# Seminars in Artificial Intelligence and Robotics: Facial Emotion Recognition

Andrea Caciolai
*Department of Computer, Automatic and Management Engineering*
*Sapienza University*
Rome, Italy
caciolai.1762906@studenti.uniroma1.it

***Abstract*—Facial Emotion Recognition (FER) is an important topic in AI due to its significant commercial potential. In this work, I try to improve a recent model for FER, with the introduction of a suitable attention mechanism, to better exploit the non-local information present in images of human faces.**

## I. Introduction

This is a report for the work I have done on the project of this year's course in *Seminars of Artificial Intelligence and Robotics*, held by Professor Napoli. The task for my project has been to develop a system able to perform *emotion recognition* and eventually set the RGB components of an ambient light according to the predicted emotion of the bystander.

Facial expression is one of the most powerful, natural and universal signals for human beings to convey their emotional states and intentions [1]. Therefore, in my project I have decided to focus on *Facial Emotion Recognition*, also called *Facial Expression Recognition* (FER) in the literature. The goal of FER is to predict the emotion of a person given an *image* of their face, in particular to classify such image as one of *six* basic emotion classes (some datasets may include more, e.g. with the addition of the *neutral* emotion class).

FER systems can be divided into two main categories according to the feature representations: *static image FER* and *dynamic sequence FER*. In the former, the feature representation is encoded with only spatial information from a single image, whereas in the latter also the temporal relation among contiguous frames in the input facial expression sequence is considered, allowing richer representations coming from video feeds [2]. Although I would have liked to work on the latter, due to limitations in the number of publicly available, openly accessible datasets, I had to settle for the former; in particular, I have worked with the *FER2013* dataset [3], effectively the only easily available dataset among the most used datasets in the literature (see section IV-A for more details).

A recent trend in deep learning research concerning FER involves the use of *attention-based* networks that can highlight the most relevant regions in the image to the FER task through *attention mechanisms* and allow models to learn expression-discriminative representations [2]. Therefore, in my project I have decided to introduce attention mechanisms in one of the most recent and successful architectures proposed for FER: *Deep-Emotion* [4].

## II. Related Work

In my project, I have consulted several previous works in the growing literature pertaining to the field of FER. I gathered most of the needed background knowledge regarding FER from two surveys on the topic [2] [5]. Then, I started researching the most recent trends in the field and found several works [6] [7] [8] employing with success attention mechanisms to solve the FER task. In [4] the authors propose a rather simple model, called *Deep-Emotion*, able to compete (and even outperform) much deeper networks in emotion recognition. They claim that such a network is able to focus on salient facial regions through a *convolutional attention network*. To be precise, this attention capability comes through the inclusion of a *Spatial Transformer* module [9], and not from explicit *self-attention* mechanisms [10], employed successfully also in other Computer Vision tasks (e.g. for GANs [11]). In my project, I consider whether the addition of self-attention mechanisms can improve the performance of the aforementioned *Deep-Emotion* model.

## III. Methodology

In this section, I present the details of the different models that I have designed, implemented and experimented with. To appreciate the effectiveness of the introduction of the aforementioned attention mechanisms, I developed different models of increasing complexities in order to draw comparisons and assess improvements, if any.

In particular, I designed a *baseline model* that is a simple, not very deep CNN classifier, then I added the *Spatial Transformer Network* (STN) module, effectively reproducing the architecture of *Deep-Emotion*, then added *Multiple Self-Attention* (MSA) module to such an architecture. Lastly, I tried to keep the STN+MSA module and change the CNN "backbone" with a deeper architecture, able to learn and extract more high-level and discriminative features; in particular, my architecture mimics *VGGFace*, one of the most popular and effective deep models for the related task of facial recognition [12].

### A. Baseline model

My baseline model has been the CNN backbone used in [2]. Figure 1 illustrates such baseline architecture: it consists of four convolutional (Conv) layers, with every two followed by

a max-pooling layer (Pool) and a rectified linear unit (ReLU) activation function to extract features from the image; they are followed by a dropout layer [13] and two fully-connected (FC) layers to compute the emotion expression class probabilities.

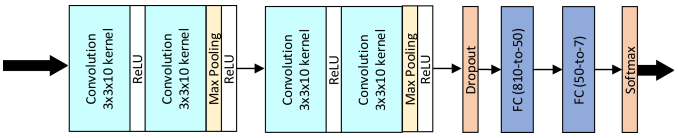


Fig. 1. Baseline architecture.

### B. Spatial Transformer Network

The *spatial transformer* is a *differentiable* module which applies a spatial transformation to a feature map, where the transformation is conditioned on the particular input, producing a single output feature map. In particular, the input feature map $U$ is passed to a localization network which regresses the transformation parameters $\theta$. The regular spatial grid $G$ over $V$ is transformed to the sampling grid $\mathcal{T}_\theta(G)$, which is applied to $U$, producing the warped output map $V$. The combination of the localization network and sampling mechanism defines a *spatial transformer* [9].

In [4] the authors employ this module as an attention mechanism to learn an *affine transformation* $\boldsymbol{A}_\theta$ to warp the input feature map, essentially trying to focus the attention of the model on the most relevant parts of the image by estimating a sample over the region of interest. In this case, for the $i^{th}$ channel of a certain hidden feature representation, the pointwise transformation of the *source* coordinates $(x_i^s, y_i^s)$ in the input feature map–that define the sample points–to the target coordinates $(x_i^t, y_i^t)$ of the regular grid in the output feature map is

$$\begin{pmatrix} x_i^s \\ y_i^s \end{pmatrix} = \mathcal{T}_\theta(G_i) = \boldsymbol{A}_\theta \begin{pmatrix} x_i^t \\ y_i^t \\ 1 \end{pmatrix} = \begin{bmatrix} \theta_{11} & \theta_{12} & \theta_{13} \\ \theta_{21} & \theta_{22} & \theta_{23} \end{bmatrix} \begin{pmatrix} x_i^t \\ y_i^t \\ 1 \end{pmatrix}. \tag{1}$$

Figure 2 illustrates the model architecture proposed by the authors.

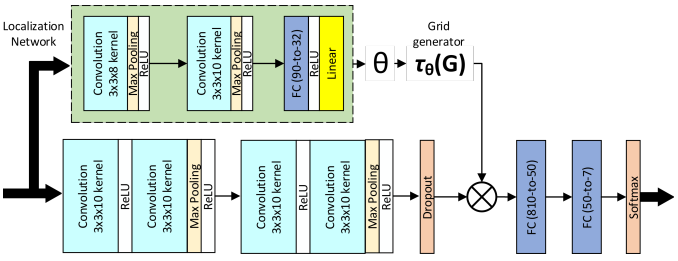


Fig. 2. *Deep-Emotion* architecture.

### C. Multiple Self-Attention

The proposed addition to the architecture is a *Multiple Self-Attention* module (MSA), to be used in conjunction with the Spatial Transformer Network (STN) module. In designing the module, I took inspiration from the self-attention (SA) layer as introduced in [11] and the mechanism of *weight learning branch* and *hybrid attention branch* for the Multiple Attention (MA) block as employed in [7].

*a) Background: SA layer:* An attention function can be described as mapping a query and a set of key-value pairs to an output, where the query, keys, values and output are all vectors [10]. Self-attention by itself is just an operation, with no learning involved; however, the model learns the association from the input feature space to the query, key, value feature spaces, so that then self-attention can be employed over these new features. Essentially, to each feature location $i$ a query, key and value vector are associated, and we transform each feature location by a proper *weighted combination* of the value vectors of every feature location. The weights are called *attention scores* and they are computed from the query vector of the feature location under consideration and the key vector of every feature location (itself included, hence *self*-attention).

In [11] the authors generalized the operation to operate on image features. Let $N = W \times H$ be the number of image feature locations and $C$ the number of channels. The SA layer (see fig. 3) receives the image features $\boldsymbol{x} \in \mathbb{R}^{C \times N}$ from the previous hidden layer and transforms them into the *query*, *key* and *value* feature spaces $\boldsymbol{f}, \boldsymbol{g}, \boldsymbol{h}$ respectively, through Conv layers with $(1 \times 1)$ receptive field (which may change the number of channels for memory efficiency, provided they all agree on the output dimensionality). Then, attention scores $\beta_{j,i}$, indicating the extent to which the model attends to the $i^{th}$ location when computing the $j^{th}$ output feature location, are computed to fill the *attention map* as follows:

$$\beta_{j,i} = \text{softmax}(\boldsymbol{f}(\boldsymbol{x}_i)^T \boldsymbol{g}(\boldsymbol{x}_j)). \tag{2}$$

The attention map is then applied to the value vectors and the resulting features are further refined using another Conv layer, to obtain the self-attention feature map $\boldsymbol{o} = (\boldsymbol{o}_1, \ldots, \boldsymbol{o}_j, \ldots, \boldsymbol{o}_N)$ where

$$\boldsymbol{o}_j = \text{Conv}_{1\times1}\left(\sum_{i=1}^{N} \beta_{j,i} \boldsymbol{h}(\boldsymbol{x}_i)\right). \tag{3}$$

In addition, the layer further multiplies the self-attention feature map by a *learnable* scale parameter $\gamma$ (initialized to 0) and add back the input feature map, to allow the network to gradually learn to use non-local evidence coming from the self-attention feature map. Therefore, the final output of the layer for the $i^{th}$ feature location is

$$\boldsymbol{y}_i = \gamma \boldsymbol{o}_i + \boldsymbol{x}_i. \tag{4}$$

*b) Contribution: MSA module:* The main idea behind the MSA module is to have multiple *attention heads*, learning different non-local features from the incoming feature maps and let the model fuse them adaptively. To do so, incoming feature maps follow two parallel branches: the *multiple attention* branch and the *weight learning* branch.

The former consists of $N$ sub-branches with an SA layer each; the output self-attention feature maps from the sub-branches are then and combined appropriately with the weights coming from the other branch.

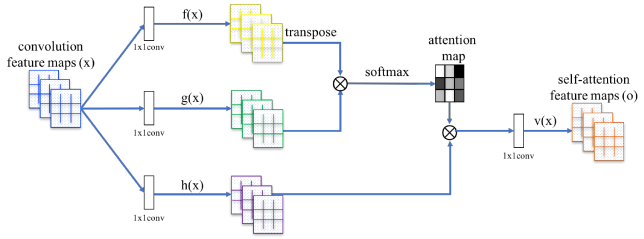


Fig. 3. Illustration of the SA module.

The latter must compute the weights to perform such a combination. These $N$ weights could be regressed using any architecture, and in my project, I followed the architecture used in [7] consisting of a Conv layer (with $(1 \times 1)$ receptive field), followed by a Pool layer (halving the dimension) and a FC layer (with sigmoid activation function). Finally, the weights are normalized to sum up to 1.

Let $\mathrm{SelfAttention}(\cdot)$ be the SA layer as presented before; the output of the MSA module is the *weighted sum* of the output of the $N$ self-attention heads in the *multiple attention branch*, with weights coming from the *weight learning branch*:

$$\boldsymbol{y} = \sum_{i=1}^{N} w_i \, \mathrm{SelfAttention}_i(\boldsymbol{x}) \tag{5}$$

with $\boldsymbol{w} = \left(w_1, \ldots, w_i, \ldots, w_N\right)$ such that

$$\boldsymbol{w} = \frac{\tilde{\boldsymbol{w}}}{\|\tilde{\boldsymbol{w}}\|}, \quad \tilde{\boldsymbol{w}} = \sigma(\boldsymbol{W} \cdot \mathrm{Pool}(\mathrm{Conv}(\boldsymbol{x})) + \boldsymbol{b}) \tag{6}$$

where $\boldsymbol{x} \in \mathbb{R}^{C \times W \times H}$ is the input feature map, with $C$ channels, $\boldsymbol{W} \in \mathbb{R}^{C \cdot W/2 \cdot H/2 \times N}$, $\boldsymbol{b} \in \mathbb{R}^{N}$ are the weight matrix and bias vector of the FC layer, respectively, and $\sigma(\cdot)$ is the sigmoid activation function. See fig. 4 for an illustration of the module.

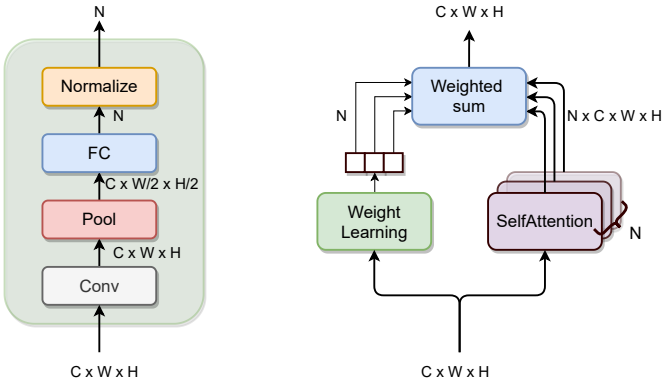


Fig. 4. MSA module. On the left a close-up look on the weight learning branch.

### D. Custom VGGFace with Multiple Self-Attention

As stated in [9], the STN module can be placed within a CNN at any stage, and so can the SA layer of [11]. For this reason, also the MSA layer can be used in combination with any CNN-backbone, and for this reason I decided to try this module with a deeper architecture, one inspired to the successful *VGGFace* (that shares its architecture with the *VGG16* model [12]). See fig. 5 for an illustration of such architecture.

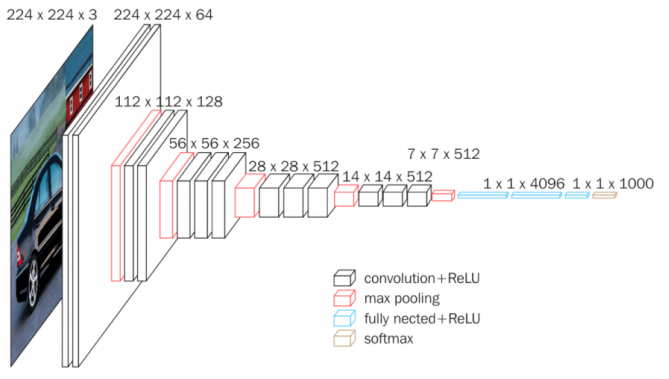


Fig. 5. VGG16 architecture.

Given the nature of the dataset I used, my architecture is not as deep since the network receives smaller sized images, and cannot have as many Pool layers as in the original architecture. In particular, my Custom VGG architecture uses 4 stacks of Conv layers, where filters with small receptive field are used: $(3 \times 3)$, which is the smallest size to capture the notion of left/right, up/down, center [12]. The convolution stride is fixed to 1 pixel, the spatial padding is such that the spatial resolution is preserved after convolution. Each stack is composed of two convolutional layers followed by a Pool layer. The first Conv layer in each stack increases the width (number of channels) by a factor of 2 (starting from 64 in the first stack and reaching 512 in the last stack), the following one maintains it as is. Batch-normalization [14] and ReLU are used after every convolution pass, and max-pooling is performed over a $(2 \times 2)$ pixel window, with stride 2. See fig. 6 for an illustration of such architecture.

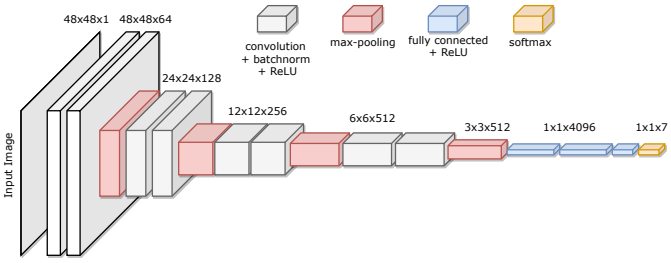


Fig. 6. My architecture.

## IV. Experiments

In this section the dataset used and the implementation details are described, then the results that the models presented in the previous section achieved are reported and discussed.

### A. Dataset

In this project, I used the *FER2013* dataset [3], which is the only publicly available dataset without any registration requirements. *FER2013* is a large-scale and unconstrained database collected automatically by the Google image search API. The dataset already provides a *train-val-test* split, in particular it contains $28,709$ training images, $3,589$ validation images and $3,589$ test images. All images are resized to $(48 \times 48)$ pixels and are automatically labelled with one of the seven expression labels [2].

Due to labels being assigned automatically, *FER2013* suffers from poor quality of its ground truth labels, leading to underperformance of the trained models. For this reason, a new set of annotations for the dataset has been released under the name of *FER+* [15], and I have employed such new annotations in my project. In *FER+*, each image has been labelled by $10$ crowd-sourced taggers, which provide better quality ground truth for still image emotion than the original FER labels.

A common problem shared by many datasets is the large class imbalance, due to the different frequency with which humans express the different emotions, and *FER+* is no exception, as can be seen in fig. 7.

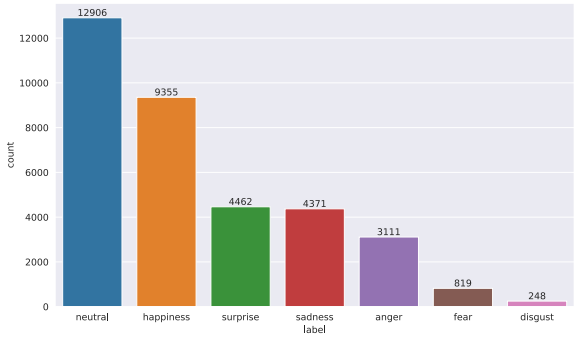


Fig. 7. Unbalanced distribution of classes in *FER+*.

For this reason, I relied on a *weighted random sampling* strategy in order to sample with increased frequency samples of classes with less support in the training dataset.

### B. Implementation details

The models presented have been implemented using PyTorch, and the training has been run on Tesla T4 and P100 GPUs, depending on the availability by Google Colab. The data fed to the model has undergone a small preprocessing phase, explained below, and I have tried to keep the hyperparameters similar among models and across training sessions. Each model has been trained on the training set for $300$ epochs, with batch size $128$ and the use of *early stopping* to mitigate overfitting; in particular, the accuracy on the validation set has been monitored to detect overfitting.

The models are trained to minimize the *cross entropy loss* with $\mathcal{L}_2$ regularization, and both *Adam* [16] and *SGD* with Nestorov momentum [17] optimizers have been tried, with the latter leading to faster convergence. Additionally, a *learning rate scheduler* to reduce the learning rate when the validation accuracy has stopped improving has been employed.

*1) Preprocessing:* An automatic deep FER pipeline usually consists of three steps: preprocessing, deep feature learning and deep feature classification [2]. The first step in the pipeline usually relies on *face alignment* to reduce variation in facial size and in-plane rotation, and *data augmentation* to alleviate overfitting and aid the generalizability of the recognition task.

Face alignment by itself entails another fundamental task: that of *face detection*. Given the nature of the dataset, consisting of images already cropped around the face of interest, during training there was no need to perform face detection. The state-of-the-art approach in face alignment [2] is to either employ *cascaded regressors* or another neural network I experimented with both on the dataset, with very poor results due to the limited resolution of the images and the extreme conditions (occlusions, non-frontal poses...) in which such images are usually taken. For this reason, I did not employ any face alignment technique in my training pipeline. However, this would not be the case in a *production* environment, as explained in section IV-D.

Finally, as for data augmentation I have employed random horizontal flipping, small rotation and small distortion to augment the data, as done in [4]. Another possibility I considered is the common choice [2] of taking ten crops of the image (input samples are cropped from the center and four corners of the image and then flipped horizontally) and averaging the prediction value over all ten crops. This did not increase the performance of the smaller models by a significant amount and led to memory issues during training of the larger one, therefore I did not pursue such strategy.

*2) Hyperparameters:* Find in table I summary of the hyperparameters used in my project, selected empirically in terms of accuracy of the resulting models.

| Parameter | Value |
|---|---|
| Num. epochs ($N$) | 300 |
| Early stopping patience | $N/10$ epochs |
| Batch size | 128 |
| Optimizer | SGD |
| Initial learning rate | $10^{-2}$ |
| Nestorov momentum | 0.9 |
| $\mathcal{L}_2$ regularization weight | $10^{-4}$ |
| Scheduler | *ReduceLROnPlateau* |
| Scheduler factor | 0.5 |
| Scheduler patience | 2 epochs |
| Weight initialization | *PyTorch* defaults |

TABLE I
HYPERPARAMETERS

### C. Results

Here I report the performance of the models described above, measured in terms of *accuracy*, and for a more in-depth analysis of the performance, I also report the *confusion matrix* for each model on the test set.

The baseline model scores an accuracy of $60.61\%$. Figure 8 shows the corresponding confusion matrix.

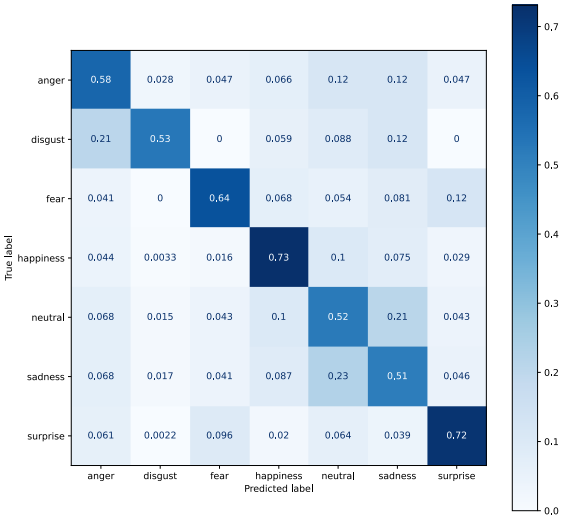


Fig. 8. Confusion matrix for the baseline model.

The reproduced Deep-Emotion model only scores an accuracy of $66.05\%$, despite the authors reporting accuracy of $70.02\%$ in [4]. Interestingly, training the model with the reported choice of hyperparameters and weight initialization resulted in slightly worse performance, but I have not been able to pinpoint the issue. Figure 9 shows the corresponding confusion matrix.

The introduction of the MSA module into the Deep-Emotion architecture resulted in a moderate increase of accuracy to reach $67.86\%$. Figure 10 shows the corresponding confusion matrix.

The custom VGGFace architecture with both STN and MSA modules instead reported accuracy of $82.54\%$. To assess again the effectiveness of the MSA module, I trained and tested also the same architecture without the MSA module, and obtained a score of $81.61\%$: again a marginal but consistent improvement over the model without the MSA module. Figure 11 shows the corresponding confusion matrix.

Find in table II the summary of the characteristics and resulting accuracies of the presented models.

| Model | STN | MSA | Accuracy (%) |
|---|---|---|---|
| Deep-Emotion | | | 60.61 |
| Deep-Emotion | ✓ | | 66.05 |
| Deep-Emotion | ✓ | ✓ | 67.86 |
| Custom VGGFace | ✓ | | 81.61 |
| Custom VGGFace | ✓ | ✓ | 82.54 |

TABLE II
ACCURACY SCORES ON THE TEST SET OF THE PRESENTED MODELS .

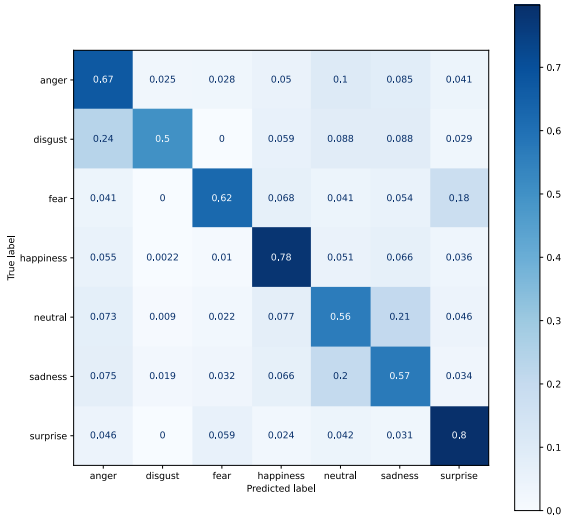


Fig. 9. Confusion matrix for the reproduced Deep-Emotion model.

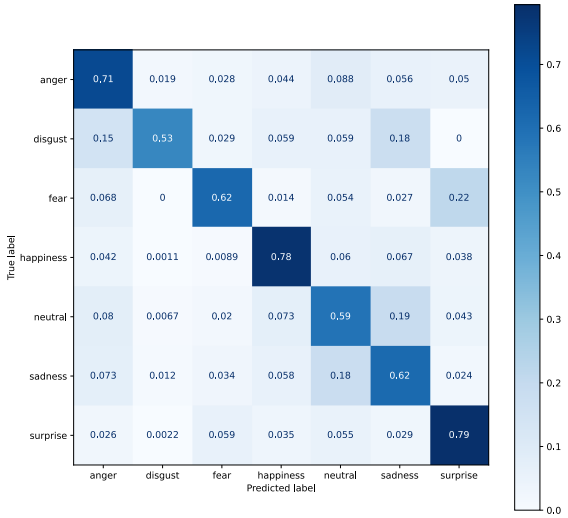


Fig. 10. Confusion matrix for the Deep-Emotion model with the introduction of the MSA module.

### D. Inference

Although the training images in the chosen dataset did not require any preprocessing, in the case of deploying the trained models for production use, the images would likely be acquired by cameras “in the wild”. In such a setting, a proper preprocessing phase becomes a necessity. At the moment,

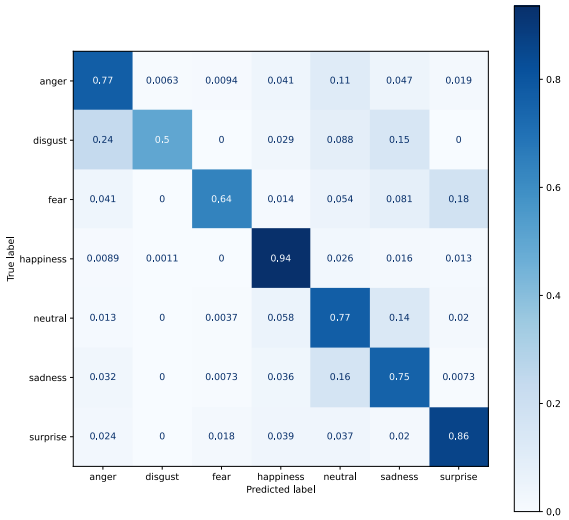


Fig. 11. Confusion matrix for the custom VGG model with the introduction of the STN+MSA module.

many different approaches exist to perform face detection and alignment, both using more traditional (generally faster but less accurate) cascaded-regressor approaches or deep-learning-based (generally slower but more accurate) approaches. To test my final model in a production-like environment, I implemented a preprocessing routine relying on the pre-trained models available in the *DeepFace* library [18] to perform both face detection and alignment. The library collects several models from the literature as available *backends*, and after experimenting with them all, I found the most consistent and reliable (both in terms of identification rate and accuracy in alignment) to be *MTCNN* [19]. Then, the production code simply consists of a loop continuously collecting frames from a camera in real-time, performing the necessary preprocessing (deep learning-based routines of DeepFace and also resizing to match the model input dimensionality), collecting probability scores for the 7 emotions and displaying them beside the identified face.

## V. Conclusion

In this report, I presented the work I have done for the project in Seminars in Artificial Intelligence and Robotics. In particular, this work has involved predicting human emotions based on face footage coming from a camera. To start this work, I first reviewed the existing literature concerning the field of Facial Emotion/Expression Recognition (FER), then decided to try attention-based mechanisms to increase the performance of a very recent paper. Therefore, I developed a rather simple attention mechanism taking inspiration from how other recent works have used attention in their models for FER, currently a very strong trend in the field. Finally, I performed a comparative test whose results have been collected and reported in this document, showing that the introduction of my attention mechanism does indeed improve marginally the accuracy rate of the preexisting model, and also of another, deeper model, showing that its effectiveness does not depend on the employed backbone CNN.

## References


[1] C. Darwin, P. Ekman, and P. Prodger, "The expression of the emotions in man and animals: Oxford university press," *USA,(1872 reprinted 2002)*, 2002.

[2] S. Li and W. Deng, "Deep facial expression recognition: A survey," *IEEE transactions on affective computing*, 2020.

[3] I. J. Goodfellow, D. Erhan, P. L. Carrier, A. Courville, M. Mirza, B. Hamner, W. Cukierski, Y. Tang, D. Thaler, D.-H. Lee, *et al.*, "Challenges in representation learning: A report on three machine learning contests," in *International conference on neural information processing*, pp. 117–124, Springer, 2013.

[4] S. Minaee, M. Minaei, and A. Abdolrashidi, "Deep-emotion: Facial expression recognition using attentional convolutional network," *Sensors*, vol. 21, no. 9, p. 3046, 2021.

[5] B. C. Ko, "A brief review of facial emotion recognition based on visual information," *sensors*, vol. 18, no. 2, p. 401, 2018.

[6] D. Meng, X. Peng, K. Wang, and Y. Qiao, "Frame attention networks for facial expression recognition in videos," in *2019 IEEE International Conference on Image Processing (ICIP)*, pp. 3866–3870, IEEE, 2019.

[7] Y. Gan, J. Chen, Z. Yang, and L. Xu, "Multiple attention network for facial expression recognition," *IEEE Access*, vol. 8, pp. 7383–7393, 2020.

[8] K. Wang, X. Peng, J. Yang, D. Meng, and Y. Qiao, "Region attention networks for pose and occlusion robust facial expression recognition," *IEEE Transactions on Image Processing*, vol. 29, pp. 4057–4069, 2020.

[9] M. Jaderberg, K. Simonyan, A. Zisserman, *et al.*, "Spatial transformer networks," *Advances in neural information processing systems*, vol. 28, pp. 2017–2025, 2015.

[10] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, pp. 5998–6008, 2017.

[11] H. Zhang, I. Goodfellow, D. Metaxas, and A. Odena, "Self-attention generative adversarial networks," in *International conference on machine learning*, pp. 7354–7363, PMLR, 2019.

[12] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[13] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[14] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*, pp. 448–456, PMLR, 2015.

[15] E. Barsoum, C. Zhang, C. Canton Ferrer, and Z. Zhang, "Training deep networks for facial expression recognition with crowd-sourced label distribution," in *ACM International Conference on Multimodal Interaction (ICMI)*, 2016.

[16] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[17] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning," in *International conference on machine learning*, pp. 1139–1147, PMLR, 2013.

[18] S. I. Serengil and A. Ozpinar, "Lightface: A hybrid deep face recognition framework," in *2020 Innovations in Intelligent Systems and Applications Conference (ASYU)*, pp. 23–27, IEEE, 2020.

[19] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao, "Joint face detection and alignment using multitask cascaded convolutional networks," *IEEE Signal Processing Letters*, vol. 23, no. 10, pp. 1499–1503, 2016.