Anchored Regression Networks applied to Age Estimation and Super Resolution

Eirikur Agustsson
D-ITET, ETH Zurich
Switzerland
aeirikur@vision.ee.ethz.ch

Radu Timofte
D-ITET, ETH Zurich
Merantix GmbH
timofter@vision.ee.ethz.ch

Luc Van Gool
D-ITET, ETH Zurich
ESAT, KU Leuven
vangool@vision.ee.ethz.ch

Abstract

We propose the Anchored Regression Network (ARN), a nonlinear regression network which can be seamlessly integrated into various networks or can be used stand-alone when the features have already been fixed. Our ARN is a smoothed relaxation of a piecewise linear regressor through the combination of multiple linear regressors over soft assignments to anchor points. When the anchor points are fixed the optimal ARN regressors can be obtained with a closed form global solution, otherwise ARN admits end-to-end learning with standard gradient based methods. We demonstrate the power of the ARN by applying it to two very diverse and challenging tasks: age prediction from face images and image super-resolution. In both cases, ARNs yield strong results.

1. Introduction

Image regression is relevant for many applications, such as biometric prediction [9, 30], image super-resolution [43, 40], bounding box regression [10, 12], facial landmark localization [19], depth estimation [24, 34], and more. The common denominator for these supervised learning tasks is that the input is an image and the output is a single- or multidimensional vector.

In contrast to image classification, where multinomial logistic regression is the dominant strategy for deep convolutional neural networks (CNNs) [21, 35, 37, 28], in general still a wide array of regression methods are deployed, without a clear winner. Some cast the task into classification [30], employ linear regression at the last layer [12, 35, 25], or apply an array of methods on top of the features extracted from a classification network. Each of these strategies has serious drawbacks: a standard linear regression layer often underfits the data [30]; casting to classification requires manually defining the classes; and working with extracted features (*e.g.* using SVR [1]) impedes end-to-end learning.

Motivated by the above issues, we propose *Anchored Regression Networks* (ARN), a novel regression architecture suitable for various regression tasks. The network, depicted in Figure 1, is easy to implement, can be seamlessly integrated into various deep or shallow networks or directly used with fixed feature representations such as handcrafted features or representations extracted from pretrained CNN architectures.

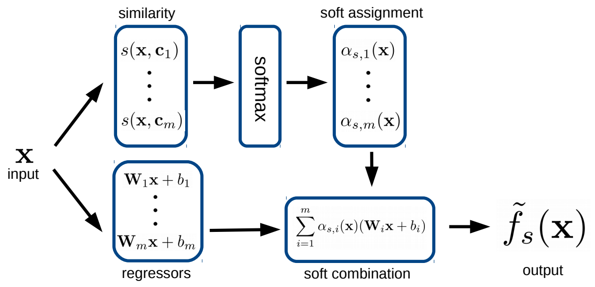

similarity
$s(\mathbf{x}, \mathbf{c}_1)$
$s(\mathbf{x}, \mathbf{c}_m)$
softmax
soft assignment
$\alpha_{s,1}(\mathbf{x})$
$\alpha_{s,m}(\mathbf{x})$
$\mathbf{x}$
input
$\mathbf{W}_1\mathbf{x} + b_1$
$\mathbf{W}_m\mathbf{x} + b_m$
regressors
$\sum_{i=1}^{m} \alpha_{s,i}(\mathbf{x})(\mathbf{W}_i\mathbf{x} + b_i)$
soft combination
$\tilde{f}_s(\mathbf{x})$
output

Figure 1. **Our proposed ARN architecture.** When an input point $\mathbf{x}$ is fed through the network, it is soft-assigned to the anchors $\mathbf{c}_1, \cdots, \mathbf{c}_m$ w.r.t. a similarity s. For each anchor $\mathbf{c}_i$, there is a corresponding regressor $(\mathbf{W}_i, b_i)$ which the input is also fed to, which are then linearly combined over the soft assignments, forming the output $\tilde{f}_s(\mathbf{x})$.

The design of our ARN is based on the assumption that the underlying features (such as raw data, handcrafted or deep features) can linearize the regression problem within local regions, such that a piecewise linear approximation of the mapping is feasible. By 'softly' partitioning the input space w.r.t. a set of anchor points, each having a corresponding linear regressor, our ARN can learn such an approximation. We show that an ARN can be learned globally w.r.t. fixed anchors via a closed form solution, and also end-to-end learned by standard gradient based methods as a layer in a shallow or deep architecture.

We validate the proposed ARN for single and multi-value regression on two very different and challenging tasks, namely age prediction from face images and single image super-resolution. Our experiments show that the corresponding ARNs improve the baseline or state-of-the-art methods and this with minimal modification.

	A+	ARN (ours)
assignments	hard (nearest neighbors)	**soft**
differentiability	non-differentiable	**differentiable**
learning features	decoupled – step-by-step hand-crafted	**end-to-end** learned or hand-crafted
anchors regressors	K-SVD fixed anchored neighborhood regression	**jointly learned with minibatch SGD**

Table 1. Differences between our ARN and A+ [39].

1.1. Related work

The assumption of the data having a piecewise linear structure has often been explored. Späth *et al.* [36] study the generic problem of learning linear regressors over partitions of the input space, and propose a greedy algorithm to learn the partitions and the regressors. DeSarbo *et al.* [5] study the same problem with fuzzy assignment in a probabilistic setting, and propose an expectation-maximization method to learn the model. More recently, Timofte *et al.* [39] built a piecewise linear model (the A+ method) for single-image super-resolution, by constraining the partitioning to a nearest neighbor assignment over a set of anchor points.

A related class of methods are based on local coordinate codings [46, 41, 45, 22]. These methods assume the data itself lies on a local manifold, which is represented with anchor points. In contrast to A+, here each anchor is associated with a function value instead of a regressor. A notable exception is the LL-SVM of Ladicky *et al.* [22] for classification, which associates a linear SVM with each anchor point, and combines them with a local coding.

In contrast to local coordinate codings [46], our ARN does not assume that the input vector lies on a manifold represented by the anchors (*i.e.* that it can be represented as a weighted sum of the anchors), but instead follows the direction of A+ [39] and use the anchors instead to partition the space.

This said, there are clear differences between ARN and A+, as summarized in Table 1. Since A+ is built on top of a discrete partition of the input space, it is not differentiable with respect to the anchors. Therefore, A+ does not admit end-to-end learning and is instead trained via a hand-defined procedure, such that it can only operate on pre-fixed feature representations. Furthermore, as the number of partitions increase, the data points per partition reduce such that robust regressors cannot be trained on each partition. Thus A+ resorts to training regressors on (possibly overlapping) fixed neighborhoods for each anchor.

Our ARN overcomes these limitations by formulating the regression model as a soft combination of linear regressors over anchor points. This enables us to get a fully differentiable model and greatly simplify the training procedure while achieving better performance than A+.

2. Proposed Method

Let $\{(\mathbf{x}_1, \mathbf{y}_1), ..., (\mathbf{x}_n, \mathbf{y}_n)\}$ be a set of n training examples, where the features $\mathbf{x}_i \in \mathbb{R}^d$ can be fixed or are the result of some layer in a neural network, and the label $\mathbf{y}_i \in \mathbb{R}^{d'}$ is real-valued and possibly multidimensional. We are interested to learn a mapping $f : \mathbb{R}^d \to \mathbb{R}^{d'}$ which approximates the relationship between $\mathbf{x}_i$ and $\mathbf{y}_i$.

To this end, we will propose a function family that is based on a network of linear regressors and anchor points. We start by motivating the design decisions of the network, and we will discuss applicable learning strategies in the following section.

Our basic assumption is that the relationship between $\mathbf{x}_i$ and $\mathbf{y}_i$ can be well approximated with partition-specific linear maps over a partition of the feature space. That is, there exists a partition $U_1, \cdots, U_m \subset \mathbb{R}^d$, $\bigcup_{i=1}^m U_i = \mathbb{R}^d$ and $U_i \cap U_j = \emptyset$ if $i \neq j$, and m linear regressors $(\mathbf{W}_1, b_1), \cdots, (\mathbf{W}_m, b_m)$ such that for $\mathbf{x}_i \in U_j$:

$$\mathbf{y}_i \approx \mathbf{W}_j \mathbf{x}_i + b_j, \tag{1}$$

with $\mathbf{W}_j \in \mathbb{R}^{d' \times d}$. The problem is to find a proper partition and corresponding regressors for the best approximation.

This problem was first studied by Späth *et al.* [36], who propose a greedy method to find such a partitioning.

Another approach, explored by Timofte *et al.* [39] in their A+ method, is to partition the space around a set of m anchor points $\mathbf{C} = \{\mathbf{c}_1, \cdots, \mathbf{c}_m\}$ using a similarity measure s, such that:

$$U_i = \left\{ \mathbf{x} \in \mathbb{R}^d \mid \forall j \neq i : s(\mathbf{x}, \mathbf{c}_i) > s(\mathbf{x}, \mathbf{c}_j) \right\}, \tag{2}$$

and to train specialized regressors for each partition U_i.

However, this formulation has two limitations.

(i) First, since the method relies on hard nearest neighbor assignments, it is not differentiable with respect to the anchors $\mathbf{C}$, preventing its use for end-to-end learning.

(ii) Second, to be able to robustly train a regressor for each partition U_i, sufficient training data is needed. Since $|U_i|$ becomes small as the number of partitions m increases, A+ resorts to instead train each regressor $\mathbf{W}_i$ on a fixed neighborhood of size N_b around $\mathbf{c}_i$. The additional hyperparameter N_b then needs to be chosen to balance the trade-off between overly generic regressors (when N_b is too large) and properly covering the training data, since in total $m \times N_b$ data points are used for training the regressors.

These limitations motivate our proposed method.

In order to obtain a differentiable formulation, we first note that in the case of hard assignments we can write the entire regression function as:

$$f_s(\mathbf{x}) = \mathbf{W}_{\gamma_s(\mathbf{x})} \mathbf{x} + b_{\gamma_s(\mathbf{x})}, \tag{3}$$

where $\gamma_s(\mathbf{x}) := \arg\max_j s(\mathbf{x}, \mathbf{c}_j)$ denotes the 'nearest' anchor to $\mathbf{x}$.

To achieve differentiability w.r.t. the anchors $\mathbf{C}$, instead of assigning $\mathbf{x}$ to a single anchor $\gamma_s(\mathbf{x})$, we perform a 'soft assignment' to multiple anchors, by using the soft-max operator:

$$\boldsymbol{\alpha}_s(\mathbf{x}) = \boldsymbol{\sigma}(\begin{pmatrix} s(\mathbf{x},\mathbf{c}_1) \\ \vdots \\ s(\mathbf{x},\mathbf{c}_m) \end{pmatrix}) = \frac{1}{\sum_{i=1}^{m} \mathrm{e}^{s(\mathbf{x},\mathbf{c}_i)}} \begin{pmatrix} \mathrm{e}^{s(\mathbf{x},\mathbf{c}_1)} \\ \vdots \\ \mathrm{e}^{s(\mathbf{x},\mathbf{c}_m)} \end{pmatrix}. \tag{4}$$

We then define our soft regression function as

$$\tilde{f}_s(\mathbf{x}) := \sum_{i=1}^{m} \alpha_{s,i}(\mathbf{x})(\mathbf{W}_i\mathbf{x} + b_i), \tag{5}$$

where $\alpha_{s,i}(\mathbf{x})$ denotes the i-th coordinate of $\boldsymbol{\alpha}_s(\mathbf{x})$.

Now, if s is differentiable, $\tilde{f}_s$ is fully differentiable w.r.t. all parameters: the input $\mathbf{x}$, the regressors $\{(\mathbf{W}_1, b_1), \cdots, (\mathbf{W}_m, b_m)\}$ and the anchor points $\{\mathbf{c}_1, \cdots, \mathbf{c}_m\}$.

Note that we can control the 'softness' of $\boldsymbol{\alpha}_s(\mathbf{x})$ by scaling the similarity s. That is, given a partitioning in (1) induced by a similarity s, using a scaled $s'(\mathbf{x},\mathbf{x}') = Ks(\mathbf{x},\mathbf{x}')$, $\boldsymbol{\alpha}_{s'}(\mathbf{x})$ will converge to a one-hot encoding of the nearest anchor w.r.t. s, as $K \to \infty$:

$$\lim_{K\to\infty} \alpha_{s',i}(\mathbf{x}) = \begin{cases} 1 & \text{if } i = \gamma_s(\mathbf{x}) \\ 0 & \text{otherwise} \end{cases}, \tag{6}$$

such that our soft formulation in (5) converges to the partition based formulation of (3):

$$\begin{aligned} \lim_{K\to\infty} \tilde{f}_{s'}(\mathbf{x}) &= \sum_{i=1}^{m} \lim_{K\to\infty} \alpha_{s',i}(\mathbf{x})(\mathbf{W}_i\mathbf{x} + b_i) & (7) \\ &= 1 \cdot (\mathbf{W}_{\gamma_s(\mathbf{x})}\mathbf{x} + b_{\gamma_s(\mathbf{x})}) & (8) \\ &= f_s(\mathbf{x}). & (9) \end{aligned}$$

We refer to $\tilde{f}_s$ as an *Anchored Regression Network* (**ARN**), since it combines multiple regressors through anchor points and, as shown in Section 3, both the anchors and the regressors can be learned in an end-to-end architecture.

3. Learning strategies

In order to learn the parameters of ARN over the training examples $\{(\mathbf{x}_1, \mathbf{y}_1), ..., (\mathbf{x}_n, \mathbf{y}_n)\}$, we need an objective, *i.e*. a loss.

We will consider first the common L_2-norm loss with L_2-norm regularization:

$$\mathcal{L} = \sum_{i=1}^{n} \|\tilde{f}_s(\mathbf{x}_i) - \mathbf{y}_i\|^2 + \lambda \sum_{j=1}^{m} \|\mathbf{W}_j\|^2. \tag{10}$$

We note that $\tilde{f}_s$ couples each data point $\mathbf{x}_i$ with all $\mathbf{W}_1, \cdots, \mathbf{W}_m$, such that we do not need fixed neighborhoods as used in A+ [39] to obtain robust regressors. Instead, we can simply optimize (10) with stochastic gradient descent (SGD).

However, it turns out that for an L_2 loss, if the anchors $\mathbf{C}$ and the similarity s are fixed, (*e.g*. obtained by clustering the training data), we have a closed form solution for the regressors of ARN. We will next explore this interesting case.

3.1. Global Solution

We will now show that we can obtain a closed form solution for the optimal regressors $\mathbf{W}_1, \cdots, \mathbf{W}_m$ that minimizes (10) when both the anchors $\mathbf{C}$ and the similarity s are fixed, such that each point $\mathbf{x}_i$ has a fixed soft assignment $\boldsymbol{\alpha}_s(\mathbf{x})$.

To simplify the notation, we write the training examples as $\mathbf{X} = [\mathbf{x}_1, \cdots, \mathbf{x}_n] \in \mathbb{R}^{d\times n}$ and $\mathbf{Y} = [\mathbf{y}_1, \cdots, \mathbf{y}_n] \in \mathbb{R}^{d'\times n}$. Furthermore, we assume the last dimension of $\mathbf{x}$ is 1, such that we can drop the bias terms b_i.

We then denote:

$$\tilde{\boldsymbol{\alpha}}_s(\mathbf{x}) := \begin{bmatrix} \alpha_{s,1}(\mathbf{x})\mathbf{I}_d \\ \vdots \\ \alpha_{s,m}(\mathbf{x})\mathbf{I}_d \end{bmatrix} \in \mathbb{R}^{(md)\times d}, \tag{11}$$

where $\mathbf{I}_d$ is the $d \times d$ identity matrix, and

$$\tilde{\mathbf{W}} := [\mathbf{W}_1, \cdots, \mathbf{W}_m] \in \mathbb{R}^{d'\times(md)}, \tag{12}$$

such that we can write

$$\begin{aligned} \tilde{f}_s(\mathbf{x}) &= (\sum_i \alpha_{s,i}(\mathbf{x})\mathbf{W}_i)\mathbf{x} & (13) \\ &= (\tilde{\mathbf{W}}\tilde{\boldsymbol{\alpha}}_s(\mathbf{x}))\mathbf{x}. & (14) \end{aligned}$$

We can then rewrite (10) as:

$$\begin{aligned} \mathcal{L} &= \sum_{i=1}^{n} \|\tilde{\mathbf{W}}\tilde{\boldsymbol{\alpha}}_s(\mathbf{x}_i)\mathbf{x}_i - \mathbf{y}_i\|^2 + \lambda \sum_{j=1}^{m} \|\mathbf{W}_j\|^2 & (15) \\ &= \|\tilde{\mathbf{W}}\,[\tilde{\boldsymbol{\alpha}}_s(\mathbf{x}_1)\mathbf{x}_1, \cdots, \tilde{\boldsymbol{\alpha}}_s(\mathbf{x}_n)\mathbf{x}_n] - \mathbf{Y}\|^2 + \lambda\|\tilde{\mathbf{W}}\|^2, & (16) \end{aligned}$$

which is a standard ridge regression problem with $\tilde{\mathbf{X}} = [\tilde{\boldsymbol{\alpha}}_s(\mathbf{x}_1)\mathbf{x}_1, \cdots, \tilde{\boldsymbol{\alpha}}_s(\mathbf{x}_n)\mathbf{x}_n] \in \mathbb{R}^{(md)\times n}$ and $\mathbf{Y} = [\mathbf{y}_1, \cdots, \mathbf{y}_n] \in \mathbb{R}^{d'\times n}$ as the observations, which has a closed form minimum

$$\tilde{\mathbf{W}} = \mathbf{Y}\tilde{\mathbf{X}}^T(\tilde{\mathbf{X}}\tilde{\mathbf{X}}^T + \lambda\mathbf{I}_{md})^{-1}. \tag{17}$$

As for standard ridge regression, the matrices $\mathbf{Y}\tilde{\mathbf{X}}^T$ and $\tilde{\mathbf{X}}\tilde{\mathbf{X}}^T$ can be efficiently computed with a single pass through the training data. Thus, we only need an $md \times md$ matrix inversion to obtain the optimal regressors $[\mathbf{W}_1, \cdots, \mathbf{W}_m] = \tilde{\mathbf{W}}$.

3.2. Gradient Based Learning

The global solution in the previous section is especially useful when the anchors $\mathbf{C}$ are fixed – *e.g.* obtained from clustering the data $\{(\mathbf{x}_1, \mathbf{y}_1), \cdots, (\mathbf{x}_n, \mathbf{y}_n)\}$.

However, in a more general setting, we would like to be able to use our ARN as a layer in a neural network, such that we can learn the anchors $\mathbf{C}$ jointly with the regressors $\mathbf{W}$ and perform end-to-end learning down to the input space of the network with gradient based learning (*i.e.* SGD).

Given a deep architecture with M layers,

$$G = g^{(M)} \circ g^{(M-1)} \circ \cdots \circ g^{(1)}, \tag{18}$$

we want to use $\tilde{f}_s$ for the last layer $g^{(M)}$. Given a training sample $(\mathbf{x}_i^{(0)}, \mathbf{y}_i)$, we denote the output of the l-th layer $g^{(l)}$ as $\mathbf{x}_i^{(l)} = g^{(l)}(\mathbf{x}_i^{(l-1)})$.

In this setting we assume that the training loss $\mathcal{L}$ takes a more general form

$$\mathcal{L} = \sum_{i=1}^{n} l(\tilde{f}_s(\mathbf{x}_i^{(M-1)}), \mathbf{y}_i) + R(G), \tag{19}$$

where $l(\mathbf{x}, \mathbf{y})$ is the sample loss function and R is the regularization term.

The model parameters of G can then be learned with backpropagation. Denoting $\mathbf{x}^{(M-1)}$ as $\mathbf{x}$, to be able to perform backpropagation through $\tilde{f}_s = g^{(M)}$, we need to compute $\frac{\partial \tilde{f}_s}{\partial \mathbf{x}}$ and $\frac{\partial \tilde{f}_s}{\partial \mathbf{c}_i}$. To this end, using numerator layout for the matrix derivatives, we compute:

$$\frac{\partial \tilde{f}_s}{\partial \mathbf{x}} = \sum_{j=1}^{m} \left(\alpha_{s,j}(\mathbf{x}) \mathbf{W}_j + \mathbf{W}_j \mathbf{x} \frac{\partial \alpha_{s,j}}{\partial \mathbf{x}} \right), \tag{20}$$

$$\frac{\partial \tilde{f}_s}{\partial \mathbf{c}_i} = \frac{\partial \tilde{f}_s}{\partial \boldsymbol{\alpha}_s(\mathbf{x})} \frac{\partial \boldsymbol{\alpha}_s(\mathbf{x})}{\partial \mathbf{c}_i}, \tag{21}$$

$$\frac{\partial \tilde{f}_s}{\partial \boldsymbol{\alpha}_s(\mathbf{x})} = [\mathbf{W}_1 \mathbf{x}, \ldots, \mathbf{W}_m \mathbf{x}]. \tag{22}$$

The gradient $\frac{\partial \alpha_{s,j}}{\partial \mathbf{x}}$ and the Jacobian $\frac{\partial \boldsymbol{\alpha}_s(\mathbf{x})}{\partial \mathbf{c}_i}$ are then computed with standard backpropagation through the similarity layer and the softmax layer (see Figure 1), thus enabling us to backpropagate through the entire layer.

Our experiments (see Table 5) show that gradient based learning enables us to improve the performance of ARN compared to the global solution over fixed anchors, since the anchors are jointly learned with the regressors.

4. Experiments

In this section we validate our proposed ARN for both single- and multi-value regression. For this we deploy our ARN on two challenging tasks: age prediction from face images and single-image super-resolution (SR). Age prediction corresponds to a single age value regression, while SR corresponds to multivalue regression as we map from low-resolution (LR) patch images to high-resolution (HR) patch images. In both cases, we derive our ARN variants within the settings of competitive baselines: the DEX method for age prediction [31, 30] and the A+ method for single image super-resolution [39]. In order to isolate the effect of ARN in our experiments from the underlying architecture, we minimally modify the baseline methods, using the same training material and pretrained models if available, and report on the same validation/test splits.

We implemented our ARN in Theano [38] using Lasagne [6].

Methods	Pretraining DEX IMDB-WIKI	Pretraining VGG-16 ImageNet
VGG-16 Regression[30]	3.650	5.586
SVR on fc7[30]	3.670	12.083
SVR on finetuned fc7[30]	3.323	9.069
DEX (10 neurons)[30]	3.505	5.369
ARN (ours) (10 anchors)	**3.237**	**4.516**
DEX (101 neurons)[30]	3.252	5.965
ARN (ours) (101 anchors)	**3.153**	**4.947**

Table 2. Age prediction results in Mean Absolute Error (MAE) of our ARN vs. DEX [30, 31] on LAP dataset. Both methods start from the same pretrained model. When using 101 neurons/anchors and DEX IMDB-WIKI pre-training, we initialize them with the classifiers of the pre-trained model.

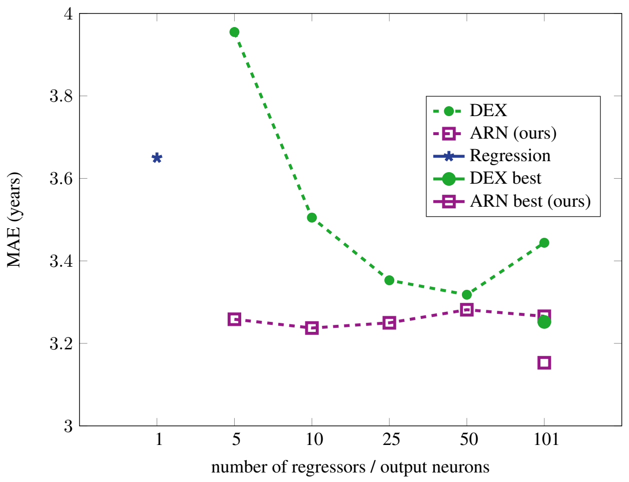


Figure 2. Age prediction results (MAE) vs. number of anchors / regressors / neurons for our ARN and DEX [30] on LAP validation split. Our ARN consistently achieves a significantly lower MAE given the same number of anchors/neurons. DEX and ARN 'best' denote the results with 101 anchors/neurons initialized with the classifiers from the DEX IMDB-WIKI pre-trained model.

4.1. Age Prediction / Single value regression

For our experiments on age prediction, *i.e.* single value regression, we compare our ARN with the DEX method of Rothe *et al.* [30]. DEX is a state-of-the-art method, winner of the ChaLearn LAP 2015 challenge on apparent age estimation at ICCV [9], and with publicly available codes

Setting	Train time	Test time/image	MAE
ARN, 101 anchors	7h 36m	0.136s	3.153
DEX[30], 101 neurons	couple of hours[30]	0.135s	3.252

Table 3. Runtimes for ARN on the LAP dataset.

Method	MORPH 2 (MAE)
Human workers [16]	6.30
DIF [16]	3.80*
AGES [11]	8.83
MTWGP [49]	6.28
CA-SVR [3]	5.88
SVR [13]	5.77
OHRank [2]	5.69
DLA [42]	4.77
[18]	4.25*
[14]	4.18*
[15]	3.92*
[44]	3.63*
[32] (CVPR2016)	3.45
OR-CNN[26] (CVPR2016)	3.27
DEX (101 neurons)[30] (IJCV2016)	3.25
ARN (ours) (101 anchors)	**3.00**

Table 4. Age prediction results (MAE) on the MORPH 2 dataset (* denotes a different split was used).

and trained models. We stay close in that we adhere to the default settings of DEX and report on the same benchmarks.

The processing pipeline of DEX is as follows. For each input image a face detector is used to obtain a robustly aligned frontal face. The detected and aligned face image is then cropped with a 40% margin around the face, and fed into a modified VGG-16 architecture. VGG-16 is a CNN introduced by Simonyan and Zisserman [35] for the task of image classification on the ImageNet challenge [33]. Since, the VGG-16 architecture has been adapted and applied to multiple vision applications. DEX adapts VGG-16 to age prediction by reformulating the age regression problem into a classification of age ranges and trains the network as such. At test time, DEX then predicts the expected age from the class probabilities of each age range. Note that treating age regression as an age range classification is not unique to DEX and is often encountered in the biometrics literature[23, 11]. DEX has shown that the combination of the network output probabilities leads to a better age prediction performance than the direct training of the same architecture for regression. Part of DEX's success is due to the pre-training on the largest crawled public dataset to date, containing face images with age labels (IMDB-WIKI dataset [30]).

To validate our ARN, we build upon the same pipeline as DEX and attach our ARN regression layer on top of the fully connected layer 7 (fc7) of VGG-16. We fix the similarity as $s(\mathbf{x}, \mathbf{c}) = \mathbf{x}^T\mathbf{c}$, such that the only hyperparameter ARN has is m, the number of anchor points / regressors.

4.1.1 Datasets

IMDB-WIKI [30] is the largest publicly available dataset for real age and gender prediction, consisting of 523,051 labeled images of people in the wild, crawled from IMDB and Wikipedia. It is noisy, as the labels come without a guarantee.
MORPH 2 [29] is a large publicly available face database, consisting of 55,134 mug shots, labeled with gender, age and other attributes. For our experiments we adhere to the setup from [2, 3, 13, 42, 32, 30], using a subset of 5,475 images with ages ranging from 16 to 77 years. For evaluation, the dataset is split into 7 random folds, using 80% for training and 20% for testing and the average performance is reported. This is perhaps the most common Morph 2 setup for validating age prediction methods. We use for ARN exactly the same splits as DEX.
ChaLearn LAP dataset [9] consists of 4699 images labeled for apparent age, the average opinion of at least 10 people on how old the subjects look. The dataset is split into 2476 images for training, 1136 for validation and 1087 for testing. Since the test split has not been released, we follow the protocol from [9, 30, 25] and report the validation errors.

4.1.2 Experimental settings

The top results of the DEX method [30] are obtained by first pretraining on the IMDB-WIKI dataset. We focus on two settings for comparison: either starting directly from a VGG-16 model pretrained on ImageNet [33], or starting from a DEX model pretrained on IMDB-WIKI as released by Rothe *et al.* [30]. In the latter case, ARN is at a slight disadvantage, since ideally we would pretrain ARN from scratch on IMDB-WIKI as well. Nonetheless, our experimental results show that we reach a better performance than DEX under all settings.

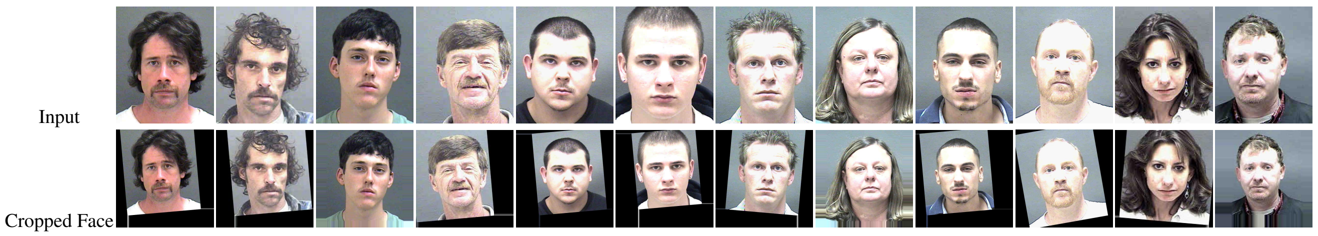
We selected the learning parameters of the fine-tuning by monitoring the validation error on a random split of the MORPH dataset, and used these for all other experiments. The training was terminated after 3000 iterations, with base learning rate 0.0001, regularization 0.0005, batch size 50, learning rate multiplier 10 for layers above fc7, and the learning rate reduced by a factor 0.9 every 2000 iterations. When using 101 anchors and finetuning from the DEX IMDB-WIKI model, we initialize the anchors and biases with the classifiers of the DEX model.

4.1.3 Results

Apparent age prediction / LAP. In Table 2 we report the mean absolute errors (MAE) of ARN in comparison with DEX for ChaLearn LAP. ARN outperforms DEX with and without the IMDB-WIKI model. The differences are more pronounced when starting from ImageNet, since ARN is at

GT Apparent	20.00	76.00	18.00	27.00	21.00	42.00	65.00	17.00	30.00	34.00	8.00	32.00
DEX	20.83	78.20	18.00	29.58	22.06	41.22	67.28	19.89	32.83	39.50	16.80	38.66
ARN	20.15	76.55	18.86	28.25	22.61	44.07	67.57	20.23	33.82	38.81	14.20	41.71

Figure 3. Representative examples of apparent age predictions when using DEX and our ARN network on the Chalearn LAP dataset. The examples are sorted from left to right, sampled according to the ARN error.



GT Real	40.00	38.00	18.00	51.00	24.00	20.00	30.00	48.00	27.00	45.00	37.00	36.00
DEX	39.08	38.80	19.45	52.75	21.84	18.19	32.49	44.75	23.18	39.21	33.15	44.48
ARN	39.83	37.47	18.88	52.24	22.43	18.03	32.38	45.19	23.71	40.93	31.78	43.51

Figure 4. Representative examples of biological age predictions when using DEX and our ARN network on the MORPH 2 dataset. The examples are sorted from left to right, sampled according to the ARN error.

a disadvantage (relative to DEX) when starting from IMDB-WIKI. Interestingly, both DEX and ARN perform better using 10 anchors/neurons instead of the default 101 when starting from VGG-16 pretrained on ImageNet. This indicates that even better results could be obtained by pretraining such a model first on IMDB-WIKI for real age prediction. In Fig. 3 we depict examples varying from very good to very bad ARN predictions, with groundtruth and DEX estimates for comparison.

Influence of the hyperparameter. In Fig. 2 we report the performance of ARN vs. DEX, when we vary the number of ranges/output neurons for DEX and of regressors for ARN, respectively. For these experiments we started from DEX pretrained on IMDB-WIKI with the replacement of the output/classification layer either by a randomly initialized new output layer with the desired number of outputs for DEX or by our randomly initialized ARN layer. The DEX results are taken from the original paper [30]. Note that this number of outputs/regressors is a critical hyperparameter given its influence on the performance of both methods. If for DEX there is a clear peak in performance for around 50 outputs and a dramatic drop below 10 outputs, for our ARN method on the other hand, the performance is quite stable regardless the number of regressors.

Runtimes. Table 3 shows the training and test times for ARN and DEX [30], for the best setting of each method for the ChaLearn LAP [9] dataset. The training time of DEX is reported from [30] while the testing times for both DEX and ARN were measured on the same GeForce GTX TITAN X GPU with Lasagne [6] without any complex optimizations (*i.e.* we do not copy data to the GPU memory, implement buffering of input images in a specific thread, etc). We note that the improved performance of ARN compared to DEX comes at almost no additional test time complexity.

Real age prediction / MORPH 2. In Table 4 we report comparison results for our ARN on the real age prediction task of MORPH 2. We adopt the standard comparison, where none of the methods are pre-trained on other face datasets.

ARN sets a new state-of-the art, clearly improving over DEX, the state-of-the-art age prediction method at the time of writing. When benefiting from the external data of the pretrained IMDB-WIKI model, ARN achieves an even lower MAE of 2.63, compared to a MAE of 2.68 for DEX.

Figure 4 shows faces with very good to very bad ARN predictions, as one moves to the right. As comparison the GT and DEX predictions are added.

4.2. Super-Resolution / Multi-value regression

Having validated the ARN performance on single-value regression for age prediction, we now focus on multi-valued ARN regression for example-based single-image super-resolution. We compare against the A+ (Adjusted Anchored Neighborhood Regression) method of Timofte *et al.* [39] and adhere to its default settings.

A+ formulates the task as a regression problem over patches, by partitioning the LR patch space around anchor points learned with K-SVD[47] to then learn anchored ridge regressors for each such partition from a fixed number of training LR and corresponding HR patches that are

Dataset	Training material						comment
	Std. 91 Train Images [43]			Other Setups			
	Set5	Set14	B100	Set5	Set14	B100	
Bicubic	30.39	27.55	27.21	30.39	27.55	27.21	
SRCNN [7](ECCV2014)	32.39	29.00	28.30				
SRCNN [8](PAMI2016)				32.75	29.30	28.41	395,909 ImageNet img.
PSyCo [27] (CVPR2016)				32.93	29.36		Std. 91 + manifold span reduction
IA [40](CVPR2016)				33.46	29.69	28.76	Std. 91 augmented = 728 img.
VDSR [20] (CVPR2016)				**33.66**	**29.77**	**28.82**	Std. 91+ BSD200 aug. = 2328 img.
JOR [4](Eurographics2015)	32.55	29.09	28.25				32 regressors, 5million anchor points
A+ [39] (ACCV2014)	32.59	29.13	28.29				1024 regressors
ARN (global solution)	32.69	29.20	28.32				1024 regressors
ARN	32.89	29.31	28.38				1024 regressors
Deep ARN	**33.01**	**29.37**	**28.45**				7×16 regressors

Table 5. Single image super resolution average PSNR (dB) results on 3 datasets with upscaling factor $\times 3$ for A+ and with ARN using the default settings of A+. ARN significantly improves over A+ (same features and training data).

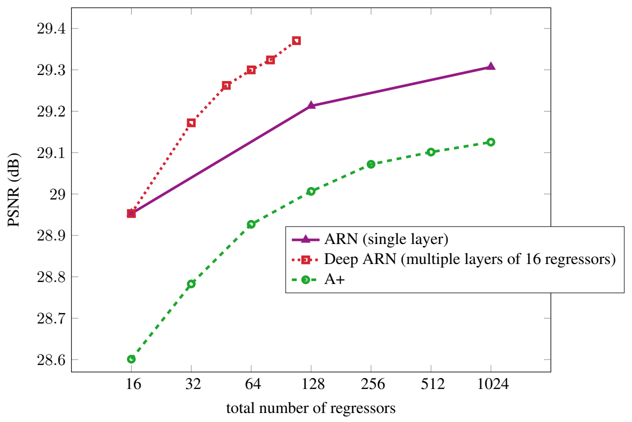


Figure 5. Average PSNR [dB] on (Set14, $\times 3$) vs. number of regressors/anchor points for A+ [39] and ARN. ARN consistently improves over A+.

the 'closest' to the anchor point. Some A+ variants were proposed for joint learning of regressors and/or anchors. Dai *et al.* [4] jointly optimize the regressors while using all the training samples as anchors. Zhang *et al.* [48] jointly learn anchors and their corresponding regressors with an expectation-maximization formulation. However, these works [4, 48] report results comparable with or inferior to A+. In contrast to A+, deep SR methods, such as Dong *et al.* [7, 8] and, the best to date, Kim *et al.* [20], use convolutional layers to regress the entire HR image with an (arguably more natural) image-to-image regression formulation.

Since our ARN is designed for "standard regression", *i.e.* mapping a vector from $\mathbb{R}^d$ to $\mathbb{R}^{d'}$, the patch based formulation of $A+$ provides a natural setting for such a multidimensional regression. Extending ARN for convolutional layers would however be an interesting direction for future works.

Experimental setup. For a fair comparison with A+ [39], we adhere to the benchmark and settings from the paper and use the publicly available code which gives us the same numbers as reported by the authors. As was the case for A+, we work only on the luminance Y channel from the YCbCr color space and for color images we bicubically interpolate the chroma channels before converting them back to the RGB color space. This is motivated by the fact that human vision is more sensitive to the intensity changes and texture than to the color. Using the A+ codes in the $\times 3$ upscaling mode, we extract the training material of the A+ method, which consists of about 5 million samples of 30-dimensional LR patch features and corresponding HR patch residuals. The LR image patch is 3×3 and is to be turned into a $9 \times 9 = 81$ dimensional HR image patch. A+ learns its anchors and regressors from these 5 million samples to at test time upscale the LR image patch by patch over a dense uniform grid and then average the overlapping restored HR patch residuals, finally added to the bicubically upscaled LR input image.

ARN learning. Our ARN model keeps the settings of A+. It works with the same training samples and patch feature representations and treats the LR input and HR output patches in the same way. We trained ARN for 90 epochs using stochastic gradient descent with momentum 0.9, learning rate 20, regularization 10^{-8}, learning rate decay 0.5 every 20 epochs, and batch size 1000. We simply randomly initialized the anchor points and the regressors, and used $s(\mathbf{x}, \mathbf{c}) = |\mathbf{x}^T\mathbf{c}|$, such that the only hyperparameter is the number of anchor points / regressors.

Deep ARN learning. With ARN we can go deeper and apply it multiple times in a layered design / architecture. Thus, ARN can also be used as a building block for new architectures and not only as the last regression layer.

When regressing from $\mathbb{R}^d$ to $\mathbb{R}^{d'}$ we compose $L-1$ layers of ARN regressors mapping from $\mathbb{R}^d$ to $\mathbb{R}^d$ along with shortcut connections [17], which are then composed with the final ARN regressor which maps from $\mathbb{R}^d$ to $\mathbb{R}^{d'}$. In total we then use $L \times m$ regressors, where m is the number of regressors in each layer. We denote this setting as L layers of deep ARN or Deep ARN with $L \times m$ layered regressors.

Results. For evaluation we use the benchmark from [39] and report average Peak Signal to Noise Ratio (PSNR) (dB)

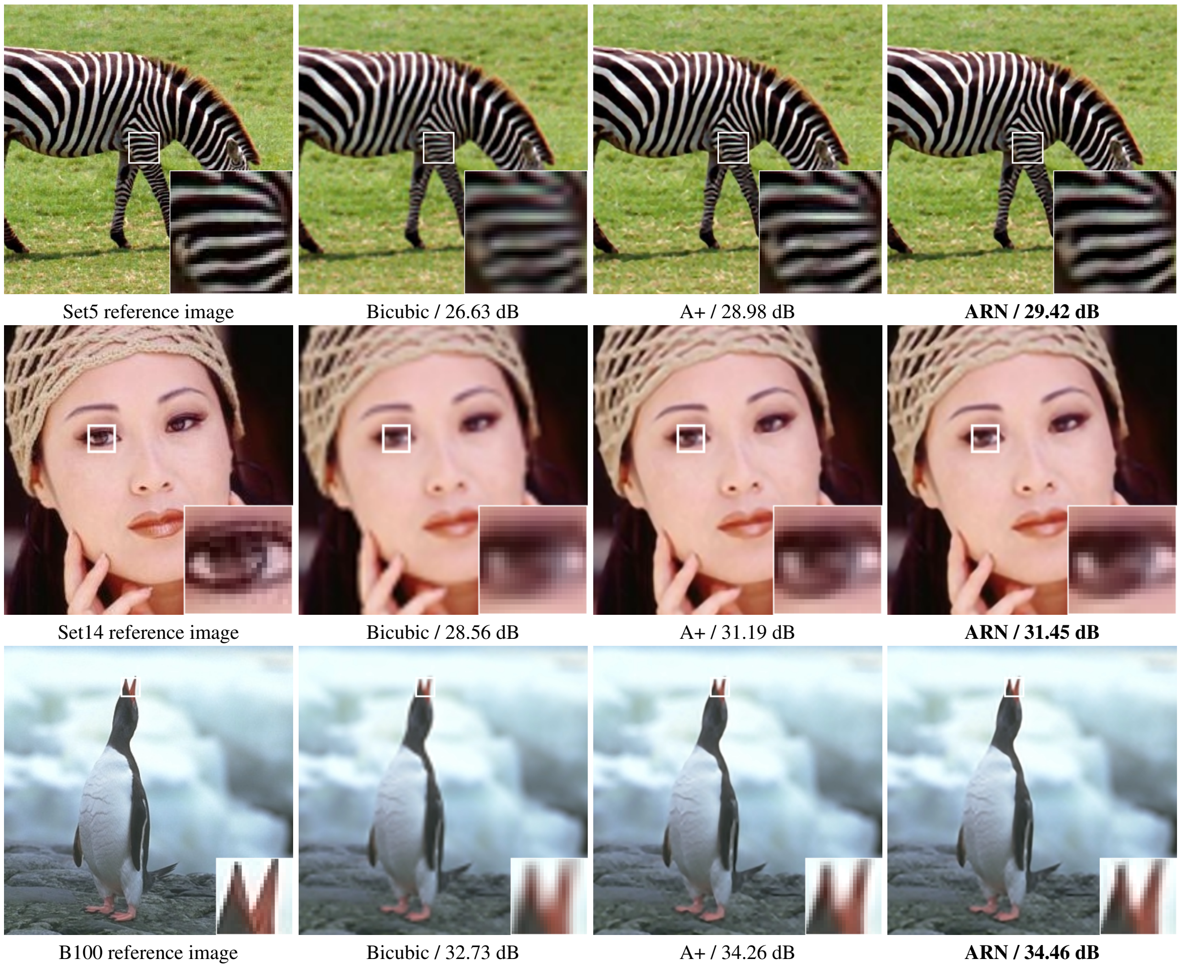


Figure 6. SR $\times 3$ results for 'zebra' (Set5), 'woman' (Set14), and '106024' (B100) images. Best zoom in.

results for the Set5, Set14, and B100 datasets, which contain 5, 14, and 100 different images, resp. The results are shown in Table 5 using the default setting of A+, namely 1024 anchor points/regressors. We significantly outperform A+ on all datasets. Figure 5 compares the performance of ARN vs. A+, as a function of the number of anchor points / regressors. The ARN gains over A+ go from above 0.35dB for 16 regressors to 0.18dB for 1024 regressors, which are significant.

For **visual assessment** we show three results in Figure 6. The ARN super-resolved images have usually fewer artifacts and sharper edges than with A+.

Deep ARN further improves over ARN using the same training settings/data (see Table 5 and Fig. 2) but fewer regressors, while the ARN with global solution performs as expected in between A+ and ARN when departing from the A+ features and under the same training conditions.

5. Conclusions

We proposed Anchored Regression Networks (ARN), a novel regression architecture that can be used standalone on fixed features or for end-to-end learning in a CNN, with standard gradient based methods. Our ARN architecture has a simple formulation motivated by basic and intuitive principles, admits a closed form solution for the regressors in the case of an L_2-loss while being easy to implement for end-to-end learning.

We have demonstrated the power of ARN for both single and multi-value regression on two challenging tasks. For age prediction from face images we set a new state-of-the art while for single image super-resolution we significantly outperform the baseline method, using the same training settings. With the former task being a high-level vision task requiring a deep understanding of the human face, and the latter being a low-level vision task operating on patches on the image, our experiments validate the generic applicability of our ARN architecture.

Thus, our ARN shows a consistent potential to easily integrate into various architectures, wherever nonlinear regression is needed.

Acknowledgments. This work was supported by the ETH Zurich General Fund (OK) and an NVIDIA GPU grant.

References

[1] D. Basak, S. Pal, and D. C. Patranabis. Support vector regression. *Neural Information Processing-Letters and Reviews*, 11(10):203–224, 2007. 1

[2] K.-Y. Chang, C.-S. Chen, and Y.-P. Hung. Ordinal hyperplanes ranker with cost sensitivities for age estimation. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 585–592, June 2011. 5

[3] K. Chen, S. Gong, T. Xiang, and C. Loy. Cumulative attribute space for age and crowd density estimation. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 2467–2474, June 2013. 5

[4] D. Dai, R. Timofte, and L. Van Gool. Jointly optimized regressors for image super-resolution. *Comput. Graph. Forum*, 34(2):95–104, May 2015. 7

[5] W. S. DeSarbo and W. L. Cron. A maximum likelihood methodology for clusterwise linear regression. *Journal of classification*, 5(2):249–282, 1988. 2

[6] S. Dieleman, J. Schlter, C. Raffel, E. Olson, S. K. Snderby, D. Nouri, D. Maturana, M. Thoma, E. Battenberg, J. Kelly, et al. Lasagne: First release., Aug. 2015. 4, 6

[7] C. Dong, C. C. Loy, K. He, and X. Tang. Learning a deep convolutional network for image super-resolution. In *European Conference on Computer Vision*, pages 184–199. Springer, 2014. 7

[8] C. Dong, C. C. Loy, K. He, and X. Tang. Image super-resolution using deep convolutional networks. *IEEE transactions on pattern analysis and machine intelligence*, 38(2):295–307, 2016. 7

[9] S. Escalera, J. Fabian, P. Pardo, X. Baro, J. Gonzalez, H. J. Escalante, D. Misevic, U. Steiner, and I. Guyon. Chalearn looking at people 2015: Apparent age and cultural event recognition datasets and results. In *The IEEE International Conference on Computer Vision (ICCV) Workshops*, December 2015. 1, 4, 5, 6

[10] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *IEEE transactions on pattern analysis and machine intelligence*, 32(9):1627–1645, 2010. 1

[11] X. Geng, Z.-H. Zhou, and K. Smith-Miles. Automatic age estimation based on facial aging patterns. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(12):2234–2240, Dec 2007. 5

[12] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014. 1

[13] G. Guo, Y. Fu, C. Dyer, and T. Huang. Image-based human age estimation by manifold learning and locally adjusted robust regression. *Image Processing, IEEE Transactions on*, 17(7):1178–1188, July 2008. 5

[14] G. Guo and G. Mu. Simultaneous dimensionality reduction and human age estimation via kernel partial least squares regression. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 657–664, June 2011. 5

[15] G. Guo and G. Mu. A framework for joint estimation of age, gender and ethnicity on a large database. *Image and Vision Computing*, 32(10):761 – 770, 2014. Best of Automatic Face and Gesture Recognition 2013. 5

[16] H. Han, C. Otto, X. Liu, and A. Jain. Demographic estimation from face images: Human vs. machine performance. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 37(6):1148–1161, June 2015. 5

[17] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016. 7

[18] I. Huerta, C. Fernández, and A. Prati. *Computer Vision - ECCV 2014 Workshops: Zurich, Switzerland, September 6-7 and 12, 2014, Proceedings, Part II*, chapter Facial Age Estimation Through the Fusion of Texture and Local Appearance Descriptors, pages 667–681. Springer International Publishing, Cham, 2015. 5

[19] V. Kazemi and J. Sullivan. One millisecond face alignment with an ensemble of regression trees. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014. 1

[20] J. Kim, J. Kwon Lee, and K. Mu Lee. Accurate image super-resolution using very deep convolutional networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016. 7

[21] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. Burges, L. Bottou, and K. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012. 1

[22] L. Ladicky and P. Torr. Locally linear support vector machines. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 985–992, 2011. 2

[23] A. Lanitis, C. Draganova, and C. Christodoulou. Comparing different classifiers for automatic age estimation. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 34(1):621–628, 2004. 5

[24] F. Liu, C. Shen, G. Lin, and I. Reid. Learning depth from single monocular images using deep convolutional neural fields. *IEEE transactions on pattern analysis and machine intelligence*, 38(10):2024–2039, 2016. 1

[25] X. Liu, S. Li, M. Kan, J. Zhang, S. Wu, W. Liu, H. Han, S. Shan, and X. Chen. Agenet: Deeply learned regressor and classifier for robust apparent age estimation. In *The IEEE International Conference on Computer Vision (ICCV) Workshops*, December 2015. 1, 5

[26] Z. Niu, M. Zhou, L. Wang, X. Gao, and G. Hua. Ordinal regression with multiple output cnn for age estimation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016. 5

[27] E. Perez-Pellitero, J. Salvador, J. Ruiz-Hidalgo, and B. Rosenhahn. Psyco: Manifold span reduction for super resolution. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016. 7

[28] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015. 1

[29] K. Ricanek and T. Tesafaye. Morph: a longitudinal image database of normal adult age-progression. In *Automatic Face and Gesture Recognition, 2006. FGR 2006. 7th International Conference on*, pages 341–345, April 2006. 5

[30] R. Rothe, R. Timofte, and L. Van Gool. Deep expectation of real and apparent age from a single image without facial landmarks. *International Journal of Computer Vision*, pages 1–14. 1, 4, 5, 6

[31] R. Rothe, R. Timofte, and L. Van Gool. Dex: Deep expectation of apparent age from a single image. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 10–15, 2015. 4

[32] R. Rothe, R. Timofte, and L. Van Gool. Some like it hot - visual guidance for preference prediction. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016. 5

[33] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015. 5

[34] A. Saxena, S. H. Chung, and A. Y. Ng. Learning depth from single monocular images. 1

[35] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014. 1, 5

[36] H. Späth. Algorithm 39 clusterwise linear regression. *Computing*, 22(4):367–373, 1979. 2

[37] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015. 1

[38] Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688, May 2016. 4

[39] R. Timofte, V. De Smet, and L. Van Gool. A+: Adjusted anchored neighborhood regression for fast super-resolution. In *Asian Conference on Computer Vision*, pages 111–126. Springer, 2014. 2, 3, 4, 6, 7

[40] R. Timofte, R. Rothe, and L. Van Gool. Seven ways to improve example-based single image super resolution. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1865–1873, 2016. 1, 7

[41] J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, and Y. Gong. Locality-constrained linear coding for image classification. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 3360–3367. IEEE, 2010. 2

[42] X. Wang, R. Guo, and C. Kambhamettu. Deeply-learned feature for age estimation. In *Applications of Computer Vision (WACV), 2015 IEEE Winter Conference on*, pages 534–541, Jan 2015. 5

[43] J. Yang, J. Wright, T. Huang, and Y. Ma. Image super-resolution as sparse representation of raw image patches. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, June 2008. 1, 7

[44] D. Yi, Z. Lei, and S. Z. Li. *Computer Vision – ACCV 2014: 12th Asian Conference on Computer Vision, Singapore, Singapore, November 1-5, 2014, Revised Selected Papers, Part III*, chapter Age Estimation by Multi-scale Convolutional Network, pages 144–158. Springer International Publishing, Cham, 2015. 5

[45] K. Yu and T. Zhang. Improved local coordinate coding using local tangents. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 1215–1222, 2010. 2

[46] K. Yu, T. Zhang, and Y. Gong. Nonlinear learning using local coordinate coding. In *Advances in neural information processing systems*, pages 2223–2231, 2009. 2

[47] R. Zeyde, M. Elad, and M. Protter. On single image scale-up using sparse-representations. In *Curves and Surfaces*, pages 711–730. Springer, 2010. 6

[48] K. Zhang, B. Wang, W. Zuo, H. Zhang, and L. Zhang. Joint learning of multiple regressors for single image super-resolution. *IEEE Signal Processing Letters*, 23(1):102–106, Jan 2016. 7

[49] Y. Zhang and D.-Y. Yeung. Multi-task warped gaussian process for personalized age estimation. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2622–2629, June 2010. 5