Combining Modality Specific Deep Neural Networks for Emotion Recognition in Video

Samira Ebrahimi Kahou[1], Christopher Pal[1], Xavier Bouthillier[2], Pierre Froumenty[1], Çağlar Gülçehre[2],* , Roland Memisevic[2], Pascal Vincent[2], Aaron Courville[2], & Yoshua Bengio[2]
[1]École Polytechique de Montréal, Université de Montréal, Montréal, Canada
[2]Laboratoire d'Informatique des Systèmes Adaptatifs, Université de Montréal, Montréal, Canada
{samira.ebrahimi-kahou, christopher.pal, pierre.froumenty}@polymtl.ca
{bouthilx, gulcehrc, memisevr, vincentp, courvila, bengioy}@iro.umontreal.ca

ABSTRACT

In this paper we present the techniques used for the University of Montréal's team submissions to the 2013 Emotion Recognition in the Wild Challenge. The challenge is to classify the emotions expressed by the primary human subject in short video clips extracted from feature length movies. This involves the analysis of video clips of acted scenes lasting approximately one-two seconds, including the audio track which may contain human voices as well as background music. Our approach combines multiple deep neural networks for different data modalities, including: (1) a deep convolutional neural network for the analysis of facial expressions within video frames; (2) a deep belief net to capture audio information; (3) a deep autoencoder to model the spatio-temporal information produced by the human actions depicted within the entire scene; and (4) a shallow network architecture focused on extracted features of the mouth of the primary human subject in the scene. We discuss each of these techniques, their performance characteristics and different strategies to aggregate their predictions. Our best single model was a convolutional neural network trained to predict emotions from static frames using two large data sets, the Toronto Face Database and our own set of faces images harvested from Google image search, followed by a per frame aggregation strategy that used the challenge training data. This yielded a test set accuracy of 35.58%. Using our best strategy for aggregating our top performing models into a single predictor we were able to produce an accuracy of 41.03% on the challenge test set. These compare favorably to the challenge baseline test set accuracy of 27.56%.

1. INTRODUCTION

Deep neural network techniques have recently yielded impressive performance gains across a wide variety of competitive tasks and challenges. For example, a number of the world's leading industrial speech recognition groups have reported significant recognition performance gains through deep network techniques [11]. The high profile ImageNet large scale image recognition challenge [15] has also recently been won (by a wide margin) through the use of deep neural networks.

The Emotion Recognition in the Wild (EmotiW) challenge [6] is based on an extended form of the Acted Facial Expressions in the Wild (AFEW) database of Dhall et al. [7] in which short video clips extracted from feature length movies have been annotated for different emotions. Thus, unlike many of the other recent high profile results with deep learning, here we must deal with not just static images and human speech, but the rich and complex nature of movie scenes with potentially multiple human actors interacting with one another, an audio track that may contain multiple voices as well as background music from the film soundtrack.

In this paper we describe our entry into the EmotiW challenge. Our approach combines multiple emotion classifiers each based on a different data source. (1) We use a deep convolutional neural network based on the model of [15] for frame-based classification of facial expressions from aligned images of faces. To use this model to classify whole video clips, we have implemented a novel video frame aggregation strategy based on the use of support vector machines (SVMs). (2) We have developed a deep belief net (DBN)-based emotion classifier from the audio signal available with the video clips. This audio signal contains both speech and background music. (3) We also used deep autoencoder-based classifier that takes an activity recognition-type approach by modeling the spatio-temporal information produced by the human actions depicted within the entire scene. (4) We have employed a shallow network architecture similar to that of [4] that focuses on extracted features of the mouth of the primary human subject in the scene and uses these features as input to an SVM emotion classifier. Finally, we present a novel technique to aggregate models based on random hyperparameter search using low complexity aggregation techniques consisting of simple weighted averages. We compare the performance of these various models and their combination.

Through this work we make a number of contributions and observations from which we are able to draw a number of conclusions that we believe will be of more general interest. A core aspect of our approach is the use of a deep convolutional neural network for frame-based facial expression classification. To train this model, we made use of additional data composed of images of faces with expressions labeled

*See section 5 for additional authors.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
ICMI'13, December 9–13, 2013, Sydney, Australia
Copyright 2013 ACM 978-1-4503-2129-7/13/12 ...$15.00.
http://dx.doi.org/10.1145/2522848.2531745.

as one of seven basic emotions (angry, disgust, fear, happy, sad, surprise and neutral). The use of this additional data seems to have made a big difference in our performance by allowing us to train high capacity models without overfitting to the relatively small EmotiW challenge training data. Importantly, a direct measure of per frame errors on the challenge data does not yield performance that is superior to the challenge baseline; however, our strategy of using the challenge training data to learn how to aggregate the per frame predictions was able to boost performance substantially. Our audio model emerged as the second most important element of our approach as it captured complementary information to our top performing image based technique. Finally, while simple averaging of all models did not yield performance superior to a combined video-audio model, we developed a novel technique to aggregate models based on random search which was able to exploit the complementary information within the predictions of all models and which yielded our highest performing technique on the challenge test set. We outline all these models in further detail in section 2, discuss techniques for their aggregation in section 3, then make more detailed observations and conclusions in section 4.

2. MODELS

We begin here with a discussion of two different convolutional neural network techniques that we explored for facial expression recognition. We present the first model in section 2.1 in greater detail as it yielded higher performance on the challenge validation set. However, we also outline an alternative deep network architecture in less detail in section 2.2. We then present our deep restricted Boltzmann machine based audio model in section 2.3, our deep autoencoder based technique inspired by activity recognition techniques in section 2.4 and a shallow neural network model based on bag of mouth features, in section 2.5. We discuss the combination of these models in section 3.

2.1 Faces & Convolutional Network #1

As discussed above, a key aspect to our approach here is that we did not use the challenge data directly to perform learning with this deep neural network. The network is shown in figure 1 and we discuss both architecture and data preprocessing in further detail below. We trained this network with two large *static* image databases of facial expressions for the seven emotion categories: the Toronto Face Dataset (TFD)[18] and a large facial expression database harvested from Google image search then cleaned and labeled by hand [2]. The TFD contains 4,178 images labeled with basic emotions and composed of a number of other standard expression datasets, essentially with only fully frontal facing poses. All these images were preprocessed based on registering the eyes and then resized to 48x48. The Google dataset consists of 35,887 images with the seven expression classes: angry, disgust, fear, happy, sad, surprise and neutral. The dataset was built by harvesting imagery returned from Google's image search using keywords related to expressions. Images are in grayscale of the size 48x48. We then aggregated the results of this model applied to all frames of the AFEW2 challenge using an SVM. For each video, the seven per frame emotion predictions of the network are combined so as to produce a fixed length vector with ten blocks of seven predictions over the duration of the video. We discuss the network architecture and the complete processing and training procedure in more detail below, including our procedure for expanding and contracting the results of the per frame predictions across the variable number of frames that constitute each video.

Before continuing with our more detailed discussion, we note at this point that we evaluated a number of training strategies for the static frame deep network and aggregation strategy before selecting the exact configuration used for our first submission to the challenge. Most importantly, one of these strategies included a procedure in which every frame containing a face extracted from the video clips of the challenge data training set (using the complete processing pipeline, deep network architecture and training presented below) was also used to train the deep network. This is in sharp contrast with the alternative that we ended up using in which the challenge training data was not used to train the deep network and was only used to train an SVM based aggregator on the predictions from the deep network. While the strategy of including the challenge data in the deep network training yielded an accuracy of 96.73% on the challenge training data - which is dramatically better than our submitted models training set accuracy of 46.87%; however, the validation set accuracy fell to 35.32% which is considerably lower than the 38.96% validation set accuracy of our first submitted model in which the challenge training data was not used to learn the deep network and was only used to train the aggregator SVM. A learning curve for the deep network trained using this strategy is shown in figure 2.

Our best network of this type was therefore trained on the two combined 'extra' datasets, using early stopping based on the challenge train and validation sets. To simplify the remainder of our discussion here, we shall refer to the challenge train, validation and test data sets as the AFEW2 train, validation and test sets as they derive from the original AFEW database discussed in our introduction.

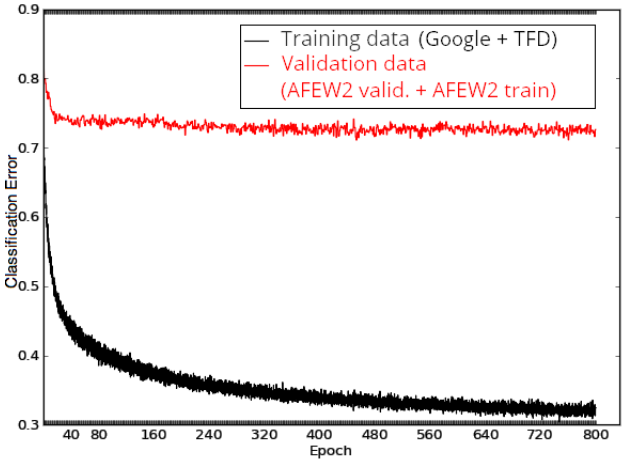


Figure 2: ConvNet 1 classification error on training and validation sets (before aggregation)

2.1.1 *Our AFEW2 Facetube Extraction Procedure*

Video frames were extracted from the AFEW2 challenge clips in a way that ensured the aspect ratio of the original movie was preserved. The Google Picasa face detector [8] was then used to detect and crop all faces detected in each frame. In order to get the bounding boxes from the output of Picasa, which consisted of cropped images we used Haar-like feature-based matching method as direct pixel to pixel

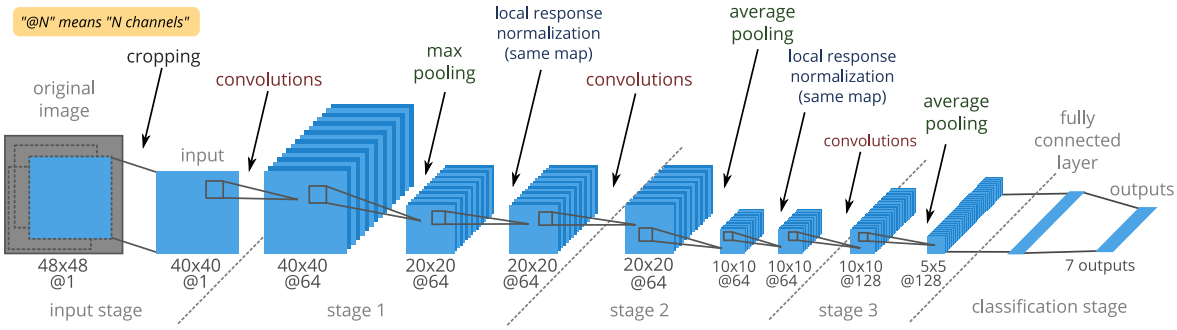


Figure 1: The convolutional neural network architecture used for our *ConvNet 1* experiments.

matching did not yield satisfactory performance. Since Picasa was not able to detect faces in all the frames, in those frames without any detected bounding box we looked in the spatial neighbourhood of the temporally closest bounding box and compared the histogram of color intensities to decide if a face was present or not. We also used a couple of basic heuristics such as the relative positions of bounding boxes, their sizes and overlap in consecutive frames to extract the final facetubes for each identity in each clip based on the bounding boxes from the previous steps.

In the AFEW2 test clips the Picasa face detector was not able to detect any face for 7 clips. We thus used the combined landmark placement and face detection technique of [22] to find faces in these clips. The facial key points returned by the model were then used to build the bounding boxes. The facetubes were then assembled with the same procedure described previously.

2.1.2 *Other Pre-processing of Datasets*

AFEW2 facetubes smoothing: In order to get images where faces sizes vary gradually in the images sequence, a smoothing procedure was applied on the AFEW2 facetube bounding boxes from section 2.1.1 For all images in a facetube, we first smooth the coordinates of the opposite corners of the bounding box with a 2-sided moving average (11 frame window size). The largest centered square is then drawn inside these smoothed bounding boxes, creating new, square bounding boxes which more tightly frame the faces. Next, a second smoothing is applied on the center of the bounding boxes with the same kind of moving average. By applying this second smoothing, the bounding boxes move gradually in the sequence of images.

Finally, in order to handle large changes of bounding box lengths that result from dramatic changes of camera position and magnification (e.g. changing from a medium shot to a close-up shot), we applied a further polynomial smoothing directly on the 2 bounding box size dimensions. We fit 2 low-order polynomials of 0 (constant) and 1 (linear) degree through the lengths of the bounding boxes sides. For a given facetube, if the 1-degree-polynomially-smoothed lengths change by a scale threshold (i.e. the *slope* $\times$ *facetube length* of the linear fit is above the threshold), then we use the values returned by the 1-degree polynomial as the lengths of the bounding box. Otherwise, the 0-degree-polynomial-smoothed bounding box lengths are used. Empirically we found a threshold of 1.5 to give reasonable results.

For each facetube, the faces are then cropped based on the smoothed bounding boxes and resized to 48x48.

Figure 3: Raw images at the top and their IS-preprocessed corresponding images below

Registration: The AFEW2 and TFD images are registered to the Google dataset using 51 facial points. These landmarks are detected by a landmark detection model which is based on mixture of trees [22] which capture pose variations. The model returns 68 points but we ignored the face contour for registration process. Images in Google dataset and AFEW2 have different poses but most of the time faces are frontal. To reduce the noise, we computed the mean shape with no-pose along each dataset and then computed the transformation between the two shapes. For transformation the Google data considered to be the base shape and the similarity transformation is used to compute the mapping. Once we have this mapping we map all other data to Google data. TFD only includes faces where Google data cropped with a small border around the face. To solve this issue we added a random noisy border to all TFD images.

Illumination normalization using isotropic smoothing : To compensate for illumination variation in all datasets, we used a diffusion-based approach [10]. We used isotropic smoothing (IS) function from INface toolbox [20, 17]. The smoothness parameter is set to default with no normalization as post-processing step. See figure 3 for an example.

2.1.3 *Convolutional Network #1*

This convolutional neural network (ConvNet) is based on the C++ and Cuda implementation of Krizhevsky et al. [14]. The architecture of our particular instantiation of this type of network is shown in Figure 1. The ConvNet input consists of images of size 40x40 that are cropped randomly from the original 48x48 images. These images are flipped horizontally with a probability of 0.5. At each epoch, the cropping and flipping are repeated and the cropped images are different. The ConvNet has 3 stages with different layers, the first 2 stages include a convolution layer followed by a max or average pooling layer, then a local response normalization layer (with the same mapping) and the third stage has a convolution layer followed by an average-pooling layer. This stage has 128000 units. The first stage has a max-

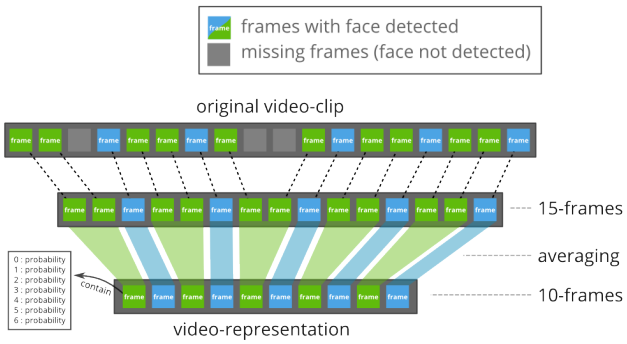


Figure 4: Frame aggregation via averaging

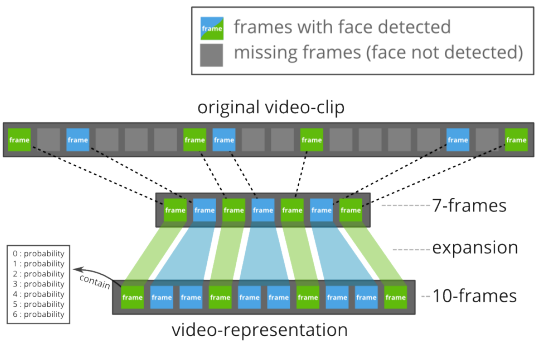


Figure 5: Frame aggregation via expansion

pooling layer whereas the second is using average-pooling. The last stage (classification) is a fully-connected layer with 7 units (classes) with a softmax layer as classifier. The test error is computed on patches cropped from center only. The early-stopping method is based on AFEW2 validation and train sets, and it was stopped at 453 epochs. The training is done on our extra data and the AFEW2 train is only used to train the SVM.

2.1.4 Per Frame Prediction Aggregation with an SVM

Using ConvNet 1, we classified all frames extracted from AFEW2 train, validation and test. For each frame the output is a 7-class probability vector. These probabilities are then aggregated to build a fixed-length representation for each video-clip. Given a video-clip, all frames extracted from the video are ordered based on time. Since, AFEW2 video-clips do not have the same number of frames and for some of the frames facetube extraction method failed to find any face, we transformed the probabilities into 10 bins by averaging the 7-class probabilities within 10 independent groups of frames. For video-clips with less than 10 frames, we expanded the frame sequence by simply repeating them uniformly. Finally, each video is represented by concatenating the averaged probabilities for each bin, which is a vector of 70 features. The processes of frame aggregation by averaging and expansion are shown in figures 4 and 5 respectively. Our aggregation method generates feature vectors for train, validation and test sets. For solving the final video classification problem, we used a support vector machine (SVM) and the implementation in [3] with a radial basis function kernel (RBF). The hyperparameters, γ and c were tuned on the AFEW2 validation set. The SVM type used in all experiments was a C-support vector classifier and the outputs are probability estimates so that we could more easily combine results with other models.

2.2 Faces & Convolutional Network #2

In this model, we used a simpler convolutional neural network that had been trained prior to the challenge on 48×48 grayscale images from the Toronto Face Dataset, preprocessed with local contrast normalization. The network has a single convolutional layer with 15 9×9 convolutional filters and 3×3 max pooling, followed by a hidden layer of 256 units, and a 7 classes softmax output that produces predicted probabilities corresponding to the 7 emotion classes of interest. Since it had been trained on TFD, whose faces are roughly aligned based on eye positions, we tried to similarly align the faces detected in the challenge video frames. We used the average of eye-related landmarks output by [22] to compute the center of the eyes, and computed the affine transformation to map the eyes to their usual position in the 48×48 TFD faces, using only rotation and equal axis scaling. This yielded 48×48 "TFD-like" grayscale faces, to which we applied local contrast normalization, and fed to the convolutional network.

For each challenge sequence, we thus collected the vector of emotion probabilities (of length 7) output by the network for the biggest face detected in each frame. Each sequence was then summarized using a few simple statistics on these emotion probabilities vectors obtained over the sequence: average, max, average of square, average of winner-take-all (one-hot vectors), average of maximum suppression vectors (keeping only the probability of the winning class, setting the others to 0). This yielded a fixed length "feature vector" for each sequence of the challenge.

In contrast to our previous convolutional network where we used an SVM for aggregation across frames, here we generated predictions to classify the clips using a multilayer perceptron (MLP) trained on sequence-level features we have produced. This MLP was composed of one rectified linear layer of 400 hidden units followed by a linear output of 7 units. The cost function used was a multiclass margin, defined as follow:

Let o be the output vector and t the target index.

$$\mathcal{L} = \text{rectifier}(1 + \max(o_{-t}) - o_t)$$

where $\max(o_{-t})$ is the maximum of all values of o except o_t and $\text{rectifier}(x) = \begin{cases} x & \text{if } x > 0 \\ 0 & \text{else} \end{cases}$ Best accuracy on validation set was found with a learning rate of 0.0001. L2 weight decay proved to be important to avoid strong overfitting on the training set with a coefficient of 0.001 for the hidden layer and 1.0 for the linear output layer. We also used early stopping based on the validation set to further avoid overfitting.

2.3 Audio & Deep RBMs

As we have noted earlier, deep learning based techniques have lead to a number of recent successes in speech recognition [11]. We followed a deep learning approach for performing emotion recognition on AFEW2 movie clips based on the audio using deep Restricted Boltzmann Machines (RBMs). In order to tune the hyperparameters of our model, we performed crossvalidation using the AFEW2 validation dataset, we used both random hyperparameter search with some manual finetuning of the hyperparameters afterwards.

Choosing the right features is a crucial aspect of the audio classification. Mel-frequency cepstral coefficients (MFCCs)

are widely used for speech recognition; however, in this task we are mainly interested in detecting emotions. Moreover emotion recognition on film audio is quite different from other audio tasks. For example, even when speech is present in the audio of a movie, background noise and the soundtrack of the movie can be significant indicators of emotion in the clip. For the EmotiW challenge, we extracted several features from the mp3 files extracted from the movie clips using the yafee library[1] with a sampling rate of 48 kHz. We extracted 29 different features from those mp3 files. We used all 27 features provided by yafee library except "Frames". Additionally 3 types of MFCC features are used, the first used 22 cepstral coefficients, the second used a feature transformation with the temporal first order derivative and the last one using second order temporal derivatives. Online PCA was applied on the extracted features, and 909 features per timescale were used [9].

2.3.1 DBN Pretraining

We used unsupervised pretraining with deep belief networks (DBN) on the extracted audio features. The DBN has two layers of RBMs, first layer RBM was a Gaussian RBM with noisy rectified linear unit (ReLU) nonlinearity [5], the second layer RBM was a Gaussian-Bernoulli RBM. We trained the RBM's using stochastic maximum likelihood and contrastive divergence with one Gibbs step (CD-1). Each RBM layer had 310 hidden units. The first and second layer RBM's were trained with learning rates of 0.0006 and 0.0005 respectively. An L2 penalty with 2×10^{-3} and 2×10^{-4} was used for the first and second layer respectively. Both the first and second layer RBM's were trained for 15 epochs on the AFEW2 training dataset. We bounded the noisy ReLU activations of the first layer Gaussian RBM, such that the activation function $min(\alpha, max(0, \mathbf{x} + \psi))$ were used where $\psi \sim N(0, \sigma(\mathbf{x}))$ with $\alpha = 6$. Otherwise large activations of the first layer RBM was causing problems training the second layer Gaussian Bernoulli RBM. We used a Gaussian model of the form $N(0, \sigma(\mathbf{x}))$, with 0 mean and standard deviation of $\sigma(\mathbf{x}) = \frac{1}{1+exp(-\mathbf{x})}$. At the end of unsupervised pretrainining, we initialized the MLP with ReLU nonlinearity for the first layer and sigmoid nonlinearity for the second layer using the weights and biases of the DBN.

2.3.2 Temporal Pooling for Audio Classification

We used a multi-time-scale learning model [9] for the MLP where we pooled the last hidden representation layer of an MLP so as to aggregates information across frames before a final softmax layer. We experimented with several different pooling methods including max pooling and mean pooling, but we obtained the best results with a specifically designed type of pooling for the MLP features discussed below. Assume that we have a matrix A for the activations of the MLP's last layer features that includes activations of all timescales in the clip where $A \in R^{d_t \times d_f}$ and d_t is the variable number of timescales, d_f is the number of features at each timescale. We sort the columns of A in decreasing order and get the top N rows using the map, $f : R^{d_t \times d_f} \rightarrow R^{N \times d_f}$. The most active N features are summarized with a weighted average of the top-N features:

$$F = \frac{1}{N} \sum_{i=0}^{N} w_i f^{(i)}(A; N) \tag{1}$$

where $f^{(i)}(A; N)$ is the i^{th} highest active features over time and weights should be: $\sum_{i=0}^{N} w_i = N$. During the supervised finetuning, we feed the reduced features to the top level softmax, we backpropagate through this pooling function to the lower layers. We only used the top 2 ($N = 2$) most active features in the weighted average. Weights of the features were not learned and they were chosen as $w_1 = 1.4, w_2 = 0.6$ for the training and $w_1 = 1.3, w_2 = 0.7$ for the test time. This kind of feature pooling technique worked best, if the features are extracted from a bounded nonlinearity such as $sigmoid(.)$ or $tanh(.)$.

2.3.3 Supervised Finetuning

Only the AFEW2 training dataset was used for supervised finetuning and we did early stopping by measuring the error on the AFEW2 validation dataset. The features were centered prior to the training. Before initiating the supervised training, we shuffled the order of clips. During the supervised finetuning phase, at each iteration on training dataset, we randomly shuffled the order of the features in the clip as well. At each training iteration, we randomly dropped out 98 clips from the training dataset and we randomly dropped out 40% of the features in the clip. 0.121 % of the hidden units are dropped out and we used a norm constraint on the weights such that the L2 norm of the incoming weights to a hidden unit do not exceed 1.2875 [12]. In addition to drop-out and maximum norm constraint on the weights, a L2 penalty with coefficient of 10^{-5} was used on the weights as well. The Rmsprop adaptive learning rate algorithm was used to tune the learning rate with a variation of Nesterov's Momentum [19]. But as opposed to the traditional rmsprop implementations, we used different learning rate per coordinate and, we divided the learning rate by the running average of root mean square of gradients after taking the momentum and gradient step. At each iteration we keep track of mean square of the gradients by:

$$RMS(\Delta_{t+1}) = \rho RMS(\Delta_t) + (1 - \rho)\Delta_t^2 \tag{2}$$

and compute the momentum, then do the stochastic gradient descent (SGD) update:

$$v_{t+1} = \mu v_t - \epsilon_0 \frac{\partial f(x^{(i)}; \theta_t)}{\partial \theta_t}, \quad \theta_{t+1} = \theta_t + \frac{\mu v_{t+1} - \epsilon_0 \frac{\partial f(x^{(i)}; \theta_t)}{\partial \theta_t}}{\sqrt{RMS(\Delta_{t+1})}} \tag{3}$$

After performing crossvalidation, we decided to use an $\epsilon_0 = 0.0005$, $\mu = 0.46$ and $\rho = 0.92$. We used early stopping based on the validation set performance, yielding an accuracy of 29.29%. Once supervised finetuning had completed 50 iterations, if the validation error continued increasing, the learning rate was decreased by a factor of 0.99.

2.4 Activity Recognition & Deep Autoencoders

Given a video sequence with a human emotion in it, most of the visual approaches that we have discussed so far do not use features derived from the temporal structure of video frames. In our approach here we considered spatio-temporal motion patterns in the video to predict an emotion label. For this task we used a recognition pipeline described in

[1]Yaafe: audio features extraction toolbox: `http://yaafe.sourceforge.net/`

[13, 16, 21] with a spatio-temporal autoencoder presented in [13] for feature learning. The model was trained on PCA-whitened input patches of size $10 \times 16 \times 16$ cropped randomly from training videos. The number of training samples is $200,000$. The number of hidden units in the spatio-temporal autoencoder is 300.

For inference, we cropped sub blocks of the same size as the patch size from $14 \times 20 \times 20$ pixel "super-blocks", using a stride of 4 [13, 16]. This yields 8 sub blocks per super block. We obtained a super block descriptor by performing PCA on the concatenation of sub block filter responses. We used K-means clustering on the super block descriptors to learn a vocabulary of 3000 spatio-temporal words. We represented each video as the histogram over spatio-temporal words, which we classified using a χ^2-kernel SVM.

2.5 Bag of Mouth Features & Shallow Networks

This model uses the aligned faces provided by the organizers, which were obtained from Zhu and Ramanan's detector [22]. The images, after being downsized by a factor of approximately 2 in each direction (71 x 90 pixels), are cropped in order to only keep a small region around the mouth. The purpose of this step, especially given the fairly small size of the training dataset, is to avoid learning irrelevant coincidences in regions less expressive than the mouth.

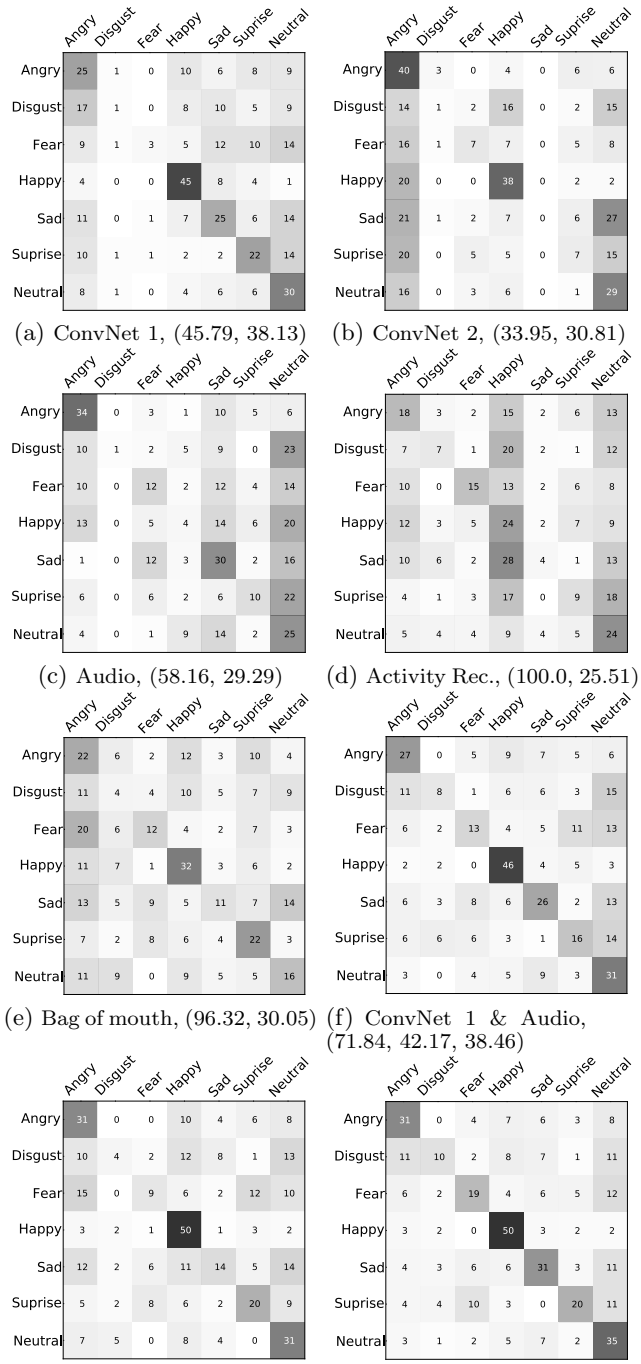
The rest of the pipeline mostly follows a procedure described by Coates [4], with a few modifications. The mouth images are subdivided into 16 regions using a 4 x 4 grid. For each of these zones, many 8 x 8 RGB patches are extracted from the training images. These patches are individually standardized by DC-centering and contrast normalization (subtracting the mean and dividing by the standard deviation of the elements). To avoid division by zero, a small value is added to the standard deviation prior to the division. Then, for each of the 192 elements, the mean across all patches from a given region is computed and subtracted from the patches. As [4] showed that whitening is often helpful in image classification tasks, the patches are whitened, keeping 90% of variance. After this preprocessing, a dictionary of size 400 is learned for each of the 16 regions by using the K-means algorithm.

For all images of the training, validation and test sets, 8 x 8 patches are extracted densely within each region and processed as in the preceding paragraph. Each patch is then assigned a 400-dimensional feature vector by comparing it to the appropriate centroids. More specifically, the so-called triangle activation [4] is used. For each centroid k ($k = 1, \ldots, 400$), the Euclidian distance z_k between it and the whitened patch is computed, and then so is the average $\mu(z)$ of these 400 distances. The activation of a given feature k is finally taken as $\max(0, \mu(z) - z_k)$.

Within each of the 16 regions, the vectors are pooled by averaging the responses of all patches. The 16 resulting vectors are subsequently concatenated. Using these 6400-dimensional feature vectors, a frame-by frame-classifier is trained using logistic regression with regularization. The probabilities for each clip are finally computed by simply averaging the probability vectors of the corresponding frames.

3. RESULTS & COMBINING MODELS

In figure 6 (a-e) we show the validation set confusion matrices from the models yielding the highest validation set accuracy for each of the techniques discussed in section 2.



(a) ConvNet 1, (45.79, 38.13) (b) ConvNet 2, (33.95, 30.81)

(c) Audio, (58.16, 29.29) (d) Activity Rec., (100.0, 25.51)

(e) Bag of mouth, (96.32, 30.05) (f) ConvNet 1 & Audio, (71.84, 42.17, 38.46)

(g) Simple Avgerage of Models, (97.11, 40.15, 37.17) (h) Random Search Weighted Avg., (92.37, 49.49, 41.03)

Figure 6: Confusion matrices on the validation set. Accuracy for each method is also given for the (training, validation & test sets, if applicable).

From this analysis one can see that ConvNet 1 yielded the highest validation set accuracy and we therefore selected this model as our first submission, yielding a test set accuracy of **35.58**%. This also indicated in table 1 which contains a summary of all our submissions. ConvNet2 was our second highest performer, followed closely by the bag of mouth and audio models at 30.81%, 30.05% and 29.29% respectively. We used the following strategies to combine predictions.

Sub.	Train	Valid	Test	Method
1	45.79	38.13	35.58	Google data & TFD used to train ConvNet 1, frame scores aggregated with SVM
2	71.84	42.17	38.46	ConvNet 1 (from submission 1) combined with Audio model using another SVM
3	97.11	40.15	37.17	Mean prediction from: Activity, Audio, Bag of mouth, ConvNet 1, ConvNet 2
4	98.68	43.69	32.69	SVM with detailed hyperparam. search: Activity, Audio, Bag of mouth, ConvNet 1
5	94.74	47.98	39.42	Short uniform random search : Activity, Audio, Bag of mouth, CN1, CN1 + Audio
6	94.74	48.48	40.06	Short local random search : Activity, Audio, Bag of mouth, CN1, CN1 + Audio
7	92.37	**49.49**	**41.03**	Moderate local random search : Activity, Audio, Bag of mouth, CN1, CN1 + Audio

Table 1: Our 7 submissions with training, validation and test accuracies.

3.1 Averaged Predictions

A simple way to make a final prediction using several models is to take the average of their predictions. We had 5 models in total, which gives $\sum_{i=1}^{n} \binom{n}{i} = 31$ possible combinations (order has no importance). In this context it is possible to test every possible combinations on the validation set to find those which are the most promising. Through this analysis we found that the average of all models yielded the highest validation set performance of 40.15%. The validation set confusion matrix for this model is shown in figure 6 (g). For our third submission we therefore submitted the results of the averaged predictions of all models, yielding **37.17**% on the test. From this analysis we also found that the exact same validation set performance was also obtained with an average not including our second convolutional network, leading us to make the intuitive conclusion that both convolutional networks were providing similar information.

The next highest performing simple average was 39.90% and consisted of simply combining ConvNet 1 and our audio model. Given this observation and the fact that the conference baselines included both video, audio and combined audio-video models we decided to submit a model in which we use just these two models. However, we first explored a more sophisticated way to perform this combination.

3.2 SVM and MLP Aggregation Techniques

To further boost the performance of our combined audio-video model we simply concatenated the results of our ConvNet 1 and audio model using vectors and learned a SVM with an RBF kernel using the challenge training set. The hyperparameters of the SVM were set via a two stage coarse, then fine grid search over integer powers of 10, then non-integer powers of 2 within the reduced region of space. The hyperparameters correspond to a kernel width term, γ and the usual c parameter of SVMs. This process yielded a model with 42.17% accuracy on the validation set, which became our second submission and produced an accuracy of **38.46**% on the challenge test set. The validation set confusion matrix for this model is shown in figure 6 (f).

Given the succes of our SVM combination strategy, we tried the same technique using the predictions of all models. However, this process quickly overfit the data and we were not able to produce any models that were able to improve upon our best validation set accuracy obtained via the ConvNet 1 and audio model. We observed a similar effect using a strategy based upon an MLP to combine the results of all model predictions.

We therefore tried a more sophisticated SVM hyperparameter search to re-weight different models and their predictions for different emotions. We implemented this via a search over discretized $[0, 1, 2, 3]$ per dimension scaling factors. While this resulted in 28 additional hyperparameters this discretization strategy allowed us to explore all combinations. This more detailed hyperparameter tuning did allow us to increase the validation set performance to 43.69%. This became our fourth submission; however, the strategy yielded a decreased test set performance at **32.69**%.

3.3 Random Search for Weighting Models

Recent work [1] has shown that random search for hyperparameter optimization can be an effective strategy, even when the dimensionality of hyperparameters can be moderate (ex. 35 dimensions). Analysis of our validation set confusion matrices shows that different models have very different performance characteristics across the different emotion types. We therefore formulated the re-weighting of per-model and per-emotion predictions as a hyperparameter search over simplexes, weighting the model predictions for each emotion type.

To perform the random search, we first generated uniform random weights and then normalized them to produce seven simplexes. This process is slightly biased towards weights that are less extreme compared to other well known procedures that are capable of generating uniform values on simplexes. After running this sampling procedure for a number of hours we used the weighting that yielded the highest validation set performance (47.98%) as our 5th submission. This yielded a test set accuracy of **39.42**%. We used the results of this initial random search to initiate a second, local search procedure which is analogous in a sense to the typical two level coarse, then fine level grid search used for SVMs. In this procedure we generated random weights using a Gaussian distribution around the best weights found so far. The weights were tested by calculating the accuracy of the so-weighted average predictions on the validation set. We also rounded these random weights to 2 decimals to help to avoid overfitting on validation set. This strategy yielded **40.06**% test set accuracy with a short duration search and **41.03**% with a longer search - our best performing submission on the test. The validation set confusion matrix for this model is shown in figure 6 (h) and the weights obtained through this process are shown in figure 7.

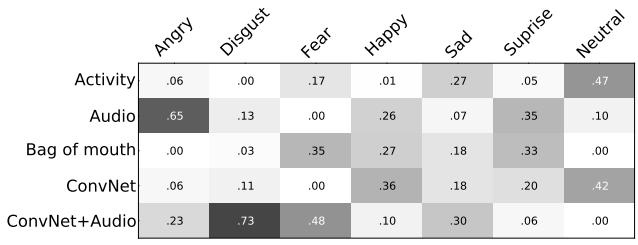


Figure 7: Final weights used for model averaging

4. FINAL CONCLUSIONS & DISCUSSION

Our efforts here have lead to a number of contributions and a number of insights which we believe may be more broadly applicable. First, we believe that our approach of using the large scale mining of imagery from Google image search to train deep neural networks has helped us avoid overfitting in our facial expression model. Perhaps counter-intuitively, we have found that our convolutional network models learned using only our additional static frame training data sets were able to yield higher validation set performance if the labeled video data from the challenge *was only used to learn the aggregation model* and the static frames of the challenge training set *were not used* to train the underlying convolutional network. We believe this effect is also explained in part by the fact that many of the video frames in isolation are not representative of the emotional tag and their inclusion in the training set for the static frame deep neural network classifier further exacerbates the problem of overfitting, adding noise to the training set.

The problem of overfitting had both direct consequences on per-model performance on the validation set as well as indirect consequences on our ability to combine model predictions. Our analysis of simple model averaging showed that no combination of models could yield superior performance to an SVM applied to the outputs of our audio-video models. Our efforts to create both SVM and MLP aggregator models lead to similar observations in that models quickly overfit the training data and no settings of hyperparameters could be found which would yield increased validation set performance. We believe this is due to the fact that the activity recognition and bag of mouth models severely overfit the challenge training set and the SVM and MLP aggregation techniques - being quite flexible - also overfit the data and in such a way that no traditional hyperparameter tuning could yield validation set performance gains.

These observation led us to develop the novel technique of aggregating the per model and per class predictions via random search over simple weighted averages. The resulting aggregation technique is therefore of extremely low complexity and the underlying prediction was therefore highly constrained - using simple weighted combinations of complex deep network models, each of which did reasonably well at this task. We were therefore able to explore many configurations in a space of moderate dimensionality quite rapidly as we did not need to re-evaluate the predictions from the neural networks and we did not adapt their parameters. As this yielded a marked increase in performance on both the challenge validation and test sets it leads us to the interpretation that given the presence of models that overfit the training data, it may be better practice to search a moderate space of simple combination models compared to more traditional approaches such as searching over the smaller space of SVM hyperparameters or even a moderately sized space of traditional MLP hyperparameters including the number of hidden layers and the number of units per layer.

5. ADDITIONAL AUTHORS

LISA - Université de Montréal: Raul Chandias Ferrari, Mehdi Mirza, Sébastien Jean, Pierre-Luc Carrier, Yann Dauphin, Nicolas Boulanger-Lewandowski, Abhishek Aggarwal, Jeremie Zumer, Pascal Lamblin, Jean-Philippe Raymond, Guillaume Desjardins, Razvan Pascanu, David Warde-Farley, Atousa Torabi, Arjun Sharma, Emmanuel Bengio. *Goethe Universität - Frankfurt*: Kishore Reddy Konda. *McGill University*: Zhenzhou Wu.

ACKNOWLEDGEMENTS: We thank the CFI, NSERC, Ubisoft and the German BMBF, project 01GQ0841.

6. REFERENCES

[1] J. Bergstra and Y. Bengio. Random search for hyper-parameter optimization. *The Journal of Machine Learning Research*, 13:281–305, 2012.

[2] P.-L. Carrier, A. Courville, I. J. Goodfellow, M. Mirza, and Y. Bengio. FER-2013 Face Database. Technical report, 1365, Université de Montréal, 2013.

[3] C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[4] A. Coates, H. Lee, and A. Y. Ng. An Analysis of Single-Layer Networks in Unsupervised Feature Learning. In *AISTATS*, 2011.

[5] G. E. Dahl, T. N. Sainath, and G. E. . Improving deep neural networks for lvcsr using rectified linear units and dropout. In *Proc. ICASSP*, 2013.

[6] A. Dhall, R. Goecke, J. Joshi, M. Wagner, and T. Gedeon. Emotion recognition in the wild challenge 2013. In *ACM ICMI*, 2013.

[7] A. Dhall, R. Goecke, S. Lucey, and T. Gedeon. Collecting large, richly annotated facial-expression databases from movies. *IEEE Multimedia*, 2012.

[8] Google. The Google picasa face detector, 2013. [Online, accessed 1-August-2013].

[9] P. Hamel, S. Lemieux, Y. Bengio, and D. Eck. Temporal pooling and multiscale learning for automatic annotation and ranking of music audio. In *ISMIR*, pages 729–734, 2011.

[10] G. Heusch, F. Cardinaux, and S. Marcel. Lighting normalization algorithms for face verification. *IDIAP Communication Com05-03*, 2005.

[11] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Sig. Proc. Magazine,*, 29(6):82–97, 2012.

[12] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

[13] K. R. Konda, R. Memisevic, and V. Michalski. The role of spatio-temporal synchrony in the encoding of motion. *arXiv preprint arXiv:1306.3162*, 2013.

[14] A. Krizhevsky. Cuda-convnet Google code home page. `https://code.google.com/p/cuda-convnet/`, Aug. 2012.

[15] A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, pages 1106–1114. 2012.

[16] Q. Le, W. Zou, S. Yeung, and A. Ng. Learning hierarchical invariant spatio-temporal features for action recognition with independent subspace analysis. In *CVPR*, 2011.

[17] V. Štruc and N. Pavešić. Gabor-based kernel partial-least-squares discrimination features for face recognition. *Informatica*, 20(1):115–138, 2009.

[18] J. Susskind, A. Anderson, and G. Hinton. The toronto face database. Technical report, UTML TR 2010-001, University of Toronto, 2010.

[19] I. Sutskever, J. Martens, G. Dahl, and G. Hinton. On the importance of initialization and momentum in deep learning. In *ICML 2013*, 2013.

[20] V. Štruc and N. Pavešić. *Photometric normalization techniques for illumination invariance*, pages 279–300. IGI-Global, 2011.

[21] H. Wang, M. M. Ullah, A. Kläser, I. Laptev, and C. Schmid. Evaluation of local spatio-temporal features for action recognition. In *BMVC*, 2009.

[22] X. Zhu and D. Ramanan. Face Detection, Pose Estimation, and Landmark Localization in the Wild. In *CVPR*, 2012.