Biternion Nets: Continuous Head Pose Regression from Discrete Training Labels

Lucas Beyer, Alexander Hermans, and Bastian Leibe

Visual Computing Institute, RWTH Aachen University

Abstract. While head pose estimation has been studied for some time, continuous head pose estimation is still an open problem. Most approaches either cannot deal with the periodicity of angular data or require very fine-grained regression labels. We introduce biternion nets, a CNN-based approach that can be trained on very coarse regression labels and still estimate fully continuous 360° head poses. We show state-of-the-art results on several publicly available datasets. Finally, we demonstrate how easy it is to record and annotate a new dataset with coarse orientation labels in order to obtain continuous head pose estimates using our biternion nets.

1 Introduction

The estimation of head poses is an important building block for higher-level computer vision systems such as social scene understanding, human-computer interfaces, driver monitoring, and security systems. For many of these tasks, a continuous head pose angle is arguably more useful than few discrete orientation classes as yielded by most current head pose systems [8,34,4].

While many face pose and gaze estimation methods have been covered in the literature, the task of regressing *head* pose is distinctly different in that it also handles people not facing the camera, resulting in poses spanning the full 360° spectrum. Thus, head pose estimators need to be able to cope with the periodicity of angular data, *i.e.* the fact that 361° corresponds to 1° and, for a head pose of 0°, a prediction of 359° is no worse than a prediction of 1°. Face pose and gaze estimators can conveniently sidestep this difficulty by constraining the prediction range to non-periodic intervals such as $[-90°, 90°]$. Another difficulty in learning a head pose regressor lies in obtaining enough training data with accurate regression labels [6,11]. All publicly available datasets, except [5], are either restricted to coarse orientation bins, or to the range of front-facing poses [6,14,2,15,9,10].

A multitude of approaches [25,32] has been proposed which solve only one of the two aforementioned problems: either they cannot cope with periodicity [26,24,34], or they need fine-grained regression data [35,33,16]. Since none of this is satisfactory, we propose a principled approach to solve both problems simultaneously.

Our approach is based on convolutional neural networks (CNNs), for which we propose a novel output layer embedding an angle into two dimensions, coupled

with a fitting cost function. It is able to handle fully periodic, continuous regression while *only requiring coarse, discrete class-labels* as training data, which are easily obtainable from video recordings. We call our approach *biternion nets*. Before demonstrating the effectiveness of the biternion output layer, we validate our CNN architecture on several publicly available datasets and show that it yields state-of-the-art results.

In summary, our contributions are threefold: (1) We present a CNN architecture that outperforms state-of-the art results on several public head pose datasets. (2) We propose a novel combination of output layer and cost function to elegantly solve the problem of periodic orientation regression, which we call *biternion nets*. (3) We show that we can learn *continuous* head-pose regression from discrete training labels. To demonstrate this, we present continuous regression results obtained from a biternion net trained on data recorded and annotated in less than two and a half hours.

2 Related Work

Head pose estimation has been a very active research field for the past 20 years [32,25]. Over time, authors have developed many different methods to approach this problem. The probably most popular direction is the functional mapping of images to a feature space where classifiers or regressors can directly be applied. These mappings range from simple gradient-based features [24,7,21], over covariance features [34], to learned functional mappings [26,33,4]. These approaches often result in a manifold embedding of the images [26,34]. However, if training data is sparse, it is hard to ensure the quality of these manifolds [19]. Another approach is to find facial landmarks, such as eye and mouth locations, and use these to determine the pose of a face [9]. It is also possible to use tracking information to get a good prior for the head pose [7,10]. Here, interactions between the body pose and the head pose can be exploited [5,8]. Several of these techniques have also been used for objects such as cars or chairs [33,28,18].

While some of these approaches work on high resolution images [14,2,10,12], the majority of them is based on low resolution images [26,24,5,34]. With the recent availability of cheap RGB-D sensors, depth information has also been used to improve head pose estimation [12].

The high activity within this field has resulted in a large number of different datasets for head pose estimation [6,14,2,1,15,5,34,9,10], most of which are face pose rather than head pose datasets and often only contain sparse head poses and fairly coarse orientation labels. As we are interested in continuous head pose estimations, most of these datasets are not suitable for our experiments.

Based on the available datasets, most approaches focus on coarse face poses, while only few *head* pose estimation approaches and datasets exist [35,5,34]. Wu and Toyama [35] estimate gradient distributions from 1024 different viewpoints and match new views to the nearest viewpoint to determine the pose. Benfold and Reid [5] use the walking direction obtained from unsupervised people tracking in a video sequence to train a regression forest for the head pose. Tosato *et al.* [34] use covariance features to classify head poses into a small set of orientation bins.

CNNs have also been used for orientation estimation before. Qi [28] fine-tunes a large pre-trained CNN to classify the orientation of chairs using a large amount of rendered chairs with precise labels. However, using CNNs pre-trained on ImageNet for low-resolution head pose estimation makes no sense due to the significantly different filter resolution, type of data, and learning task. Most similar to our approach is the one by Osadchy *et al.* [26], which also uses a CNN for continuous head pose estimation. They learn a face manifold on (non-public) data with regression labels, which enables them to jointly detect and estimate the pose of faces. In contrast to us, they focus on using face pose data to improve face detection and do not address the periodicity problem.

Some approaches also aim at solving the periodicity problem [33,16,18]. However, their approaches are typically based on nearest-neighbor matching or kernel density estimation, meaning that they require dense orientation labels for training. All three of the above approaches use fine grained face datasets [14,1] and it is unclear how well they could perform for head pose estimation.

To the best of our knowledge, only Huang *et al.* [19] aim at learning continuous regressors from a discrete face pose dataset. They learn a mixture of local tangent subspaces that are robust to regression regions with bad coverage in the training set. Their representation is based on HOG features and they use high resolution images. It is questionable whether their approach can deal with head poses, as HOG features are not very expressive for the back of a head. Furthermore, they do not evaluate how continuous their regression really is.

In conclusion, based on existing approaches, the task of continuous periodic head pose estimation is still unsolved. Here our approach comes into play.

3 CNNs for Head Pose Estimation

Throughout this paper, we work in the framework of deep convolutional networks and stochastic, gradient-based optimization. In this section, we present the specific network architecture we use for all experiments, changing only the output layer and cost function to match the task at hand. We then apply it to multiple publicly available datasets, consistently outperforming current state-of-the-art methods on those datasets.

3.1 The Network Architecture

We use a moderately deep, batch-normalized [20], VGG-style network architecture [30] consisting of six convolutional layers with 24, 24, 48, 48, 64 and 64 feature channels, respectively, followed by a single hidden layer of 512 units, and train it for a fixed duration of 50 epochs in all our experiments. For all details about the network and the training procedure, please refer to the supplementary material. We implemented the network in Theano [3] using IPython notebook [27]. All numbers reported within this paper are averages over five runs. While we will show that this architecture already performs very well, it is likely possible to reduce the error even further by using deeper networks with

Table 1: Class-average accuracies on the four classification datasets from [34]. The sample counts refer to the provided train/test splits. We obtain state-of-the-art results on all datasets.

	HIIT	HOCoffee	HOC	QMUL	
# Samples	12 000/12 007	9522/8595	6860/5021	7603/7618	9813/8725
# Classes	6	6	4	4	4 + 1
Tosato *et al.* [34]	96.5%	81.0%	78.69%	94.25%	91.18%
Lallemand *et al.* [21]	-	-	79.9%	-	-
Our CNN	**98.70%**	**86.99%**	**83.97%**	**95.58%**	**94.30%**

more careful regularization and a bag of other well-known tricks [23,13,36,29,17]. We do not further go down that road, since the goal of this section is simply to demonstrate the suitability of CNNs in general, and our architecture in particular, for predicting head poses on low-resolution images.

3.2 Experimental Validation

We use the collection of datasets provided by Tosato *et al.* [34] to validate our approach. First, we show results on those datasets that treat pose estimation as a classification task in Table 1. These datasets contain very rough pose bins, such as `Front`, `Back`, `Left` and `Right`, with the addition of `FrontLeft` and `FrontRight` for HIIT and HOCoffee, and `Background` for the 5-class version of the QMUL dataset.

In this case, the network's output layer is a softmax-layer and the cost being optimized is the negative log-likelihood. While the accuracies obtained by state-of-the-art methods are already high, we show that our CNN architecture achieves a significant improvement as it reduces the error by about a third across all datasets.

We next turn to the datasets with continuous regression labels. Statistics about the datasets are shown in Table 2, together with our results. The IDIAP Head Pose dataset, which stems from a video recording of few people in a meeting room, has a very restricted range of angles; specifically, 94 % of the pan angles lie within the rather narrow, front-facing range of $[-60°, 60°]$. For this experiment, the output of our network is computed by a fully-connected layer with three outputs and the cost function is the mean absolute deviation. This simple approach to pan-tilt-roll regression outperforms the state-of-the art in all three dimensions. Please note that with a linear output layer and the MAD cost function, the network does not learn the pan, tilt and roll angles jointly; they merely share a common feature representation.[1]

The CAVIAR dataset comes in both a clean version containing only fully-visible heads, and an occluded version containing *only* partially-occluded heads. While they do come in the full range of angles, almost 40 % of the training samples lie within $\pm 4°$ of the four canonical orientations. A major downside of

[1] This becomes evident by computing the derivatives of the cost w.r.t. the parameters: the tilt and roll terms are absent from the derivative w.r.t. the pan and vice-versa.

Table 2: A comparison to two regression datasets from [34]. The first number is the mean absolute angular deviation, the second its standard deviation across test-samples. We obtain state-of-the-art results on all datasets.

	IDIAP Head Pose			CAVIAR-c	CAVIAR-o
# Samples	42 304/23 991			10 660/10 665	10 802/10 889
Pose range	pan	tilt	roll	pan	pan
	[-101,101]	[-73,23]	[-46,65]	[0, 360]	[0, 360]
Tosato *et al.* [34]	10.3°±10.6°	4.5°±5.3°	4.3°±3.8°	22.7°±18.4°	35.3°±24.6°
Ba & Odobez [2]	8.7°±9.1°	19.1°±15.4°	9.7°±7.1°	-	-
Our CNN	**5.9°±7.2°**	**2.8°±2.6°**	**3.5°±3.9°**	**19.2°±24.2°**	**25.2°±26.4°**

this dataset is that most images have been upscaled to 50-by-50 pixels from their original size of, on average, 7-by-7 pixels. We still perform the comparison for the sake of completeness, and our network manages to beat the current state-of-the-art on such a difficult dataset.

These experiments show that the network architecture we use forms a solid basis by itself and we can now use it to further investigate continuous, periodic orientation regression.

4 Periodic Orientation Regression

None of the datasets in the previous section really uncover a crucial problem for full head-orientation regression: periodicity. We can demonstrate that this is a real problem by adding 360° to all negative pan values of the IDIAP dataset. With this semantically identical dataset, the exact same (naive) network used in the previous section becomes very unstable and only reaches errors of 12.9°, 4.5° and 5.3° for pan, tilt and roll, respectively.

For memory-based models such as k-NN and kernel-methods, periodicity only plays a role during the voting part of the algorithm, where it can easily be solved by a modulo operation. But this kind of model suffers from the inherent need of fine-grained training data, hence our focus on parametric models.

For parametric models such as CNNs, periodicity may cause problems in two different ways: (1) The cost function to be optimized is unaware of the fact that a prediction of 359° for a ground truth orientation of 0° should incur the same loss as 1°. Unfortunately, simply applying a `mod` operator to the output of the network results in a discontinuous error function that can no longer be optimized robustly. (2) A regression output which results from a matrix-vector product, such as performed in most parametric models, is an inherently linear operation, while we ideally want a circular output.

Our biternion approach solves both of these problems in an elegant way.

4.1 Von Mises Cost Function

The first problem of discontinuity in the cost function can be addressed by turning to the von Mises distribution [22], which is a close approximation to the normal distribution on the unit circle:

$$p_{\mathrm{VM}}(\varphi \mid \mu, \kappa) = \frac{e^{\kappa \cos(\varphi - \mu)}}{2\pi I_0(\kappa)} \ . \tag{1}$$

Equation (1) defines its probability density function, where φ is an angle, μ is the mean angle of the distribution, κ is inversely related to the variance of the approximated Gaussian, and $I_0(\kappa)$ is the modified Bessel function of order 0, which is a constant for fixed κ. Since it leverages the cosine function to avoid any discontinuity, it is well-suited for gradient-based optimization and we can derive the following cost function by inverting and scaling it accordingly:

$$C_{\mathrm{VM}}(\varphi \mid t; \kappa) = 1 - e^{\kappa(\cos(\varphi - t) - 1)}. \tag{2}$$

In the cost formulation, we call t the target value and κ is a simple hyperparameter that controls the tails of the loss function.

4.2 Biternion Representation for Orientation Regression

While the von Mises cost presented above solves the first issue, the fundamental problem of predicting a periodic value using a linear operation persists. Also, $\|\mathbf{y}\| = 1$Inspired by the quaternion representation often found in computer graphics, we propose a natural alternative representation of an angle by the two-dimensional vector consisting of its sine and cosine $\mathbf{y} = (\cos\varphi, \sin\varphi)$, which we call the *biternion representation*. Surprisingly, the only use of a similar encoding we found in the related literature is that by Osadchy *et al*. [26], who also embed angles into a similar, albeit different, higher-dimensional space. Unfortunately, their approach does not solve the periodicity problem since it uses the discontinuous `atan2` function.

The biternion representation immediately suggests the use of the continuous, cyclic cosine cost widely used in the NLP literature [31]:

$$C_{\cos}(\mathbf{y} \mid \mathbf{t}) = 1 - \frac{\mathbf{y} \cdot \mathbf{t}}{\|\mathbf{y}\| \, \|\mathbf{t}\|} \ . \tag{3}$$

Implementing a biternion output-layer in any framework for neural networks is relatively straightforward, since all that is needed is a fully-connected layer and a normalization layer. For clarity, Equation 4 gives the operation performed by a biternion-layer during the forward pass, where $\mathbf{W} \in \mathbb{R}^{n \times 2}$ and $\mathbf{b} \in \mathbb{R}^2$ are the learnable parameters from the fully-connected layer:

$$f_{\mathrm{BT}}(\mathbf{x}; \mathbf{W}, \mathbf{b}) = \frac{\mathbf{W}\mathbf{x} + \mathbf{b}}{\|\mathbf{W}\mathbf{x} + \mathbf{b}\|} \tag{4}$$

The derivative of the normalization, necessary for the backward pass, can then be stated as

$$\partial_{x_i} \frac{\mathbf{x}}{\|\mathbf{x}\|} = \partial_{x_i} \frac{\mathbf{x}}{\sqrt{\sum_j x_j^2}} = \frac{\sum_{j \neq i} x_j^2}{\left(\sum_j x_j^2\right)^{\frac{3}{2}}} = \frac{\sum_{j \neq i} x_j^2}{\|\mathbf{x}\|^3} \ . \tag{5}$$

Notice how (1) the normalization in the biternion layer makes sure the output values are learned *jointly* and (2) the normalization terms in $C_{\cos}$ can subsequently be omitted.

Finally, the ensembling of multiple biternion predictions, as needed by some augmentation techniques, can simply be performed by averaging the vectors, since the average of unit vectors is again a unit vector, a fact also used by Hara *et al*. [16].

Biternions are Restricted Quaternions. We now show that biternions correspond to unit-quaternions restricted to a single reference axis of rotation. Let Q_φ be the quaternion $\left(a_x \sin(\frac{\varphi}{2}), a_y \sin(\frac{\varphi}{2}), a_z \sin(\frac{\varphi}{2}), \cos(\frac{\varphi}{2})\right)$ representing a rotation of φ around the axis $\mathbf{a}$ and Q_θ the quaternion representing a rotation of θ around the same axis. A quaternion representing the immediate rotation from Q_φ to Q_θ can be computed as $\frac{Q_\varphi}{Q_\theta}$, which corresponds to:

$$\begin{pmatrix} -\cos(\frac{\varphi}{2})a_x \sin(\frac{\theta}{2}) + a_x \sin(\frac{\varphi}{2}) \cos(\frac{\theta}{2}) - a_y \sin(\frac{\varphi}{2})a_z \sin(\frac{\theta}{2}) + a_z \sin(\frac{\varphi}{2})a_y \sin(\frac{\theta}{2}) \\ -\cos(\frac{\varphi}{2})a_y \sin(\frac{\theta}{2}) + a_y \sin(\frac{\varphi}{2}) \cos(\frac{\theta}{2}) - a_z \sin(\frac{\varphi}{2})a_x \sin(\frac{\theta}{2}) + a_x \sin(\frac{\varphi}{2})a_z \sin(\frac{\theta}{2}) \\ -\cos(\frac{\varphi}{2})a_z \sin(\frac{\theta}{2}) + a_z \sin(\frac{\varphi}{2}) \cos(\frac{\theta}{2}) - a_x \sin(\frac{\varphi}{2})a_y \sin(\frac{\theta}{2}) + a_y \sin(\frac{\varphi}{2})a_x \sin(\frac{\theta}{2}) \\ \cos(\frac{\varphi}{2}) \cos(\frac{\theta}{2}) + a_x \sin(\frac{\varphi}{2})a_x \sin(\frac{\theta}{2}) + a_y \sin(\frac{\varphi}{2})a_y \sin(\frac{\theta}{2}) + a_z \sin(\frac{\varphi}{2})a_z \sin(\frac{\theta}{2}) \end{pmatrix}$$

Using the fact that $\|\mathbf{a}\| = 1$, the last entry of the quaternion —which encodes the cosine of half the angle represented by the quaternion— simplifies to $\cos(\frac{\varphi}{2}) \cos(\frac{\theta}{2}) + \sin(\frac{\varphi}{2}) \sin(\frac{\theta}{2}) = \cos(\frac{\varphi-\theta}{2})$. The other entries can similarly be simplified, resulting in a quaternion representing a rotation of the angle from φ to θ around the same axis $\mathbf{a}$. This shows that biternions can be seen as quaternions around a fixed reference axis $\mathbf{a}$ and the cosine cost corresponds to the amplitude of the direct rotation between the predicted and the target biternions.

Relationship to the von Mises Cost. By comparing C_{VM} and $C_{\cos}$, it is visible that they do *not* compute the same expression, *i.e.*, the biternion-layer coupled with the cosine cost does *not* optimize the von Mises cost. The von Mises cost for the biternion layer can be written as:

$$C_{\mathrm{VM,BT}}(\mathbf{y} \mid \mathbf{t}) = 1 - e^{\kappa(\mathbf{y} \cdot \mathbf{t} - 1)}. \tag{6}$$

Notice the similarity to Equation 3; the main difference is the presence of e, which "pushes down" the error around the target value, in effect penalizing small mistakes less strongly.

4.3 Experimental Results

In order to investigate the relative usefulness of the von Mises cost and the biternion representation for periodic regression, we now turn to the TownCentre dataset [5]. This dataset contains heads of tracked pedestrians in a shopping district, annotated with head pose regression labels. The prior distribution of the pose angle is shown in the middle of Fig. 1. For all experiments, we train on 7920 heads of 3960 persons and evaluate on 774 heads of 387 random but *different* persons. The results can be seen in Table 3.

As a first baseline, we train a shallow linear regressor on raw pixel values. We then train a deep CNN using a naive regression output and cost, as described in Section 3.2. While the depth of the architecture allows it to perform much better, it is still plagued by the two problems of cyclic regression. Using the von Mises cost solves the first problem in the cost function; this reduces the error by a significant amount, showing that the more appropriate cost function indeed does aid optimization. Following this, we evaluate the performance of a biternion net both with the cosine cost and the von Mises cost. As can be seen, the expressive power of the biternion layer solves both problems encountered in periodic regression and produces the best results.

Table 3: Quantitative regression results for the TownCentre dataset [5].

Method	MAE
Linear Regression	$64.1° \pm 45.0°$
Naive Regression	$38.9° \pm 40.7°$
Von Mises	$29.4° \pm 31.3°$
Biternion	$21.6° \pm 25.2°$
Biternion+Von Mises	$\mathbf{20.8}° \pm \mathbf{24.7}°$
Benfold&Reid [5]	25.6° / 64.9°

It should be noted that we cannot fairly compare to most of the related work for various reasons: the results in [8] have been computed on only 15 persons, which is far from representative for this dataset. Chamveha *et al.* [7] use a tracker and scene-specific orientation priors. Even the numbers from Benfold and Reid [5] are not a fair comparison since they use walking direction as a prior. The first of their numbers in Table 3 is achieved by a regressor which has seen all persons and their walking direction during training[2], while the second of their numbers has not seen any of the persons since it has been trained on a different dataset.

5 Continuous Regression from Discrete Training Labels

We have shown that biternion nets are well-suited to fully-periodic head pose regression. We now turn to the third contribution of this paper, namely the ability to perform continuous head pose regression using only discrete pose labels for training. To simulate discrete pose labels, we discretize the continuous annotations of the TownCentre dataset. By varying the number of discrete bins, we generate multiple datasets on which we train various approaches using only *the centers of the bins* as training labels. We then evaluate the predictions made by these approaches by computing their mean angular deviation w.r.t. the full regression annotations of the test set. All results are reported in Table 4. We first apply two classification-based baselines, followed by all regression-based approaches introduced in Section 4.

In order to train a regressor using discrete pose labels, a first rather simplistic approach commonly found in the literature is to train a classifier which outputs the class center as prediction. For probabilistic classifiers, a natural extension of this approach is to output the argmax of a quadratic interpolation of the class with the highest posterior probability and its neighboring classes. On average, this improves the results by about 2°.

[2] Their setup is justified for their task, but makes a fair comparison impossible.

Table 4: Regression results from different approaches for different discretizations. Here infinity represents no discretization. Note that the Biternion layer handles the discrete labels very well, both with the cosine and the von Mises cost.

Class bins	Class center	Class interpolation	Naive regression	Von Mises	Biternion	Biternion + Von Mises
3	37.2°±32.8°	35.5°±30.4°	45.5°±39.7°	36.6°±34.5°	**32.1°±28.1°**	32.2°±28.8°
4	34.9°±30.5°	31.7°±29.3°	43.0°±40.6°	33.4°±32.2°	27.1°±27.3°	**26.9°±27.4°**
6	26.1°±28.4°	24.1°±27.6°	38.3°±38.5°	31.8°±33.1°	**22.1°±25.5°**	22.7°±26.7°
8	24.5°±28.6°	22.6°±28.0°	40.6°±39.7°	30.2°±32.3°	21.8°±24.9°	**21.3°±25.2°**
10	23.8°±27.5°	21.9°±26.9°	37.6°±38.3°	28.8°±30.8°	**21.4°±24.6°**	21.8°±25.5°
12	23.6°±29.4°	22.2°±28.8°	39.0°±38.2°	29.7°±31.5°	**21.4°±25.3°**	21.8°±25.3°
∞	-	-	38.9°±40.7°	29.4°±31.1°	21.6°±25.2°	**20.8°±24.7°**

CNNs compute a continuous function of their input and, during training, each sample *pulls* the parameters of the CNN slightly into a direction leading to a better prediction of its pose. This intuition suggests that it should be possible for CNNs to learn a continuous mapping from images to pose angles even when only given very rough pose labels. This is shown in the last four columns of Table 4. As can be seen, this idea hardly works at all in the naive regression case and is only somewhat improved by the von Mises cost. Biternion nets, on the other hand, have no difficulty being trained this way and in fact outperform the class-based approaches with any number of realistically annotable classes, whether the cosine or the von Mises cost is used

Unfortunately, looking only at numbers representing an average error over a large amount of images does not reflect the real advantage of biternion nets over the classifier approach. For this reason, we plotted heatmaps of the predictions made by a CNN classifier with quadratic interpolation and the predictions made by a biternion net in Figure 1. These heatmaps clearly show that, while the class-interpolation approach and biternion nets give similar scores, the predictions of the biternion nets are vastly superior because they are more continuous and similar to the distribution of the ground-truth angles.

5.1 Practicality

To show the potential of our approach, we recorded a small dataset using a common smartphone camera and annotated it with eight class labels. For this, we recorded 24 people in our lab and asked them to rotate on the spot. We then manually cropped a square region in the resulting videos containing their head and rescaled it to 50×50 pixels to make it compatible to our network

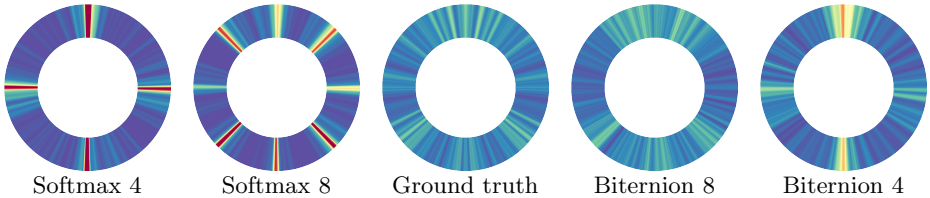

Fig. 1: Prediction distributions for softmax and biternion output layers trained on different discretizations. The classification results include the interpolation.

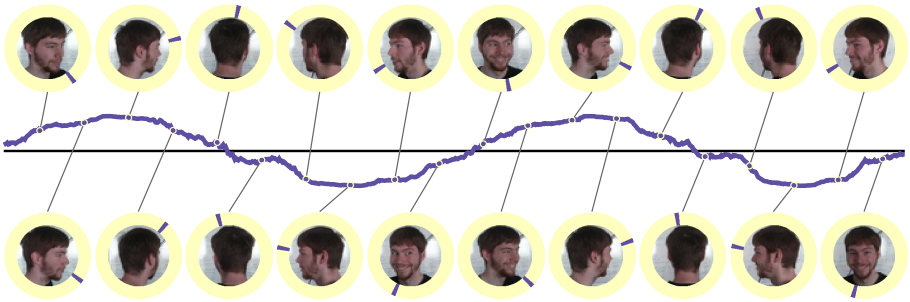

Fig. 2: Qualitative results. The purple line shows the sine of the predicted orientation angle across two full turns. For each head, the purple mark shows the orientation as seen from above. Results are equally spaced and not cherry-picked, more densely sampled results can be seen in the supplementary material.

architecture. In our scenario, the image sequence of a single person can easily be annotated based on temporal constraints. We split up the full annotation task into two annotation runs of four classes. First we annotate `Front`, `Left`, `Back` and `Right`, followed by the same annotation with boundaries shifted by 45°. We select temporal regions in the video through their start and end frames and mark any such region as one class. The resulting pair of annotations can then easily be merged into an eight-class annotation. The whole process, including the cropping of the head regions and the annotation itself, was done by a single person and took no longer than two and a half hours.

We train a biternion net on the resulting dataset except for one person, which we set aside for qualitative evaluation. We only train this network for five epochs since the number of people in this dataset is orders of magnitude smaller than in all previous datasets. We then let the biternion net predict the head pose of the left-out person *for each frame individually*. The result, which can be seen in Figure 2, clearly shows that the network estimates a fairly smooth sinusoidal pose across the two turns the person made, despite having been trained on only eight discrete pose annotations.

6 Conclusion

In this paper, we have introduced biternion nets, a CNN based approach. We have validated our architecture on several public datasets and have shown that our biternion layer is essential for continuous periodic orientation regression. Our obtained results redefine the state of the art on all used datasets. We furthermore show that, using biternion nets, it becomes possible to collect data with discrete and coarse orientation labels, which can be annotated quickly and cheaply, in order to train a continuous and precise head pose regressor. This suggests that fine-grained regression annotations are no longer necessary for continuous orientation estimation. The work in this paper was funded by the EU projects STRANDS (ICT-2011-600623) and SPENCER (ICT-2011-600877). Code is available at `http://github.com/lucasb-eyer/BiternionNet`.

References

1. Aghajanian, J., Prince, S.: Face Pose Estimation in Uncontrolled Environments. In: BMVC (2009)
2. Ba, S.O., Odobez, J.M.: Evaluation of Multiple Cue Head Pose Estimation Algorithms in Natural Environements. In: ICME (2005)
3. Bastien, F., Lamblin, P., Pascanu, R., Bergstra, J., Goodfellow, I.J., Bergeron, A., Bouchard, N., Bengio, Y.: Theano: new features and speed improvements. Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop (2012)
4. Baxter, R.H., Leach, M.J., Mukherjee, S.S., Robertson, N.M.: An Adaptive Motion Model for Person Tracking with Instantaneous Head-Pose Features. IEEE Signal Processing Letters 22(5), 578–582 (2015)
5. Benfold, B., Reid, I.: Unsupervised Learning of a Scene-Specific Coarse Gaze Estimator. In: ICCV (2011)
6. Black, Jr., J.A., Gargesha, M., Kahol, K., Kuchi, P., Panchanathan, S.: A Framework for Performance Evaluation of Face Recognition Algorithms . In: Proc. SPIE. vol. 4862, pp. 163–174 (2002)
7. Chamveha, I., Sugano, Y., Sugimura, D., Siriteerakul, T., Okabe, T., Sato, Y., Sugimoto, A.: Head direction estimation from low resolution images with scene adaptation. CVIU 117(10), 1502–1511 (2013)
8. Chen, C., Odobez, J.M.: We are not Contortionists: Coupled Adaptive Learning for Head and Body Orientation Estimation in Surveillance Video. In: CVPR (2012)
9. Dantone, M., Gall, J., Fanelli, G., Van Gool, L.: Real-time Facial Feature Detection using Conditional Regression Forests. In: CVPR (2012)
10. Demirkus, M., Precup, D., Clark, J.J., Arbel, T.: Probabilistic Temporal Head Pose Estimation Using a Hierarchical Graphical Model. In: ECCV (2014)
11. Dollár, P., Welinder, P., Perona, P.: Cascaded Pose Regression. In: CVPR (2010)
12. Fanelli, G., Dantone, M., Gall, J., Fossati, A., Van Gool, L.: Random Forests for Real Time 3D Face Analysis. IJCV 101(3), 437–458 (2013)
13. Goodfellow, I., Warde-Farley, D., Mirza, M., Courville, A., Bengio, Y.: Maxout Networks. In: ICML (2013)
14. Gourier, N., Hall, D., Crowley, J.L.: Estimating Face orientation from Robust Detection of Salient Facial Structures. In: ICPR'04 FG Net Workshop (2004)
15. Gross, R., Matthews, I., Cohn, J., Kanade, T., Baker, S.: Multi-Pie. Image and Vision Computing 28(5), 807–813 (2010)
16. Hara, K., Chellappa, R.: Growing Regression Forests by Classification: Applications to Object Pose Estimation. In: ECCV (2014)
17. He, K., Zhang, X., Ren, S., Sun, J.: Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. arXiv preprint arXiv:1502.01852 abs/1502.01852 (2015)
18. He, K., Sigal, L., Sclaroff, S.: Parameterizing Object Detectors in the Continuous Pose Space. In: ECCV (2014)
19. Huang, D., Storer, M., De la Torre, F., Bischof, H.: Supervised Local Subspace Learning for Continuous Head Pose Estimation. In: CVPR (2011)
20. Ioffe, S., Szegedy, C.: Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. arXiv preprint arXiv:1502.03167 (2015)
21. Lallemand, J., Ronge, A., Szczot, M., Ilic, S.: Pedestrian Orientation Estimation. In: GCPR (2014)
22. Mardia, K.V., Jupp, P.E.: Directional Statistics, vol. 494. John Wiley & Sons (2009)

23. Montavon, G., Orr, G.B., Müller, K. (eds.): Neural Networks: Tricks of the Trade - Second Edition. Springer (2012)
24. Murphy-Chutorian, E., Doshi, A., Trivedi, M.M.: Head Pose Estimation for Driver Assistance Systems: A Robust Algorithm and Experimental Evaluation. In: ITSC (2007)
25. Murphy-Chutorian, E., Trivedi, M.M.: Head Pose Estimation in Computer Vision: A Survey. PAMI 31(4), 607–626 (2009)
26. Osadchy, M., Cun, Y.L., Miller, M.L.: Synergistic Face Detection and Pose Estimation with Energy-Based Models. JMLR 8, 1197–1215 (2007)
27. Pérez, F., Granger, B.E.: IPython: a system for interactive scientific computing. Computing in Science and Engineering 9(3), 21–29 (May 2007), http://ipython.org
28. Qi, R.: Learning 3D Object Orientations From Synthetic Images (2015)
29. Saxe, A.M., McClelland, J.L., Ganguli, S.: Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. In: ICLR (2014)
30. Simonyan, K., Zisserman, A.: Very Deep Convolutional Networks for Large-Scale Image Recognition. In: ICLR (2015)
31. Singhal, A.: Modern Information Retrieval: A Brief Overview. IEEE Data Eng. Bull. 24(4), 35–43 (2001)
32. Siriteerakul, T.: Advance in Head Pose Estimation from Low Resolution Images: A Review. IJCSI 9(2) (2012)
33. Torki, M., Elgammal, A.: Regression from Local Features for Viewpoint and Pose Estimation. In: ICCV (2011)
34. Tosato, D., Spera, M., Cristani, M., Murino, V.: Characterizing Humans on Riemannian Manifolds. PAMI 35(8), 1972–1984 (2013)
35. Wu, Y., Toyama, K.: Wide-Range, Person- and Illumination-Insensitive Head Orientation Estimation. In: Int. Conf. on Automatic Face and Gesture Recognition (2000)
36. Zeiler, M.D., Rob, F.: Stochastic Pooling for Regularization of Deep Convolutional Neural Networks. In: ICLR (2013)

Biternion Nets: Continuous Head Pose Regression from Discrete Training Labels

Supplementary Material

Lucas Beyer, Alexander Hermans, and Bastian Leibe

Visual Computing Institute, RWTH Aachen University

Abstract. In this supplementary material, we show additional details about both the training procedure and the architecture of our CNN. Furthemore, we show more detailed quantitative and qualitative results.

1 CNN Training

We use Theano [1] as a framework for our implementation. The details of the architecture used throughout the paper can be seen in Table 1. The weights of all convolutional and all but the output's fully-connected layers are initialized using "Xavier"-initialization [2] and all biases are initialized to zero. The weights of both the softmax and the angle output layers are initialized to zero, while those of the biternion output layer are initialized to random standard normal values multiplied by 0.01 to break symmetry. Our implementation of batch-normalization [3] uses a second forward pass through the data after each epoch for collecting exact mini-batch statistics. Its γ weights are initialized to one and its β weights to zero.

For the optimization, we implemented AdaDelta [5] for its stability so we could use the same hyperparameters $\rho = 0.95$ and $\epsilon = 1 \times 10^{-7}$ for all experiments. We only perfromed data augmentation in two ways. First, we horizontally flip all training images and adjust their label accordingly. Second, we use random 46×46 crops during training and average the output of five such crops (center and four corners) during prediction. The size of all our minibatches is 100 and we divide the accumulated parameter gradients by that same number. We ran all experiments five times with random seeds taken from `/dev/urandom`, resetting the optimizer's state between runs, and report all our results as the average of those five runs. Finally, as reported in the main paper, we use early-stopping at epoch 50 for all but the last experiment, for which we stop after the fifth epoch due to the comparatively very small size of the dataset.

Table 1: The CNN architecture used throughout the paper, with two special cases for differently shaped datasets. All Conv layers have a stride of 1 and all MaxPool layers have a stride equal to their size, *i.e.* are non-overlapping.

General		IDIAP		HOC	
Type	Size	Type	Size	Type	Size
Crop size	46×46×3	Input	68×68×3	Input	54×123×3
Conv	24×(3×3×3)	Conv	24×(3×3×3)	Conv	24×(3×3×3)
Batch Norm	24	Batch Norm	24	Batch Norm	24
ReLU		ReLU		ReLU	
Conv	24×(3×3×24)	Conv	24×(3×3×24)	Conv	24×(3×3×24)
Batch Norm	24	Batch Norm	24	Batch Norm	24
ReLU		ReLU		ReLU	
				Conv	24×(3×3×24)
				Batch Norm	24
				ReLU	
MaxPool	2×2	MaxPool	2×2	MaxPool	2×3
Conv	48×(3×3×24)	Conv	48×(3×3×24)	Conv	48×(3×3×24)
Batch Norm	48	Batch Norm	48	Batch Norm	48
ReLU		ReLU		ReLU	
Conv	48×(3×3×48)	Conv	48×(3×3×48)	Conv	48×(3×3×48)
Batch Norm	48	Batch Norm	48	Batch Norm	48
ReLU		ReLU		ReLU	
				Conv	48×(3×3×48)
				Batch Norm	48
				ReLU	
MaxPool	2×2	MaxPool	2×2	MaxPool	3×3
Conv	64×(3×3×48)	Conv	64×(3×3×48)	Conv	64×(3×3×48)
Batch Norm	64	Batch Norm	64	Batch Norm	64
ReLU		ReLU		ReLU	
Conv	64×(3×3×64)	Conv	64×(3×3×64)	Conv	64×(3×3×64)
Batch Norm	64	Batch Norm	64	Batch Norm	64
ReLU		ReLU		ReLU	
		MaxPool	2×2		
Dropout $p = 0.2$		Dropout $p = 0.2$		Dropout $p = 0.2$	
FullyConn	512	FullyConn	512	FullyConn	512
ReLU		ReLU		ReLU	
Dropout $p = 0.5$		Dropout $p = 0.5$		Dropout $p = 0.5$	
Softmax/Angle/Biternion		Softmax/Angle/Biternion		Softmax/Angle/Biternion	

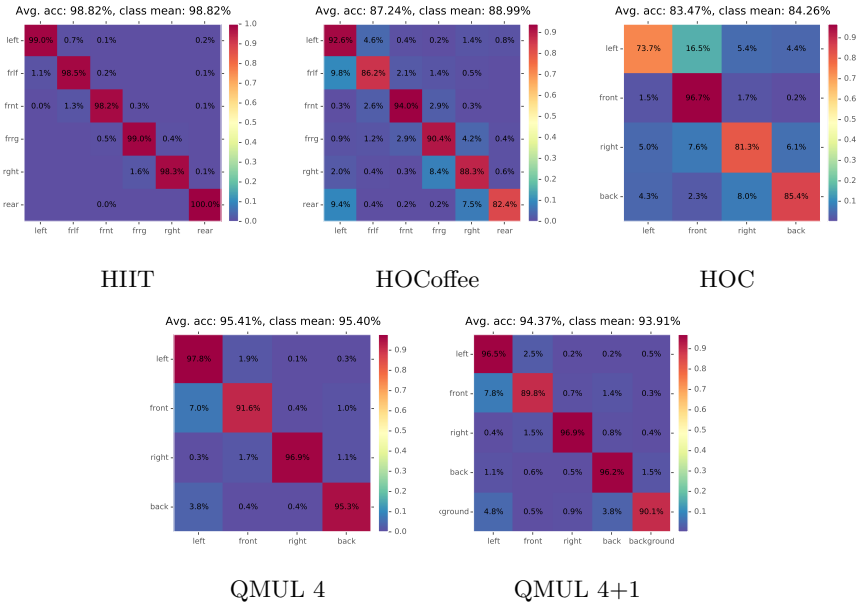

Fig. 1: Confusion matrices of the last run of the classification task.

2 Confusion Matrices of the Classification Tasks

The confusion matrices obtained for the classification task of [4] described in Section 3.2 of the main paper are shown in Fig. 1. The average accuracies may slightly differ from those reported in Table 1 of the main paper because the confusion matrices are those of the last run only, while the numbers in Table 1 of the main paper are the average of five runs.

3 Qualitative Results

Figure 2 shows further qualitative results based on our test person. The blue line in the visualization represents the head pose as seen when looking down onto the person from above it, *i.e.* the line being at the top signifies that the person is looking away from the camera. Due to both anonymity and privacy reasons, further persons will only be published in the final version. None of the pictures are cherry-picked; we show every ninth frame of the full recording. A small mistake of the biternion net is visible in the first three pictures of the second row. Note how the biternion net makes very continuous, stable predictions even though *it works on a frame-by-frame* basis, with no notion of time or order.

References

1. Bastien, F., Lamblin, P., Pascanu, R., Bergstra, J., Goodfellow, I.J., Bergeron, A., Bouchard, N., Bengio, Y.: Theano: new features and speed improvements. Deep

Fig. 2: Qualitative results on our own dataset. The small purple mark indicates the orientation of the person as seen from above, *i.e.* the top corresponds to *back* and the bottom to *front*.

Learning and Unsupervised Feature Learning NIPS 2012 Workshop (2012)
2. Bengio, Y., Glorot, X.: Understanding the difficulty of training deep feedforward neural networks. In: Proceedings of AISTATS. vol. 9, pp. 249–256 (2010)
3. Ioffe, S., Szegedy, C.: Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. arXiv preprint arXiv:1502.03167 (2015)
4. Tosato, D., Spera, M., Cristani, M., Murino, V.: Characterizing Humans on Riemannian Manifolds. PAMI 35(8), 1972–1984 (2013)
5. Zeiler, M.D.: Adadelta: an adaptive learning rate method. arXiv preprint arXiv:1212.5701 (2012)