2017-11-01

Head pose estimation in the wild using Convolutional Neural Networks and adaptive gradient methods

Patacchiola, M

10.1016/j.patcog.2017.06.009
Pattern Recognition

Head Pose Estimation in the Wild using Convolutional Neural Networks and Adaptive Gradient Methods

Massimiliano Patacchiola*, Angelo Cangelosi

Centre for Robotics and Neural Systems, School of Computing Electronics and Mathematics, Plymouth University, Plymouth, UK, PL4 8AA.

Abstract

Head pose estimation is an old problem that is recently receiving new attention because of possible applications in human-robot interaction, augmented reality and driving assistance. However, most of the existing work has been tested in controlled environments and is not robust enough for real-world applications. In order to handle these limitations we propose an approach based on Convolutional Neural Networks (CNNs) supplemented with the most recent techniques adopted from the deep learning community. We evaluate the performance of four architectures on recently released in-the-wild datasets. Moreover, we investigate the use of dropout and adaptive gradient methods giving a contribution to their ongoing validation. The results show that joining CNNs and adaptive gradient methods leads to the state-of-the-art in unconstrained head pose estimation.

Keywords: convolutional neural networks, head pose estimation, adaptive gradient, deep learning

1. Introduction

In the last few years major advancements in robotics, augmented reality and driving assistance have highlighted the need for robust methods to estimate the head pose in real-world scenarios. For instance, robots are gradually leaving factories and becoming part of our lives as companions and as assistants. It has been shown that in human-robot interaction a coarse pose estimation of the head is a fundamental prerequisite for building trust with users during joint-attention tasks [1]. In the context of autonomous cars a driving assistance system could

*Corresponding author

Email address: massimiliano.patacchiola@plymouth.ac.uk (Massimiliano Patacchiola)

take advantage of head pose estimation for decelerating the car when pedestrians do not notice the presence of the vehicle [2]. Moreover a similar system can be installed inside the vehicle and used to monitor the driver's awareness. The need of a robust head pose estimation is not limited to these domains. There have been significant applications in surveillance and anomaly detection, human-computer interaction and crowd behavioural dynamics analysis [3]. All of these unconstrained scenarios need an estimator which is resistant to variable environmental conditions, and which can evaluate the focus of attention in absence of more accurate information such as the gaze. Here it is necessary to specify what we consider as a wild environment. We define as taken in a wild environment those face images exhibiting a large variety in appearance (pose, expression, ethnicity, age, gender, etc), environmental conditions (artificial light, shadows, etc), and containing relevant occlusions (sunglasses, masks, scarves, etc). We will show how Convolutional Neural Networks (CNNs) can be considered one of the best algorithms for robust head pose estimation in a wild environment.

We can summarise the main contribution of our work in three points:

1. As far as we know this is the first work that has deeply investigated the use of CNNs in head pose estimation. Our main contribution is a rigorous evaluation of multiple CNN models and factors. The results are compared with other algorithms, and show how an approach based on CNNs, dropout and adaptive gradient methods represents the state of the art in head pose estimation.
2. Deep learning is a rapidly growing field, which is bringing new techniques that can significantly improve the performance of CNNs. Because these techniques have been released in the last few years, there is still a validation process for establishing their cross-domain usefulness. We explored the role of adaptive gradient methods and we gave a valuable contribution to their ongoing validation.
3. The results obtained in this work have been used to implement a Python library called Deepgaze. The library includes pre-trained CNNs based on Tensorflow [4] which can run in real-time on GPUs and mobile devices. Deepgaze is released under an open-source license and is available for both academic and commercial purposes. The software is available on the author's repository [1].

2. Related Work

The head pose estimation problem has been investigated from different points of view and with different techniques. Devices such as laser pointers, camera arrays, stereo-cameras, magnetic and inertial sensors, have been used to get a stable estimation in controlled situations [5]. More recently some good

[1] https://github.com/mpatacchiola/deepgaze

results have been obtained with commercial depth cameras [6]. However the use of these devices is not always feasible due to space constraints and to technical problems when operating outdoors. Our work made use of RGB images taken from monocular cameras which permits the greatest portability in real-world applications, given the proliferation of such cameras in mobile phones and laptops. This introduction is thus limited to this kind of approach. A complete description of all the methods available is out of the scope of this article so we refer the reader to a recent survey [5].

The main branches of a functional taxonomy in head pose estimation can be considered to be appearance-based methods, model-based methods, manifold embedding methods and nonlinear regression methods. Appearance-based methods compare the view of a person's head with discrete models which represent pose labels [7, 8]. With different kinds of template matching techniques it is possible to evaluate the similarity of the input features with the exemplar set. Appearance-based methods are quite simple to implement but suffer from some serious limitations. For instance, they cannot estimate discrete pose locations without using interpolation methods. They assume that there is a close similarity between the image and the pose space, and they can easily associate a pose based on the resemblance with a wrong model. A common solution to compensate these errors is to filter and convolve the models [9]. The effect of these manipulations is to highlight some features (e.g. vertical and horizontal lines) and to remove part of the variations across the models. However this solution is expensive because it requires to manually find the right filters and to test the effects on the pose estimation. For all these limitations the use of appearance-based algorithms has been decreasing over time.

Model-based methods use geometric information, non-rigid facial models or landmark locations to estimate the head pose. The most common model-based approach consists in finding coplanar facial key-points and to estimate the distance from a reference coordinate system [10]. This method requires high precision and does not work for certain degenerative angles. Another common approach consists in evaluating the position of multiple non-coplanar key-points. The head pose is estimated assuming fixed geometric relationships between the landmarks and comparing the position of the points with an average mask obtained through anthropometric measurements [11]. Although there have been improvements in key-points detection and tracking in real world conditions [12, 13], the landmarks detection is the major limitation of this approach. In general we can say that the accuracy of model-based methods is correlated with the quality and quantity of the geometric cues extrapolated from the image. In real world scenarios occlusions can often obscure facial landmarks having a negative effect on the head pose prediction.

Manifold embedding methods consider an high-dimensional image to be constrained in a low-dimensional manifold in which the head pose is estimated. Dimensionality reduction techniques can be considered as part of the manifold embedding category. In the standard approach principal component analysis (PCA) is used in order to project the input images in a subspace and compare them to the models [14]. The main problem of manifold embedding is that the

pose must be recovered across multiple sources of variation. In this sense, other manifold embedding techniques have recently shown to be promising. In [15] the problem of the source variation has been tackled learning a similarity kernel through geometric invariant features. This method showed a good reliability on benchmark datasets. However further research is needed in order to reach state of the art performances.

Nonlinear regression methods use a labelled training set to create a nonlinear mapping from images to poses. We can consider CNNs as part of these methods. Nonlinear methods have many advantages. These algorithms work properly with high and low resolution images, and they have demonstrated their representational ability in tolerating systematic errors in the training set data. For instance, an approach based on a multilayer perceptron [16] had for long time the lowest reported mean angular error on the Prima head pose dataset [5]. The main disadvantages of these methods are two. The first is the need of a consistent dataset in order to train the parameters. The second is the need of a precise head localisation before the pose estimation step. The first issue can be overcome thanks to recently released datasets containing a large number of images. The second issue can be attenuated using CNNs instead of multi layer perceptrons which guarantee a good shift and distortion invariance. In this sense CNNs could be the elective technique for mastering the head pose estimation problem, since recent advances in deep learning made possible to easily train complex CNNs on large datasets. Despite their potential, the use of CNNs for head pose estimation has been sporadic. An approach based on CNNs and energy-based models has been proposed for simultaneous face detection and pose estimation [17]. The authors trained the model on a dataset containing images taken in laboratory conditions, and validated it on datasets containing frontal faces with in-plane rotations and faces in profile. The prediction of the network was limited to two degrees of freedom. This work is valuable but it is ten years old and at that time advanced techniques for training deep networks were not available. Moreover the results relative to the head pose estimation accuracy were not included. A more recent work investigated the use of CNNs with low-resolution images from monocular cameras [18], obtaining the best reported result on the Biwi Kinect Head Pose dataset [19]. The authors did not use any of the most recent techniques available such as dropout or adaptive methods, and their results have been validated on a single dataset that does not contain in-the-wild images. However the results obtained confirm the validity of CNNs in head pose estimation. In [20] CNNs have been used for monitoring the driver alertness. The authors used a six-layer CNN to classify five discrete head poses: left, right, up, down and frontal. Because of the limits implicit in a discrete classification with only five poses we cannot compare this work with the others presented here. In [21] the authors used deep convolutional networks to estimate the head pose in multimodal RGB-D videos. Depth information is not always available due to space constraints and to problems operating outdoors. The authors did not test the effect of different numbers of layers or parameters, moreover they did not use any kind of adaptive gradient method or regularisation to improve the performance of the network.

Given the lack of satisfying work we wanted to add something more complete and modern to the existing literature. In the next sections we shortly explain how CNNs work and how dropout and adaptive gradient methods can boost their performance. The next section is not intended to be an exhaustive description of the mechanics behind CNNs, instead it is a way to introduce some key concepts and to define the notation used in the rest of the article.

3. Convolutional Neural Networks

In recent years deep convolutional networks have showed their strength in numerous pattern recognition contests. Some remarkable achievements have been recently obtained in object detection [22], facial expression recognition [23] and scene classification [24]. This technology is increasingly used in commercial applications such as content filtering in social networks, recommendation systems in e-commerce websites or image classifiers in web-search engines. The deep learning revolution has been driven by the diffusion of cheap graphics processing units (GPUs), originally used for video games. The use of GPUs speeds up matrix and vector multiplications, which are the core operations in neural network training. Because the general introduction of CNNs is well established by now, we will not extend this section any further. Instead we refer the reader to the following article [25]. In the next section we give a brief description of the main building blocks of a CNN, introducing the notation used in the rest of the article.

3.1. Notation

We define a CNN as an ensemble of many single units grouped in three-dimensional layers. The units inside a layer are connected to a small region of the layer before it with connections called kernels (or filters, weights, parameters). The input to a CNN is a matrix X of dimension $m \times m \times r$, where m is the height and width of the matrix and r is the number of channels. The convolutional layer has k kernels of size $n \times n \times q$, where $n < m$ and $q \leq r$. The convolution multiplies each element of X with its local neighbours, weighted by the kernel W, generating k feature maps of size $m - n + 1$. The convolutional layer is often followed by a mean or max pooling layer which permits subsampling the maps over a $p \times p$ local region, with $2 \leq p \leq 5$. During the training phase the kernels are adjusted following a well-known algorithm called backpropagation [26]. The backpropagation minimises a loss (or error) function $J(w)$ with an iterative process of gradient descent that updates w at $t + 1$ using the gradient information at t, as expressed in the following equation:

$$w_{t+1} = w_t - \alpha \nabla E|J(w_t)| + \mu v_t \tag{1}$$

The expectation is approximated with the cost and gradient over the full training set. The value α is called learning rate and corresponds to the step taken by the algorithm in the direction of the gradient. The value $\mu \in [0, 1]$ is called momentum and is a technique for accumulating a velocity vector v

in the direction of persistent reduction of the loss function. A variation of the standard gradient descent is called Stochastic Gradient Descent (SGD). The SGD computes the gradient using a few training examples or mini-batches instead of the whole training set. Using the stochastic approach the variance is reduced leading to a more stable convergence.

There are different kinds of loss functions. In our experiments we used the sum of squares of the differences between the target value y and the estimated value $\hat{y}$:

$$J(w) = \sum_{n=1}^{N} (y_n - \hat{y}_n)^2 + \lambda \sum_{l=1}^{L} w_l^2 \tag{2}$$

The additive factor λ is an $L2$ regularisation term, used in each hidden layer l to prevent a very large growth of the parameters during the minimisation process.

In the next sections we will focus on some recent techniques developed by the deep learning community in the very last few years, which made the training of deep architectures more manageable leading to better results. These techniques are dropout and adaptive gradient methods.

3.2. Dropout

In networks with a large number of parameters the generalisation on a new set of data can be compromised due to memorisation of the training set which leads to overfitting. Dropout [27] addresses this problem by randomly dropping units and connections during the training phase, preventing co-adaptation. We can summarise the operations involved in dropout in a few lines. Let's define r as a vector of Bernoulli random variables, where each variable has probability p of being 1 and a probability $1 - p$ of being 0. For each hidden layer l the vector r is sampled and then multiplied elementwise with y the output vector of that layer. The result is a thinned vector $\tilde{y}$:

$$\begin{aligned} r^{(l)} &\sim Bernoulli(p) \\ \tilde{y}^{(l)} &= r^{(l)} \circ y^{(l)} \end{aligned} \tag{3}$$

Experimental evidence shows how dropout introduces noise in the gradients and produces a cancellation effect [27]. To deal with this issue it is recommended to use a higher learning rate and momentum. In order to use a higher learning rate without a very large growth of the weights, another form of regularization can be used at the same time, such as $L2$ or max-norm. The probability p is another hyperparameter to tune. However numerous experimental results suggest that a value of $p = 0.5$ produces the best performance [27], so we used this value in our experiments. A graphical representation of dropout is presented in Fig. 1.

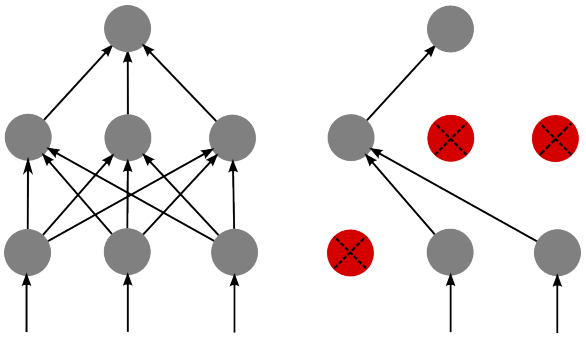

Figure 1: Graphical representation of the dropout process. The figure on the left represents a generic neural network composed of three layers. The figure on the right represents the same network after applying the dropout with 50% probability of keeping a unit.

3.3. Adaptive Gradient Methods

Neural networks are not off-the-shelf algorithms, and generally the research of the best hyperparameters can be extremely time consuming. One of the most important parameters is the learning rate. When the learning rate is too high the optimisation can diverge, on the contrary if it is too low the optimisation can be slow.To solve these problems some adaptive gradient methods have been proposed recently. Adaptive gradient methods use first order information to approximate second order information and then find an optimal step size.

One of the best adaptive methods introduced recently is Adagrad [28]. This optimiser incorporates information about the features to control the gradient step. The procedure associates a low learning rate with frequently occurring features and high learning rate with infrequent features. Therefore, the adaptation facilitates identifying the most predictive features and it is well suited for sparse data. The authors tested the algorithm on different image and text databases [28], showing how it outperforms the non-adaptive counterpart. The updating rule for the weights w following the Adagrad algorithm can be expressed with the following equation:

$$w_{t+1} = w_t - \frac{\alpha}{\sqrt{G_t + \epsilon}} \odot g_t \tag{4}$$

The matrix G is a diagonal matrix where each entry in the main diagonal is the sum of the squares of the previous gradients up to time t. The value ϵ is a small value used to avoid division by zero. The symbol $\odot$ represent a matrix-vector multiplication. The main problem with Adagrad is the accumulation of squared gradients in G that lead to an infinitesimally small learning rate α and then to a loss in knowledge accumulation.

To solve the problem related with Adagrad an extension called Adadelta [29] has been proposed. Adadelta does not accumulate all the past gradients but it constrains the window to a fixed interval. The denominator of the Equation 4 is then replaced by a moving average $E[g^2]$ which represents all the past squared

gradients. The advantage of this solution is that the moving average depends only on the previous average and the current gradient, as follows:

$$w_{t+1} = w_t - \frac{\alpha}{\sqrt{E[g^2]_t + \epsilon}} g_t \tag{5}$$

As experimentally shown in [29] Adadelta is particularly robust and it guarantees convergence with different learning rate values. Another effective method to solve the Adagrad issue is an unpublished algorithm called RMSProp [30]. RMSProp has an updating rule which is similar to Equation 5 with the only difference being that it introduces a decaying value γ which specifies how long the old gradients in $E[g^2]$ are kept. The value $E[g^2]$ at time t is then updated in this way:

$$E[g^2]_t = \gamma E[g^2]_{t-1} + g_t^2 \tag{6}$$

The authors suggest some default values for the learning rate and the decaying value which should be closer to $\alpha = 0.001$ and $\gamma = 0.9$. In our experiments we used this configuration.

Finally we want to introduce another method called Adaptive Moment Estimation (Adam) [31]. Like Adadelta and RMSProp, Adam stores a decaying average of past gradients and squared gradients. The two decaying averages are then used to estimate m_1 and m_2 which are the first and second moment (mean and variance) of the gradient. The updating rule for Adam can be expressed as follows:

$$w_{t+1} = w_t - \frac{\alpha}{\sqrt{m_2} + \epsilon} m_1 \tag{7}$$

The two moments m_1 and m_2 are taken at time t and before the weights update they are corrected to limit a bias toward zero during the first steps. The moments are regulated by two decaying factors β_1 and $\beta 2$. The authors suggest to initialise these parameters to standard values $\beta_1 = 0.9$ and $\beta_2 = 0.999$. We used these values in our experiments.

4. Experiments

In this section we report the results obtained using CNNs, dropout and adaptive gradient methods on three public datasets: the Prima head-pose dataset [32], the Annotated Facial Landmarks in the Wild (AFLW) dataset [33], and the Annotated Face in the Wild (AFW) dataset [34]. The former is a well-known dataset which has been around for more than ten years, and it is considered a classic benchmark for head pose algorithms. The second is a recently released in-the-wild dataset, and it has the largest number of annotated poses currently available. The third is a small in-the-wild dataset used mainly for benchmarks. Other details about these datasets are available in Sec. 4.1, 4.2 and 4.3.

In these experiments we used a total of four networks: A, B, C, D. The architecture A is a standard LeNet-5 [35] and with six layers and 4.3×10^6

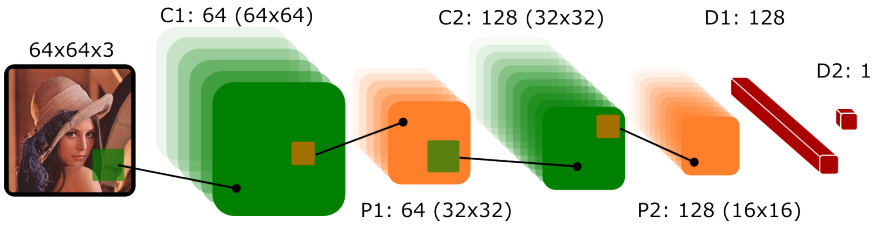

Figure 2: Grapichal representation of a Convolutional Neural Network with two convolutional (C1 and C2), two subsampling (P1 and P2) and two fully connected (D1 and D2) layers. The label above each layer specifies number of elements and size (rows × columns) of the future maps. For the dense layers we reported the number of units. We used the following colour convention to identify the different layers: green for convolution, orange for subsampling, and red for dense layers.

parameters. The graphical representation of this architecture is presented in Fig. 2. The architecture B has one more convolutional layer and one more pooling layer. The number of parameters is slightly higher (4.6×10^6). The idea is to keep the architectures as similar as possible to isolate the effect of the additional layers. It is hard to define a priori how many parameters lead to a good performance, for this reason the third and fourth networks have more weights. The architecture C has a high number of parameters (8.5×10^6). The fourth architecture (D) is similar to a standard AlexNet [22] and it has a total of 9.0×10^6 parameters. A graphical comparison of the four architectures is reported in Fig. 3.

The approach we used is based on a divide-and-conquer strategy. We trained different CNNs for each degree of freedom. This kind of strategy has the advantage of splitting the main problem into different sub-problems which are easier to manage. Having a specialised network for roll, pitch and yaw, permits fine tuning the network for a specific degree of freedom without losing the predictive power obtained on another one. Our methodology consisted of two parts. First, we evaluate which optimiser was better suited to a specific dataset. Second, we tested CNNs with a variable number of layers and parameters to understand the impact of deeper architectures. As discussed in [5] the Mean Absolute Error (MAE) and the Standard Deviation (STD) are the best metric of accuracy in head pose datasets with discrete or continuous labels. We report both MAE and STD and we use them for comparing results. In all of the experiments we used the same hardware configuration: a multi-core workstation with 32 GB of RAM and a GPU NVIDIA Tesla K-40. The experiments have been implemented in Python using the TensorFlow library [4].

4.1. Prima head-pose dataset

This dataset consists of 2790 monocular face images of 15 subjects. The subjects range in age from 20 to 40 years old, five possessing facial hair and seven wearing glasses. Pitch and yaw angles are in the range $[-90°, 90°]$ for a total of 93 discrete poses for each person (Fig. 4). Two series of pictures with different lighting conditions are provided increasing the total number of

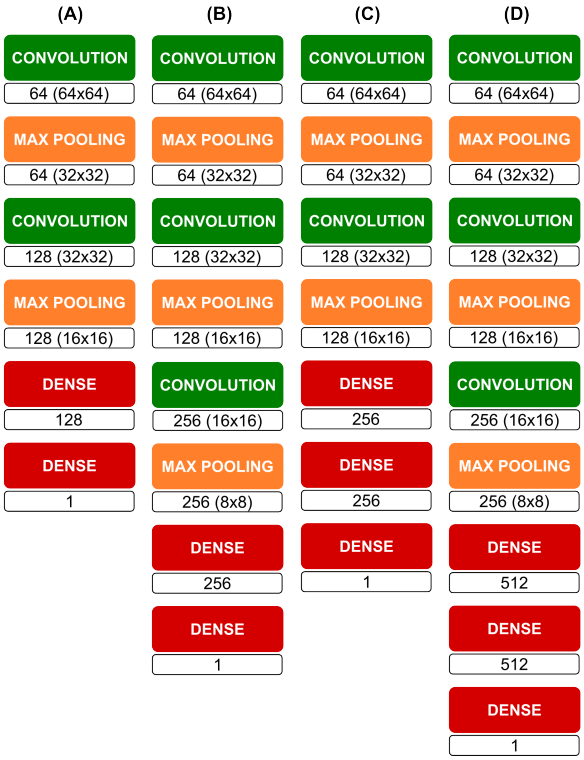

Figure 3: Comparison of the four architectures used in our experiments. The label below each layer represents the number and size (rows $\times$ columns) of the future maps. For the dense layers we reported only the number of units. The networks are organised in order of complexity, on the left there is the network with less parameters and on the right the network with more. Network A has 4.3×10^6 parameters, whereas network B has two more layers and a total of 4.6×10^6. Network C has 8.5×10^6 parameters, whereas network D has two more layers for a total of 9.0×10^6 parameters. We used the following colour convention to identify the different layers: green for convolution, orange for subsampling, and red for dense layers.

available pictures to 186 per person. The head pose is estimated by directional evaluation, as a result this dataset is extremely challenging because the predictor must deal with substantial errors and poor uniformity between subjects.

Many different methods have been tested on the Prima dataset. In [14] a performance comparison is made between high-order Singular Value Decomposition (SVD), Principal Component Analysis (PCA) and locally embedded analysis. Neural networks-based methods have been tested in [36] and [16]. This dataset is the only one in which the human performance has been measured [32].

4.1.1. Methods

This experiment investigated the influence of three factors (network type, optimiser, dropout) and it consisted of two phases:

1. Optimiser selection. In the first phase we used a standard network to select

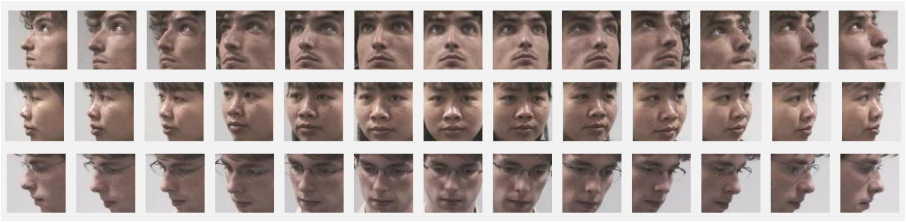

Figure 4: This figure represents a collection of images taken from the Prima dataset as they appear after cropping and scaling.

the best optimiser for this dataset. The standard network used is LeNet-5 [35] shown in Figure 2. The optimisers used were the four described in Section 3.3 plus SGD and SGD with momentum.

2. Network selection. In the second phase we used the best optimiser selected previously for finding the best architecture for each degree of freedom.

In this experiment we used the sigmoid activation function which produced a continuous output in the range $[0, 1]$. As a loss function we used the sum of squares of the differences between the target value y and the estimated value $\hat{y}$, reported in Eq. 2, with $\lambda = 5 \times 10^{-4}$. We trained the networks for 20000 epochs, using mini-batches of size 64. The network weights were sampled from a Gaussian distribution ($\mu = 0$, $\sigma = 0.1$), and any values that had a magnitude more than two standard deviations from the mean were dropped and re-sampled. The weights were updated using Eq. 1. For each optimiser we used the learning rate value recommended by the authors (see Sec. 3.3). When the authors did not suggest any standard value or when the recommended values did not lead to convergence we used a grid-search procedure. Starting from $\alpha = 0.1$ we observed the loss function in the first 1000 epochs. In case of divergence the learning rate was divided by 2 and the procedure repeated. The value which permitted the fastest convergence was then selected and used for the training. Because we used dropout we set the value of the momentum to 0.95 as recommended in [27]. The faces were isolated using the bounding boxes included in the dataset and then resized to 64×64 pixels.

Two kinds of cross-validation tests were applied to this dataset, in accordance with the procedures reported in [37]. Testing on known faces was done by dividing the images per series into two separate folds. The two folds contained images of the same subjects taken under different lighting conditions. Testing on unknown faces is done using the Jack-Knife (leave-one-out) procedure, which consists of training the algorithm on all the subjects but one, which is used for testing. The procedure was repeated fifteen times, leaving out each subject. The mean value of all the measurements is considered to be the final score. We used the leave-one-out procedure to select the best optimiser and the best network. In a second moment the best configuration was tested with the known-subjects procedure. During the training we did not use any kind of early stopping technique. The use of the early stopping requires to monitor the error

on a validation set taken from the test set, but the standard procedure reported in [37] does not take it into account. Considering that the other methods used the standard procedure, we decided to keep the test set unchanged in order to have a fair comparison. However the observation of the error can be useful for understanding if there is overfitting. For this reason, in a separate phase, we generated a validation set randomly picking 50% of the images contained in the test set. We observed the root mean square error (RMSE) for both training and validation set.

Training the architecture A for 20000 epochs on the two-fold dataset took 4.2 hours, while training it on the fifteen-fold dataset took 18.75 hours.

4.1.2. Results

The results in term of MAE for the first phase (optimiser selection) are reported in Table 1. We analysed the convergence speed, observing the loss value during each of the 20000 epochs. For each optimiser we considered the loss values obtained from the mean of the 15 subject in the leave-one-out test, and we plotted the resulting graphs for both pitch (Fig. 5) and yaw (Fig. 6). The Adam optimiser had the lowest MAEs for both pitch and yaw ($10.71 \pm 11.04, 7.74 \pm 8.03$). A similar performance has been obtained with RMSProp ($10.75 \pm 10.51, 8.3 \pm 8.17$).

In the second phase (network selection) we used Adam to train the four architectures shown in Figure 3. The best performances for both pitch and yaw are obtained with the architecture A ($10.71 \pm 11.04, 7.74 \pm 8.03$). A comparison of the four architectures is shown in Figure 7. Mapping the continuous output in 9 discrete categories for pitch and 13 discrete categories for yaw, we were able to interpret the results in terms of classification. The best architecture (A) have an accuracy of 60.93% (pitch) and 62.33% (yaw) on unknown subjects. The accuracy is even higher on known subjects where it reaches 69.26% (pitch) and 67.61% (yaw). The normalised confusion tables for the best architecture are reported in form of heatmaps in Figure 8.

We used the best architecture also for the know-subjects test. We obtained a MAE of 8.06 ± 8.88 for the pitch estimation (accuracy 73.91%), and a MAE of 6.93 ± 7.32 for the yaw estimation (accuracy 66.6%).

To investigate the presence of overfitting, we trained the best architecture using the Adam optimiser on the unknown-subjects test and we generated a validation set randomly picking 50% of the images from the test set. We monitored the RMSE for both training and validation set. The results for the pitch and yaw estimation are reported in Figure 9 and 10 and represent the mean RMSE for each one of the 15 subjects considered in the leave-one-out procedure.

We did some experiments to check if data augmentation of the images (horizontal flip) had an impact on the final score. We did not notice any significant improvement.

The best results on the Prima dataset have been obtained with the smallest network (A). However it is possible that networks with less parameters could perform better than our best architecture. This conclusion is reasonable since the Prima datasets contains a few thousands images and overfitting

problems could deteriorate the performances of larger networks. To further investigate this hypothesis we used the Adam optimiser to train two networks. The networks were similar to LeNet-5 [35] with six layers organised as in Figure 2. The first network had 2.1×10^6 parameters and the following architecture: C1=32(64×64), P1=32(32×32), C2=64(32×32), P2=64(16×16), D1=128, D2=1. The second network had 0.5×10^6 parameters and the following architecture: C1=16(64×64), P1=16(32×32), C2=32(32×32), P2=32(16×16), D1=64, D2=1. The results did not show any major improvement except for the first architecture that got a slightly lower MAE (10.57 ± 10.78) and an higher accuracy (61.4%) for the pitch estimation on unknown subjects. The same network had a worse performance in yaw estimation with an MAE of 8.36 ± 8.42 and accuracy of 57.67%. The network with 0.5×10^6 parameters reported a score of 11.14 ± 11.24 (accuracy 58.02%) for pitch and 8.38 ± 8.57 (accuracy 57.53%) for yaw, which is significantly below the score obtained with architecture A. Also in the known-subjects test we did not observe any major improvement. The network with 2.1×10^6 parameters obtained an MAE of 8.24 ± 9.81 (accuracy 71.15%) for pitch and 7.23 ± 7.43 (accuracy 64.51%) for yaw. The network with 0.5×10^6 parameters obtained an MAE of 8.9 ± 10.35 (accuracy 70.82%) for pitch and 7.23 ± 7.42 (accuracy 64.19%) for yaw. We hypothesise that the use of dropout and L2 regularization helped to prevent overfitting in larger architectures leading to stabler solutions.

The comparison between our approach and other methods is reported in Tab. 2. CNNs perform better than any other methods on the two-fold test for known subjects and in the leave-one-out test for unknown subjects. The comparison in terms of MAE between human and our method shows how CNNs outperform naive humans in both pitch and yaw, and pre-trained humans only for the yaw estimation. The comparison in terms of accuracy (Tab. 2) shows that our method has the best reported scores for tests on known and unknown subjects. In this case the performance of pre-trained humans for pitch estimation on unknown subjects (59%) is lower than our score (60.93%).

4.2. Annotated Facial Landmarks in the Wild

The AFLW dataset [33] provides a collection of 21997 annotated images gathered from an image hosting website. The dataset contains a large variety of appearances (age, ethnicity, occlusions, expressions, etc), lighting and environmental conditions for both genders (56% female, 44% male). Images compatible with the datasets are shown in Figure 11. As reported by the authors the ratio of non-frontal faces (66%) is higher than in other databases. The head pose is estimated from 21 manually annotated landmarks using the POSIT algorithm [38].

Different methods have been tested on this dataset [39, 40, 15, 41, 34]. To compare our work with the others we used the data reported in [15], where the authors describe the performance in terms of MAE for all of these methods, although limited to the yaw angle. The accuracy has been measured by dividing the range $[-90°, 90°]$ into steps of 15°, and it is intended as the percentage of

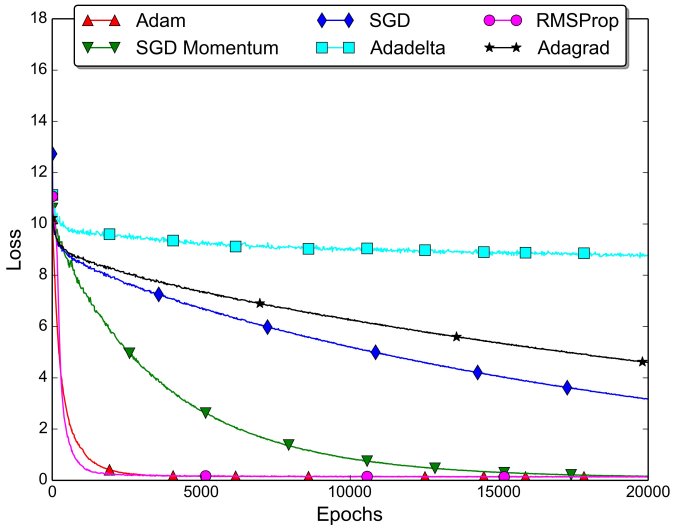

Figure 5: Comparison of different optimisers (trained on network A) for the estimation of the pitch angle on the Prima dataset.

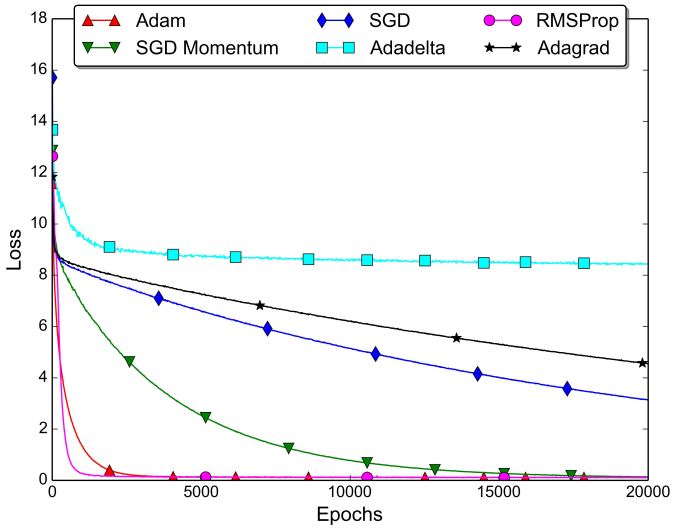

Figure 6: Comparison of different optimisers (trained on network A) for the estimation of the yaw angle on the Prima dataset.

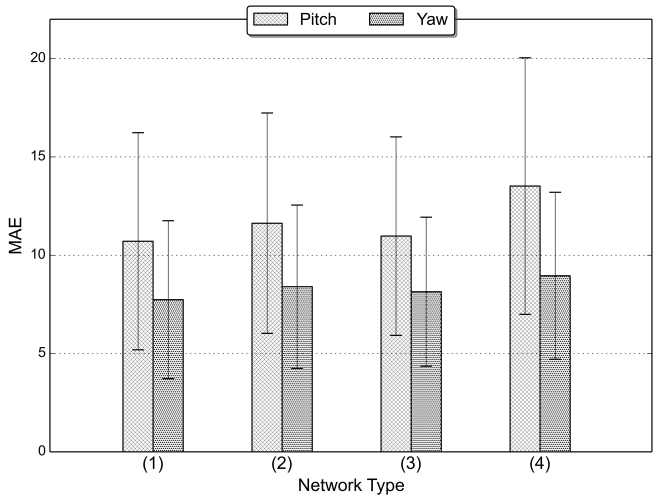

Figure 7: Comparison of the performances of the four networks trained with the Adam optimiser in pitch and yaw estimation of unknown subjects (Prima dataset). The STD has been shrunk by a factor of two for graphical reason.

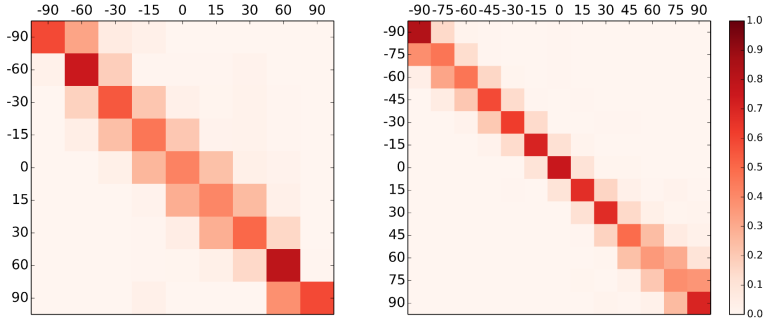

Figure 8: Representation of the confusion tables for pitch (left) yaw angle (right) of the best architecture (A) on unknown subjects (Prima dataset). Each row of the tables represents the instances in a predicted class while each column represents the instances in an actual class.

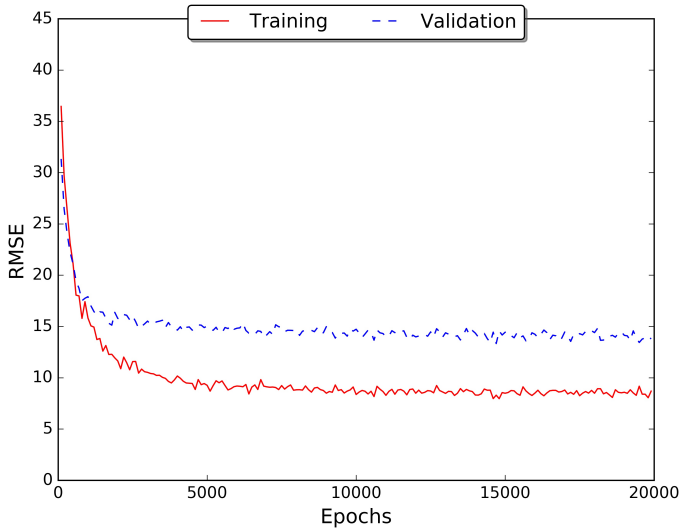

Figure 9: Performance of the best architecture (A) in terms of RMSE (degrees) on the training and validation set for the estimation of the pitch angle on the Prima dataset (unknown subjects test).

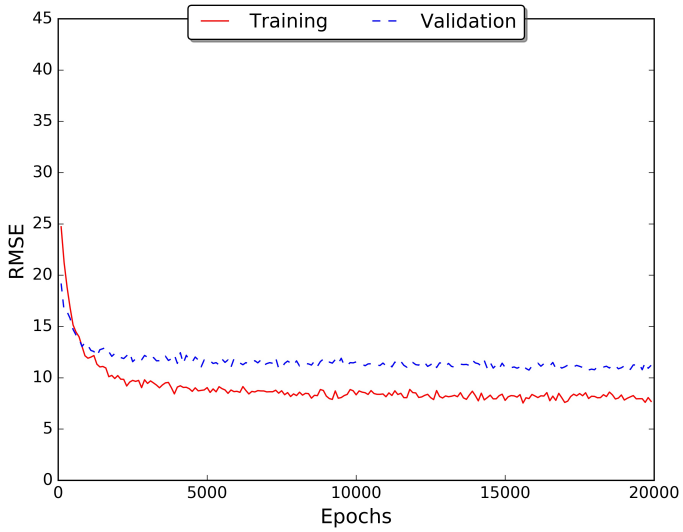

Figure 10: Performance of the best architecture (A) in terms of RMSE (degrees) on the training and validation set for the estimation of the yaw angle on the Prima dataset (unknown subjects test).

Figure 11: This figure represents a collection of images which are compatible with the AFLW and the AFW datasets (the licenses do not allow publishing the original images). The AFLW and the AFW datasets contain a large variety of appearances (age, ethnicity, occlusions, expressions, etc), lighting and environmental conditions for both genders.

images within ±15 degrees of error. In our measurement of the yaw accuracy we used the same metric to enable a comparison with these works.

4.2.1. Methods

In this experiment we manipulated two factors: network type and optimiser. The experiment was divided into two phases:

1. Optimiser selection. In the first phase we used a standard network to select the best optimiser for this dataset. We trained the LeNet-5 [35] using the four optimisers described in Section 3.3 plus SGD and SGD with momentum.
2. Network selection. In the second phase we used the optimiser selected previously for finding the best architecture among the four previously described (Fig. 3).

The AFLW dataset is extremely challenging because the distribution of poses is not uniform. Roll has a mean and standard deviation of 1.07 ± 14.04 degrees, pitch -8.1 ± 13.4 degrees, and yaw 1.91 ± 41.8 degrees. Because of this asymmetrical distribution it is difficult to uniformly map the angles in a continuous interval without losing precision. For this reason we decided to take only images above and below 2.698 standard deviations (1.5 the interquartile range of the lower and higher quartile). The final distribution contained poses in the following ranges: roll=$[-25°, 25°]$, pitch=$[-45°, 45°]$, yaw=$[-100°, 100°]$. To further compensate for the high variability in the angle distribution, we decided to use the hyperbolic tangent activation function. Using the hyperbolic function the output of the networks was constrained to within the range $[-1, 1]$ instead of $[0, 1]$. We considered only faces with a bounding box greater or equal to 64×64 pixels. In total we discarded very few images (less than 5%) because the mean and standard deviation of the bounding boxes was 300 ± 343 pixels, with 84% of the samples distributed between 110 and 647 pixels. In the first phase we trained the networks for 20000 epochs and in the second phase for 30000 (mini-batches of size 64). In both phases we used dropout whit $p = 0.5$. In the second phase we decided to extend the number of epochs to 30000 because the training

was much faster compared to the Prima dataset (e.g. 8.0 vs 18.75 hours for network A). Training the network for more epochs generally leads to stabler solutions and it is recommended when using dropout without time constraint [22].

We applied a five-fold cross-validation procedure to test our networks on the dataset. After randomly dividing the dataset into five folds, we trained the models on four of the five folds and we tested them on the remaining one. The procedure was repeated five times, independently testing the model on each one of the five folds. The mean of the five tests was considered to be the final score. The total number of images included in each one of the five folds was 16696, whereas the number of images left for the test was 4173. To test the classification accuracy we split the yaw poses range $[-100°, 100°]$ into different categories using steps of 15°. The accuracy is intended as the percentage of images with ±15 degrees error. This measure is in line with the one used in [15] and makes it possible to compare the results obtained with the different methods reported in that article. To have more reliable results we took the mean of the five folds.

The hyper-parameters selection followed the same methodology described in Section 4.1.1. The training of roll, pitch and yaw for a single experimental condition (20000 epochs) took 8.0 hours on the smallest architecture (A) and 9.8 hours on the largest architecture (D).

4.2.2. Results

The results for the first phase (optimiser selection) are reported in Table 3. The results show that the RMSProp had the lowest reported MAE for roll, pitch and yaw (4.4 ± 4.35, 7.15 ± 6.0, 11.04 ± 10.86). As it is possible to see in Figure 12 the RMSProp had the fastest convergence rate and it reached the lowest loss values.

The results for the second phase (network selection) are reported in Figure 13. The architecture B had the lowest MAE (4.15 ± 3.87, 6.8 ± 5.64, 9.51 ± 9.21). Mapping the continuous output in three discrete categories for roll, nine for pitch and 13 for yaw, we were able to interpret the results in terms of classification. The accuracy for roll, pitch and yaw was originally $76.55\%, 57.68\%$ and 48.55%. If we consider an error of ±15° the accuracy drastically increases to $99.66\%, 97.24\%$ and 92.47%. To better visualise the accuracy we report in Figure 14 the confusion table of the best network (second architecture, RMSProp optimiser) for roll, pitch and yaw classification.

Similarly to what we did for the Prima dataset, we investigated the presence of overfitting in a separate phase. We trained the best architecture using RMSProp on the five-fold test and we generated a validation set randomly picking 50% of the images from the test fold. We monitored the RMSE for both training and validation set. The results for the yaw estimation are reported in Figure 15 and represent the mean RMSE for each one of the five folds in 20000 epochs.

The comparison with other methods (Tab. 4) shows that CNNs perform better than any other algorithm in terms of MAE and accuracy.

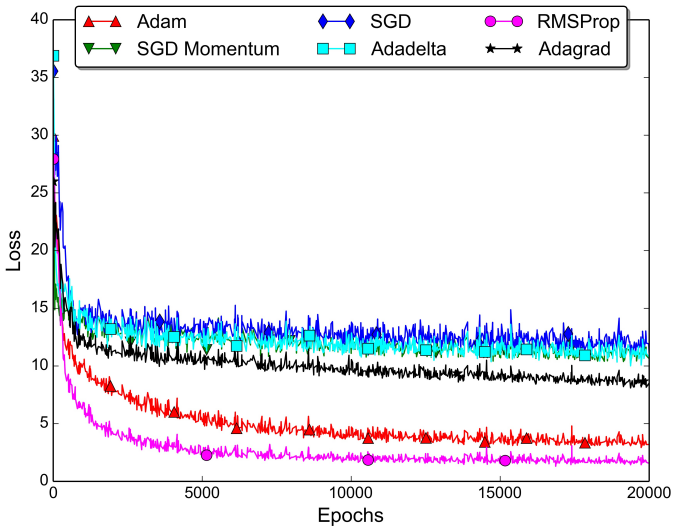

Figure 12: Comparison of the convergence speed between the six optimisers used to train architecture A for yaw estimation on the AFLW dataset. The loss values are the mean of the five fold.

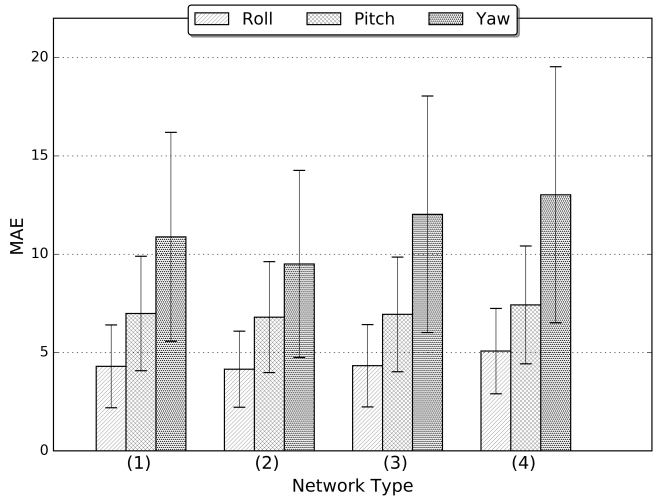

Figure 13: Comparison in term of MAE between four architectures for roll pitch and yaw using the RMSProp optimiser on the AFLW dataset. The STD has been shrunk by a factor of two for graphical reason.

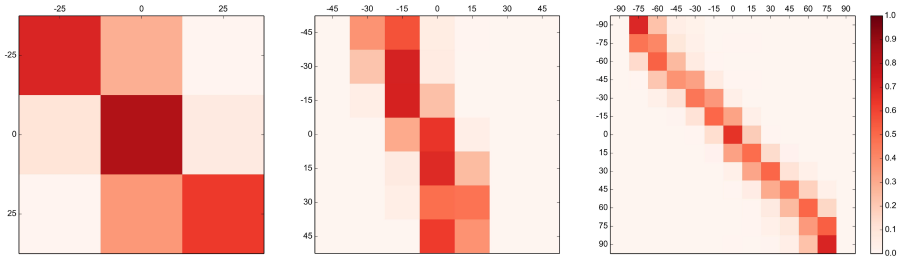

Figure 14: Representation of the confusion tables for roll (left), pitch (centre) and yaw (right) of the best architecture (B) trained with the RMSProp optimiser on the AFLW dataset. Each row of the tables represents the instances in a predicted class while each column represents the instances in an actual class.

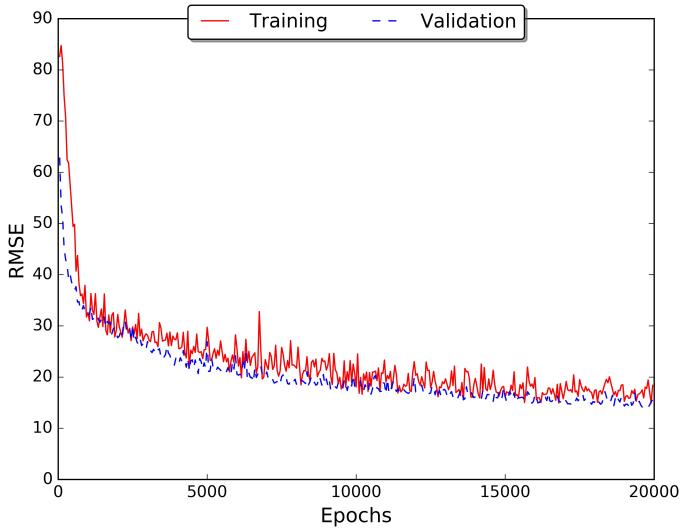

Figure 15: Performance of the best architecture (A) in terms of RMSE (degrees) on the training and validation set for the estimation of the yaw angle on the AFLW dataset.

4.3. Annotated Face in the Wild

The AFW dataset is a benchmark proposed in [34] to test the performance of face detection and head pose estimation methods. Similarly to the AFLW dataset, the AFW is composed of images sampled from social networks. It contains 205 images with 468 faces. Most of the images contain cluttered backgrounds with large variations in both face viewpoint and appearance (age, occlusion, expression, etc). Each face is labeled with a bounding box, 13 discrete viewpoints (range $[-90°, 90°]$) along pitch and yaw directions, and three discrete viewpoints along the roll direction (left, centre, right). This dataset differs from similar collections in its annotation of multiple, non-frontal faces in a single image. Pictures compatible with this dataset are shown in Figure 11. Given the low number of images this dataset is generally used only for testing [15, 34]. In our case we used the AFW dataset to test the best architectures trained on the AFLW dataset. In this way we have one more in-the-wild benchmark for comparing our work with other approaches.

4.3.1. Methods

In this experiment we used the whole AFLW dataset (20869 faces) to train the architecture B. The network was trained for 30000 epochs. We used the RMSProp optimiser with the same hyper-parameters as used in the previous experiment: $\alpha = 0.001$ and $\gamma = 0.9$. Each face presented in the images was cropped and resized to 64×64 pixels. We removed smaller images (only five in total). The trained network was tested on the AFW dataset. To measure the network performance we used the average of five learning cycles. The CNN was initialised five times with random weights, trained on the AFLW dataset and tested on the AFW dataset. The average of the five MAEs has been used as the final score.

4.3.2. Results

We obtained an MAE and STD of 16.73 ± 17.17 and an accuracy of 32.96% which reaches 75.29% when considering an error of $\pm 15°$. This is the best score ever reported on this dataset. The comparison of CNNs with other methods is reported in Table 4. We did some additional tests augmenting the data in the AFLW dataset by horizontal flipping. These experiments did not lead to any major improvement in the final performance.

4.4. Discussion

We now detail the results achieved. The first phase of each experiment consisted of the optimiser selection. The results on the Prima dataset show the superiority of Adam on the other methods. The use of adaptive methods has been extremely important also for in-the-wild datasets where they significantly outperform SGD. In the AFLW dataset the RMSProp had the lowest reported MAE and the highest accuracy. It must be pointed out that in the last epochs the SGD with momentum reaches similar loss values of Adam and RMSProp, however the results in terms of MAE are higher. We hypothesise that the

flexibility of Adam and RMSProp allowed exploring the space better than SGD with momentum, especially because of the high variability of the input features. Another advantage of the adaptive methods is the convergence speed. As it is possible to observe from Figures 5, 6 and 12, Adam and RMSProp converge to a very low loss value during the first epochs.

The second phase of the experiments consisted of the architecture selection. In the Prima dataset, comparing architecture A and its deeper counterpart (B) we see that the second architecture had an highest MAE for pitch ($+0.92°$) and yaw ($+0.66°$) estimation. The same effect has been observed between network C and D ($+2.54°, +0.81°$). We can conclude that adding another convolutional layer or adding more parameters did not lead to any improvement. The reasons could be the small size of the Prima dataset and the controlled environment where the pictures have been taken. In the AFLW dataset, comparing architecture A with architecture B we can see that the second architecture had lowest MAEs for roll, pitch and yaw ($-0.15°, -0.18°, -1.38°$). Augmenting the number of parameters did not lead to any improvement. In this case having an additional convolutional layer made a significant difference, and we can conclude that estimating the head pose in unconstrained datasets requires more complex architectures.

In a separate phase we investigated the presence of overfitting during the training. In the Prima dataset we monitored the RMSE of the best architecture trained with the Adam optimiser for pitch (Fig. 9) and yaw (Fig. 10). In both cases the RMSE on the validation set decreased stably without significantly diverging from the training error. This result seems to indicate that there is an extremely low overfitting and that the CNN has a good generalisation ability for unknown subjects. In the AFLW dataset we observed the RMSE of the best architecture for the yaw prediction 15. Also in this case the validation error decreased stably in accordance with the training curve meaning that the network could effectively generalise to unseen data. Comparing the RMSE of the Prima and AFLW datasets it is possible to notice that in the AFLW the gap between training and validation is reduced thanks to the large amount of images available.

We stated in the introduction that the use of nonlinear regression methods can tolerate systematic errors in the training set. Our experimental results confirm this statement. The accuracy of CNNs is higher than any other method meaning that the networks can grab the general rules and are not heavily affected by mislabelling. To visualise the accuracy we reported the confusion tables as heatmaps for both Prima (Fig. 8) and AFLW (Fig. 14) datasets. The darker cells are disposed along the main diagonal, meaning that the prediction has a high accuracy with false positives constrained to within the proximity of the correct category. The main difference between Prima and AFLW is in the prediction of values which are close to $\pm 90°$. The reason for this difference is the AFLW dataset does not have a uniform distribution and values distant from the mean are very limited.

5. Conclusions

In this article we introduced the use of dropout and adaptive gradient methods for head pose estimation with CNNs. Our approach is significantly different from previous research [18, 20, 21, 17], and show how using the most recent deep learning techniques leads to the state-of-the-art in constrained and unconstrained datasets. Our method should be considered as part of a broader system, in particular it can be used in conjunction with a face detector. We implemented the system in Python using TenworFlow [4] and OpenCV [42]. We used the Viola-Jones object detection framework [43] as a face detector and two CNNs of type B trained on the AFLW dataset as head pose estimator (pitch and yaw). We acquired camera frames with a resolution of 640×480 from a commercial webcam, and then we isolated the faces using the Viola-Jones algorithm. Finally we gave as input the isolated faces to the CNNs obtaining the head pose. The whole system runs at 15 frames per second on a standard laptop (intel core i5, 8 GB RAM) without the use of a GPU. It must be pointed out that an inadequate face detector can be a significant bottleneck. In our case removing the face detector from the loop allows executing the head pose estimation at 20 frames per second. Going to the real-world is not straightforward and different problems can arise. The major problem we experienced during our tests was the increasing in pose estimation errors when the face detector returned a frame which was not well centred on the subjects face. In this case we found useful to train the networks on an augmented version of the AFLW dataset, where the centre of the frame was randomly shifted in one of four directions.

Although our approach is highly competitive, it can be further improved. Other factors can play an important role as pointed out by recent literature. In our opinion future research should particularly focus on the impact of weight initialization [44] and sample selection [45]. Moreover the results can be further improved once in-the-wild datasets with more images and an extended range of poses are available.

Acknowledgment

This material is based upon work supported by the Air Force Office of Scientific Research, Air Force Materiel Command, USAF under Award No. FA9550-15-1-0025.

We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Tesla K40 GPU used for this research.

References

[1] D. Zanatto, M. Patacchiola, J. Goslin, A. Cangelosi, Priming anthropomorphism: Can the credibility of humanlike robots be transferred to non-humanlike robots?, in: Proceedings of the Eleventh Annual ACM/IEEE International Conference on Human-Robot Interaction., IEEE Press, Christchurch, New Zealand, 2016, pp. 543–544.

[2] D. Geronimo, A. M. Lopez, A. D. Sappa, T. Graf, Survey of pedestrian detection for advanced driver assistance systems, IEEE Transactions on Pattern Analisys and Machine Intelligence 32 (2010) 1239–1258.

[3] R. H. Baxter, M. J. V. Leach, S. S. Mukherjee, N. M. Robertson, An adaptive motion model for person tracking with instantaneous head-pose features, IEEE Signal Processing Letters 22 (2015) 578–582.

[4] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, X. Zheng, TensorFlow: Large-scale machine learning on heterogeneous systems, software available from tensorflow.org (2015).
URL `http://tensorflow.org/`

[5] E. Murphy-Chutorian, M. M. Trivedi, Head pose estimation in computer vision: A survey, IEEE Transactions on Pattern Analysis and Machine Intelligence 31 (2009) 607–626.

[6] G. Fanelli, T. Weise, J. Gall, L. V. Gool, Real time head pose estimation from consumer depth cameras, Pattern Recognition 6835 (2011) 101–110.

[7] C. Wang, X. Song, Robust head pose estimation via supervised manifold learning, Neural Networks 53 (2014) 15–25.

[8] J. Wu, M. M. Trivedi, A two-stage head pose estimation framework and evaluation, Pattern Recognition 41 (2008) 1138–1158.

[9] J. Sherrah, S. Gong, E.-J. Ong, Face distributions in similarity space under varying head pose, Image and Vision Computing 19 (12) (2001) 807–819.

[10] D. Lia, W. Pedrycz, A central profile-based 3d face pose estimation, Pattern Recognition 47 (2014) 525–534.

[11] P. Martins, J. Batista, Monocular head pose estimation, in: International Conference Image Analysis and Recognition, Springer, 2008, pp. 357–368.

[12] Q. Liu, J. Yang, J. Deng, K. Zhang, Robust facial landmark tracking via cascade regression, Pattern Recognition.

[13] X. Jin, X. Tan, Face alignment by robust discriminative hough voting, Pattern Recognition 60 (2016) 318–333.

[14] J. Tu, Y. Fu, Y. Hu, T. Huang, Evaluation of head pose estimation for studio data, Lecture Notes in Computer Science 4122 (2007) 281–290.

[15] K. Sundararajan, D. L. Woodard, Head pose estimation in the wild using approximate view manifolds, in: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, Boston, Massachusetts, 2015, pp. 50–58.

[16] R. Stiefelhagen, Estimating head pose with neural networks - results on the pointing04 icpr workshop evaluation data, in: Proceedings of Pointing, ICPR, International Workshop on Visual Observation of Deictic Gestures, Cambridge, UK, 2004, pp. –.

[17] M. Osadchy, Y. L. Cun, M. L. Miller, Synergistic face detection and pose estimation with energy-based models, The Journal of Machine Learning Research 8 (2007) 1197–1215.

[18] B. Ahn, J. Park, I. S. Kweon, Real-time head orientation from a monocular camera using deep neural network, in: 12th Asian Conference on Computer Vision, Springer International Publishing, Singapore, 2014, pp. 82–96.

[19] G. Fanelli, M. Dantone, J. Gall, A. Fossati, L. V. Gool, Random forests for real time 3d face analysis, International Journal of Computer Vision 101 (2012) 437–458.

[20] N. Malagavi, V. Hemadri, U. Kulkarni, Head pose estimation using convolutional neural networks, International Journal of Innovative Sciencem Engineering and Technology 1 (2014) 470–475.

[21] S. S. Mukherjee, N. M. Robertson, Deep head pose: Gaze-direction estimation in multimodal video, IEEE Transactions On Multimedia 17 (2015) 2094–2107.

[22] A. Krizhevsky, I. Sutskever, G. E. Hinton, Imagenet classification with deep convolutional neural networks, in: F. Pereira, C. J. C. Burges, L. Bottou, K. Q. Weinberger (Eds.), Advances in Neural Information Processing Systems 25, Curran Associates, Inc., 2012, pp. 1097–1105.

[23] A. T. Lopes, E. de Aguiar, A. F. De Souza, T. Oliveira-Santos, Facial expression recognition with convolutional neural networks: Coping with few data and the training sample order, Pattern Recognition.

[24] K. Nogueira, O. A. Penatti, J. A. d. Santos, Towards better exploiting convolutional neural networks for remote sensing scene classification, arXiv preprint arXiv:1602.01517.

[25] J. Schmidhuber, Deep learning in neural networks: An overview, Neural Networks 61 (2015) 85–117.

[26] D. E. Rumelhart, G. E. Hinton, R. J. Williams, Learning internal representation by error propagation., Nature 323 (1986) 318–362.

[27] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: A simple way to prevent neural networks from overfitting, Journal of Machine Learning Research 15 (2014) 1929–1958.

[28] J. Duchi, E. Hazan, Y. Singer, Adaptive subgradient methods for online learning and stochastic optimization, Journal of Machine Learning Research 12 (Jul) (2011) 2121–2159.

[29] M. D. Zeiler, Adadelta: an adaptive learning rate method, arXiv preprint arXiv:1212.5701.

[30] T. Tieleman, G. Hinton, Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude, COURSERA: Neural Networks for Machine Learning 4 (2).

[31] D. Kingma, J. Ba, Adam: A method for stochastic optimization, arXiv preprint arXiv:1412.6980.

[32] N. Gourier, D. Hall, J. L. Crowley, Estimating face orientation from robust detection of salient facial features, in: Proceedings of Pointing 2004, ICPR, International Workshop on Visual Observation of Deictic Gestures, Cambridge, UK, 2004, pp. –.

[33] M. Koestinger, P. Wohlhart, P. M. Roth, H. Bischof, Annotated facial landmarks in the wild: A large-scale, real-world database for facial landmark localization, in: First IEEE International Workshop on Benchmarking Facial Image Analysis Technologies, 2011.

[34] X. Zhu, D. Ramanan, Face detection, pose estimation, and landmark localization in the wild, in: Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on, IEEE, 2012, pp. 2879–2886.

[35] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, in: Proceedings of the IEEE, IEEE, 1998, pp. 2278–2324.

[36] M. Voit, K. Nickel, R. Stiefelhagen, Neural network-based head pose estimation and multi-view fusion, in: 1st international evaluation conference on Classification of events, activities and relationships, 2007, pp. 291–298.

[37] N. Gourier, J. Maisonnasse, D. Hall, J. L. Crowley, Head pose estimation on low resolution images, Lecture Notes in Computer Science 4122 (2006) 270–280.

[38] D. F. Dementhon, L. S. Davis, Model-based object pose in 25 lines of code, International Journal of Computer Vision 15 (1995) 123–141.

[39] J. Aghajanian, S. Prince, Face pose estimation in uncontrolled environments., in: BMVC, Vol. 1, 2009, p. 3.

[40] C. Hegde, A. C. Sankaranarayanan, R. G. Baraniuk, Learning manifolds in the wild, Preprint, July 1 (2) (2012) 4.

[41] M. Torki, A. Elgammal, Regression from local features for viewpoint and pose estimation, in: 2011 International Conference on Computer Vision, IEEE, 2011, pp. 2603–2610.

[42] Itseez, Open source computer vision library, `https://github.com/itseez/opencv` (2015).

[43] P. Viola, M. Jones, Rapid object detection using a boosted cascade of simple features, in: Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on, Vol. 1, IEEE, 2001, pp. I–511.

[44] D. Ribeiro, J. C. Nascimento, A. Bernardino, G. Carneiro, Improving the performance of pedestrian detectors using convolutional learning, Pattern Recognition.

[45] W. Yang, L. Jin, D. Tao, Z. Xie, Z. Feng, Dropsample: A new training method to enhance deep convolutional neural networks for large-scale unconstrained handwritten chinese character recognition, Pattern Recognition 58 (2016) 190–203.

[46] E. Ricci, J.-M. Odobez, Learning large margin likelihoods for realtime head pose tracking, in: 16th IEEE International Conference on Image Processing (ICIP), IEEE, Cairo, 2009, pp. 2593–2596.

[47] N. Alioua, A. Amine, M. Rziza, A. Bensrhair, D. Aboutajdine1, Head pose estimation based on steerable filters and likelihood parametrized function., in: 21st European Signal Processing Conference (EUSIPCO 2013), IEEE, Marrakech, 2013, pp. 1–5.

Table 1: In this table we report the results obtained on the Prima dataset for leave-one-out (unknown subjects) test using different optimisers. These results have been obtained training the architecture A for 20000 epochs (18.75 hours). The results are in terms of MAE (accuracy) with MAE expressed in degrees and Accuracy expressed in percentage. The best scores are in bold.

Optimiser	**Pitch**	**Yaw**
Adadelta [29]	20.71(35.3)	13.7(35.23)
Adagrad [28]	12.57(53.69)	9.23(54.73)
Adam [31]	**10.71(60.93)**	**7.74(62.33)**
RMSProp [30]	10.75(57.67)	8.30(58.92)
SGD	12.87(52.11)	9.06(56.06)
SGD (Momentum)	13.26(49.35)	8.66(56.81)

Table 2: In this table we compare the results of our method with the result obtained by other authors on the Prima head pose dataset. These results have been obtained training architecture A for 20000 epochs on both unknown (18.75 hours) and know (4.2 hours) test with the Adam optimiser. The results are in term of MAE(accuracy), with MAE expressed in degrees and Accuracy expressed in percentage. The best scores are in bold.

	Unknown Subjects		**Known Subjects**	
Method	**Pitch**	**Yaw**	**Pitch**	**Yaw**
Human Performance [37]	12.6(48)	11.9(42.4)	–	–
Human Performance (training) [37]	9.4(59)	11.8(40.7)	–	–
Locally Embedded Analysis [14]	–	–	17.44(50.61)	15.88(45.16)
High-order SVD [14]	–	–	17.97(54.84)	12.9(49.25)
PCA [14]	–	–	14.98(57.99)	14.11(55.2)
Neural Network [36]	–	–	12.77(52.1)	12.3(41.8)
Large Margin Likelihoods [46]	–	–	10.5	9.1
Associative Memories [37]	15.9(43.9)	10.3(50.04)	10.1(61.7)	8.5(60.8)
Steerable Filters [47]	13.8	11.0	12.4	9.6
Steerable Filters (manual) [47]	11.4	9.97	10.1	8.7
CNNs [our]	**10.57(61.4)**	**7.74(62.33)**	**8.06(73.91)**	**6.93(66.6)**

Table 3: In this table we report the results obtained on the AFLW dataset for the five-fold cross validation test. These results have been obtained training the architecture A for 30000 epochs (14.6 hours). The results are in terms of MAE(accuracy). MAE is expressed in degrees and Accuracy is expressed in percentage. The best scores are in bold.

Optimiser	**Roll**	**Pitch**	**Yaw**
Adadelta [29]	6.26(22.2)	9.98(42.67)	21.87(22.19)
Adagrad [28]	5.27(69.23)	8.24(51.29)	17.83(28.8)
Adam [31]	4.81(72.41)	7.78(53.49)	13.5(38.56)
RMSProp [30]	**4.4(75.14)**	**7.15(55.98)**	**11.04(44.54)**
SGD	7.0(58.64)	9.89(42.15)	22.98(22.23)
SGD momentum	6.17(63.19)	10.3(42.37)	21.31(22.52)

Table 4: In this table we report the results in term of MAE (degrees) and accuracy (percentage) obtained with different methods for the yaw estimation on the AFLW and the AFW datasets. Our results have been obtained training the architecture B for 30000 epochs (14.6 hours) with the RMSProp optimiser. MAE is expressed in degrees and Accuracy is expressed in percentage. The best scores are in bold.

Method	**AFLW**	**AFW**
Mixture of trees [34]	46.54(15.72)	40.17(26.07)
Patch Based [39]	38.39(23.87)	41.67(21.36)
Feature Embedding [41]	33.01(32.82)	28.15(40.38)
Learning Manifold [40]	16.31(63.13)	18.26(58.33)
Approximate View Manifold [15]	17.48(58.05)	17.2(58.33)
CNNs [our]	**9.51(92.47)**	**16.73(75.29)**