models, could be employed. Lastly, the application of these models to real-world scenarios could provide valuable insights and practical benefits. Deep learning has found its application in almost every sector of business, including fraud detection, image sorting, and predictive analytics. In the context of FER, these applications could range from enhancing human-computer interactions to improving surveillance systems. By exploring these directions, we can continue to push the boundaries of what is possible in FER and contribute to the advancement of deep learning applications in this field.

References

[1] Ozioma Collins Oguine1*, Kanyifeechukwu Jane Oguine, Hashim Ibrahim Bisallah Daniel Ofuani.: *Hybrid facial expression recognition (fer2013) model for real-time emotion classification and prediction*.

[2] Shima Alizadeh, Azar Fazel: *Convolutional neural networks for facial expression recognition*.

[3] Giannopoulos, P., Perikos I. Hatzilygeroudis I.: *Deep learning approaches for facial emotion recognition: A case study on fer-2013. smart innovation, systems and technologies*. 2017.

[4] Answers, Educative: *What is googlenet?*, 2023. https://www.educative.io/answers/what-is-googlenet, Accessed: yyyy-mm-dd.

[5] Geeks, Geeks for: *Understanding googlenet model cnn architecture*, 2024. https://www.geeksforgeeks.org/understanding-googlenet-model-cnn-architecture/, Accessed: yyyy-mm-dd.

[6] Kaggle: *Emotion detection: Different techniques*, 2024. https://www.kaggle.com/code/keerthanas57/emotiondetection-different-techniques, Accessed: yyyy-mm-dd.

[7] Paperspace: *Popular deep learning architectures: Alexnet, vgg, googlenet*, 2024. https://blog.paperspace.com/popular-deep-learning-architectures-alexnet-vgg-googlenet/, Accessed: yyyy-mm-dd.

[8] Code, Papers with: *Googlenet explained*, 2023. https://paperswithcode.com/method/googlenet, Accessed: yyyy-mm-dd.

[9] Science, Towards Data: *Deep learning: Googlenet explained*, 2023. https://towardsdatascience.com/deep-learning-googlenet-explained-de8861c82765?gi=b60ce63e0549, Accessed: yyyy-mm-dd.

[10] Mohammad Rahimzadeh, Abolfazl Attar: *A modified deep convolutional neural network for detecting covid-19 and pneumonia from chest x-ray images based on the concatenation of xception and resnet50v2*. 2020.

[11] viso.ai: *Vision transformers (vit) in image recognition - 2024 guide*, 2023. https://viso.ai/deep-learning/vision-transformer-vit/.

[12] DataGen: *Resnet-50: Architecture, advantages and applications*, 2023. https://datagen.tech/guides/computer-vision/resnet-50/.

[13] Mastery, Machine Learning: *The vision transformer model*, 2023. https://machinelearningmastery.com/the-vision-transformer-model/.

[14] Face, Hugging: *Vision transformer (vit)*, 2023. https://huggingface.co/docs/transformers/model_doc/vit.

[15] Lee, Y and S Nam: *Performance comparisons of alexnet and googlenet in cell growth inhibition ic50 prediction*. Int J Mol Sci, 22(14):7721, 2021.

[1] [2] [3] [4] [5] [6] [7] [8] [9] [10] [11] [11] [12] [13] [14] [15]