**CNN architecture:**

1. **Model Overview and Architecture Details:**

The deep learning model comprises two parts: convolutional layers for detecting image features and an artificial neural network. The neural network receives the output of the last convolutional layer as a flattened vector and predicts the emotional class.

The architectural specifications of the convolutional neural network for the main model are as follows:

* The model consists of three convolutional layers, each with 64, 128, and 256 neurons or applied filters, respectively.
* In each convolutional layer, three steps should be followed, consisting of padding, filtering, batch normalization, applying the activation function, and maximum pooling.
* There is no padding in the main model.
* The filters should be applied with a kernel size of three and a step size of one.
* Apply the non-linear activation function ReLU to the normalized output of the convolutional operation.
* Apply the max-pooling algorithm with a moving window size and a stride of two.
* After passing through the convolutional layers and obtaining the 2D recognized features, they should be flattened into one vector and fed into the last block, which is an artificial neural network for predicting the emotional label.
* The artificial neural network contains four layers. The input vector to the two middle layers and the output layer should be normalized, and then the ReLU function is applied as the activation function for the neurons.

Regarding controlling the number of parameters, it is known that having many parameters leads to overfitting. To avoid this, 50% of the weights in each layer should be discarded.

To understand the impacts of changing the kernel size and learning depth, the following modifications have been applied to the main architecture of the model to create different variations.

|  |  |  |
| --- | --- | --- |
| **Models** | **Kernel size** | **Dimensions of the convolutional layers** |
| **Main model** | **3** | **Three convolutional layers with 64, 128, and 256 neurons, respectively.** |
| Variation one | 2 | Three convolutional layers with 64, 128, and 256 neurons, respectively. |
| Variation two | 5 | Three convolutional layers with 64, 128, and 256 neurons, respectively. |
| Variation three | 3 | Two convolutional layers with 64, and 128 neurons, respectively. |
| Variation four | 3 | Four convolutional layers with 64, 128, 256, and 512 neurons, respectively. |

1. **Training Process:**

The dataset, which contains images of four emotional classes (angry, bored, focused, and neutral), should be divided into three groups: training, validation, and test datasets. The 'train\_test\_split' function from scikit-learn is used for this purpose, allocating 70%, 15%, and 15% to the training, test, and evaluation datasets, respectively. In this phase, the training and validation datasets are utilized.

For the training procedure, various hyperparameters must be considered during the model's design. These parameters are as follows:

* Number of epochs: In each epoch, the entire training dataset should be seen once. Consequently, the epoch number indicates how many times the entire training dataset will be observed. For this model, the epoch value is set to 100.
* Number of batches: This parameter specifies how many batches the entire training dataset should be divided into during each epoch. The weights are updated after calculating the output error for one batch. In this model, the batch size has been set to 10.
* Learning rate: The learning rate determines the speed of weight adjustments. For this model, the learning rate is set to 0.01, providing a balance between rapid and gradual changes.
* Activation function of the neurons: ReLU, a non-linear function, is chosen as the activation function. One advantage of this function compared to sigmoid is that it leads to sparse activations, meaning that the output of many neurons is zero. This is beneficial for the efficiency and resource utilization of the neural network.
* The chosen loss function is categorical cross-entropy. This loss function aims to minimize the difference between the true and predicted occurrence probabilities of different classes.

The output of this phase includes two models saved in a specified path after running the stage: the best model resulting in the optimal performance for the validation data within 100 epochs and the final model after 100 epochs. The best weights contribute to the evaluation part and serve as the output of this phase.

The following images display the loss function versus the number of epochs for both the training and validation datasets, showing results for the main model and various variations.

A graph showing the difference between train and validation loss

Description automatically generated

Figure Loss/Epochs for the main model (kernel size:3, the dimension of convolutional layers: 64, 128, and 256)

A graph of a graph showing the difference between train and validation loss

Description automatically generatedA graph of a graph showing the difference between train and validation loss

Description automatically generated

Figure 2 Loss/Epochs for variations 1 (left side image) and 2 (right side image)

A graph of a graph showing a number of different colored lines

Description automatically generated with medium confidenceA graph of a graph showing the difference between train and validation loss

Description automatically generated

Figure 3 Loss/Epochs for variations 3 (left side image) and 4 (right side image)

**Evaluation:**

In this section, we assess the evaluation results of two variations in addition to the main model. As demonstrated in the previous section, we explored two kernel sizes (one larger and one smaller than the kernel size of the main model) and two learning depths (one greater and one smaller than the number of convolution layers in the main model).

Based on the figures presented above, the best model with a kernel size of two exhibits a lower loss value compared to the kernel size of five. Regarding learning depth, having four convolution layers is more effective than two layers. Consequently, variations one and four are chosen among the four examined variations.

1. **Performance metrices**

**Micro**

**Macro**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Config | P | R | F1-measure | P | R | F2-measure | Accuracy |
| Main model | 79.63% | 76.69% | 75.19% | 74.61% | 74.61% | 74.61% | 74.61% |
| Variation one | 68.45% | 66.53% | 65.93% | 65.38% | 65.38% | 65.38% | 65.38% |
| Variation four | 64.52% | 62.84% | 62.52% | 63.78% | 63.78% | 63.78% | 63.78% |

1. **Confusion matrices**

|  |  |
| --- | --- |
| **Recall** | **Precision** |
| 39.5% | 84.33% |
| 91.33% | 90.13% |
| 92.02% | 58.36% |
| 83.87% | 85.71% |

**Real**

**Prediction**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Angry | Bored | Focused | Neutral |
| Angry | 70 | 0 | 107 | 0 |
| Bored | 0 | 137 | 0 | 13 |
| Focused | 13 | 0 | 150 | 0 |
| Neutral | 0 | 15 | 0 | 78 |

Table Main model

**Prediction**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Angry | Bored | Focused | Neutral |
| Angry | 36 | 0 | 1 | 56 |
| Bored | 0 | 38 | 21 | 0 |
| Focused | 0 | 6 | 72 | 0 |
| Neutral | 24 | 0 | 0 | 58 |

|  |  |
| --- | --- |
| **Recall** | **Precision** |
| 38.70% | 60% |
| 64.40% | 86.36% |
| 92.30% | 76.59% |
| 70.73% | 50.87% |

**Real**

Table 2 Variation one

|  |  |
| --- | --- |
| **Recall** | **Precision** |
| 75.26% | 58.82% |
| 52.54% | 70.45% |
| 83.33% | 69.89% |
| 40.24% | 58.92% |

**Real**

**Prediction**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Angry | Bored | Focused | Neutral |
| Angry | 70 | 0 | 0 | 23 |
| Bored | 0 | 31 | 28 | 0 |
| Focused | 0 | 13 | 65 | 0 |
| Neutral | 49 | 0 | 0 | 33 |

Table 3 Variation four

**Reasons behind class misclassification?**

Several points extracted from the misclassification analysis are listed below.

* In terms of datasets that are not classified properly, it is observed that for the 'angry' dataset in both the main and the first variation models, only approximately 40% of the actual angry test images are correctly labeled. This may be attributed to the nature of the data, as both the 'angry' and 'neutral' datasets have been obtained from the FER dataset. The issue with this dataset lies in its significant noise and low resolution. Despite the cleaning phase's efforts to mitigate noise, the problem persists even with a smaller kernel size (an attempt to learn more detailed features). The same issue is observed with the neutral dataset, and increasing the number of layers has not provided a solution.
* For the 'bored' dataset, with the variation four model, it can be seen that roughly half of the real bored images can be recognized. Regarding the other classes, except for the class of 'angry,' it can be seen that raising the number of learning layers does not improve performance. The figure related to the loss shows that by increasing the number of layers, the best-found model within ten epochs significantly decreases the difference between the true and predicted occurrence probabilities (roughly the same as the main model); however, the precision metrics worsen. The reason behind that could be overfitting, even when randomly discarding 50% of the weights.

**Motives behind perfect classifications?**

Good classifications are highlighted in green, and these accurate predictions occur with the main model and the first variation, both of which have three convolutional layers and kernel sizes of three and two, respectively. However, increasing the depth of learning (variation number four) results in uninteresting outcomes due to overfitting. It may be beneficial to experiment with increasing the number of layers concurrently with adjusting the ratio of discarded weights to address this issue.