BABEŞ BOLYAI UNIVERSITY, CLUJ NAPOCA, ROMÂNIA
FACULTY OF MATHEMATICS AND COMPUTER SCIENCE

Detection of emotions in preschoolers' behavior

MIRPR report

Pintea Carmen, CSen, 936
Pletosu Cosmin, CSen, 936
Ventaniun Iustinian, CSro, 237
Darie George, CSro, 233

2020

Abstract

With the transition of facial expression recognition (FER) from laboratory controlled to challenging in-the-wild conditions and the recent success of deep learning techniques in various fields, deep neural networks have increasingly been leveraged to learn discriminative representations for automatic FER. Recent deep FER systems generally focus on two important issues: overfitting caused by a lack of sufficient training data and expression-unrelated variations, such as illumination, head pose and identity bias. In this paper, we provide details about facial expression recognition in preschooler's behavior and implementation insights regarding the "Emotion Detection" application. This application aims to be a useful tool for software developers interested in getting a non-verbal feedback from users under the age of seven which do not possess writing and reading abilities.

Contents

Chapter 1

Introduction

1.1. What? Why? How?

What?

The present paper describes the application "Emotion Detection" which offers an objective analysis of children's facial emotions as users of an IT app. The main user of the application is the software developer interested in the non-verbal feedback from the users of his products.

Why?

The success or failure of an interactive application is determined by the usability of the product. One of the components of usability is user satisfaction. Satisfaction assessment for adult users is done through observation, post-interaction interviews or quantitative methods (questionnaires) that cannot be applied to preschool children, who have limited capacity for self-analysis and communication, cannot read and cannot write.

Thus, besides the observation (which may be subjective, depending on the interpretation of the preschooler's reactions by the expert who makes the application), we want an objective measurement of the emotions that children experience during the interaction. This requires the development of an application that allows the identification of the emotional states of a preschooler during the course of an activity (e.g. 30% of the time he smiled or associations between tasks that children do and the frequency of an emotion).

How?

For this purpose, the application uses an intelligent solution based on machine learning technology and offers an UX friendly graphical interface that facilitates the rapid analysis of emotions in real time.

As can be seen from the experiments to be presented, the best solution is to use a Deep Convolutional Neural Network trained on the datasets "CK+, Fer2013 and CAFE" augmented by various methods. A deep CNN has an architecture that consists of filter layers and a classification layer. A filter stage involves a convolutional layer, followed by a temporal pooling layer and a soft max unit.

The intelligent algorithm detects the facial emotions with **81% accuracy**.

During the implementation phase we followed the “guideline” illustrated below:

Chapter 2

Scientific problem

2.1. Problem definition

The main problems to be solved for the intelligent part of the application are: face detection and emotion classification. Face Detection is the first and essential step for emotion classification, and it is used to detect faces in the images. It is a part of object detection and can use in many areas such as security, bio-metrics, law enforcement, entertainment, personal safety, etc. Emotion detection consists of analyzing the facial expression and classifying the emotion into one of the 7 categories defined for preschoolers: neutral, happy, surprise, sad, angry, fear, disgust.

Neutral Angry Disgust

Why it must be solved by an intelligent algorithm?

These problems can only be solved by using intelligent algorithms. Neural networks have proven to be the best methods in terms of visual perception results due to their self-learning ability, in contrast to traditional methods that would require human-defined metrics and complex mathematical reasoning. For instance, if we try to establish similarities between two photographs of the same emotion expressed by two different children, we would notice that in reality the pictures differ greatly (due to the light, position, facial features, colors, hair length and so on). Therefore, defining a distance for classification is not deductible.

Advantages and disadvantages of using either a linear method or an intelligent method.

	Intelligent method	Linear method
Metrics have to be defined	✗	✓
Self-learning	✓	✗
Large data set needed	✓	✗
Handle the variation of data	✓	✗

This paper is aimed at solving the problems of face detection and classification of emotion in preschooler's behavior. The face detection algorithm takes as **input** an image and **outputs** a cropped image which frames only the face and the emotion classification algorithm takes as **input** the cropped image and **outputs** the percentages for each of the seven emotions: neutral, happy, neutral, happy, surprise, sad, angry, fear, disgust.

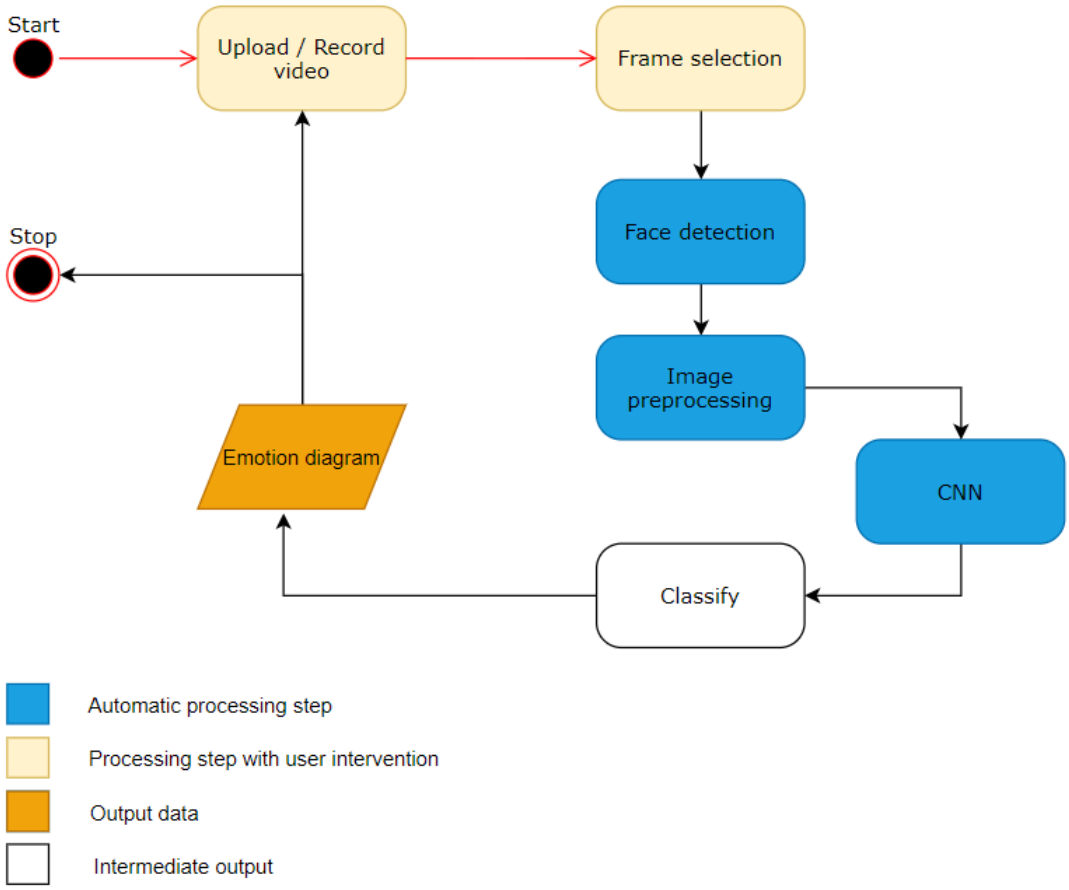
Why is this problem challenging?

Datasets with children faces are difficult to collect (parental consent is required, children find it difficult to pretend), which is why the datasets available online contain few images and are not evenly distributed in relation to emotions (the predominant emotions being neutral, sad and happy). It is important to find solutions to define a model with the highest accuracy, as the training data for our objective are few:

- Combining the data set with children with a data set with adults and a diverse one
- Data augmentation (especially for the data set with children)

As an example, the following child is sad, but his mouth can express a happy face.

2.2. Graphic abstract



Chapter 3

Related work / State-of-the-art

Some of the relevant solutions to this problem are as follows:

Method 1 (J.Cai, Z.Meng, A.S. Khan, Z. Li) [1]

Dataset

CK+

Intelligent algorithm

Network type: CNN

Network size: 6

Preprocessing: Discriminative response map fitting

Results

7 classes: 90.66%

Description

A CNN consists of three types of heterogeneous layers: convolutional layers, pooling layers, and fully connected layers. The convolutional layer has a set of learnable filters to convolve through the whole input image and produce various specific types of activation feature maps. The pooling layer follows the convolutional layer and is used to reduce the spatial size of the feature maps and the computational cost of the network. Average pooling and max pooling are the two most commonly used nonlinear down-sampling strategies for translation invariance. The fully connected layer is usually included at the end of the network to ensure that all neurons in the layer are fully connected to activations in the previous layer and to enable the

Method 2 (G.E. Hinton, R.R. Salakhutdinov) [2]

Dataset

JAFFE

Intelligent algorithm

Network type: DAE (Deep autoencoder)

Network size: 3

Preprocessing: Active Appearance Mode

Results

7 classes: 95.79%

Description

In contrast to the traditional networks, which are trained to predict target values, the DAE is optimized to reconstruct its inputs by minimizing the reconstruction error. Variations of the DAE exist, such as the denoising autoencoder, which recovers the original undistorted input from partially corrupted data; the sparse autoencoder network (DSAE), which enforces sparsity on the learned feature representation; the convolutional autoencoder, which uses convolutional (and optionally pooling) layers for the hidden layers in the network; and the variational auto-encoder (VAE), which is a directed graphical model with certain types of latent variables to design complex generative models of data.

Method 3 (Y. Tang) [3]

Dataset

CK+

Intelligent algorithm

Learning model: SVM (Support Vector Machine)

Preprocessing: Bag of words, Scale-Invariant Feature Transform

Results

7 classes: 84.65%

Description

Support-vector machine is a supervised learning model that analyze data used for classification and regression analysis. Given a set of training examples, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier. An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on the side of the gap on which they fall. In addition to performing linear classification, SVMs can efficiently perform a non-linear classification using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces.

Method 4 (H. Yang, U. Ciftci, and L. Yin) [6]

Dataset

CK+

Intelligent algorithm

Network type: cGAN

Preprocessing: Mixtures of trees

Results

7 classes: 96.30%

Description

GAN was first introduced by Goodfellow et al [4] in 2014, which trains models through a minimax two-player game between a generator G(z) that generates synthesized input data by mapping latent z to data space with $z \sim p(z)$ and a discriminator D(x) that assigns probability $y = Dis(x) \in [0, 1]$ that x is an actual training sample to tell apart real from fake input data. The generator and the discriminator are trained alternatively and can both improve themselves by minimizing/maximizing the binary cross entropy LGAN . Extensions of GAN exist, such as the cGAN [5] that adds a conditional information to control the output of the generator, the DCGAN that adopts deconvolutional and convolutional neural networks to implement G and D respectively.

Chapter 4

Proposed approach

The application that will be deeply described in the next paragraphs uses Haar Cascade algorithm for the detection of girls and a CNN for the classification of preschool emotion.

In the following we will exemplify the steps for the next input:

Step 1. Face detection

The application takes a video as an input and divides it into frames. In each frame is identified the face of the subject by using the Haar cascade algorithm.

Firstly, we have to convert to grayscale the image:

Haar Cascade

Haar Cascade is a machine learning object detection algorithm used to identify objects in an image or video and is based on the concept of features proposed by Paul Viola and Michael Jones in their paper “Rapid Object Detection using a Boosted Cascade of Simple Features” in 2001.

It is a machine learning based approach where a cascade function is trained from a lot of positive and negative images. It is then used to detect objects in other images.

The algorithm has four stages:

1. Haar Feature Selection (A Haar-like feature considers adjacent rectangular regions at a specific location in a detection window, sums up the pixel intensities in each region and calculates the difference between these sums)
2. Creating Integral Images (An integral image is a summed-area table for quickly and efficiently generating the sum of values in a rectangular subset of a grid)
3. Adaboost training (AdaBoost training process selects only those features known to improve the predictive power of the model, reducing dimensionality)
4. Cascading Classifiers

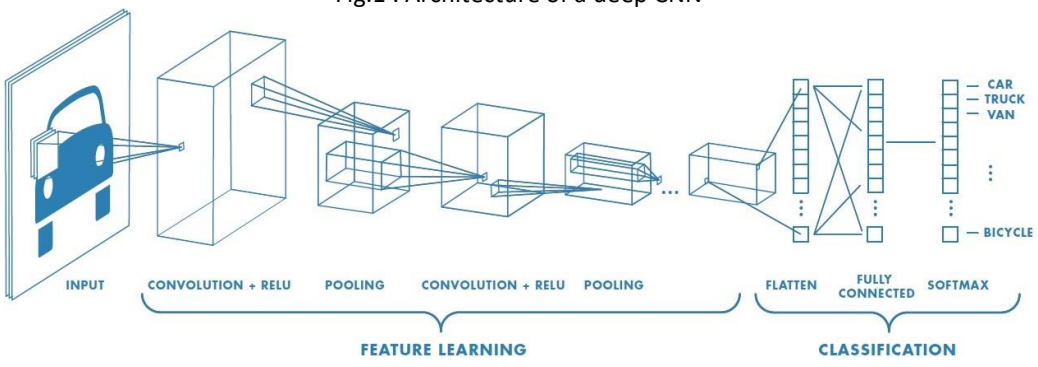
The result of applying Haar Cascade on the image:

Step 2. Emotion classification

In neural networks, Convolutional neural network (ConvNets or CNNs) is one of the main categories to do image recognition and image classification.

CNN image classifications take an input image, process it and classify it under certain categories (E.g., happy kid, angry kid, sad kid). Computers sees an input image as array of pixels and it depends on the image resolution. Based on the image resolution, it will see **h** x **w** x **d**
(h = Height, w = Width, d = Dimension).

Fig.1 : Architecture of a deep CNN

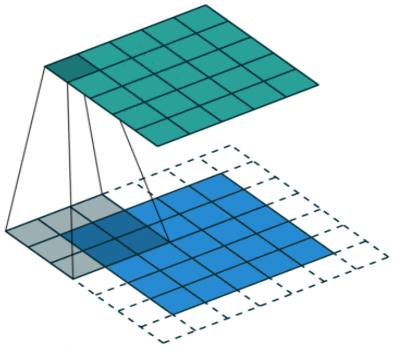


As we can see in Fig.1 a CNN has two parts: feature learning and classification.

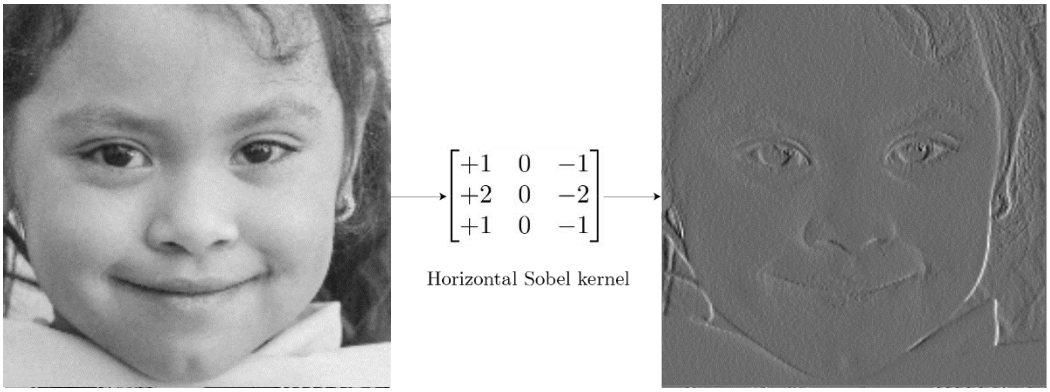
A) Feature learning

a. Convolution Layer

Convolution is the first layer to extract features from an input image. Convolution preserves the relationship between pixels by learning image features using small squares of input data. It is a mathematical operation that takes two inputs such as image matrix and a filter or kernel.



Example of Kernels: Fig.3 Vertical edge detector kernel



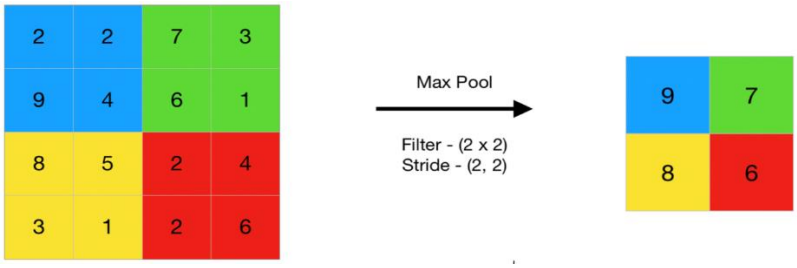
b. Pooling layer

A pooling layer is a new layer added after the convolutional layer. Specifically, after a nonlinearity (e.g. ReLU) has been applied to the feature maps output by a convolutional layer; for example the layers in a model may look as follows:

The addition of a pooling layer after the convolutional layer is a common pattern used for ordering layers within a convolutional neural network that may be repeated one or more times in a given model.
The pooling layer operates upon each feature map separately to create a new set of the same number of pooled feature maps.
Pooling involves selecting a pooling operation, much like a filter to be applied to feature maps. The size of the pooling operation or filter is smaller than the size of the feature map; specifically, it is almost always 2×2 pixels applied with a stride of 2 pixels.

This means that the pooling layer will always reduce the size of each feature map by a factor of 2, e.g. each dimension is halved, reducing the number of pixels or values in

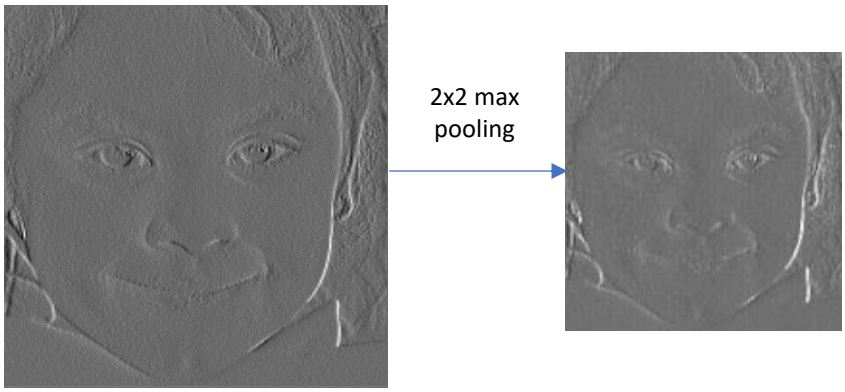
each feature map to one quarter the size. For example, a pooling layer applied to a feature map of 6×6 (36 pixels) will result in an output pooled feature map of 3×3 (9 pixels).



The pooling operation is specified, rather than learned. Two common functions used in the pooling operation are:

Average Pooling: Calculate the average value for each patch on the feature map.

Maximum Pooling (or Max Pooling): Calculate the maximum value for each patch of the feature map.

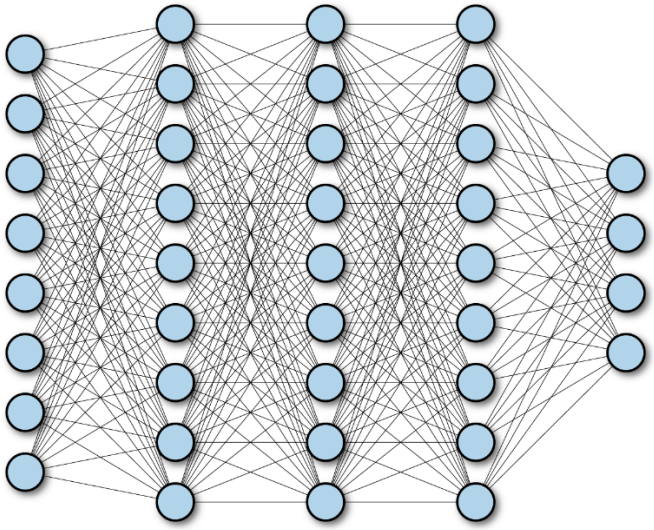


Now, the main features are extracted and the image can be passed through the classification neural network.

B) Classification

a. Fully connected layers

A fully connected neural network consists of a series of fully connected layers. A fully connected layer is a function from $\mathbb{R}$ m to $\mathbb{R}$ n . Each output dimension depends on each input dimension. Pictorially, a fully connected layer is represented as follows in Fig. 4.



Let's dig a little deeper into what the mathematical form of a fully connected network is. Let x ∈ $\mathbb{R}$ m represent the input to a fully connected layer. Let $y_i \in \mathbb{R}$ be the i -th output from the fully connected layer. Then $y_i \in \mathbb{R}$ is computed as follows:

$y_i = \sigma (w_{1 \times 1} + \cdots + w_{m \times m})$

Here, σ is a nonlinear function (for now, think of σ as the sigmoid function introduced in the previous chapter), and the w_i are learnable parameters in the network.

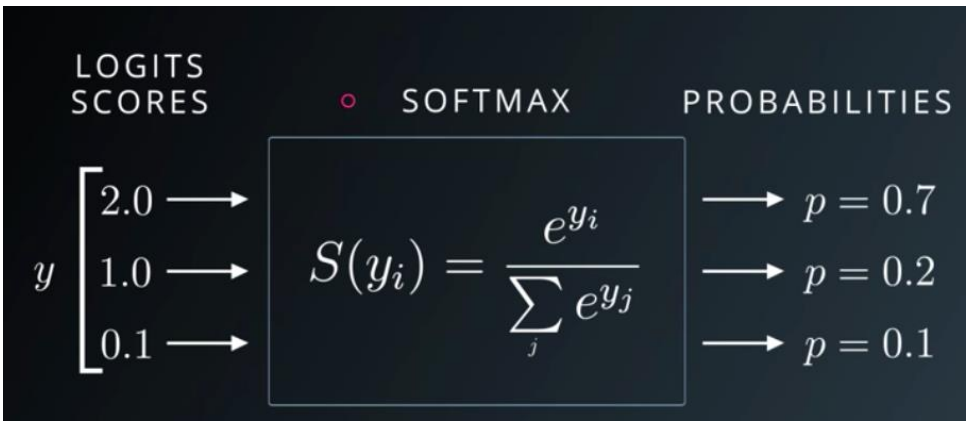
"Neurons" in Fully Connected Networks

The nodes in fully connected networks are commonly referred to as "neurons." Consequently, elsewhere in the literature, fully connected networks will commonly be referred to as "neural networks."

b. Softmax activation function

The Softmax regression is a form of logistic regression that normalizes an input value into a vector of values that follows a probability distribution whose total sums up to 1. The output values are between the range [0,1] which is nice because we are able to avoid binary classification and accommodate as many classes or dimensions in our neural network model.

This is why softmax is sometimes referred to as a multinomial logistic regression.



As an aside, another name for Softmax Regression is Maximum Entropy (MaxEnt) Classifier.

The function is usually used to compute losses that can be expected when training a data set. Known use-cases of softmax regression are in discriminative models such as Cross-Entropy and Noise Contrastive Estimation. These are only two among various techniques that attempt to optimize the current training set to increase the likelihood of predicting the correct word or sentence.

If you look at it from the outset, the definition may sound off as trivial but in the realm of Machine Learning and NLP, this regression function has been useful as a baseline comparator. Researchers who design new solutions have to carry out experimentation keeping the softmax results as a reference.

However, it should be noted that softmax is not ideally used as an activation function like Sigmoid or ReLU (Rectified Linear Units) but rather between layers which may be multiple or just a single one.

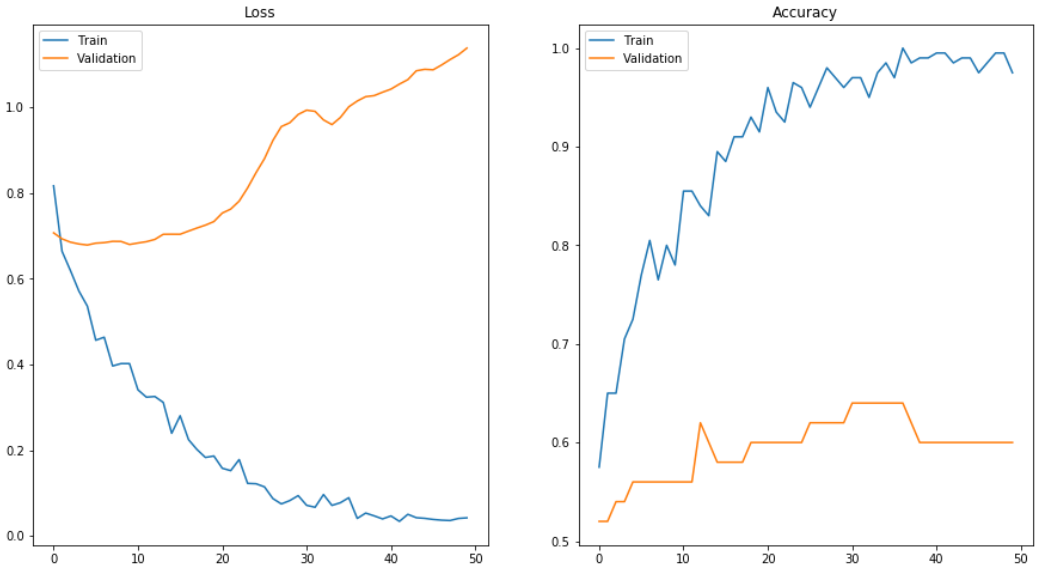
For our example, the model predicted the following distribution of emotions:

Angry Disgust Fear Happy Sad Surprise Neutral

[[0.000700 0.000000 0.000800 0.803800 0.004500 0.000400 0.189800]]

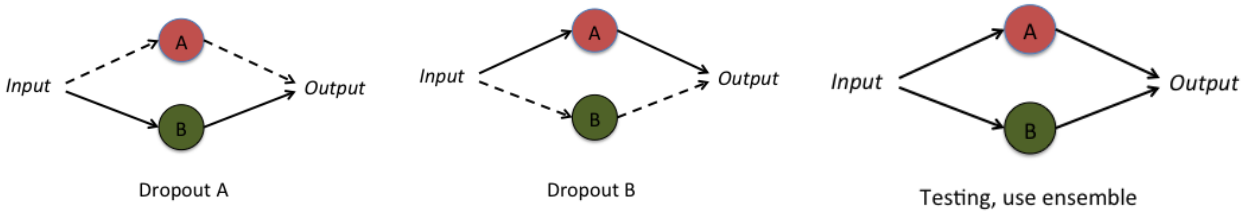
Overfitting prevention

In statistics, overfitting is "the production of an analysis that corresponds too closely or exactly to a particular set of data, and may therefore fail to fit additional data or predict future observations reliably". An overfitted model is a statistical model that contains more parameters than can be justified by the data. The essence of overfitting is to have unknowingly extracted some of the residual variation (i.e. the noise) as if that variation represented underlying model structure.



A) Dropout layer

Dropout layers provide a simple way to avoid overfitting. The primary idea is to randomly drop components of neural network (outputs) from a layer of neural network. This results in a scenario where at each layer more neurons are forced to learn the multiple characteristics of the neural network.



B) Early stopping

Too many epochs can lead to overfitting of the training dataset, whereas too few may result in an underfit model. Early stopping is a method that allows you to specify an arbitrary large number of training epochs and stop training once the model performance stops improving on a holdout validation dataset.

Improving the AI algorithm

A) Batch normalization

First introduced in the paper: Accelerating Deep Network Training by Reducing Internal Covariate Shift (https://arxiv.org/pdf/1502.03167.pdf).

As the data flows through a deep network, the weights and parameters adjust those values, sometimes making the data too big or too small again - a problem the authors refer to as "internal covariate shift". By normalizing the data in each mini-batch, this problem is largely avoided. Batch Normalization normalizes each batch by both mean and variance reference.

We normalize the input layer by adjusting and scaling the activations. Batch normalization reduces the amount by what the hidden unit values shift around (covariance shift).

How does batch normalization work?

To increase the stability of a neural network, batch normalization normalizes the output of a previous activation layer by subtracting the batch mean and dividing by the batch standard deviation.

However, after this shift/scale of activation outputs by some randomly initialized parameters, the weights in the next layer are no longer optimal. SGD (Stochastic gradient descent) undoes this normalization if it’s a way for it to minimize the loss function.

Consequently, batch normalization adds two trainable parameters to each layer, so the normalized output is multiplied by a “standard deviation” parameter (gamma) and add a “mean” parameter (beta). In other words, batch normalization lets SGD do the denormalization by changing only these two weights for each activation, instead of losing the stability of the network by changing all the weights.

Benefits of Batch Normalization

- Networks train faster and converge much more quickly by eliminating the Internal Covariate Shift
- We can use higher learning rates because batch normalization makes sure that there’s no activation that’s gone really high or really low. And by that, things that previously couldn’t get to train, it will start to train.
- It reduces overfitting because it has a slight regularization effect. Similar to dropout, it adds some noise to each hidden layer’s activations. Therefore, if we use batch normalization, we will use less dropout, which is a good thing because we are not going to lose a lot of information. However, we should not depend only on batch normalization for regularization; we should better use it together with dropout.
- Makes more activation functions viable. Because batch normalization regulates the values going into each activation function, non-linarites that don't seem to work well in deep networks actually become viable again.

B) Progressive resizing

Progressive resizing is a technique for building CNNs that can be very helpful during the training and optimization phases of a machine learning project. The technique appears most prominently in Jeremy Howard's work, using it in the fast.ai course, "Deep Learning for Coders".

One trouble is that a single neural network can only work with standardly-sized images; too-small images must be scaled up and too-large images must be scaled down.

If your goal is model accuracy, larger images are obviously better. But there is a lot of advantage to starting with small images.

To understand why, we must first understand that the most important features of an image classification problem are "large". Properly tuned gradient descent naturally favors robust, well-supported features in its decision-making. In the image classification case this translates into features occupying as many pixels in as many of the sample images as possible.

The practical result is that while a model trained on very small images will learn fewer features than one trained on very large images, it will still learn. Thus, a model architecture that works on small images will generalize to larger ones.

Meanwhile, small-image models are much faster to train. Since small-image models generalize well to larger input sizes, and since they take less time to train, and since the first batch of models we build are going to be highly experimental anyway, we can train our first model on small data and worry about scaling up the images and the models later.

We started by building a classifier that performs well on tiny n x n (48 x 48) images. The next step is scaling our model up to 2n x 2n (96 x 96) images. We do this using transfer learning. Transfer learning is the technique of re-using layers and weights from previous models when building new ones. In our case, this will mean taking the model we just built, freezing it (so that further training won't make any changes to our existing weights), and injecting its layers into a new model (one which takes up scaled 96 x 96 images as input).

The work we do at this stage is limited to finding a configuration of good "feeder layers" we can prefix our old model with. We instantiated a new model with a new 96 x 96 convolutional layer and appended a max pooling layer, which down samples 96 x 96 -> 48 x 48 (the expected input size of our old model). We reattach the rest of the layers of our old model onto the new one and freeze the old convolutional and fully-connected layer weights in place.

We have created a new model that trains on 96 x 96 pixels images which reuses our old 48 x 48 classifier internally. This new model improves performance from ~73% to ~81%.

Chapter 5

Experimental methods obtained

1. Data

For the made experiments, in the training and testing phases the following datasets were used:

1.1. Fer 2013

The FER2013 database was introduced during the ICML 2013 Challenges in Representation Learning. All images have been registered and resized to 48*48 pixels after rejecting wrongfully labeled frames and adjusting the cropped region. FER2013 contains 28,709 training images, 3,589 validation images and 3,589 test images with seven expression labels (anger, disgust, fear, happiness, sadness, surprise and neutral).

It does not include children in the dataset.

Samples:

1.2. CK+

The Extended Cohn-Kanade (CK+) database is the most extensively used laboratory-controlled database for evaluating FER systems. CK+ contains 593 video sequences from 123 subjects. 327 sequences from 118 subjects are labeled with seven basic expression labels (anger, contempt, disgust, fear, happiness, sadness, and surprise). For static-based methods, the most common data selection method is to extract the last one to three frames with peak formation and the first frame (neutral face) of each sequence.

It does not include children in the dataset.

Samples:

1.3. Jaffe

The Japanese Female Facial Expression (JAFFE) database is a laboratory-controlled image database that contains 213 samples of posed expressions from 10 Japanese females. Each person has 3 to 4 images with each of six basic facial expressions (anger, disgust, fear, happiness, sadness, and surprise) and one image with a neutral expression.

We used this database for testing only. It does not include children.

Samples:

1.4. CAFE

The Child Affective Facial Expression Set (CAFE) is a collection of photographs taken of 2- to 8-year-old children posing for 6 emotional facial expressions—sadness, happiness, surprise, anger, disgust, and fear—plus a neutral face. In total, 154 child-models pose each of these 7 expressions. The result is 1192 total photographs.

This set has only children 😊. It was our reference point for a greater accuracy.

Samples:

Data augmentation

Because the data sets with children are not large enough and unevenly distributed, we expanded the data by increasing them. For data augmentation, we used the following methods:

Additive Gaussian Noise

Additive Poisson Noise

Pepper noise

Image flip

Gaussian Blur

Histogram Equalization

2. Methodology

Used metrics for evaluating the quality of the machine learning algorithm:

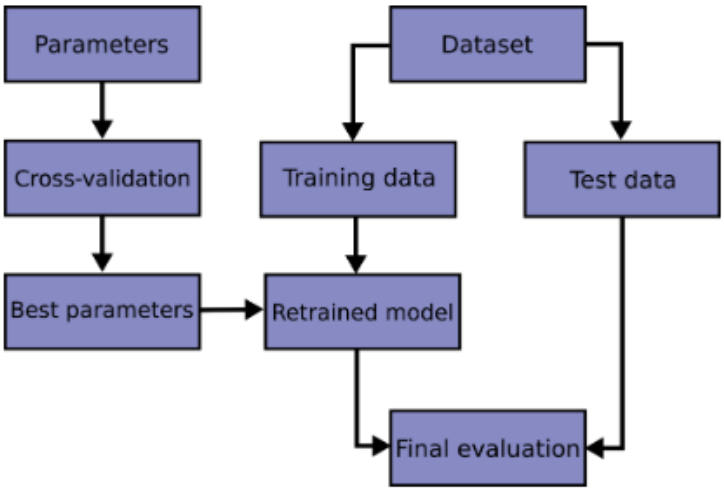
1. Classification Accuracy

Classification Accuracy is what we usually mean, when we use the term accuracy. It is the ratio of number of correct predictions to the total number of input samples.

$$Accuracy = \frac{Number\ of\ Correct\ predictions}{Total\ number\ of\ predictions\ made}$$

It works well only if there are equal number of samples belonging to each class.

To ensure that the model did not only learn the training data, we used the cross-validation method. **Cross validation (CV)** is one of the techniques used to test the effectiveness of a machine learning models, it is also a re-sampling procedure used to evaluate a model if we have a limited data. To perform CV, we need to keep aside a sample/portion of the data on which is do not use to train the model, later use this sample for testing/validating. We randomly split the complete data into training and test sets. Then Perform the model training on the training set and use the test set for validation purpose, ideally split the data into 70:30 or 80:20.



2. Categorical crossentropy

Categorical crossentropy is a loss function that is used for single label categorization. This is when only one category is applicable for each data point. In other words, an example can belong to one class only.

$$L(y,\hat{y}) = -\sum_{j=0}^{M}\sum_{i=0}^{N}(y_{ij} * log(\hat{y}_{ij}))$$

Categorical crossentropy will compare the distribution of the predictions (the activations in the output layer, one for each class) with the true distribution, where the probability of the true class is set to 1 and 0 for the other classes. To put it in a different way, the true class is represented as a one-hot encoded vector, and the closer the model's outputs are to that vector, the lower the loss.

3. Experiments and results

Experiment 1

Dataset: Fer2013
Input size: 48x48
Nr. Of features: 64
Data distribution:
Angry – 4593 images
Disgust – 547 images
Fear – 5121 images
Happy – 8989 images
Sad – 6077 images
Surprise – 4002 images
Neutral – 6198 images
Training set: 30887 photos
Validation set: 5233 photos
Training: 36 epochs
Test on:
Fer2013 (test data) – 70% accuracy
Jaffe – 50%
CAFE + CK+ – 49%
Architecture:
[2 x CONV (3 x 3 x nr_of_features), RELU] — MAXP (2x2) — DROPOUT (0.5)
[2 x CONV (3 x 3 x 2 x nr_of_features), RELU] — MAXP (2x2) — DROPOUT (0.5)
[2 x CONV (3 x 3 x 2 x 2 x nr_of_features), RELU] — MAXP (2x2) — DROPOUT (0.5)
[2 x CONV (3 x 3 x 2 x 2 x 2 x nr_of_features), RELU] — MAXP (2x2) — DROPOUT (0.5)
Dense (512), RELU — DROPOUT (0.5)
Dense (256), RELU — DROPOUT (0.5)
Dense (128), RELU — DROPOUT (0.5)
Dense (7), RELU -> Softmax
Optimizer: Adam (learning_rate = 0.001, beta_1 = 0.9, beta_2 = 0.999, epsilon = 1e-7)
Loss function: Categorical cross entropy
Overfitting prevention:
ReduceLROnPlateau(factor = 0.9, patience = 3)
EarlyStopping (min_delta = 0, patience = 8)

Experiment 2

Dataset: Fer2013
Input size: 48x48
Nr. Of features: 64
Data distribution:
Angry – 4593 images
Disgust – 547 images
Fear – 5121 images
Happy – 8989 images
Sad – 6077 images
Surprise – 4002 images
Neutral – 6198 images
Training set: 30887 photos
Validation set: 5233 photos
Training: 32 epochs
Test on:
Fer2013 (test data) – 71% accuracy (close to state-of-the-art)
Jaffe – 52%
CAFE + CK+ – 50%
Architecture:
[CONV (3 x 3 x nr_of_features), RELU] — MAXP (2x2) — DROPOUT (0.5)
[CONV (3 x 3 x 2 x nr_of_features), RELU] — MAXP (2x2) — DROPOUT (0.5)
[CONV (3 x 3 x 2 x 2 x nr_of_features), RELU] — MAXP (2x2) — DROPOUT (0.5)
[CONV (3 x 3 x 2 x 2 x 2 x nr_of_features), RELU] — MAXP (2x2) — DROPOUT (0.5)
Dense (512, RELU) — DROPOUT (0.3)
Dense (256, RELU) — DROPOUT (0.3)
Dense (128, RELU) — DROPOUT (0.3)
Dense (64, RELU) — DROPOUT (0.5)
Dense (7, RELU) -> Softmax
Optimizer: Adam (learning_rate = 0.001, beta_1 = 0.9, beta_2 = 0.999, epsilon = 1e-7)
Loss function: Categorical cross entropy
Overfitting prevention:
ReduceLROnPlateau(factor = 0.9, patience = 3)
EarlyStopping (min_delta = 0, patience = 4)

Observations

With the new architecture the training phase was faster. We can see that the accuracy is slightly greater.

Experiment 3

Dataset: Fer2013 + CAFE + CK+
Input size: 48x48
Nr. Of features: 64
Data distribution:
Angry – 6224 images
Disgust – 6408 images
Fear – 4648 images
Happy – 6864 images
Sad – 4844 images
Surprise – 5468 images
Neutral – 5846 images
* **IMPROVEMENT**: Uniform distribution of classes + data augmentation (only on **CAFE** and CK+)
Training set: 32241 photos
Validation set: 8061 photos
Training: 38 epochs
Test on:
Fer2013 + CAFE + CK+ (test data) – 78% accuracy
Architecture:
[CONV (3 x 3 x nr_of_features), RELU] — MAXP (2x2) — DROPOUT (0.5)
[CONV (3 x 3 x 2 x nr_of_features), RELU] — MAXP (2x2) — DROPOUT (0.5)
[CONV (3 x 3 x 2 x 2 x nr_of_features), RELU] — MAXP (2x2) — DROPOUT (0.5)
[CONV (3 x 3 x 2 x 2 x 2 x nr_of_features), RELU] — MAXP (2x2) — DROPOUT (0.5)
Dense (512, RELU) — DROPOUT (0.3)
Dense (256, RELU) — DROPOUT (0.3)
Dense (128, RELU) — DROPOUT (0.3)
Dense (64, RELU) — DROPOUT (0.5)
Dense (7, RELU) -> Softmax
Optimizer: Adam (learning_rate = 0.001, beta_1 = 0.9, beta_2 = 0.999, epsilon = 1e-7)
Loss function: Categorical cross entropy
Overfitting prevention:
ReduceLROnPlateau(factor = 0.9, patience = 3)
EarlyStopping (min_delta = 0, patience = 4)

Observations

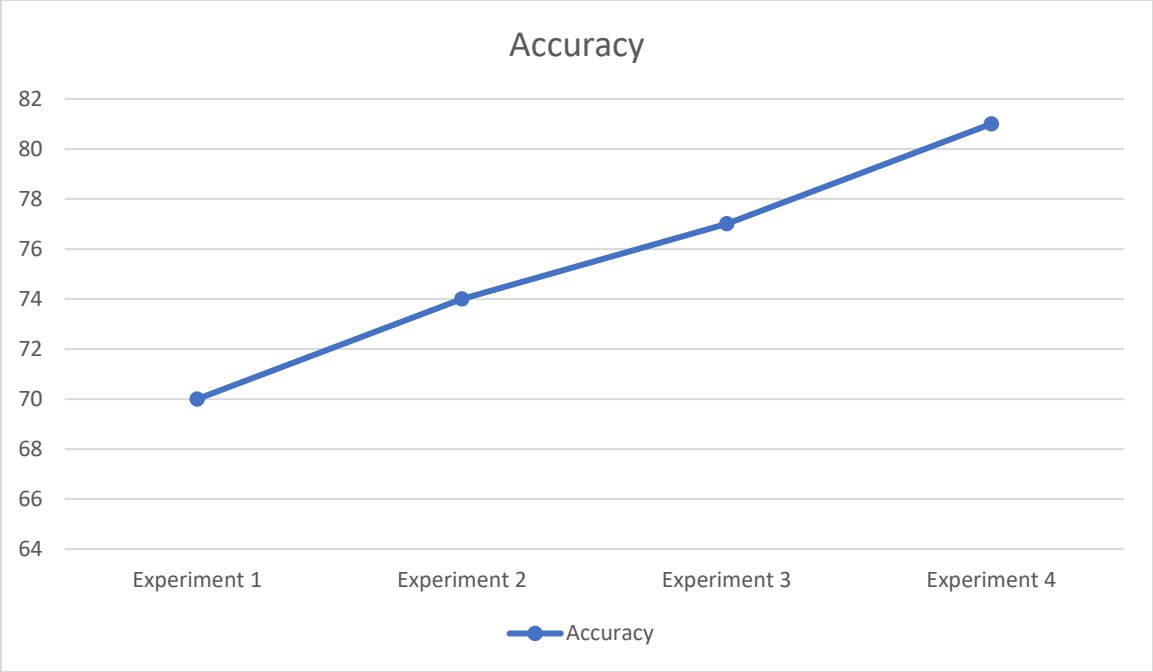
Accuracy increased with 7 percentages.

Experiment 4

Dataset: Fer2013 + CAFE + CK+
Input size: 96x96 (the new layer was trained only on CAFE + CK+)
Nr. Of features: 64
Data distribution:
Angry – 6224 images
Disgust – 6408 images
Fear – 4648 images
Happy – 6864 images
Sad – 4844 images
Surprise – 5468 images
Neutral – 5846 images
*** IMPROVEMENT**: Uniform distribution of classes + data augmentation (only on **CAFE** and CK+)
Training set: 32241 photos
Validation set: 8061 photos
Training: 36 epochs
Test on:
Fer2013 + CAFE + CK+ (test data) – 81% accuracy
Architecture:
* [CONV (3 x 3 x nr_of_features), RELU] – MAXP (3x3) – Batch Normalization()
[CONV (3 x 3 x nr_of_features), RELU] — MAXP (2x2) — DROPOUT (0.5)
Batch Normalization
[CONV (3 x 3 x 2 x nr_of_features), RELU] — MAXP (2x2) — DROPOUT (0.5)
Batch Normalization
[CONV (3 x 3 x 2 x 2 x nr_of_features), RELU] — MAXP (2x2) — DROPOUT (0.5)
Batch Normalization
[CONV (3 x 3 x 2 x 2 x 2 x nr_of_features), RELU] — MAXP (2x2) — DROPOUT (0.5)
Dense (512, RELU) — DROPOUT (0.3)
Dense (256, RELU) — DROPOUT (0.3)
Dense (128, RELU) — DROPOUT (0.3)
Dense (64, RELU) — DROPOUT (0.5)
Dense (7, RELU) -> Softmax
Optimizer: Adam (learning_rate = 0.001, beta_1 = 0.9, beta_2 = 0.999, epsilon = 1e-7)
Loss function: Categorical cross entropy
Overfitting prevention:
ReduceLROnPlateau(factor = 0.9, patience = 3)
EarlyStopping (min_delta = 0, patience = 4)

Observations

*** This convolutional layer was added for progressive resizing.**
This experiment led us to the best result.



4. Discussion

It should be noted that the experimental results do not totally coincide with those observed during the use of the application by a child. The reasons why it differs may be:

- The position of the camera towards the subject (training data are frontal pictures)
- Children are constantly moving
- Quality of the images differ
- Overfitting (the model behaves well only on data similar to CAFE)
- Much of the data set was with adults

During the experiments, we noticed the importance of the size and diversity of the data set. Data augmentation increased accuracy by 7 percent through image diversification.

Also, an important step was the usage of progressive resizing. It further increased the accuracy with 3 percentages.

In comparison to other methods, we obtained less accuracy. What we did not consider is a preprocessing of the images different from the one made by the convolutional layer. We can conclude that a preprocessing would have improved the accuracy of the algorithm and would have helped reduce the overfitting.

However, on the Fer2013 dataset, the results are close to the state-of-the-art solution.

Chapter 6

Emotion detection application doc

Problem statement:

Whether you are developing an application or making a presentation, feedback from users will always be helpful to improve your project. Collecting feedback from your users can sometimes be difficult, even more so if your target audience is composed of young kids. The most common practice is to ask your users to complete a questionnaire, but this can not be applied for young kids.

Proposed solution:

Our project's objective is to offer an easy way for anyone interested in developing a project for a young target audience to collect feedback from their users.

The concept is simple, our users will have access to an easy to use interface where they can upload a video record of a kid using their application or watching their presentation. The video is processed using a neural network specialized in kids emotion recognition. After the processing is done they will be able to view a detailed report of the detected emotions. All the data that was processed will be saved in a database, allowing our users to view older reports at will.

Our application's features are:

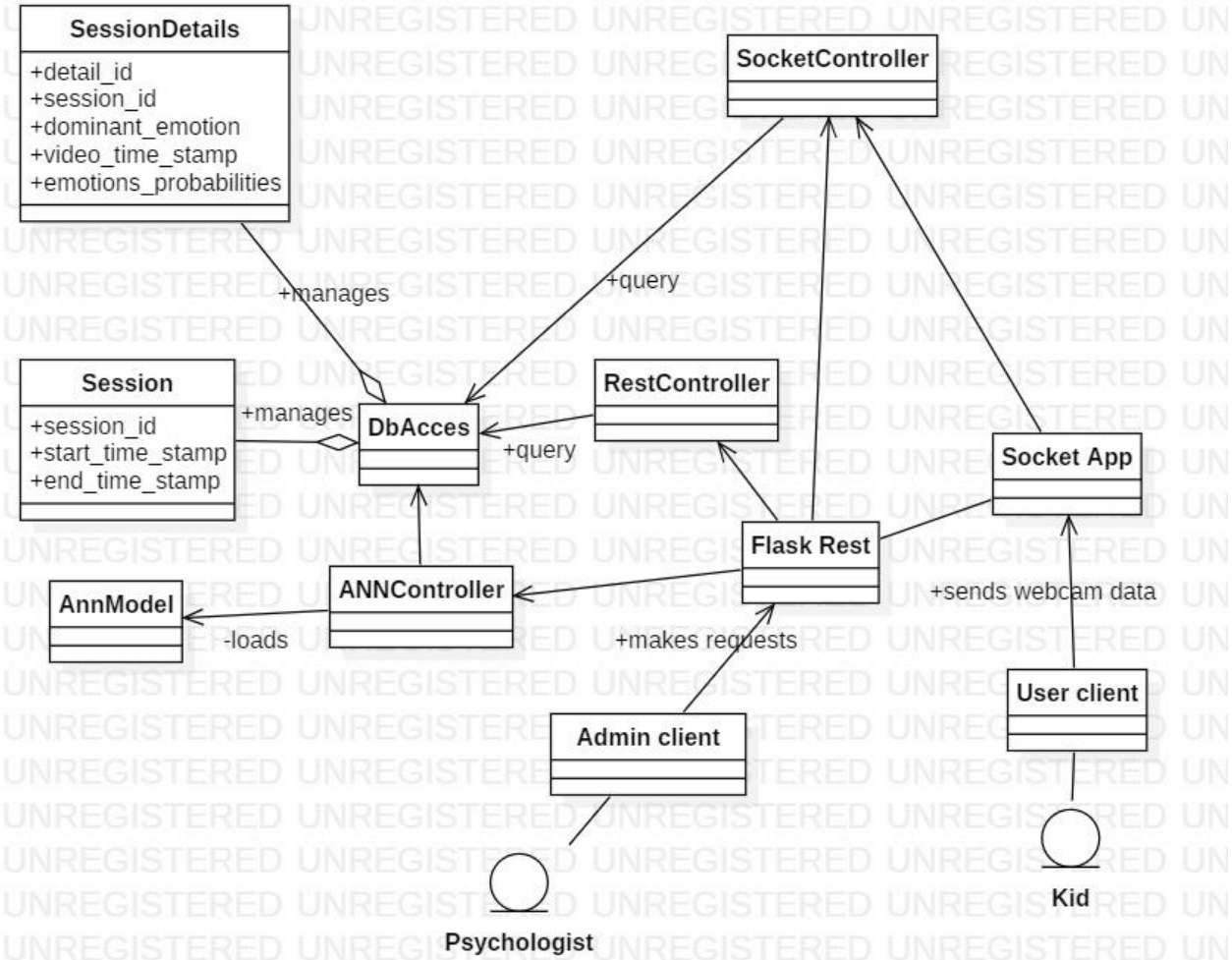
F1. Video upload: Users select the video upload tab in their user interface, select the video they are willing to analyze and click send. They will be notified via a pop-up when the video has been processed.

F2. Review report: Users select the video they want to review from a list on their main tab and they are presented with a pie chart showing an overview of the emotions recognized through the video. They will also be able to navigate through the clip in order to see a pie chart of the most dominant emotions second by second.

Solution architecture:

The application will be developed using React, Redux for front-end and Python, Flask for back-end and will load the neural network using Keras.

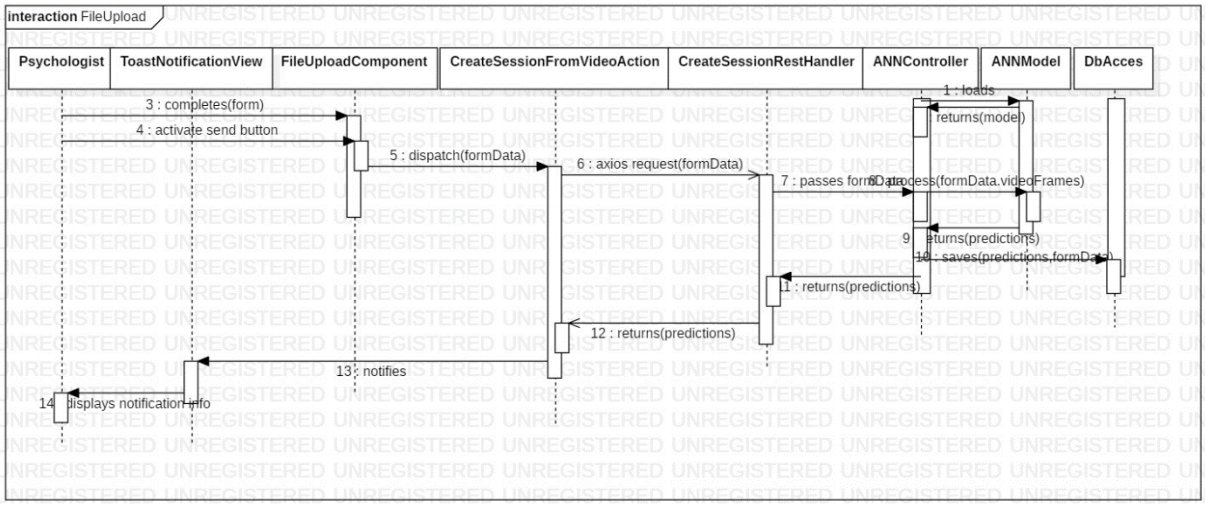
Backend classes diagram:



Note: Diagram has been simplified in order to make it easier to visualize.

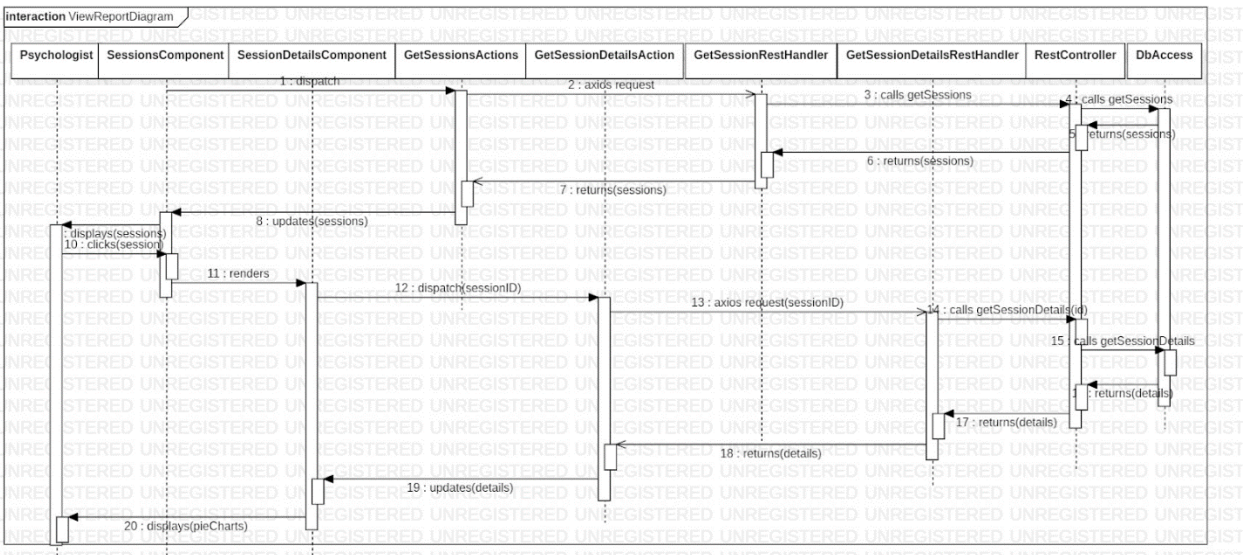
Sequence diagrams:

F1. Video upload:



Note: Diagram has been simplified in order to make it easier to visualize.

F2. Review report:



Note: Diagram has been simplified in order to make it easier to visualize.

Chapter 7

Conclusions and future work

Despite the fact that we have not been able to obtain an accuracy as great as the references mentioned, we can see that we have managed to make improvements from the starting point: an architecture taken from an online article. Through the development of this project and our attempts to improve it, we have gained a broader view on the various technologies and methods in the machine learning field.

After trying and succeeding to recognize facial expressions, we are really enthusiast about Keras. Our previous experience with AI was purely theoretical, nevertheless I could apply the knowledge to build a practical example.

Major shortcomings of our method are:

- Limited hardware for testing more varied models. Even if on the experimentation phase we trained on small data, each new training step after a small change was very time consuming
  Future work: To overcome this problem we would have needed a larger RAM capacity or a better video card for efficient parallelization.
- A better image preprocessing could have led to greater accuracy.
  Future work: Some alternative methods would be:
  - Autoencoder
  - Principal Component Analysis
  - Discriminative response map fitting
  - Bag of words
  - Mixture of trees
- The data set with children was quite small
  Future work: A larger data set can feed the CNN better for a greater accuracy and less overfitting
- Several more augmentation methods can be used
  We limited it to 6 due to hardware limitations
- In real data, children are agitated
  Future work: A better data set containing photos from many angles
- Progressive resizing was used with a basic format; some improvements can be made.
  Future work:
  - Increasing or decreasing the number of new convolutional layers you add.
  - Tuning hyper-parameters like activation functions and learning rates.
  - Unfreezing one or more preexisting convolutional layers and (re-)training those as well.

Chapter 8

References

[1] J. Cai, Z. Meng, A. S. Khan, Z. Li, J. OReilly, and Y. Tong, “Island loss for learning discriminative features in facial expression recognition,” in Automatic Face & Gesture Recognition (FG 2018), 2018 13th IEEE International Conference on. IEEE, 2018, pp. 302–309.

[2] G. E. Hinton and R. R. Salakhutdinov, “Reducing the dimensionality of data with neural networks,” science, vol. 313, no. 5786, pp. 504–507, 2006.

[3] Y. Tang, “Deep learning using linear support vector machines,” arXiv preprint arXiv:1306.0239, 2013.

[4] I. Goodfellow, J. Pouget - Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial nets,” in Advances in neural information processing systems, 2014, pp. 2672– 2680.

[5] M. Mirza and S. Osindero, “Conditional generative adversarial nets,” arXiv preprint arXiv:1411.1784, 2014.

[6] H. Yang, U. Ciftci, and L. Yin, “Facial expression recognition by de expression residue learning,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 2168–2177.

[7]https://towardsdatascience.com/build-a-simple-image-retrieval-system-with-an-autoencoder-673a262b7921

[8] https://arxiv.org/pdf/1804.08348.pdf?fbclid=IwAR29EfU401IM2WJg-4gMOEf847n1R9WZff2mzITJdse2BUQgOGzMJZvShUU

[9] https://medium.com/datadriveninvestor/real-time-facial-expression-recognition-f860dacfeb6a

[10]https://towardsdatascience.com/classifying-facial-emotions-via-machine-learning-5aac111932d3

[11]https://github.com/opencv/opencv/blob/master/data/haarcascades/haarcascade_frontalface_alt.xml

[12]https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_photo/py_non_local_means/py_non_local_means.html

[23]https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_feature2d/py_sift_intro/py_sift_intro.html

[14] https://towardsdatascience.com/bag-of-visual-words-in-a-nutshell-9ceea97ce0fb

[15] https://en.wikipedia.org/wiki/Support-vector_machine

[16] Cohn-Kanade http://www.consortium.ri.cmu.edu/ckagree/

[17] FER https://www.kaggle.com/c/challenges-in-representation-learning-facial-expression-recognition-challenge/overview

[18] CAFE (for kids) link](http://databrary.org/volume/30) https://www.childstudycenter-rutgers.com/the-child-affective-facial-expression-se