ML Project Documentation

Convolutional Neural Network

Feature set

The chosen feature set in the code is the raw pixel values of the images - *pixel intensity features*. The images are read using the **`preprocess_image`** function, which reads the image file, decodes the PNG-encoded image file into a tensor, and normalizes the pixel values to the range [0, 1].

Model

The model used in this code is a **`convolutional neural network (CNN)`**, because CNNs are highly effective for image classification tasks. The model architecture consists of several **`convolutional layers`** (the primary building blocks of CNNs) followed by **`max-pooling layers`**. The final part of the model includes **`fully connected layers`**. Convolutional layers apply a set of operations called kernels. This consists of a matrix (in our code, the size of this matrix is 3x3) sliding across each image and performing some mathematical operations. The activation function used for these layers is **`relu`**, which stands for rectified linear unit. The function sets negative values to zero and keeps positive values unchanged. Padding is used to ensure that the filters can convolve over the entire image.

The model is compiled with the Adam optimizer, using the sparse categorical cross-entropy loss function and accuracy as the metric.

Data preprocessing and Augmentation

The training and validation data are loaded from CSV files (`train.csv` and `val.csv`) that contain the image names and corresponding class labels. The class labels are converted to numerical values using the `LabelEncoder` from `scikit-learn`.

Data augmentation is applied using the `ImageDataGenerator` from `TensorFlow`. The augmentation includes random rotations, shifts, shearing, zooming, and horizontal flipping. This helps increase the diversity of the training data.

Hyperparameter Tuning

The code uses several callbacks for saving the best model at different epochs during training. Three checkpoints are defined, and the model weights are saved as `best_model_1.h5`, `best_model_2.h5`, and `best_model_3.h5` when the validation accuracy reaches certain epochs (15, 22, and 30, respectively). The numbers 15 and 22 were chosen through experimenting and noticing that those are generally the peak epochs of the model.

Training and Evaluation

The training and validation datasets are created using `Dataset` from `TensorFlow.data` from the preprocessed images and labels. The training data is shuffled and batched with a batch size of 64. The validation data is batched with the same batch size.

The `fit` method is called on the model using the `datagen.flow` function, which generates augmented images during training. The model is trained for 30 epochs, and the validation data is used for monitoring the model's performance. The defined callbacks are used to save the best models based on validation accuracy at specific epochs.

Performance fluctuation

An important role was played by choosing the number of **epochs**, the **batch size** and the number of **layers**.

The optimal number of epochs for our specific model and training dataset seemed to be between 15 and 25. More than 25 did not bring improvement and might have been a little overfitted.

I tried training the model with batch sizes 32, 64 and 128. There was not a significant difference, but out of the three 64 worked the best.

As for the layers, I experimented with multiple layers combinations, starting from a low number of layers and going up until the performance would not increase any more. Generally, adding more layers made a meaningful difference (by far the most meaningful, followed by data augmentation).

Evolution of accuracy rate:

	8 layers	14 layers	16 layers
10 epochs	0,05	0,55	0,57
15 epochs	0,20	0,70	0,56
22 epochs	0,36	0,70	0,60

Neural Network

Alternatively, I also tried an approach using regular neural networks (NN). The implementation was very similar to the CNN, as NN was the base for it. The only difference was that the regular network did not include convolutional layers. The accuracy was lower, around 0.55 at best, even though data augmentation and other parameters were tried.