A comparison between SIFT and CNN for Facial Expression Recognition

Corentin Carteau, Riad Ghorra, Arthur Lindoulsi, Lucile Saulnier
CentraleSupélec
corentin.carteau@student.ecp.fr
riad.ghorra@student.ecp.fr
lucile.saulnier@student.ecp.fr

Abstract

For a few years, classifying facial expressions using Deep Convolutional Neural Networks to identify human emotions has been the subject of many papers and advances are made each year in the domain. In this paper we compare a Computer Vision approach using SIFT descriptors with a Light CNN architecture as well as a hybrid SIFT-CNN model. Our goal is to reach a classification accuracy close to that of current state-of-the-art architectures and at the same time achieve fast prediction and training speeds. Our best model turns out to be the pure CNN achieving decent results in terms of accuracy on the FER (62.92%), FER+ (83.96%) and FEC (78.50%) data sets while at the same time boasting an above average prediction speed. We argue that the simplicity of our model makes it trainable relatively fast and enables us to classify emotions in real-time on videos or on any other image-based medium. Furthermore this model being very modular could be integrated easily in many types of applications that need facial expression recognition algorithms.

1. Introduction

Human beings use different forms of communication like speech, gestures and facial expressions. Understanding these vectors of communication is a scientific, social and cultural issue. Researchers in fields such as human-computer interaction, neuroscience or computer vision have tried to identify emotions through these vectors. Such studies would allow creating computers that are able to understand emotions as well as humans thus leading to a seamless interaction between the two.

Though there are many ways of identifying emotions, one of the most promising means is facial expression recognition. The link between expressions and emotions stems at Darwin's Theory of Evolution but has been formalized by Paul Ekman [5] who argued that all human emotions derive from 6 basic micro-expressions that are universal across all cultures : happiness, fear, anger, disgust, sadness and surprise. These expressions usually appear during one twenty-fifth of a second and are thus hard to identify for the untrained eye. That is why automating facial expression and emotions recognition is a real challenge in Deep Learning, Computer Vision and Artificial Intelligence nowadays.

In this paper, we train three models to perform an emotions classification task : a light Convolutional Neural Network, a Random Forest algorithm using SIFT descriptors as features and tow versions of a hybrid SIFT-CNN architecture. These models were trained on images from the FER [9] and FER+ [12] data sets which labels the 6 basic emotions mentioned earlier as well as a "Neutral" emotion. Our goal is to use as light a model as possible in order to be able to use it on video inputs while maintaining an good level of accuracy. Since we want to be able to generalize our predictions to "real-world videos" we used images extracted from Google's FEC data set as our test set reference [19].

The rest of the paper is organized as follows. In Section 2 we discuss related works and we present the FER and FEC data sets in Section 3. Afterwards we expose the architecture we chose for our Network in Section 4, we outline the framework we followed regarding training our model in Section 5 and finally discuss results and next steps in Sections 6 and 7.

2. Related Work

The first work on facial expression classification done on the FER data set was published by Yichuan Tang [18] who introduced a support vector machine instead of the final softmax layer in his CNN thus minimizing a margin-based loss instead of the cross-entropy loss. He achieved an accuracy of 71.2% on the FER data set.

In 2016 Emad Barsoum *et al*. [2] built a new version of the FER data set they called FER+ by re-labeling images with 10 crowd taggers. With a custom version of the VGG13 network and cost functions based on four different schemes(majority voting, multi-label learning, probabilistic label drawing, and cross-entropy loss), they were able to

outperform state of the art solutions.

Studies have also been conducted on other data sets such as that of Peter Burkert *et al*. [3] who use a Deep Convolutional Neural Network (DCNN) with a Feature Extraction Block inspired by GoogleNet. They achieved an average accuracy of 99.6% on the Extended Cohn-Kanade data set (CKP) and of 98.63% on the MMI Facial Expression Database.

In their paper published in 2018, Dinesh Acharya *et al*. [1] argue that introducing second-order statistics such as covariance is a better way to capture distortions in human facial expressions.Their DCNN architecture coupled with Coviariance Pooling was able to achieve an accuracy of 58.14% on the Static Facial Expressions in the Wild data set (SFEW 2.0) and of 87% on the Real-World Affective Faces data set (RAF).

A comparison between SIFT features and Deep Learning was done by Florian Rançon *et al*. [15] while studying the spread of diseases in Bordeaux vineyards. They wanted to develop a classification model that was able to distinguish healthy vines from infected ones and thus compared a Computer Vision approach with a Deep Learning one in order to obtain the best results in terms of accuracy, precision and recall. For the Computer Vision model they extracted SIFT descriptors from each image and then encoded them into Bags of Visual Words using unsupervised clustering. As for the Deep Learning network, they performed transfer leaning from the pre-trained ImageNet network. Results showed that the deep learning features yielded a better outcome for this classification task.

Finally, in 2017, Mundher Al-Shabi*et al*. [4] worked on a neural network that combined both SIFT descriptors and CNN features to perform emotion classification on facial expressions. They obtained state-of-the-art results on the FER (accuracy of 73.4%) and the CK+ (accuracy of 99.1%) data sets. Their model consists of two networks working in parallel: a Dense SIFT detector on the one hand and a CNN architecture on the other. For each image, the two types are features are first computed separately, then concatenated and then fed to two fully-connected layers. While this network constitutes an interesting approach to the problem their best performance is obtained by using an aggregator that averages predictions from a simple CNN, a CNN with SIFT and a CNN with Dense SIFT.

3. Data sets

3.1. Facial Expression Recognition (FER)

This competition/challenge was hosted by the ICML 2013 workshop on representation learning [6], organized by the LISA at University of Montreal. The contest itself was hosted on Kaggle with over 120 competing teams during the initial development period.

Figure 1: Examples of images of each class in the FER data set.

Emotion	Number of images	Proportion
Anger	4954	13.8%
Sadness	6077	16.9%
Disgust	547	1.5%
Happiness	8989	25.0%
Fear	5121	14.3%
Surprise	4002	11.2%
Neutral	6198	17.3%
Total	35888	100%

Table 1: Distribution among classes in FER

The data set consists of approximately 35 000 48x48 images of faces with 7 different emotions: anger, sadness, disgust, happiness, fear, surprise and neutral. See Figure 1 for examples and their corresponding expression category.

The data set is slightly unbalanced, see the exact distribution between classes in Table 1.

3.2. The FER+ data set

This data set can be seen as an amendment to the FER data set as the latter was re-labelled by 10 crowd taggers [2] who were tasked to choose one single emotion for each image. It contains the same images as FER but with more trustworthy labels in the form of a number of votes in each category for all the data set. This opened the way to multi-class classification models. Moreover an eighth emotion has been added as a category: contempt.

3.3. Google Facial Expression Comparison data set (FEC)

FEC [19] is a large-scale facial expression data set consisting of face image triplets along with human annotations that specify which two faces in each triplet form the most similar pair in terms of facial expression. Each triplet in this data set was annotated by six or more human raters. This data set is quite different from existing expression data sets that focus mainly on discrete emotion classification or action unit detection. It contains approximately 500K triplets

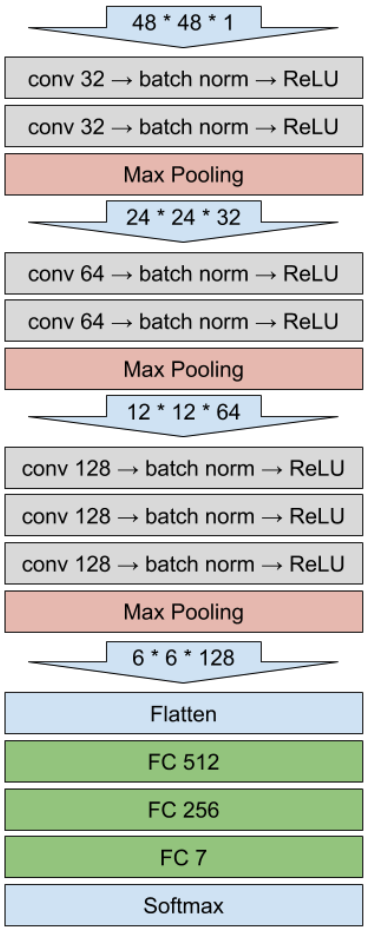


Figure 2: CNN Network architecture

with about 156K colored face images of different sizes.

4. Architecture of the compared models

We have implemented 3 models with different architectures to perform an emotions classification task : a light Convolutional Neural Network, a Random Forest algorithm using SIFT descriptors as features and a hybrid SIFT-CNN architecture.

4.1. LightCNN (pure CNN)

Our use case involves human face images which are relatively small. For this reason, we did not use directly standard CNN architectures such as the ones achieving state of the art performances on the ImageNet challenge [16], which typically use an input of size 224*224 on 3 channels. Instead, we built a model adapted to our use case and to the FER data set, with an input size of 48*48 on 1 channel (grayscale images).

The architecture (see Figure 2) we used is inspired from the VGG network [17]. It is composed of 7 convolution layers and 3 max-pooling layers. For the convolution layers, the filters are 3*3, the stride is of 1 pixel and we use same padding (1 pixel as we use 3*3 filters). Spacial pooling is using a 2*2 pixels window and a stride of 2 pixels. These parameters are the same as the ones use in the numerous VGG network architectures.

The first convolution block has 32 output channels. Each of the following blocks increase this number by a factor 2 until we reach 128 channels after the seven blocks. Meanwhile, each max-pooling layer reduces the width and height of the input by 2, reaching 6*6 from 48*48 after 3 max-pooling layers. The convolution blocks are followed by 3 fully connected layers and a *softmax* layer. The output is the 7*1 probability vector for our 7 classes. We use the *ReLU* non-linearity for all the hidden layers.

We also used batch normalisation [8] after each convolution layer. Using batch normalisation decreased the duration per epoch by 30% We have 2 928 103 parameters to learn with this architecture, 2 492 160 of which are parameters of the fully connected layers. For comparison, the smallest architecture (11 layers) described in [17] had 133 millions parameters to learn.

4.2. Random Forest with SIFT features

We also wanted to implement an automatic classification model for facial expressions of emotions based on descriptors. We could read in several research papers that SIFT descriptors were particularly efficient for classification tasks. But we could also have tested other feature extractors such as SURF or DAISY.

4.2.1 General information on SIFT and Dense SIFT methods

SIFT descriptor is introduced by D.Lowe, University of British Columbia, in his paper *Distinctive Image Features from Scale-Invariant Keypoints* [11]. This algorithm named Scale Invariant Feature Transform (SIFT) aims to extract keypoints and compute its descriptors.

This algorithm follows 4 main steps:

1. Scale-space Extrema Detection. Since the keypoints detected aim to be scale invariant, this step aims to find points of interest (x,y) that are local extrema at the scale of σ. To find this list of potential key points (x,y,σ), the SIFT algorithm uses the Gaussian difference (DoG) which is an approximation of the Gaussian Laplacian (LoG).

2. Keypoints list refining. This step aims to select only real points of interest. The DoG method returns a

higher response for edges, so edges should be eliminated. For this, a concept similar to the Harris corner detector is used. In addition, points with a contrast below a threshold are also removed.

3. Orientation Assignment. This step aims to assign an orientation to the point of interest to make it rotationally invariant. To do this, a neighbourhood is taken around the location of the key point as a function of scale, and the magnitude and direction of the gradient is calculated in this region. An orientation histogram with 36 boxes covering 360 degrees is created. The highest peak of the histogram is taken and any peak greater than 80% of the histogram is also taken into account to calculate the orientation.

4. Descriptors creation. The last step is the creation of a descriptor for each key point. A 16x16 neighbourhood around one keypoint is taken. It is divided into 16 sub-blocks of 4x4 size. For each sub-block, an orientation histogram of 8 values of 0 or 1 is created. In addition to this, several measures are taken to achieve robustness against changes in illumination, rotation, etc. As a result, a total of 128 values are concatenate in a vector to form the descriptor of the key point.

A variant of this method, the Dense SIFT, is particularly interesting for our classification problem. Indeed, with the SIFT method presented above, the number of descriptors will be different from one image to another. On the contrary, the dense SIFT method allows us to fix the number of SIFT descriptors calculated per image. The dense method greatly simplifies steps 1 to 3 of the method outlined above. The difference concerns only the identification of the points of interest. With this method, points of interest are sampled according to a grid of fixed size and orientation. This method is interesting if we think that each parcel has valuable information to represent the content of the image, in other words, that areas with little contraction can carry precisely the information needed for classification.

4.2.2 Random Forest with SIFT features

We have therefore implemented 3 automatic classification models based exclusively on these SIFT descriptors. In the 3 cases, we used a Random Forest as classifier, the difference is thus only found in the SIFT features given to the classifier.

1. **Regular SIFT - Random Forest** : in the first case, we used regular SIFT. Since the number of keypoints varies for each image, we had to paddle the values of each image to an arbitrary value so that the input has size of $48 \times 48 \times 128 = 294912$.

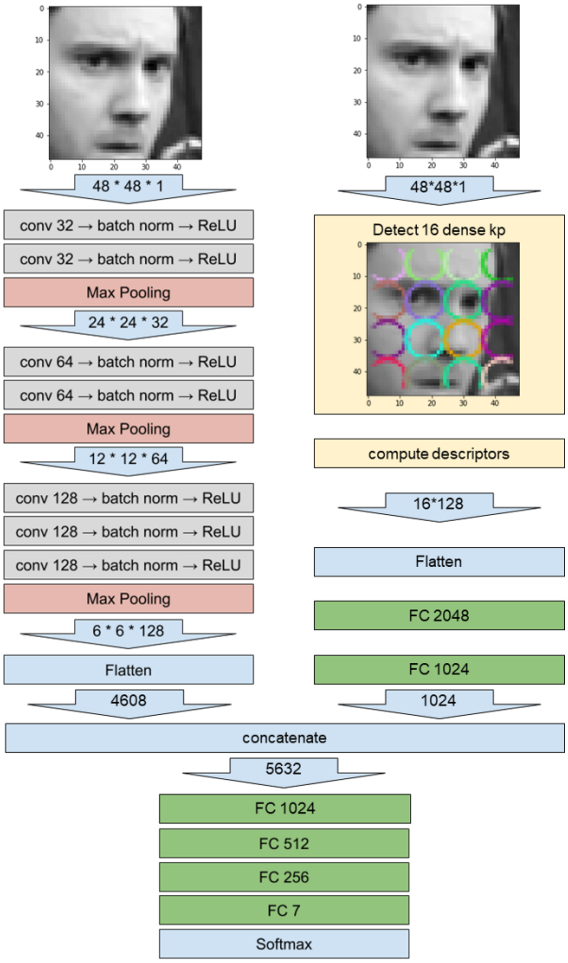


Figure 3: DenseHybrid Network architecture

2. **16 Regular SIFT - Random Forest** : in the second case, we also used regular SIFT but set the limit of keypoints per frame to 16. Each image will thus have a corresponding input of size $16 \times 128 = 2048$. If the number of keypoints detected by the SIFT method is less than 16, the values are set to an arbitrary integer.

3. **Dense SIFT - Random Forest** : finally, we used a dense generation of keypoints. We chose a 12x12 pixels grid, distinguishing 16 regions and thus 16 descriptors per image. In this case, all the images are associated with an input of size $16 \times 128 = 2048$.

4.3. DenseHybrid (Dense SIFT - CNN)

For this first hybrid model we kept the same architecture as for the light CNN by adding only the features of the descriptors to the input data of the first fully connected layers. The general architecture is described in Figure 3.

For this model, the keypoints of each image are extracted

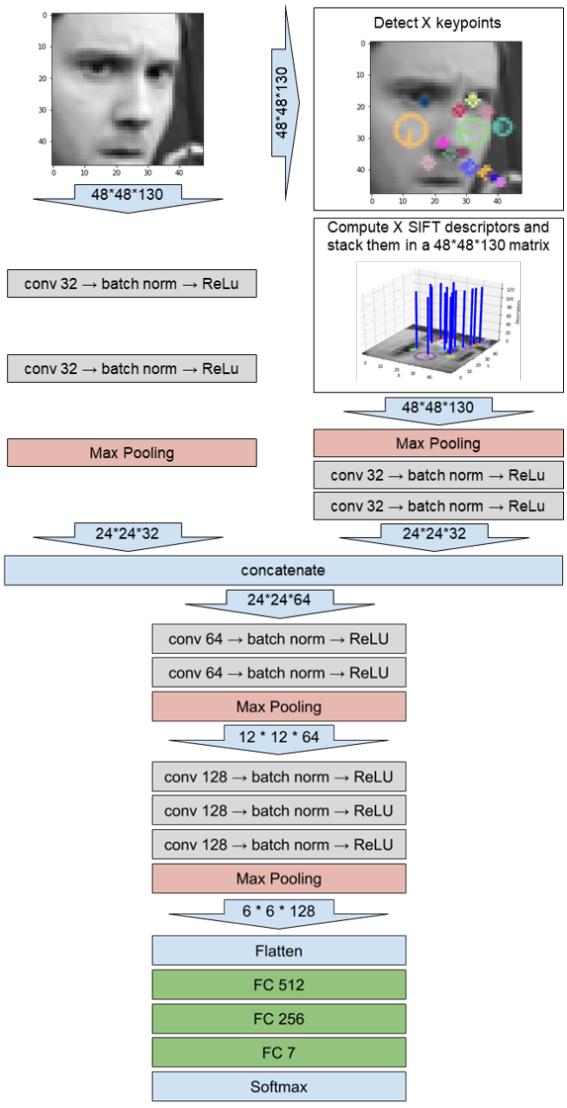


Figure 4: Hybrid Network architecture

by a dense sampling with a 12x12 pixels grid size. Then the SIFT descriptors are calculated for each of these 16 descriptors. As a result, 16 vectors of size 128 are extracted from each image. They are then flattened into a vector of size 2048 before being passed through 2 fully connected layers to be reduced to size 1024.

These SIFT features are then concatenated to the resulting CNN features described in section 4.1 before being passed through 4 successive fully connected layers and then the softmax.

4.4. Hybrid (regular SIFT - CNN)

For this second hybrid model, we integrate the SIFT features earlier in the light CNN than for the first model. The overall architecture is shown in Figure 4.

In addition, the SIFT keypoints are extracted by the classical SIFT method. There is therefore a variable number of them from one image to another. To compensate for this variability, each 48*48*1 image is associated with a 48*48*130 matrix in which the default values are initialized to 0. If a keypoint is detected at the coordinates (x,y) then the 128 values of the SIFT descriptors, the size of the keypoint and its angle are concatenated in a vector of size 130 which will be placed in the 48*48*130 matrix at the position x*y.

5. Classification Framework

We trained the models detailed in section 4 both on the original FER data set and on the FER+ data set. By doing this we have multiple goals in mind :

1. Compare performances on the 2 data sets and measure the improvements brought by using FER+ and training using probability vector labels (detailed below) compared to training using one-hot labels on FER.

2. Compare all models to decide which of the pure CNN, the pure SIFT or the hybrid approach performs best for this classification task.

3. Compare the training and prediction speeds of each model.

To effectively compare these different approaches, we divided the data sets into 3 parts: 70% used for training, 15% for hyper-parameter tuning and 15% for generalization testing. In this section, we will first go through the general training approach, then explain our testing protocol.

5.1. General training approach for Neural Networks models

NB: *This section concerns our LightCNN (pure CNN), DenseHybrid (Dense SIFT - CNN) and Hybrid (regular SIFT - CNN) models.*

As pre-processing for each step, we normalise each image to have pixel values between 0 and 1, with a mean of 0.5 and a standard deviation of 0.5.

The training is carried out by reducing the loss function using mini-batch Adam optimization, with $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 1e^{-8}$ (see [10]). The actual loss functions used will be detailed in the following subsections as they differ on FER and FER+. Given the relatively small training set size (24000 faces), we used a small batch size of 64 images. The training data is shuffled between epochs. We use an original learning rate of 0.001 that is reduced over training using a learning rate scheduler. Namely, the learning rate is divided by 2 when we see no improvements of the loss value for 3 consecutive epochs.

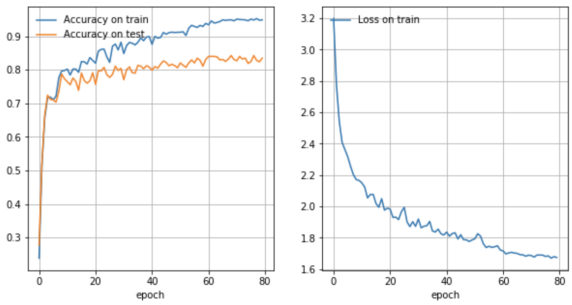


Figure 5: Train accuracy, loss and test accuracy during training on the FER+ data set with the LightCNN model

The network architecture was small enough to use random weight initialisation for each training. For the fully connected layers, the weights are sampled from $U(-\sqrt{k}, \sqrt{k})$ where $k = \frac{1}{in\_features}$ and for the convolution layers the weights are sampled from $U(-\sqrt{k}, \sqrt{k})$ where $k = \frac{1}{in\_channels*n*n}$ (with $n = 3$ the kernel size).

We used data augmentation following a simple procedure. When a batch is constituted, each face image is randomly flipped around the vertical axis, which artificially augments the training data by a factor 2.

In terms of regularization, as specified in the batch normalization original paper [8]:

> "A training example is seen in conjunction with other examples in the mini-batch, and the training network no longer producing deterministic values for a given training example"

When using batch normalisation, the normalised value for a training example depends on its mini-batch and the mini-batches change between epochs. This acts as a regularizing effect similarly to dropout. Using batch normalisation on all the convolution layers (see Section 4) reduces the need for regularization. We used no L2 penalty term and only 2 dropout layers with a probability to zero an element of 0.33, placed after the last convolution layer and the first fully connected layer. We did not actually need more regularization: the test accuracy curve was following the train accuracy curve during training even after the network converged, there was no over-fitting zone were the test accuracy would go down and the train accuracy continue to go up (see Figure 5 for the curves on FER+, the shape of the curves were similar for the training on FER). As expected, adding L2 regularization during training did not improve the performances.

5.2. Training on the FER data set

The FER data set has only one label per face. For training, we used the negative log-likelihood loss. As the FER data set is unbalanced (see Figure 1) we added a weight to the loss for each class equals to the inverse of the proportion of the class in the data set. The loss with weights is expressed as

$$loss(x, y) = -\sum_j \frac{1}{f_j} y_j \log x_j$$

where f_j is the frequency of the class j, x is the output of the *softmax* layer and y is the one-hot encoding indicating the actual FER label.

5.3. Training on the FER+ data set

The FER+ data set has 10 labels per face, annotated by 10 crowd-sourced annotators. The annotations for each face were saved in FER+ as an integer representing the number of annotators choosing each emotion. We transformed the annotations for each face to get a target label vector defined as the proportion of annotators choosing each emotion. The label is then a probability vector. We trained the model using such labels instead of just taking the majority vote. We expected a better accuracy with this training procedure, as it is often hard to categorize a face with a single emotion (one can be happily surprised for example). To be able to fit the model on this probability vector labels, we used the cross-entropy loss expressed as

$$loss(x, y) = -\sum_j \frac{1}{f_j} (y_j \log x_j + (1 - y_j) \log (1 - x_j))$$

where f_j is the frequency of the class j, x is the output of the *softmax* layer and y is the probability vector label built from FER+ annotations.

5.4. Specifics of using SIFT features

5.4.1 Random Forest with SIFT

The idea behind this model was to measure the ability of SIFT descriptors to classify emotions "on their own". Thus we fed SIFT then Dense SIFT features to two Random Forests models using Python's Scikit-Learn library [14]. For every image of the FER+ data set we computed both SIFT and Dense SIFT features and built our training, validation and test sets. However in the case of regular SIFT, the number of descriptors and key points found in each image is not the same, thus yielding vectors of different sizes. We solved this by using zero padding to even out vector dimensions. Finally, we normalized all feature vectors with a Standard Scaler before feeding it to the Random Forest algorithm.

5.4.2 Neural Networks with SIFT

Building the DenseHybrid (Dense SIFT - CNN) and the Hybrid (regular SIFT - CNN) networks required to integrate SIFT descriptors into the existing base LightCNN.

The first thing we did was to adapt the pre-processing function we described in section 5.1. to integrate the detection and processing of SIFT features for each image during every epochs. The reason we have to do this every time and cannot do it beforehand is that since we use data augmentation, we have to compute SIFT features during every iteration. Integrating this function in the pipeline also proved useful when we tested the performances of our model on the external FEC data set (see section 5.5.). Moreover, when using regular SIFT, since the number of descriptors is not constant across images, we added a zero-padding function to make sure the dimension of the input was the same for every observation.

5.5. Testing approach

We split both FER and FER+ data sets in 3 sections: train (70%), validation (15%) and test (15%). Train and validation are used during training to train the model and tune the parameters. The evolution of accuracy during training is computed using the training set and the validation set. We keep the test set to evaluate the generalization of the trained model. The faces in train, validation and test are the same for FER and FER+ to be able to compare the performances. Still for comparison purposes, we keep the top 1 predicted class accuracy as a common metric for both data sets. To better reflect the overall performances of the model, we also use confusion matrices to plot the rates of predicted labels against those of true labels. Early on, we noticed that our models do very poorly on certain classes such as "disgust" or "fear" mostly because they are very under-represented in our data sets (notwithstanding the weights we used to compensate this imbalance). Thus we used another confusion matrix to visualize our performances by grouping "negative" emotions together *i.e.* sadness, fear, anger and disgust. Do note that this grouping was only employed to compute statistics on predictions and was not used in the training part of our work.

In order to test our model on external data (*i.e.* not provided in the FER/FER+ data sets) while keeping a standardized face input, we needed to integrate the model in a face detection pipeline. In order to use our model to classify emotions in videos, or at least generalize it to any image, we needed a way to detect faces on images and video frames. We thus integrated to our model a face detection algorithm developed with *OpenCV* [7] that crops an image around a face when it detects one. The parameters chosen for this algorithm were optimized based on the number of faces on FER we were able to detect with a set of parameters. The final parameters allowing to detect the maximum number of faces on FER were $minNeighbors = 3$, $scaleFactor = 1.01$, $minSize = (20, 20)$. We were able to apply this algorithm to many types of images containing a face and not just data set images. We could also use it on videos after sampling some frames to reduce the number of images to process. We thus had at our disposal a pipeline that could extract facial expressions from most images or videos. In order for the model to perform well in this new pipeline, we trained other versions using a pre-processed version of FER, with the faces cropped with the same algorithm as we use for real-world testing. This enforces the faces during training and validation to come from the same pipeline as the faces during test and thus improving generalization. This is at the cost of reducing our amount of training data as some faces are not detected by the face detection algorithm (see Table 2).

Class	**Images kept**	**Images dropped**
Anger	3286 (66.3%)	1668 (33.7%)
Sadness	3310 (54.5%)	2767 (45.5%)
Disgust	362 (66.2%)	185 (33.8%)
Happiness	6454 (71.8%)	2535 (28.2%)
Fear	3206 (62.6%)	1915 (37.4%)
Surprise	2927 (73.1%)	1075 (26.9%)
Neutral	4622 (74.6%)	1576 (25.4%)
Total	24167 (67.3%)	11721 (32.7%)

Table 2: Proportion of the FER data set kept per class after face detection algorithm

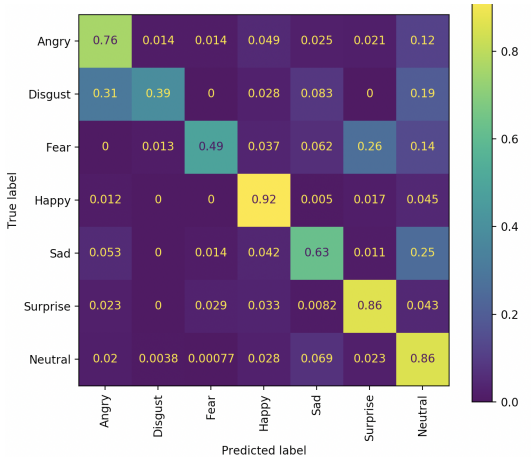
Once we had models trained to be used in a face detection pipeline, we wanted to test the performances on real-world data. We used for this purpose the FEC data set (see 3.3.). We detail the results obtained in the Section 6 below.

All the code and instructions for testing are available at `https://github.com/riadghorra/facial-expression-recognition`. This includes an image viewer to be able to quickly visualise the model predictions.

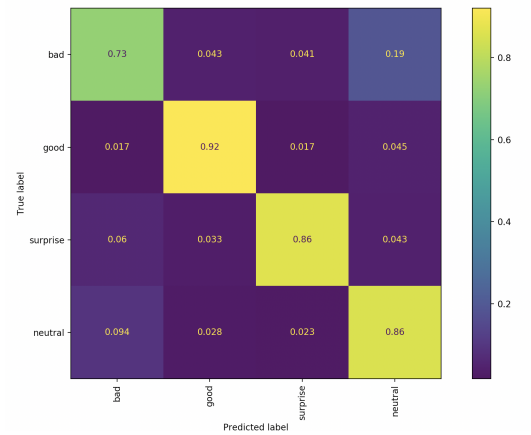
5.6. Implementation

To implement the architecture and the pipeline and train the model, we used Python3 and the PyTorch [13] library. As detailed in the previous section, we used the OpenCV library to crop faces, to detect the keypoints and to compute SIFT descriptors. More specifically, in order to use the SIFT implementation proposed by openCV we had to install 3.4.2.16 version of the opencv-python and opencv-contrib-python libraries. To manipulate the CSV data sets, we used the Pandas library. All the code and instructions for testing are available at `https://github.com/riadghorra/facial-expression-recognition`.

To train the models, we used pytorch support for CUDA and Google Colab, which allowed us to quickly iterate through different versions of the model.



(a) Confusion matrix with the 7 regular classes.



(b) Confusion matrix with the 4 grouped classes.

Figure 6: Confusion matrices to visualize model predictions on the FER+ data set with the LightCNN model.

6. Classification results

6.1. Metrics

In order to evaluate the performance of our models we looked at different indicators. Even though accuracy is not an ideal measure in a multi-class classification problem, we still used it in our case as it is a global metric to compare the performances obtained. The second indicator we used is a confusion matrix that plots the rates of predicted labels against those of true labels. It is a good way of identifying the classes on which our model performs best and where it struggles. We used another confusion matrix to evaluate performances while considering grouped emotions (see 5.2.). You can see in Figure 6 the two types of confusion matrices.

Trained on \ Tested on	FER	FEC
FER	62.92%	50.37%
Cropped FER	61.04%	58.78%
FER+	**83.96%**	72.95%
Cropped FER+	83.04%	**78.50%**

Table 3: Performances of our models in terms of accuracy

6.2. Results

Using these metrics, we evaluated our models through different testing approaches with the architectures proposed in section 4. As a reference, state of the art accuracy on the FER data set is around 71%, whereas random guessing would yield an accuracy of approximately 14%.

6.3. Influence of the FER vs FER+ training dataset

At first we wanted to see on which dataset it was better to train our model. To do this we compared the results of a CNN light when trained on FER and FER+.

The accuracy on the test sets are computed for all the models using the FER+ with cropped faces test set as it is the most restrictive. All these results are summarized in Table 3.

The best model trained on the FER data set and achieved a peak evaluation set accuracy of 63.12% in 80 epochs. Evaluated on the FER test set, the generalization accuracy is at 62.92%.

As explained in the testing approach section, we also applied our face detection algorithm to the FER data set in order to remove observations that do not clearly show a face and thus hopefully improve performances. The model trained on this cropped version of FER achieved an evaluation accuracy of 66.11% and a generalization accuracy on the test set of 61.04%.

Then we trained a model on the FER+ data set and achieved a validation set accuracy of 81.44%. The test accuracy for this model is of 83.96%. The model on the cropped FER+ data-set achieved a peak accuracy of 83.99% on the validation set after around 60 epochs. The test set accuracy for this final model is of 83.04%. The confusion matrices shown in Figures 6a and 6b illustrate the predictions made by this last version of the model. We explain this consequent difference between the models trained on FER and FER+ by the better annotation of FER+ compared to FER and the fact that we took into account probabilities for all classes rather than just a single correct class for each example.

As a way to test this model on real-world images, we labelled by hand about 600 images from the FEC data set. However, since FEC data are RGB images of varying di-

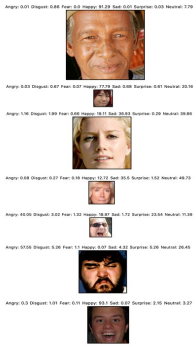


Figure 7: Some examples of correctly classified images from FEC

mensions, we had to develop a pipeline to convert them to grayscale and resize them to fit the input layer of our network. On this small portion of FEC we reached an accuracy of **78.50%**. This result being slightly worse than our test set's can be explained by the fact that FEC contains random internet images from many different sources that differ a bit from the images in the FER data set. For instance, a great number of images in the former set are of people making funny faces, sticking out their tongue, or images of babies, the expressions of which are hardly identifiable even for humans. You can see on Figure 7 a few examples of successfully classified FEC images.

6.3.1 Influence of the model architecture

Another goal of our study was to compare different model architectures performing an emotions classification task : a light Convolutional Neural Network (Light CNN), 3 Random Forest algorithms using SIFT descriptors as features (Dense SIFT x Random Forest, Regular SIFT x Random Forest, 16 regular SIFT x Random Forest) and 2 hybrid SIFT-CNN architecture (DenseHybrid and Hybrid). All these results are summarized in Table 4 below.

Random Forest's performance proves that SIFT features are capable of summarizing information needed for sentiment classification. Among these Random Forest, the most efficient is the one based on Dense SIFT features. Given that we work on the FER+ cropped dataset, all the images are already almost at the same scale and this particularity can explain that the dense sampling is particularly interesting in this case.

We also note that the performance of Light CNN and hybrid networks are relatively similar when tested them on the FER+ dataset. This observation leads us to believe that the Light CNN network would already have this information synthesized in its network and that adding this feature by concatenation does not bring any additional information.

In addition to these metrics, one of the objectives of this paper was to develop a model, easily trainable and which could predict emotions fast. As for training time, our best model, the LightCNN, reached its peak accuracy after 70 epochs which was just under an hour on Google Colab. As for prediction speed when evaluating the models on the FER+ with cropped faces test set, all the models were taking 32s (+- 1s) to make predictions for the 3540 images with a batch size of 64, i.e. only 9ms per image on average. This was evaluated on a standard Mac Book 2.9 GHz Intel Core i5 CPU, not a GPU. On the contrary, the hybrid model requiring the computation of SIFT features for each image considerably increases the training and prediction time.

7. Conclusion and future work

In this paper we build different model architectures to classify facial emotions in order to identify emotions on images. We show that (1) a simple model composed of a feature extractor, such as SIFT features, and a simple classifier, such as a Random Forest, could achieve very respectable classification performance and that (2) the architecture providing the best performance both on the datasets used for its training and on external datasets such as Google's FEC is the Light CNN architecture.

During this project we have learned that adding SIFT features to our CNN neural network did not improve its performance and that building our own network from scratch is better (in this case) than using an existing one and applying Transfer Learning on it. Developing our Light CNN enabled us to fit it better to our needs in terms of data sets, performances and the desired pipeline in general. The latter part is especially important since using our model on any kind of image or videos required additional bricks like face detection algorithms or video sampling.

Architecture	Trained on	Tested on	
		FER+	FEC
LightCNN	FER+	**83.96%**	72.95%
	Cropped FER+	83.04%	**78.50%**
DenseHybrid	FER+	82.09%	62.5%
	Cropped FER+	83.96%	62.5%
Hybrid	FER+	80.88%	-
	Cropped FER+	81.89%	-
Dense SIFT x Random Forest	Cropped FER+	67.18%	-
Regular SIFT x Random Forest	Cropped FER+	45.95%	-
16 regular SIFT x Random Forest	Cropped FER+	45.11%	-

Table 4: Performances of our models in terms of accuracy

References

[1] Dinesh Acharya, Zhiwu Huang, Danda Pani Paudel, and Luc Van Gool. Covariance Pooling for Facial Expression Recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2018.

[2] 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.

[3] Peter Burkert, Felix Trier, Muhammad Zeshan Afzal, Andreas Dengel, and Marcus Liwicki. DeXpression: Deep Convolutional Neural Network for Expression Recognition, 2016.

[4] Tee Connie, Mundher Al-Shabi, Wooi Ping Cheah, and Michael Goh. Facial Expression Recognition Using a Hybrid CNN–SIFT Aggregator. In Somnuk Phon-Amnuaisuk, Swee-Peng Ang, and Soo-Young Lee, editors, *Multi-disciplinary Trends in Artificial Intelligence*, pages 139–149, Cham, 2017. Springer International Publishing.

[5] Paul Ekman. Universals and cultural differences in facial expressions of emotions. *Nebraska Symposium on Motivation*, 19(1):207–282, 1972.

[6] Ian J. Goodfellow, Dumitru Erhan, Pierre Luc Carrier, Aaron Courville, Mehdi Mirza, Ben Hamner, Will Cukierski, Yichuan Tang, David Thaler, Dong-Hyun Lee, Yingbo Zhou, Chetan Ramaiah, Fangxiang Feng, Ruifan Li, Xiaojie Wang, Dimitris Athanasakis, John Shawe-Taylor, Maxim Milakov, John Park, Radu Ionescu, Marius Popescu, Cristian Grozea, James Bergstra, Jingjing Xie, Lukasz Romaszko, Bing Xu, Zhang Chuang, and Yoshua Bengio. Challenges in Representation Learning: A report on three machine learning contests. In *ICONIP 2013: Neural Information Processing*, 2013.

[7] Joseph Howse and Steven Puttemans. Cascade Classifier on OpenCV, 2015.

[8] Sergey Ioffe and Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift, 2015.

[9] Source: Kaggle. Challenges in Representation Learning: Facial Expression Recognition Challenge, 2013.

[10] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization, 2014.

[11] David Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60:91–, 11 2004.

[12] Source: Microsoft. FER+ data set download, 2017.

[13] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[14] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[15] Florian Rançon, Lionel Bombrun, Barna Keresztes, and Christian Germain. Comparison of SIFT Encoded and Deep Learning Features for the Classification and Detection of Esca Disease in Bordeaux Vineyards. *Remote Sens*, 11(1):1–26, 2019.

[16] Olga Russakovsky, Hao Su Jia Deng, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115(1):211–252, 2014.

[17] Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition, 2014.

[18] Yichuan Tang. Deep Learning using Linear Support Vector Machines, 2013. ICML 2013 Challenges in Representation Learning Workshop.

[19] R. Vemulapalli and A. Agarwala. Google facial expression comparison data set, 2018. A Compact Embedding for Facial Expression Similarity.