

# Biternion Nets: Continuous Head Pose Regression from Discrete Training Labels

Anonymous GCPR 2015 submission

Paper ID 52

**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  $360^\circ$  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, 33, 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  $360^\circ$  spectrum. Thus, head pose estimators need to be able to cope with the periodicity of angular data, *i.e.* the fact that  $361^\circ$  corresponds to  $1^\circ$  and, for a head pose of  $0^\circ$ , a prediction of  $359^\circ$  is no worse than a prediction of  $1^\circ$ . Face pose and gaze estimators can conveniently sidestep this difficulty by constraining the prediction range to non-periodic intervals such as  $[-90^\circ, 90^\circ]$ . 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, 31] has been proposed which solve only one of the two aforementioned problems: either they cannot cope with periodicity [26, 24, 33], or they need fine-grained regression data [34, 32, 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 [31, 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 [33], to learned functional mappings [26, 32, 4]. These approaches often result in a manifold embedding of the images [26, 33]. 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 [32, 27, 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, 33]. 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, 33, 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 [34, 5, 33]. Wu and Toyama [34] 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.* [33] use covariance features to classify head poses into a small set of orientation bins.

84 CNNs have also been used for orientation estimation before. Qi [27] fine- 84  
 85 tunes a large pre-trained CNN to classify the orientation of chairs using a large 85  
 86 amount of rendered chairs with precise labels. However, using CNNs pre-trained 86  
 87 on ImageNet for low-resolution head pose estimation makes no sense due to 87  
 88 the significantly different filter resolution, type of data, and learning task. Most 88  
 89 similar to our approach is the one by Osadchy *et al.* [26], which also uses a CNN 89  
 90 for continuous head pose estimation. They learn a face manifold on (non-public) 90  
 91 data with regression labels, which enables them to jointly detect and estimate 91  
 92 the pose of faces. In contrast to us, they focus on using face pose data to improve 92  
 93 face detection and do not address the periodicity problem. 93

94 Some approaches also aim at solving the periodicity problem [32, 16, 18]. How- 94  
 95 ever, their approaches are typically based on nearest-neighbor matching or kernel 95  
 96 density estimation, meaning that they require dense orientation labels for train- 96  
 97 ing. All three of the above approaches use fine grained face datasets [14, 1] and 97  
 98 it is unclear how well they could perform for head pose estimation. 98

99 To the best of our knowledge, only Huang *et al.* [19] aim at learning con- 99  
 100 tinuous regressors from a discrete face pose dataset. They learn a mixture of 100  
 101 local tangent subspaces that are robust to regression regions with bad coverage 101  
 102 in the training set. Their representation is based on HOG features and they use 102  
 103 high resolution images. It is questionable whether their approach can deal with 103  
 104 head poses, as HOG features are not very expressive for the back of a head. 104  
 105 Furthermore, they do not evaluate how continuous their regression really is. 105

106 In conclusion, based on existing approaches, the task of continuous periodic 106  
 107 head pose estimation is still unsolved. Here our approach comes into play. 107

### 108 3 CNNs for Head Pose Estimation 108

109 Throughout this paper, we work in the framework of deep convolutional networks 109  
 110 and stochastic, gradient-based optimization. In this section, we present the spe- 110  
 111 cific network architecture we use for all experiments, changing only the output 111  
 112 layer and cost function to match the task at hand. We then apply it to multiple 112  
 113 publicly available datasets, consistently outperforming current state-of-the-art 113  
 114 methods on those datasets. 114

#### 115 3.1 The Network Architecture 115

116 We use a moderately deep, batch-normalized [20], VGG-style network architec- 116  
 117 ture [29] consisting of six convolutional layers with 24, 24, 48, 48, 64 and 64 117  
 118 feature channels, respectively, followed by a single hidden layer of 512 units, and 118  
 119 train it for a fixed duration of 50 epochs in all our experiments. For all details 119  
 120 about the network and the training procedure, please refer to the supplement- 120  
 121 ary material. We implemented the network in Theano [3]. All numbers reported 121  
 122 within this paper are averages over five runs. While we will show that this archi- 122  
 123 tecture already performs very well, it is likely possible to reduce the error even 123  
 124 further by using deeper networks with more careful regularization and a bag 124  
 125 of other well-known tricks [23, 13, 35, 28, 17]. We do not further go down that 125

Table 1: Class-average accuracies on the four classification datasets from [33]. 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 <i>et al.</i> [33]	96.5%	81.0%	78.69%	94.25%	91.18%
Lallemant <i>et al.</i> [21]	-	-	79.9%	-	-
Our CNN	<b>98.70%</b>	<b>86.99%</b>	<b>83.97%</b>	<b>95.58%</b>	<b>94.30%</b>

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.* [33] 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  $[-60^\circ, 60^\circ]$ . 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.<sup>1</sup>

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  $\pm 4^\circ$  of the four canonical orientations. A major downside of this dataset is that most images have been upsampled to 50-by-50 pixels from their

<sup>1</sup> 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 [33]. 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.

# Samples	IDIAP Head Pose			CAVIAR-c	CAVIAR-o
	42 304/23 991			10 660/10 665	10 802/10 889
	pan	tilt	roll	pan	pan
Pose range	[-101,101]	[-73,23]	[-46,65]	[0, 360]	[0, 360]
Tosato <i>et al.</i> [33]	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	<b>5.9°±7.2°</b>	<b>2.8°±2.6°</b>	<b>3.5°±3.9°</b>	<b>19.2°±24.2°</b>	<b>25.2°±26.4°</b>

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:

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

Equation (1) defines its probability density function, where  $\varphi$  is an angle,  $\mu$  is the mean angle of the distribution,  $\kappa$  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  $\kappa$ . 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:

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

In the cost formulation, we call  $t$  the target value and  $\kappa$  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. 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 [30]:

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

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:

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

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

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

215 Notice how (1) the normalization in the biternion layer makes sure the output 215  
 216 values are learned *jointly* and (2) the normalization terms in  $C_{\text{cos}}$  can subse- 216  
 217 quently be omitted. 217

218 Finally, the ensembling of multiple biternion predictions, as needed by some 218  
 219 augmentation techniques, can simply be performed by averaging the vectors, 219  
 220 since the average of unit vectors is again a unit vector, a fact also used by 220  
 221 Hara *et al.* [16]. 221

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

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

222 Using the fact that  $\|\mathbf{a}\| = 1$ , the last entry of the quaternion—which en- 222  
 223 codes the cosine of half the angle represented by the quaternion—simplifies 223  
 224 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 224  
 225 simplified, resulting in a quaternion representing a rotation of the angle from  $\varphi$  to 225  
 226  $\theta$  around the same axis  $\mathbf{a}$ . This shows that biternions can be seen as quaternions 226  
 227 around a fixed reference axis  $\mathbf{a}$  and the cosine cost corresponds to the amplitude 227  
 228 of the direct rotation between the predicted and the target biternions. 228

229 **Relationship to the von Mises Cost.** By comparing  $C_{\text{VM}}$  and  $C_{\text{cos}}$ , it is 229  
 230 visible that they do *not* compute the same expression, *i.e.*, the biternion-layer 230  
 231 coupled with the cosine cost does *not* optimize the von Mises cost. The von 231  
 232 Mises cost for the biternion layer can be written as: 232

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

233 Notice the similarity to Equation 3; the main difference is the presence of 233  
 234  $e$ , which “pushes down” the error around the target value, in effect penalizing 234  
 235 small mistakes less strongly. 235

### 236 4.3 Experimental Results 236

237 In order to investigate the relative usefulness of the von Mises cost and the 237  
 238 biternion representation for periodic regression, we now turn to the TownCentre 238  
 239 dataset [5]. This dataset contains heads of tracked pedestrians in a shopping 239  
 240 district, annotated with head pose regression labels. The prior distribution of 240  
 241 the pose angle is shown in the middle of Fig. 1. For all experiments, we train 241  
 242 on 7920 heads of 3960 persons and evaluate on 774 heads of 387 random but 242  
 243 *different* persons. The results can be seen in Table 3. 243

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.

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<sup>2</sup>, while the second of their numbers has not seen any of the persons since it has been trained on a different dataset.

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

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

## 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  $2^\circ$ .

<sup>2</sup> 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°	<b>32.1°±28.1°</b>	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°	<b>26.9°±27.4°</b>
6	26.1°±28.4°	24.1°±27.6°	38.3°±38.5°	31.8°±33.1°	<b>22.1°±25.5°</b>	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°	<b>21.3°±25.2°</b>
10	23.8°±27.5°	21.9°±26.9°	37.6°±38.3°	28.8°±30.8°	<b>21.4°±24.6°</b>	21.8°±25.5°
12	23.6°±29.4°	22.2°±28.8°	39.0°±38.2°	29.7°±31.5°	<b>21.4°±25.3°</b>	21.8°±25.3°
∞	-	-	38.9°±40.7°	29.4°±31.1°	21.6°±25.2°	<b>20.8°±24.7°</b>

286 CNNs compute a continuous function of their input and, during training, 286  
 287 each sample *pulls* the parameters of the CNN slightly into a direction leading to 287  
 288 a better prediction of its pose. This intuition suggests that it should be possible 288  
 289 for CNNs to learn a continuous mapping from images to pose angles even when 289  
 290 only given very rough pose labels. This is shown in the last four columns of 290  
 291 Table 4. As can be seen, this idea hardly works at all in the naive regression 291  
 292 case and is only somewhat improved by the von Mises cost. Biternion nets, on 292  
 293 the other hand, have no difficulty being trained this way and in fact outperform 293  
 294 the class-based approaches with any number of realistically annotatable classes, 294  
 295 whether the cosine or the von Mises cost is used 295

296 Unfortunately, looking only at numbers representing an average error over a 296  
 297 large amount of images does not reflect the real advantage of biternion nets over 297  
 298 the classifier approach. For this reason, we plotted heatmaps of the predictions 298  
 299 made by a CNN classifier with quadratic interpolation and the predictions made 299  
 300 by a biternion net in Figure 1. These heatmaps clearly show that, while the class- 300  
 301 interpolation approach and biternion nets give similar scores, the predictions of 301  
 302 the biternion nets are vastly superior because they are more continuous and 302  
 303 similar to the distribution of the ground-truth angles. 303

304 **5.1 Practicality** 304

305 To show the potential of our approach, we recorded a small dataset using a 305  
 306 common smartphone camera and annotated it with eight class labels. For this, 306  
 307 we recorded 24 people in our lab and asked them to rotate on the spot. We 307  
 308 then manually cropped a square region in the resulting videos containing their 308  
 309 head and rescaled it to 50 × 50 pixels to make it compatible to our network 309



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

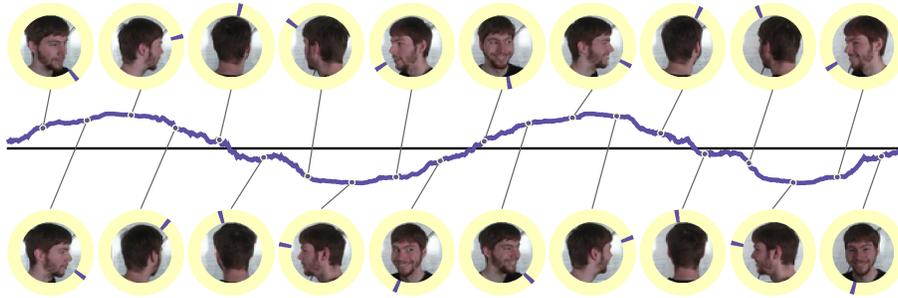


Fig. 2: Qualitative results. The purple line shows the sine of the predicted orientation angle across two full turns. For each head, the purple mark shows the orientation as seen from above. Results are equally spaced and not cherry-picked, more densely sampled results can be seen in the supplementary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

310 architecture. In our scenario, the image sequence of a single person can easily be 310  
 311 annotated based on temporal constraints. We split up the full annotation task 311  
 312 into two annotation runs of four classes. First we annotate `Front`, `Left`, `Back` 312  
 313 and `Right`, followed by the same annotation with boundaries shifted by  $45^\circ$ . 313  
 314 We select temporal regions in the video through their start and end frames and 314  
 315 mark any such region as one class. The resulting pair of annotations can then 315  
 316 easily be merged into an eight-class annotation. The whole process, including 316  
 317 the cropping of the head regions and the annotation itself, was done by a single 317  
 318 person and took no longer than two and a half hours. 318

319 We train a biternion net on the resulting dataset except for one person, which 319  
 320 we set aside for qualitative evaluation. We only train this network for five epochs 320  
 321 since the number of people in this dataset is orders of magnitude smaller than 321  
 322 in all previous datasets. We then let the biternion net predict the head pose of 322  
 323 the left-out person *for each frame individually*. The result, which can be seen 323  
 324 in Figure 2, clearly shows that the network estimates a fairly smooth sinusoidal 324  
 325 pose across the two turns the person made, despite having been trained on only 325  
 326 eight discrete pose annotations. 326

## 327 6 Conclusion 327

328 In this paper, we have introduced biternion nets, a CNN based approach. We 328  
 329 have validated our architecture on several public datasets and have shown that 329  
 330 our biternion layer is essential for continuous periodic orientation regression. Our 330  
 331 obtained results redefine the state of the art on all used datasets. We furthermore 331  
 332 show that, using biternion nets, it becomes possible to collect data with discrete 332  
 333 and coarse orientation labels, which can be annotated quickly and cheaply, in 333  
 334 order to train a continuous and precise head pose regressor. This suggests that 334  
 335 fine-grained regression annotations are no longer necessary for continuous orien- 335  
 336 tation estimation. 336

337 **References**

- 338 1. Aghajanian, J., Prince, S.: Face Pose Estimation in Uncontrolled Environments. 338  
 339 In: BMVC (2009) 339
- 340 2. Ba, S.O., Odobez, J.M.: Evaluation of Multiple Cue Head Pose Estimation Algo- 340  
 341 rithms in Natural Environments. In: ICME (2005) 341
- 342 3. Bastien, F., Lamblin, P., Pascanu, R., Bergstra, J., Goodfellow, I.J., Bergeron, A., 342  
 343 Bouchard, N., Bengio, Y.: Theano: new features and speed improvements. Deep 343  
 344 Learning and Unsupervised Feature Learning NIPS 2012 Workshop (2012) 344
- 345 4. Baxter, R.H., Leach, M.J., Mukherjee, S.S., Robertson, N.M.: An Adaptive Motion 345  
 346 Model for Person Tracking with Instantaneous Head-Pose Features. IEEE Signal 346  
 347 Processing Letters 22(5), 578–582 (2015) 347
- 348 5. Benfold, B., Reid, I.: Unsupervised Learning of a Scene-Specific Coarse Gaze Es- 348  
 349 timator. In: ICCV (2011) 349
- 350 6. Black, Jr., J.A., Gargsha, M., Kahol, K., Kuchi, P., Panchanathan, S.: A Frame- 350  
 351 work for Performance Evaluation of Face Recognition Algorithms . In: Proc. SPIE. 351  
 352 vol. 4862, pp. 163–174 (2002) 352
- 353 7. Chamveha, I., Sugano, Y., Sugimura, D., Siriteerakul, T., Okabe, T., Sato, Y., 353  
 354 Sugimoto, A.: Head direction estimation from low resolution images with scene 354  
 355 adaptation. CVIU 117(10), 1502–1511 (2013) 355
- 356 8. Chen, C., Odobez, J.M.: We are not Contortionists: Coupled Adaptive Learning 356  
 357 for Head and Body Orientation Estimation in Surveillance Video. In: CVPR (2012) 357
- 358 9. Dantone, M., Gall, J., Fanelli, G., Van Gool, L.: Real-time Facial Feature Detection 358  
 359 using Conditional Regression Forests. In: CVPR (2012) 359
- 360 10. Demirkus, M., Precup, D., Clark, J.J., Arbel, T.: Probabilistic Temporal Head 360  
 361 Pose Estimation Using a Hierarchical Graphical Model. In: ECCV (2014) 361
- 362 11. Dollár, P., Welinder, P., Perona, P.: Cascaded Pose Regression. In: CVPR (2010) 362
- 363 12. Fanelli, G., Dantone, M., Gall, J., Fossati, A., Van Gool, L.: Random Forests for 363  
 364 Real Time 3D Face Analysis. IJCV 101(3), 437–458 (2013) 364
- 365 13. Goodfellow, I., Warde-Farley, D., Mirza, M., Courville, A., Bengio, Y.: Maxout 365  
 366 Networks. In: ICML (2013) 366
- 367 14. Gourier, N., Hall, D., Crowley, J.L.: Estimating Face orientation from Robust 367  
 368 Detection of Salient Facial Structures. In: ICPR'04 FG Net Workshop (2004) 368
- 369 15. Gross, R., Matthews, I., Cohn, J., Kanade, T., Baker, S.: Multi-Pie. Image and 369  
 370 Vision Computing 28(5), 807–813 (2010) 370
- 371 16. Hara, K., Chellappa, R.: Growing Regression Forests by Classification: Applica- 371  
 372 tions to Object Pose Estimation. In: ECCV (2014) 372
- 373 17. He, K., Zhang, X., Ren, S., Sun, J.: Delving Deep into Rectifiers: Sur- 373  
 374 passing Human-Level Performance on ImageNet Classification. arXiv preprint 374  
 375 arXiv:1502.01852 abs/1502.01852 (2015) 375
- 376 18. He, K., Sigal, L., Sclaroff, S.: Parameterizing Object Detectors in the Continuous 376  
 377 Pose Space. In: ECCV (2014) 377
- 378 19. Huang, D., Storer, M., De la Torre, F., Bischof, H.: Supervised Local Subspace 378  
 379 Learning for Continuous Head Pose Estimation. In: CVPR (2011) 379
- 380 20. Ioffe, S., Szegedy, C.: Batch Normalization: Accelerating Deep Network Training 380  
 381 by Reducing Internal Covariate Shift. arXiv preprint arXiv:1502.03167 (2015) 381
- 382 21. Lallemand, J., Ronge, A., Szczot, M., Ilic, S.: Pedestrian Orientation Estimation. 382  
 383 In: GCPR (2014) 383
- 384 22. Mardia, K.V., Jupp, P.E.: Directional Statistics, vol. 494. John Wiley & Sons 384  
 385 (2009) 385

- 386 23. Montavon, G., Orr, G.B., Müller, K. (eds.): Neural Networks: Tricks of the Trade 386  
387 - Second Edition. Springer (2012) 387
- 388 24. Murphy-Chutorian, E., Doshi, A., Trivedi, M.M.: Head Pose Estimation for Driver 388  
389 Assistance Systems: A Robust Algorithm and Experimental Evaluation. In: ITSC 389  
390 (2007) 390
- 391 25. Murphy-Chutorian, E., Trivedi, M.M.: Head Pose Estimation in Computer Vision: 391  
392 A Survey. PAMI 31(4), 607–626 (2009) 392
- 393 26. Osadchy, M., Cun, Y.L., Miller, M.L.: Synergistic Face Detection and Pose Esti- 393  
394 mation with Energy-Based Models. JMLR 8, 1197–1215 (2007) 394
- 395 27. Qi, R.: Learning 3D Object Orientations From Synthetic Images (2015) 395
- 396 28. Saxe, A.M., McClelland, J.L., Ganguli, S.: Exact solutions to the nonlinear dy- 396  
397 namics of learning in deep linear neural networks. In: ICLR (2014) 397
- 398 29. Simonyan, K., Zisserman, A.: Very Deep Convolutional Networks for Large-Scale 398  
399 Image Recognition. In: ICLR (2015) 399
- 400 30. Singhal, A.: Modern Information Retrieval: A Brief Overview. IEEE Data Eng. 400  
401 Bull. 24(4), 35–43 (2001) 401
- 402 31. Siriteerakul, T.: Advance in Head Pose Estimation from Low Resolution Images: 402  
403 A Review. IJCSI 9(2) (2012) 403
- 404 32. Torki, M., Elgammal, A.: Regression from Local Features for Viewpoint and Pose 404  
405 Estimation. In: ICCV (2011) 405
- 406 33. Tosato, D., Spera, M., Cristani, M., Murino, V.: Characterizing Humans on Rie- 406  
407 mannian Manifolds. PAMI 35(8), 1972–1984 (2013) 407
- 408 34. Wu, Y., Toyama, K.: Wide-Range, Person- and Illumination-Insensitive Head Ori- 408  
409 entation Estimation. In: Int. Conf. on Automatic Face and Gesture Recognition 409  
410 (2000) 410
- 411 35. Zeiler, M.D., Rob, F.: Stochastic Pooling for Regularization of Deep Convolutional 411  
412 Neural Networks. In: ICLR (2013) 412