Face Emotion Detection on FER+ Dataset

Gustavo Zanoni Felipe
Machine Learning Engineer

CONTENTS

LIST OF FIGURES

LIST OF TABLES

Face Emotion Detection on FER+ Dataset

***Abstract*—This report presents the initial results obtained for a Face Emotion Detection task. In this project, we employ the usage of Convolution Neural Networks (CNNs) to classify face expressions to detect emotions in them. By the end of this work, we could reach a weighted average F1-Score of 0.7895 by classifying the input images using an EfficientNetV2B1 model combined with data augmentation and learning rate scheduling techniques.**

I. Problem Definition

This project presents an approach for face emotion detection based on 2D images. The used dataset is called FER+ [1]. The FER+ annotations introduce a fresh set of labels for the conventional Emotion FER dataset. In FER+, every image undergoes labeling by 10 crowd-sourced taggers, offering superior quality ground truth for emotion in still images compared to the original FER labels. With 10 taggers assigned to each image, researchers can compute an emotion probability distribution for each face. The training set comprises 28,709 examples, and for validation and testing two different sets of 3,589 examples each are made available.

All images are in grayscale (single-channel) and resized to 48×48. The dataset includes a total of 10 classes (only the first 9 are considered during testing):

- (0) Neutral;
- (1) Happiness;
- (2) Surprise;
- (3) Sadness;
- (4) Anger;
- (5) Disgust;
- (6) Fear;
- (7) Contempt;
- (8) Unknown;
- (9) Not a Face.

The dataset may be used for both multi-label training and multi-class training. Multi-label expects that one or more labels may be attributed to a singular input image. As the multi-class considers only one label per image. In this project, since we are simulating a real-time environment on a single user, we employ multi-class training. To do so, the class with the higher number of votes is considered as the final label (majority rule).

II. Proposed Solution

This section presents a brief overview of the proposed approach to the face emotion detection task and clarifies the decisions made throughout the project development.

A. Base Architecture Selection

In the first moment, we evaluate the usage of different Convolutional Neural Network (CNN) architectures for the here approached problem. The experimented backbones were selected considering the final deployment scenario. Since the final developed model will be used in real-time and possibly embedded on a device, we opted for testing models with a low quantity of parameters, i.e. small sizes. With a smaller model, we expect to have a faster latency. Therefore, five models were approached. These may be found listed in Table I. All the information presented in such table may be found available at the Keras Applications[1] page.

TABLE I: Models used during experimentation. Top-1 Accuracy is measured on ImageNet validation set.

Model	Size (MB)	# of Parameters	Top-1 Accuracy
MobilenetV2 [2]	14	3.5M	0.713
EfficientNetV2B0 [3]	29	7.2M	0.787
EfficientNetV2B1 [3]	34	8.2M	0.798
EfficientNetV2B2 [3]	42	10.2M	0.805
ResNet50V2 [4]	98	25.6M	0.760

The final model architecture is based on the backbone model initialized on ImageNet [5] weights; followed by a Global Average Pooling 2D layer; and a final Dense layer with dimension equivalent to the number of classes and SoftMax activation. To reduce overfitting, a dropout layer was placed between the Global Average Polling and Dense layers.

Grayscale input images were reshaped from 1-channel to 3-channel images to match the expected input format of the models. Before being fed to the model, images were normalized and pixels were re-scaled to the range of $[-1, 1]$.

To evaluate the most promising base model, we performed training sessions with fixed hyperparameters. The training configuration for such experiments was:

- Loss Function: Categorical Cross-entropy;
- Optimizer: Adam [6];
- Learning Rate: 0.001;
- Batch Size: 512;
- Number of Epochs: 50.

It is worth mentioning that all experiments were executed on a MacBook Pro equipped with a M3 Pro chip and 18GB of unified memory. Initial results may be seen in Table II. Metrics are presented as averages of each class, weighted by their proportions.

In such Table, it is possible to observe the EfficientNetV2 architectures achieved higher F1-Score metrics when compared to both MobilenetV2 and ResNet50V2. Such a family of models was designed to be computationally efficient and in

[1] https://keras.io/api/applications/

TABLE II: Results obtained from varying the model's backbone.

Model	Precision	Recall	F1-Score	Latency p/ Batch
MobilenetV2	0.6208	0.5875	0.5853	**0.1154**
EfficientNetV2B0	0.7508	0.7557	0.7510	0.2244
EfficientNetV2B1	0.7524	**0.7610**	**0.7535**	0.2424
EfficientNetV2B2	**0.7526**	0.7593	0.7516	0.2585
ResNet50V2	0.2217	0.2939	0.2370	0.1379

the here found results, presented a good balance between performance and latency.

Among the tested EfficientNetV2 models, EfficientNetV2B1 presented the best F1-Score, equivalent to 0.7335. From the latency and the total number of images in the test set, we may estimate that such a model can operate around 2100 frames per second. From such a number, we may assume that it is suitable for real-time applications.

B. Data Augmentation

After establishing the base model architecture, we aim our efforts to implement a data augmentation pipeline to apply variations to the training data artificially. Such an approach usually present benefits to the overall model generalization, by applying noises that may occur during production. Keeping that in mind, we propose the usage of two data augmentation configurations.

The first one, here called "weak" data augmentation, aims to add noises that will increase the model general robustness. Such data augmentation pipeline is based on randomly applying horizontal flips, brightness contrasts, and down-scaling the image to simulate lower-resolution face images. Such noises may co-exist in different image inputs and are applied online during training.

The second one, "strong" data augmentation, aims to add higher-intensity operations to the image inputs. Therefore, in addition to the "weak" data augmentations, its pipeline randomly applies angle rotations (from -30° to 30°), affine, color jitter, and random crops. A comparison between F1-Scores obtained from applying both data augmentation configurations may be seen in Table III.

TABLE III: F1-Score values for each class considering the data augmentation configurations.

Class	No Data Aug.	Weak Data Aug.	Strong Data Aug.
Neutral	0.7955	0.8222	**0.8256**
Happiness	0.8727	0.8815	**0.8974**
Surprise	0.7891	**0.81045**	0.7972
Sadness	0.5601	0.5859	**0.6174**
Anger	0.6833	0.7242	**0.7376**
Disgust	**0.4242**	0.3333	0.3428
Fear	0.4078	0.4358	**0.4431**
Contempt	0.3589	**0.4651**	0.3809
Unknown	0.0555	0.0588	**0.1142**
Macro Avg.	0.5497	0.5686	**0.5729**
Weighted Avg.	0.7535	0.7758	**0.7846**

When analyzing such table, it is possible to notice that by applying data augmentation, we were able to increase the F1-Score metrics for 8 of all 9 classes, in which the strong data augmentation protocol achieved the highest results on 6 of them. Such an increase may be addressed to the fact that data augmentation enhances the variability of information present in the dataset. To put it in perspective, the strong data augmentation may apply a total of 128 different combinations of operations, that are different for each image and vary for each epoch.

One other factor is that data augmentation may be used as a regulator for overfitting. When analyzing the loss plots (presented in Figure 1), it becomes clear the the model overfits (Figure 1a) the training data. But, after employing data augmentation, we can reduce the model's overfitting (Figure 1b). Although we still have space for improvement.

C. Learning Rate Scheduling

After defining the data augmentation protocol, we experimented with creating a learning rate scheduler. The usage of schedulers becomes useful to dynamically adjust the learning rate during training and improve generalization, preventing overfitting. Here, we apply an event-based learning rate scheduler. Whenever the network did not improve for a patience of 10 epochs, we applied a factor of 0.1 to the learning rate. By doing so, we could improve our results once more, as presented in Table IV.

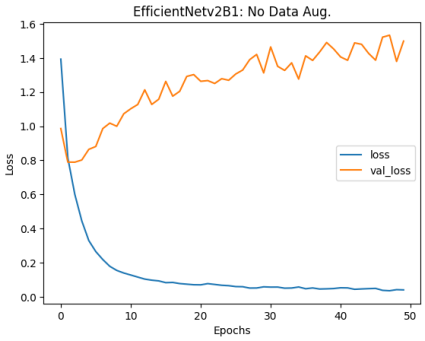
Class	No Data Aug.	Strong Data Aug.	Strong + LR Sched.
Neutral	0.7955	**0.8256**	0.8255
Happiness	0.8727	0.8974	**0.9027**
Surprise	0.7891	0.7972	0.8031
Sadness	0.5601	**0.6174**	0.6159
Anger	0.6833	0.7376	**0.7403**
Disgust	0.4242	0.3428	**0.4736**
Fear	0.4078	0.4431	**0.4907**
Contempt	0.3589	0.3809	**0.4782**
Unknown	0.0555	**0.1142**	0.1111
Macro Avg.	0.5497	0.5729	**0.6046**
Weighted Avg.	0.7535	0.7846	**0.7895**

TABLE IV: F1-Score values for each class considering the strong data augmentation configuration and a learning rate scheduler.

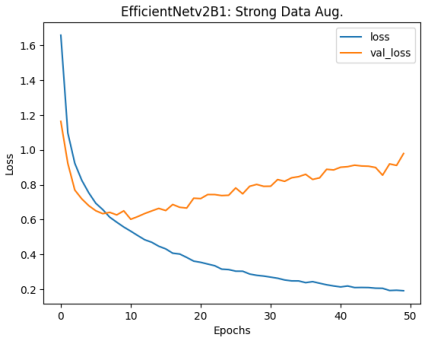
By observing the aforementioned table, it is possible to notice that we were able to keep the metrics of the higher-performing classes (e.g., neutral, happiness, surprise, etc.) while improving the performance of lower-performing classes (i.e., disgust, fear, and contempt). The macro average F1-Score reached a value above 0.6, while the weighted average F1-Score was close to 0.79. It is also possible to observe that the scheduler reduced the validation loss degradation in the later epochs (as shown in Figure 1c).

III. Final Results

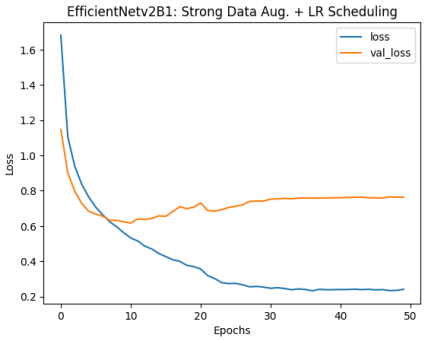
By the end of this project, we were able to reach the higher evaluation metrics by using the EfficientNetV2B0 network, strong data augmentation protocol, and learning rate scheduling. Figure 2 presents the final normalized confusion matrix for each class presented in the test set.



(a) No data augmentation.



(b) Strong data augmentation.



(c) Strong data augmentation and learning rate scheduling.

Fig. 1: Loss curves from training with and without data augmentation.

When analyzing such a confusion matrix, it is possible to observe that the lowest-performing class was ”unknown”. It becomes evident that such a class represents the uncertainty of a given set of image faces, unifying the image samples that could not be labeled accordingly to the pre-determined classes. In the final model, most of them were misclassified as neutral (46%), followed by happiness (14%) and sadness (11%). In the future, such images could be re-visited and have their label reviewed.

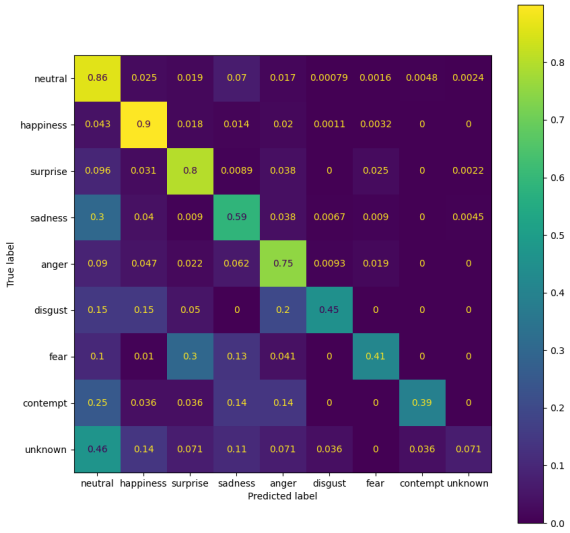


Fig. 2: Final normalized confusion matrix.

When verifying the overall predictions, it is also possible to notice that most of the miss-predicted image samples, were detected as part of the neutral class. This could be justified by the fact that in some cases (e.g., sadness), those facial expressions may share similar features. This is also the case of ”fear”, which was mislabeled as ”surprise” in 30% of all predictions. In some scenarios, surprise comes followed by fear (e.g., in a car crash incident, people are first surprised by the situation and then may encounter fear of the outcome).

Finally, it is possible to verify that three of the nine classes have reached at least 80% of the total number of correct predictions: happiness (90%), neutral (86%), and surprise (80%). In the first moment, we could agree that those facial expressions are the most distinguishable among the set. But, it is also worth noting that they are also the most representative in the training set, with ”neutral”, ”happiness”, and ”surprise” having a respective total of 35.85%, 26.21%, and 12.38% of image samples on the dataset.

Similarly, four of the nine classes did not reach at least 50% of correct predictions: disgust (45%), fear (41%), contempt (39%), and unknown (7%). Aside from the motivation mentioned in previous paragraphs, those classes are also the ones with the least amount of representation in the dataset, with proportions of 2.28%, 0.66%, 0.59%, and 0.58%, for respectively ”disgust”, ”fear”, ”contempt” and ”unknown”. From such numbers, we conclude that the final results could have been affected by the dataset classes’ unbalance.

IV. Next Steps

From the here obtained results, we may define the next steps as:

1) build the model using face embedding generation backbones (e.g., Facenet, ArcFace, etc.);
2) oversample the minority class using imbalanced classification methods (e.g., SMOTE);
3) evaluate the usage of new the classification heads;
4) creating classification ensembles;
5) build the model using tiny vision transformers encoders;
6) build a personalized model for the task;
7) test model compression methods.

It is also worth mentioning, that for a production model, a face detector should be included.

REFERENCES

[1] Emad Barsoum, Cha Zhang, Cristian Canton Ferrer, and Zhengyou Zhang. Training deep networks for facial expression recognition with crowd-sourced label distribution. In *ACM International Conference on Multimodal Interaction (ICMI)*, 2016.

[2] Mark Sandler, Andrew G. Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation. *CoRR*, abs/1801.04381, 2018.

[3] Mingxing Tan and Quoc V. Le. Efficientnetv2: Smaller models and faster training. *CoRR*, abs/2104.00298, 2021.

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. *CoRR*, abs/1603.05027, 2016.

[5] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[6] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.