LABEL ENCODING FOR REGRESSION NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Deep neural networks are used for a wide range of regression problems and are typically trained by minimizing the squared or absolute error of output labels. Solving a regression problem with a set of binary classifiers can improve the accuracy and also utilizes well-studied binary classification algorithms. However, a limited design space of regression by binary classification has been studied by prior works. This work introduces a novel approach using *binary-encoded labels (BEL)* to generalize the application of binary classification to regression, and a taxonomy identifying the key design aspects of such formulation. Of these design aspects, we primarily explore and identify the desirable properties of suitable encoding and decoding functions used for the conversion between real-valued and binary-encoded labels based on theoretical and empirical analysis. BEL can be combined with off-the-shelf task-specific feature extractors and trained end-to-end. We further propose a series of sample encoding, decoding, and training loss functions applicable in this domain and demonstrate that our generic regression approach results in lower error than direct regression and specialized approaches while being suitable for various regression problems, network architectures, and evaluation metrics.

1 INTRODUCTION

Deep regression networks, in which a continuous output is predicted for a given input, are traditionally trained by minimizing squared/absolute error of a logit, which we refer to as *direct regression*. More recently, several works have proposed specialized regression approaches for regression problems such as head pose estimation, age estimation, and facial landmark estimation, and shown significant accuracy improvements over direct regression. However, developing task-agnostic generic approaches to improve accuracy for regression is desirable.

A regression problem can be posed as a set of binary classification sub-problems. A similar approach has been applied to other domains such as ordinal regression (Li & Lin, 2006) and multiclass classification (Dietterich & Bakiri, 1995). Such formulation allows the use of well-studied binary classification approaches. Further, new generalization bounds for ordinal regression or multiclass classification can be derived from the known generalization bounds of binary classification. This reduces the efforts for design, implementation, and theoretical analysis significantly (Li & Lin, 2006). Dietterich & Bakiri (1995) demonstrated that posing multiclass classification as a set of binary classification problems can increase error tolerance and improve accuracy. However, the proposed approaches for multiclass classification do not apply to regression due to the differences in task objective and properties of the error probability distribution of classifiers (Section 2). On the other hand, prior works on ordinal regression have explored the application of binary classifiers in a more restricted way which limits its application to a wide range of complex regression problems (Section 2). *There exists a lack of a generic framework that unifies possible formulations for using binary classification to solve regression.*

In this work, we propose *binary-encoded labels (BEL)* which improves regression by generalizing the application of binary classification. In BEL, a target label is quantized and converted to a binary code of length M, and M binary classifiers are then used to learn binary-encoded labels. An encoding function is introduced to convert the target label to a binary code, and a decoding function is introduced to decode the output of binary classifiers to the predicted label. BEL allows using an adjustable number of binary classifiers depending upon the quantization, encoding, and decoding

functions. BEL opens possible avenues to improve the accuracy of regression problems with a large design space of the quantization function, encoding/decoding functions, and optimization algorithm.

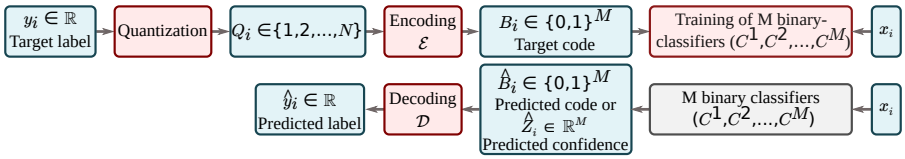
We focus on the encoding and decoding functions and theoretically study the relations between the absolute error of label and binary classifiers' errors for sample encoding and decoding functions. This analysis demonstrates the impact of binary classifiers' error distribution over the numeric range of target labels on the suitability of different encoding and decoding functions. Based on our analysis and empirically observed binary classifiers' error distribution, we propose properties of suitable encoding functions for regression and explore various encoding functions on a wide range of tasks. We also propose an expected correlation-based decoding function for regression that can effectively reduce the quantization error introduced by the encoding.

A deep regression network consists of a feature extractor and a regressor and is trained end-to-end. Our proposed regression approach (BEL) can be combined with off-the-shelf task-specific feature extractors by increasing the regressor's output logits. We find that the correlation between multiple binary classifiers' outputs can be exploited to reduce the size of the feature vector and consequently reduce the number of parameters in the regressor. The training loss function can be formulated according to the decoding function used; we study the binary cross-entropy, cross-entropy, MAE, and MSE loss functions for different decoding functions. We evaluate our approach on four complex regression problems: head pose estimation, facial landmark detection, age estimation, and end-to-end autonomous driving. We make the following contributions in this work:

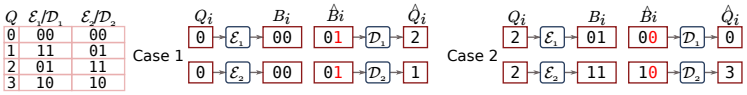
- We propose binary-encoded labels for regression and introduce a general framework and a taxonomy for the design aspects of regression by binary classification. We propose desirable properties of encoding and decoding functions suitable for regression problems.
- We present a series of suitable encoding, decoding, and loss functions for regression with BEL. We present an end-to-end learning approach and regression layer architecture for BEL. We combine BEL with task-specific feature extractors for four tasks and evaluate multiple encoding, decoding, and loss functions. BEL outperforms direct regression for all the problems and specialized approaches for several tasks.
- We theoretically and empirically demonstrate the effect of different design parameters on the accuracy, how it varies across different tasks, datasets, and network architectures, and provide preliminary insights and motivation for further study.

2 Related Work

Binary classification for regression: Prior works have proposed binary classification-based approaches for ordinal regression (Crammer & Singer, 2001; Chu & Keerthi, 2005; Li & Lin, 2006). Ordinal regression is a class of supervised learning problems, where the samples are labeled by a rank that belongs to an ordinal scale. Ordinal regression approaches can be applied to regression by discretizing the numeric range of the real-valued labels (Fu et al., 2018; Berg et al.; Niu et al., 2016). In the existing works on ordinal regression by binary classification, $N-1$ binary classifiers are used for target labels $\in \{1, 2, ..., N\}$, where classifier-k predicts if the label is greater than k or not for a given input. Li & Lin (2006) provided a reduction framework and generalization bound for the same. However, the proposed binary classification formulation is restricted. It requires several binary classifiers if the numeric range of output is extensive, whereas reducing the number of classifiers by using fewer quantization levels increases quantization error. Thus, a more generalized approach for using binary classification for regression is desirable to allow the flexible design of output classifiers.
Binary classification for multiclass classification: Dietterich & Bakiri (1995) proposed the use and properties of error-correcting output codes to convert a multiclass classification to a set of binary classification sub-problems. This improves the accuracy of the network as it introduces tolerance to errors in output logits depending upon the hamming distance (i.e., number of bits changed between two binary strings) between two codes. Allwein et al. (2001) provided a unifying framework and multiclass loss bounds in terms of binary classification loss for this formulation. More recent works have also used Hadamard code, a widely used error-correcting code (Song et al., 2021; Verma & Swami, 2019). However, the proposed encoding and decoding approaches do not consider the ordinality of labels for regression. Further, the binary classifiers possess distinct error probability distribution properties for regression problems as observed empirically (Section 3.1), which can be exploited to design codes suitable for regression.



(a) Training and inference flow



(b) Encoding and decoding examples

Figure 1: (a) The training (top) and inference (bottom) flow of binary-encoded labels (BEL) for deep regression. Red colored blocks represent design aspects we focus on. (b) Examples of two lookup table-based encoding and decoding functions. $\hat{Q}_i$ represents the decoded quantized level.

Multiclass classification and ordinal regression by binary classification can be viewed as special cases falling under the BEL framework. As shown in Section 4, other BEL designs yield improvements in accuracy over these approaches. Task-specific regression techniques are well explored as summarized below (see also Appendix D-G). While effective, task-specific approaches lack generality by design.

Head pose estimation: SSR-Net (Rothe et al., 2015) and FSA-Net (Yang et al., 2019) used softmax to refine labels. HopeNet (Ruiz et al., 2018) used a combination of classification and regression loss. QuatNet (Hsu et al., 2019) used a combination of regression and ordinal regression loss.
Facial landmark detection: Wang et al. (2020) minimize L2 loss between predicted and target 2D heatmaps with the latter formed using small variance Gaussians centered on ground truth landmarks. AWing (Wang et al., 2019) modified loss for different pixels in the heatmap. LUVLi (Kumar et al., 2020) proposed a landmark's location, uncertainty, and visibility likelihood-based loss. Bulat & Tzimiropoulos (2016) used binary heatmaps with pixel-wise binary cross-entropy loss.
Age estimation: OR-CNN (Niu et al., 2016) used ordinal regression via binary classification. CORAL-CNN (Cao et al., 2020) refined this approach by enforcing the ordinality of the model output. MV-Loss (Pan et al., 2018) extended the soft regression approach by penalizing the output of the model based on the variance of the age distribution, while DLDL (Gao et al., 2018) proposed to use the KL-divergence between the softmax output and a generated label distribution to train a model.

3 Binary-encoded Labels for Regression (BEL)

We consider regression problems where the goal is to minimize the error between real-valued target labels y_i and predicted labels $\hat{y}_i$, over a set of training samples i. We transform this problem to a set of binary classification sub-problems by converting a real-valued label to a binary code.

Figure 1a shows the training and inference flow for BEL. The red-colored blocks highlight functions that vary under BEL. A real-valued label $y_i \in \mathbb{R}$ is quantized to level $Q_i \in \{1, 2, ..., N\}$. The quantized label is then converted to a binary sequence $B_i \in \{0, 1\}^M$ using an encoding function $\mathcal{E}$. The binary-encoded labels B_i are used to train M classifiers. In the prediction phase, the classifiers predict a binary code $\hat{B}_i \in \{0, 1\}^M$ for an input x_i. The predicted binary code ($\hat{B}_i$) or the real-valued confidence vector of classifiers' prediction ($\hat{Z}_i$) is then decoded to a predicted label $\hat{y}_i \in \mathbb{R}$ using a decoding function $\mathcal{D}$. We explore decoding functions that yield either quantized or continuous predicted outputs. For the latter, we employ an expected correlation-based decoding function that decodes classifiers' output to a continuous value, reducing the quantization error (Section 3.3).

There are five key design parameters when applying BEL: quantization, encoding, decoding, regressor network architecture, and training loss formulation yielding a large design space. Figure 1b illustrates how encoding and decoding functions $\{\mathcal{E}, \mathcal{D}\}$ can impact accuracy. The figure shows two distinct lookup table based encoding/decoding functions $\{\mathcal{E}_1, \mathcal{D}_1\}$ and $\{\mathcal{E}_2, \mathcal{D}_2\}$. Case 1 and 2 illustrate examples of encoding a given target value and decoding an erroneous predicted code (red bits indicate misclassification). For a given classification error, $\{\mathcal{E}_2, \mathcal{D}_2\}$ exhibits lower regression error versus $\{\mathcal{E}_1, \mathcal{D}_1\}$. In this work we consider only uniform quantization while leaving nonuniform quantization (Fu et al., 2018) to future work. Section 3.2 and 3.3 explore the characteristics of suitable

Q_i
1
2
3
4
5
6
7
8

(a) Label

$C^1C^2C^3C^4C^5C^6C^7$
0 0 0 0 0 0 0
1 0 0 0 0 0 0
1 1 0 0 0 0 0
1 1 1 0 0 0 0
1 1 1 1 0 0 0
1 1 1 1 1 0 0
1 1 1 1 1 1 0
1 1 1 1 1 1 1

(b) Unary

$C^1C^2C^3C^4$
0 0 0 1
0 0 1 1
0 1 1 1
1 1 1 1
1 1 1 0
1 1 0 0
1 0 0 0

(c) Johnson

C^1	C^2C^3
0	0 0
0	0 1
0	1 1
0	1 0
1	0 0
1	0 1
1	1 1
1	1 0

(d) B1JDJ (NRB)

C^1	C^2C^3
0	0 0
0	0 1
0	1 1
0	1 0
1	1 0
1	1 1
1	0 1
1	0 0

(e) B1JDJ

C^1C^2	C^3
0 0	0
0 0	1
0 1	1
0 1	0
1 1	0
1 1	1
1 0	1
1 0	0

(f) B2JDJ

Q_i	Hex of Q_i	$C^1C^2.....C^8\ C^9....C^{15}C^{16}$
0	00	00000000 00000000
1	01	00000000 00000001
2	02	00000000 00000011
15	0F	00000000 10000000
32	20	00000011 00000000
33	21	00000011 00000001
34	22	00000011 00000011
35	23	00000011 00000111

(g) HEXJ

Figure 2: Examples of different codes proposed for deep regression network. (a) represents the corresponding numeric values and quantized values of the labels for different codes in (b)-(f). (g) represents examples of quantized and encoded values of labels for HEXJ code. Space between bits is added to differentiate between base and displacement, or digits. Red lines represent decision boundaries.

encoding and decoding functions. Section 3.4 explore the impact of regression network architecture and training loss for given encoding/decoding functions. We find varying any of these aspects can improve regression accuracy. While BEL provides a framework and some codes appear generally better than others, the most suitable BEL code to employ appears to vary across task, dataset, and network architecture, as we show both theoretically (Section 3.1) and empirically (Section 4).

3.1 Analysis of Encoding/Decoding Functions

This section analytically analyzes the potential impact of $\{\mathcal{E}, \mathcal{D}\}$ on regression accuracy based upon empirically observed error distributions for the underlying classifiers. We divide our analysis into three parts. First, we derive the relation between the expected error of predicted labels and classifiers' errors for two $\{\mathcal{E}, \mathcal{D}\}$ functions. Next, we propose an approximate classifier's error probability distribution over the numeric range of target labels for regression based on empirical study. Last, we compare the expected error of sample $\{\mathcal{E}, \mathcal{D}\}$ functions based on our analysis. We use labels $y_i \in [1, N-1]$, with quantization levels $Q_i \in \{1, 2, ..., N-1\}$. Quantization error is not included as it is not affected by $\{\mathcal{E}, \mathcal{D}\}$ functions. We compare the impact of a unary (Figure 2b) and a Johnson code (Figure 2c). The intuition for trying these codes is discussed in Section 3.2.

Expected absolute error bounds in terms of classification error: First, we analyze the expected absolute error given the classification error for a unary code, BEL-U. The encoding function $\mathcal{E}^{\text{BEL-U}}$ converts Q_i to $B_i = b_i^1, b_i^2, ..., b_i^{N-2}$, where $b_i^k = 1$ for $k < Q_i$, else 0. The decoding function is:

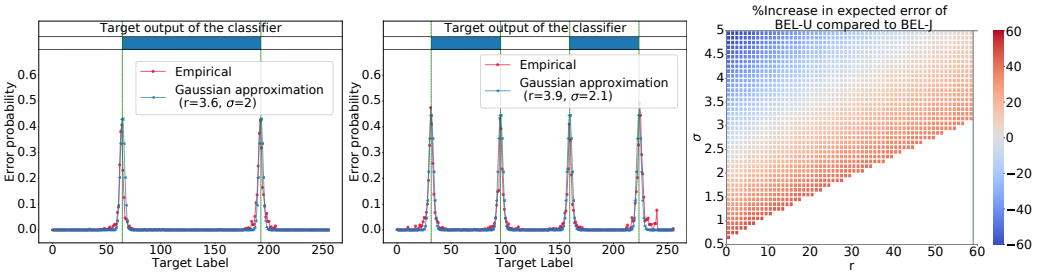
$$\mathcal{D}^{\text{BEL-U}}(\hat{b}_i^1, \hat{b}_i^2, ..., \hat{b}_i^{N-2}) = \sum_{k=1}^{N-2} \hat{b}_i^k + 1 \tag{1}$$

Let $e_k(n)$ be the error probability of classifier C^k for target quantized label $Q_i = n$. For a uniform distribution of y_i in the range $[1, N-1]$, the expected error for BEL-U is:

$$\mathbb{E}(|\hat{y} - y|) \leqslant \frac{1}{N-1} \sum_{n=1}^{N-1} \Big(\sum_{k=1}^{N-2} e_k(n) \Big) \tag{2}$$

A similar analysis can be applied for the Johnson code in which the encoding function $\mathcal{E}^{\text{BEL-J}}$ converts Q_i to a binary code $B_i = b_i^1, b_i^2, ..., b_i^{N/2}$, where, $b_i^k = 1$ for $\frac{N}{2} - Q_i < k \leqslant N - Q_i$, else 0 (Appendix B).

Error probability of binary classifiers: A classifier's target output is 0 or 1 for different quantized target labels on an ordinal scale, and consists of one or more decision boundaries. For example, for the unary code shown in Figure 2b, the target output of classifier C^2 is 1 for $Q_i > 2$ and 0 otherwise. This classifier should learn a decision boundary between target labels in $[1, 2]$ and $[3, 8]$. Thus, ideally, a classifier learns a decision boundary for each $0 \rightarrow 1$ or $1 \rightarrow 0$ transition in the classifier's target output over the range of target labels. It is known that the error/misclassification probability $e_k(n)$ of a classifier tends to increase as the target label value n is closer to the classifier's decision boundaries (Cardoso & Pinto da Costa, 2007). Thus, we approximate $e_k(y)$ for a classifier C^k with t



(a) Classifier A (b) Classifier B (c) Expected error comparison

Figure 3: (a),(b) Classification error probability versus target label y for different classifiers. The top horizontal bar represent target output of the classifier. Blue color represents output 1. The area under the error probability distribution can exceed 1 as the labels are independent. (c) Percentage increase in the expected error for BEL-U compared to BEL-J based on equation 2- equation 4.

decision boundaries as a linear combination of t Gaussian distributions. Here, each Gaussian term is centered around a decision boundary. Let $f_{\mathcal{N}(\mu,\sigma^2)}(y)$ denote the probability density of a normal distribution with mean μ and variance σ^2. Each classifier for BEL-U encoding has one decision boundary, whereas, each classifier for BEL-J encoding has two decision boundaries (except the first and last classifiers). $e_k(y)$ of a classifier C^k for BEL-U and BEL-J encoding is approximated as:

$$e_k^{BEL-U}(y) = r f_{\mathcal{N}(\mu_k,\sigma^2)}(y), \text{where, } \mu_k = k + 0.5 \tag{3}$$

$$e_k^{BEL-J}(y) = r f_{\mathcal{N}(\mu_{1k},\sigma^2)}(y) + r f_{\mathcal{N}(\mu_{2k},\sigma^2)}(y), \text{where, } \mu_{1k} = \frac{N}{2} - k + 0.5, \mu_{2k} = N - k + 0.5 \tag{4}$$

Here, r is a variable that scales up/down the error probabilities. Figure 3a and Figure 3b show the empirically observed and proposed approximate error probability distributions for two classifiers trained using HRNetV2-W18 (Wang et al., 2020) and COFW facial landmark detection dataset (Burgos-Artizzu et al., 2013) for labels $y \in [1, 256]$.

Comparison of expected absolute error for BEL-U and BEL-J: Based on equation 2-equation 4, we compare expected absolute errors of BEL-U and BEL-J encoding/decoding functions. Figure 3c represents the percentage increase in absolute error for BEL-U compared to BEL-J for valid values of variance σ^2 ($y-$axis) and scaling factor r ($x-$axis) from equation 3 and equation 4. Here, BEL-J has a lower error in the red-colored region (%increase> 0), whereas BEL-U has a lower error in the blue-colored region (%increase > 0). The figure shows that BEL-J or BEL-U has lower error depending upon the values σ and r used to approximate the classifiers' error probability distribution. We leave the derivation of these parameters from the first principles to future work. Further theoretical study of the properties of classifiers for ordinal labels can provide bounds on these values, which will also help with encoding/decoding design. Thus, the choice of $\{\mathcal{E}, \mathcal{D}\}$ functions heavily depends upon the classifiers' error probability distribution, which depends upon many factors, including the task objective, dataset, label distribution, and optimization approach. We provide the derivation of expected error for BEL-U and BEL-J and classifiers' empirical error probability distributions for different architectures, datasets, and encodings in Appendix C-D.

3.2 Design of Encoding Functions

The objective of regression is to minimize the absolute/squared error between predicted and target values, and the encoding function can be designed to take advantage of this fact. Thus, a desirable property for a BEL encoding function is that: *the distance between two quantized values should be proportional to the distance between their codes.*

Further, the error probability of classifiers tends to increase as the target gets closer to a decision boundary (Figure 3a-3b). If the target value is far from the decision boundaries of a given classifier, its output is less likely to be erroneous. We note that a hamming distance does not account for which classifiers are more likely to mispredict. This gives the second property of a suitable encoding: *the distance metric must consider how likely each classifier is to mispredict for some target value to account for classifiers' nonuniform error probability distribution for a regression problem.*

We evaluate several encodings that to greater or lesser extent exhibit these desirable properties. We focus on encodings that reduce the number of classifiers (bits) while balancing the separation between the decision boundaries per classifier. Below we empirically evaluate the impact of these different encodings on previously proposed deep regression networks. We leave the development of procedures for automatically discovering optimized encodings to future work. Figure 2 shows example codes for encodings explored. We explain all encodings in detail below.

Unary code (U): The unary code (Wikipedia, b) (Figure 2b) is a simple code with one decision boundary per classifier and requires $N-1$ bits to encode N values. Prior works on ordinal regression by binary classification (Li & Lin, 2006; Fu et al., 2018; Niu et al., 2016) uses a similar formulation of classifiers as the unary code.

Johnson code (J): The Johnson code sequence (Figure 2c) is used in the design of counters in digital circuits and is based on Libaw-Craig code (Libaw & Craig, 1953). We select this code as it has well-separated decision boundaries and requires $\frac{N}{2}$ bits compared to N required for unary code. This code exemplifies the difference between hamming distance and a distance that considers classifiers' error probability distribution. Although the hamming distance between 1 and 7 is small, since the differing bits (C^1 and C^4) are far from decision boundaries and less likely to be mispredicted, a 1 is less likely to be mispredicted as a 7.

Base+displacement based code (B1JDJ/B2JDJ): We further reduce the number of bits using a base+displacement-based representation. In this representation, a value is represented in base-k using a radix-term b and displacement d via b * k + d. We represent both the base and displacement using Johnson codes. Further, to minimize the misclassification probability around the design boundary for the radix-term, we adapt reflected binary codes (Gray, 1953). We evaluate base-2 (B1JDJ - Figure 2e) and base-4 representations (B2JDJ - Figure 2f).

Binary coded hex - Johnson code (HEXJ): A number k is converted to its hex value in this code, and each hex digit ($0-$F) is encoded separately using an $8-$bit Johnson code. Figure 2g represents examples of quantized labels, their hex values, and encoded values. A 16-bit HEXJ code can represent 256 values, reducing the number of classifiers significantly for a large number of quantization levels.

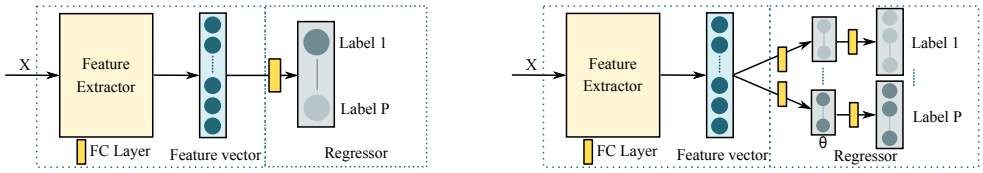
Hadamard code (HAD): Hadamard codes (Wikipedia, a) are widely used as error-correcting codes due to their fault-tolerant properties and also have been used to encode classes in multiclass classification to improve the accuracy and robustness to adversarial attacks (Dietterich & Bakiri, 1995; Verma & Swami, 2019). It requires N bits to encode N values. We evaluate the Hadamard code to show that it is not suitable for regression.

3.3 Design of Decoding Functions

We explore three decoding functions: custom decoding, correlation-based decoding, and expected value-based decoding. Custom decoding functions are specific to the encoding function, and are only evaluated for unary and Johnson codes. In contrast, correlation-based decoding, first explored in prior work studying ECOC for multiclass classification (Allwein et al., 2001), can be applied to all codes. For labels in $\{1, 2, ..., N\}$, we define a code matrix $\boldsymbol{C}$ of size $N \times M$, where M is the number of bits/classifiers used for the binary-encoded label. Each row $\boldsymbol{C}_{k,:}$ in this matrix represents the binary code for label $Q_i = k$. For example, Figure 2b can be considered a code matrix, where the first row represents code for label $Q_i = 1$. Let $\hat{Z}_i \in \mathbb{R}^M$ denote the output logit values of the classifiers. For decoding, the row with the highest correlation with the output $\hat{Z}_i$ is selected as the decoded label. Output logit value vector $\hat{Z}_i$ is used instead of output binary code $\hat{B}_i$ to find the correlation as it uses the confidence of a classifier to make a more accurate prediction. For target quantized labels $Q_i \in 1, 2, ..., N$, the decoding function is defined as:

$$\mathcal{D}^{\text{GEN}}(\hat{Z}_i) = \underset{k \in \{1,2,...,N\}}{\text{argmax}} \left(\hat{Z}_i \cdot \boldsymbol{C}_{k,:} \right) \tag{5}$$

However, $\mathcal{D}^{\text{GEN}}$ outputs a quantized prediction, introducing quantization error. To remedy this concern and demonstrate the potential of more sophisticated decoding rules, we propose and evaluate an expected value-based decoding function, which allows prediction of real-valued label $\hat{y}_i$. For



(a) Direct regression

(b) Binary-encoded labels

Figure 4: Network architecture for direct and binary-encoded label based regression. The feature extractor architecture is not modified for binary-encoded labels.

Table 1: Benchmarks used for evaluation

Task	Feature Extractor	Existing Approach	Dataset	Benchmark	Output range/Quantization levels	θ
Landmark-free 2D head pose estimation	ResNet50	Regression+classification (Ruiz et al., 2018)	BIWI	HPE1	-100-100/200	10
			300LP/AFLW2000	HPE2	-100-100/200	10
	RAFANet	Direct regression (Behera et al., 2021)	BIWI	HPE3	-180-180/360	50
			300LP/AFLW2000	HPE4	-180-180/360	50
Facial Landmark Detection	HRNetV2-W18	Heatmap regression (Wang et al., 2020; Xu et al., 2020)	COFW	FDL1	0-256/256	10
			300W	FDL2	0-256/256	10
			WFLW	FDL3	0-256/256	10
			AFLW	FDL4	0-256/256	30
Age estimation	ResNet50 /ResNet34	Ordinal regression (Cao et al., 2020)	AFAD	AE1	0-64/64	10
			MORPH-II	AE2	0-32/32	10
End-to-end autonomous driving	PilotNet	Direct regression (Bojarski et al., 2017)	PilotNet	PN	0-670/670	10

target labels $y_i \in [1, N]$, the decoding function is defined as:

$$\mathcal{D}^{\text{GEN-EX}}(\hat{Z}_i) = \sum_{k=1}^{N} k\sigma_k, \text{ where } \sigma_k = \frac{e^{\hat{Z}_i \cdot \boldsymbol{C}_{k,:}}}{\sum_{k=1}^{N} e^{\hat{Z}_i \cdot \boldsymbol{C}_{k,:}}} \tag{6}$$

Training loss functions: A deep neural network with multiple output binary classifiers can be trained using the binary cross-entropy (BCE) loss $\mathcal{L}_{\text{BCE}}\big(\hat{Z}_i, \mathcal{E}(Q_i)\big)$. However, this loss minimizes the mismatch between predicted and target code but does not directly minimize the error between the target and predicted values. For decoding function $\mathcal{D}^{\text{GEN}}$, cross-entropy loss $\mathcal{L}_{\text{CE}}\big(\mathcal{C}\hat{Z}_i, Q_i\big)$ can be used for training. $\mathcal{C}$ represents the code matrix for given encoding. Similarly, for decoding function $\mathcal{D}^{\text{GEN-EX}}$, which predicts a continuous value, L1 or L2 loss $\mathcal{L}_{\text{L1/L2}}\big(\mathcal{D}^{\text{GEN-EX}}(\hat{Z}_i), y_i\big)$ can also be used for training. We further evaluate multiple combinations of decoding and loss functions in Section 4.

3.4 Regression Network Architecture for BEL

A deep regression network typically consists of a feature extractor and regressor. Figure 4a represents a network architecture for direct regression. The regressor consists of a fully connected layer between the output of the feature extractor (i.e., feature vector) and output logits. Figure 4b shows the modified neural network architecture for BEL. The number of logits is increased to the number of classifiers (bits) used. The size of the fully connected regression layer increases with the number of output logits. However, empirically, we find small feature vectors suffice as the output logits are highly correlated for the encoding functions we explore. Adding a fully connected layer to reduce feature vector size to θ can reduce the number of parameters and provide a trade-off between the model size and accuracy.

4 Evaluation

Table 1 summarizes the tasks, datasets, and network architectures used for the evaluation of BEL. These tasks are commonly used for evaluation of regression approaches by prior works due to the complexity of problem and network architectures (Díaz & Marathe, 2019). Landmark-free 2D head pose estimation (HPE) aims to find a human head's pose in terms of three angles: yaw, pitch, and roll from a 2D image without landmarks. Facial landmark detection (FDL) is a problem of detecting the (x, y) coordinates of keypoints in a given face image. Age estimation aims to predict the age of a

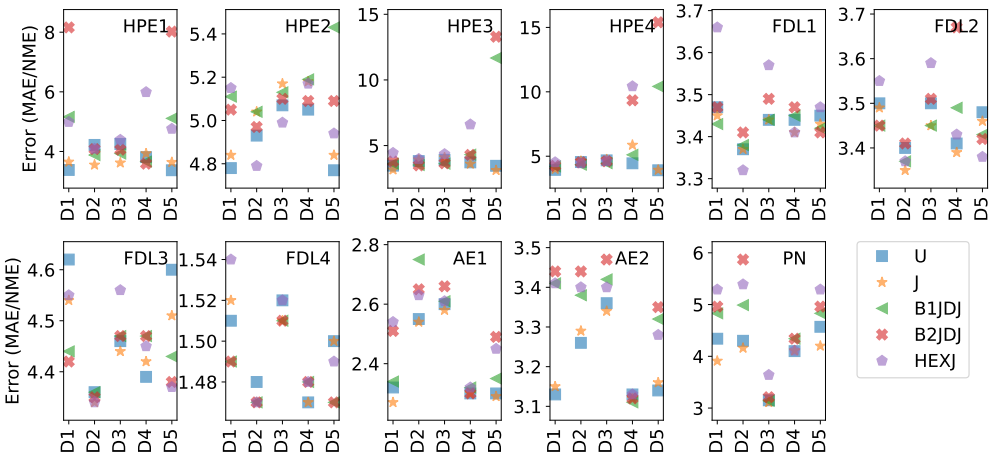


Figure 5: Mean Absolute Error(MAE)/Normalized Mean Error (NME) for different decoding and loss functions for BEL. D1-D5 on the horizontal axis represents different combinations of decoding and loss functions. D1: BCE loss with BEL-U/BEL-J/GEN deocding for U/J/others, D2: CE loss with GEN-EX decoding, D3: CE loss with GEN decoding, D4: L1 or L2 loss with GEN-EX decoding, D5: BCE loss with GEN-EX decoding.

person from an image. In end-to-end autonomous driving, the steering wheel's next angle is predicted from an image of the road. Normalized Mean Error (NME) and Mean Absolute Error (MAE) with resepct to raw real-valued labels are used as the evaluation metric for FDL and the rest, respectively.

Our main goal is to compare the regression approach; hence we keep the feature extractor, data augmentation, evaluation protocols, and the number of training iterations uniform across different methods for each benchmark. We report average of five training runs and error margin of 95% confidence interval. Details on datasets, training parameters, related work for specific tasks, and other evaluation metrics are provided in Appendix D-G. BEL introduces several design parameters for regression by binary classification. We evaluate different encoding ($\mathcal{E}$), decoding ($\mathcal{D}$), and training loss ($\mathcal{L}$) functions for BEL across all the benchmarks as shown in Figure 5. We study if design parameters impact accuracy and analyze the relationship between design parameters and error.

Encoding function ($\mathcal{E}$): Figure 5 plots MAE/NME using different encodings. We do not show results for Hadamard codes in the figure as it results in significantly higher error than other encodings. On average, Hadamard codes result in 60% higher error than J encoding for GEN decoding and CE loss function, which shows that these codes are unsuitable for regression. The results show the encoding function has a significant impact on the accuracy of several benchmarks and the best-performing encoding function varies across datasets and network architectures for a given task (e.g., HPE1 and HPE3). In Section 3.1 we observed that which encoding/decoding functions result in lower error depends upon the error probability distribution of the classifiers. For HPE4, FDL1, and AE1, Johnson does better than unary; we attribute this to misclassification errors occurring more frequently near decision boundaries in these tasks.

The encoding function impacts the number of output classifiers and the complexity of function to be learned by a classifier (i.e., number of decision boundaries). We observe a trade-off between these two parameters. For some benchmarks, the availability of sufficient training data and network capacity facilitates the learning of complex classifiers such as B2JDJ. In such a case, a reduced number of binary classifiers compared to U, J, or B1JDJ codes results in a lower error for B2JDJ. We provide empirical results for the same in Appendix B.

Decoding ($\mathcal{D}$) and training loss ($\mathcal{L}$) functions: We explore three decoding functions (Section 3.3). The training loss functions can be selected from BCE, CE, L1, or L2 loss functions. However, not all the combinations of decoding and loss functions perform well. For example, CE, L1, or L2 losses do not use BEL-J or BEL-U decodings. Therefore, optimizing the network for these losses does not minimize the absolute error between the target and decoded predictions. In contrast, GEN-EX decoding is used for L1/L2 loss function calculation; in such a case, the training loss directly

Table 2: Comparison of BEL with different regression approaches. Existing approaches for each benchmark are described in Table 1

	Error (MAE or NME) / Model size			
	HPE1	HPE2	HPE3	HPE4
Specialized approach	-	6.16 / 23.6M	3.40 / 69.8M	4.14 / 69.8M
Direct regression	4.76 $\pm$0.35 / 23.5M	6.71 $\pm$0.13 / 23.5M	3.40 / 69.8M	4.14 / 69.8M
Multiclass classification	4.49 $\pm$0.24 / 24.2M	5.31 $\pm$0.05 / 24.8M	5.22 $\pm$0.04 / 72.0M	5.14 $\pm$0.08 / 72.0M
BEL	**3.37** $\pm$0.09 / 23.6M	**4.77** $\pm$0.05 / 23.6M	**3.12** $\pm$0.12 / 69.8M	**3.90** $\pm$0.03 / 69.8M
BEL $\mathcal{E}/\mathcal{D}/\mathcal{L}$ functions	U/GEN-EX/BCE	U/GEN-EX/BCE	J/GEN-EX/BCE	U/GEN-EX/BCE
	FDL1	FDL2	FDL3	FDL4
Specialized approach	3.45 / 9.6M	3.32 / 9.6M	**4.32** / 9.6M	1.57 / 9.6M
Direct regression	3.60 $\pm$0.02 / 10.2M	3.54 $\pm$0.03 / 10.2M	4.64 $\pm$0.03 / 10.2M	1.51 $\pm$0.01 / 10.2M
Multiclass classification	3.58 $\pm$0.03 / 25.4M	3.51 $\pm$0.02 / 45.2M	4.50 $\pm$0.01 / 61.3M	1.56 $\pm$0.01 / 20.1M
BEL	**3.34** $\pm$0.02 / 10.6M	**3.36** $\pm$0.02 / 11.2M	4.33 $\pm$0.03 / 11.7M	**1.47** $\pm$0.00 / 10.8M
BEL $\mathcal{E}/\mathcal{D}/\mathcal{L}$ functions	HEXJ/GEN-EX/CE	J/GEN-EX/CE	B2JDJ/GEN-EX/CE	J/GEN-EX/CE
	AE1	AE2	PN	
Specialized approach	2.49 / 21.3M	3.47 / 21.3M	4.24 / 1.8M	
Direct regression	2.44 $\pm$0.01 / 23.1M	3.21 $\pm$0.02 / 23.1M	4.24 $\pm$0.45 / 1.8M	
Multiclass classification	2.75 $\pm$0.03 / 23.1M	3.38 $\pm$0.05 / 23.1M	13.17 $\pm$0.00 / 1.9M	
BEL	**2.27** $\pm$0.01 / 23.1M	**3.11** $\pm$0.00 / 23.1M	**3.11** $\pm$0.01 / 1.8M	
BEL $\mathcal{E}/\mathcal{D}/\mathcal{L}$ functions	J/BEL-J/BCE	B1JDJ/GEN-EX/L1	HEXJ/GEN-EX/BCE	

minimized the error between target values and decoding values. Out of such twelve combinations, we present results for five combinations. Figure 5 compares MAE/NME achieved by different decoding/loss functions combinations across several benchmarks and highlights the range of error variations across decoding/loss functions. $\mathcal{D}^{\text{GEN-EX}}$ results in the lowest error for the majority of the benchmarks as it reduces quantization error and also takes advantage of the output logit confidence values to decode the output code. $\mathcal{D}^{\text{GEN-EX}}$ consistently perform better than $\mathcal{D}^{\text{GEN}}$ function that has been used for multiclass classificaiton by prior works (Allwein et al., 2001). For loss function, we observe that the use of CE or L1/L2 loss results in a lower error with $\mathcal{D}^{\text{GEN-EX}}$ as the training loss function objective minimized the error between decoded labels and target labels. In contrast, BCE loss does not directly minimize the error between decoded and target labels.

Comparison of BEL with regression approaches: Table 2 compares BEL with other approaches for different benchmarks (Table 1). We explore and evaluate multiple combinations of encoding ($\mathcal{E}$), decoding ($\mathcal{D}$), and loss ($\mathcal{L}$) functions for BEL in this work. The best combination of these functions and its error is reported in Table 2. BEL results in lower error than direct regression and multiclass classification and even outperforms task-specific regression approaches for several benchmarks.

The results show no single combination of encoding/decoding/loss functions evaluated was best for all benchmarks but also demonstrate BEL enables accuracy improvements across a range of regression problems.

5 Conclusion

This work proposes binary-encoded labels to pose regression as binary classification. We propose a taxonomy identifying the key design aspects for regression by binary classification and demonstrate the impact of classification error and encoding/decoding functions on the expected label error. Different encoding, decoding, and loss functions are explored to evaluate our approach using four complex regression tasks. Binary-encoded labels result in an average 9.1%, 8.4%, and 12.9% lower error than direct regression, multiclass classification, and task-specific regression approaches, respectively. BEL improves accuracy over state-of-the-art approaches for head pose estimation (BIWI, AFLW2000), facial landmark detection (COFW), age estimation (AFAD), and end-to-end autonomous driving (PilotNet). Our analysis and empirical evaluation in this work demonstrate the potential of the vast design space of BEL for regression problems and the importance of finding suitable design parameters for a given task. The best performing encoding/decoding function pair may be task, dataset, and network specific. A possibility this suggests, which we leave to future work, is that it may be beneficial to develop automated approaches to optimizing these functions,

Reproducibility: We have provided a detailed discussion about training hyperparameters, experimental setup, and modifications made in publicly available network architectures in Appendix D-E for all benchmarks. We will add a comment directed to the reviewers and area chairs to provide a link to our code.

Code of Ethics Some of the major applications of regression problems are artificial intelligence and autonomous machines, and regression improvement can accelerate the development of autonomous systems. However, depending upon the use, autonomous systems can have some negative societal impacts, such as job loss in some sectors and ethical concerns.

REFERENCES

Erin L. Allwein, Robert E. Schapire, and Yoram Singer. Reducing multiclass to binary: A unifying approach for margin classifiers. *J. Mach. Learn. Res.*, 1:113–141, September 2001. ISSN 1532-4435. doi: 10.1162/15324430152733133. URL https://doi.org/10.1162/15324430152733133.

Ardhendu Behera, Zachary Wharton, Pradeep Hewage, and Swagat Kumar. Rotation axis focused attention network (rafa-net) for estimating head pose. In *Computer Vision – ACCV 2020*, pp. 223–240, 2021. ISBN 978-3-030-69541-5.

Axel Berg, Magnus Oskarsson, and Mark O Connor. Deep Ordinal Regression with Label Diversity. *arXiv*, pp. 1–8.

Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D. Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, Xin Zhang, Jake Zhao, and Karol Zieba. End to End Learning for Self-Driving Cars. *arXiv*, pp. 1–9, 2016.

Mariusz Bojarski, Philip Yeres, Anna Choromanaska, Krzysztof Choromanski, Bernhard Firner, Lawrence Jackel, and Urs Muller. Explaining how a deep neural network trained with end-to-end learning steers a car. *arXiv*, pp. 1–8, 2017.

Adrian Bulat and Georgios Tzimiropoulos. Human pose estimation via convolutional part heatmap regression. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling (eds.), *Computer Vision – ECCV 2016*, pp. 717–732, Cham, 2016. Springer International Publishing. ISBN 978-3-319-46478-7.

Xavier Burgos-Artizzu, Pietro Perona, and Piotr Dollár. Robust face landmark estimation under occlusion. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1513–1520, 12 2013. doi: 10.1109/ICCV.2013.191.

Wenzhi Cao, Vahid Mirjalili, and Sebastian Raschka. Rank consistent ordinal regression for neural networks with application to age estimation. *Pattern Recognition Letters*, 140:325–331, 2020. ISSN 0167-8655. doi: https://doi.org/10.1016/j.patrec.2020.11.008.

Jaime S. Cardoso and Joaquim F. Pinto da Costa. Learning to classify ordinal data: The data replication method. *J. Mach. Learn. Res.*, 8:1393–1429, December 2007. ISSN 1532-4435.

Sully Chen. driving-datasets. https://github.com/SullyChen/driving-datasets.

Wei Chu and S. Sathiya Keerthi. New approaches to support vector ordinal regression. In *Proceedings of the 22nd International Conference on Machine Learning*, ICML '05, pp. 145–152, New York, NY, USA, 2005. Association for Computing Machinery. ISBN 1595931805.

Koby Crammer and Yoram Singer. Pranking with ranking. In *Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic*, NIPS'01, pp. 641–647, Cambridge, MA, USA, 2001. MIT Press.

T. G. Dietterich and G. Bakiri. Solving Multiclass Learning Problems via Error-Correcting Output Codes. *Journal of Artificial Intelligence Research*, 2:263–286, 1995. ISSN 1076-9757. doi: 10.1613/jair.105.

Xuanyi Dong, Yan Yan, Wanli Ouyang, and Yi Yang. Style aggregated network for facial landmark detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 379–388, 2018.

Raúl Díaz and Amit Marathe. Soft labels for ordinal regression. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4733–4742, 2019. doi: 10.1109/CVPR.2019.00487.

Gabriele Fanelli, Matthias Dantone, Juergen Gall, Andrea Fossati, and Luc Gool. Random forests for real time 3d face analysis. *International Journal of Computer Vision*, 101(3):437–458, February 2013. ISSN 0920-5691.

Huan Fu, Mingming Gong, Chaohui Wang, Kayhan Batmanghelich, and Dacheng Tao. Deep Ordinal Regression Network for Monocular Depth Estimation. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 2002–2011, 2018. ISSN 10636919.

Bin-Bin Gao, Hong-Yu Zhou, Jianxin Wu, and Xin Geng. Age estimation using expectation of label distribution learning. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pp. 712–718. International Joint Conferences on Artificial Intelligence Organization, 7 2018. doi: 10.24963/ijcai.2018/99. URL https://doi.org/10.24963/ijcai.2018/99.

Frank Gray. Pulse code communication, 1953.

Heng-Wei Hsu, Tung-Yu Wu, Sheng Wan, Wing Hung Wong, and Chen-Yi Lee. Quatnet: Quaternion-based head pose estimation with multiregression loss. *IEEE Transactions on Multimedia*, 21(4): 1035–1046, 2019. doi: 10.1109/TMM.2018.2866770.

Marek Kowalski, Jacek Naruniec, and Tomasz Trzcinski. Deep alignment network: A convolutional neural network for robust face alignment. In *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 2034–2043, 07 2017. doi: 10.1109/CVPRW.2017.254.

Abhinav Kumar, Tim K. Marks, Wenxuan Mou, Ye Wang, Michael Jones, Anoop Cherian, Toshiaki Koike-Akino, Xiaoming Liu, and Chen Feng. LUVLi face alignment: Estimating Landmarks' location, uncertainty, and visibility likelihood. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 8233–8243, 2020. ISSN 10636919. doi: 10.1109/CVPR42600.2020.00826.

M. Köstinger, P. Wohlhart, P. M. Roth, and H. Bischof. Annotated facial landmarks in the wild: A large-scale, real-world database for facial landmark localization. In *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pp. 2144–2151, 2011.

Ling Li and Hsuan-Tien Lin. Ordinal regression by extended binary classification. In *Proceedings of the 19th International Conference on Neural Information Processing Systems*, pp. 865–872, 2006.

William Libaw and Leonard Craig. A photoelectric decimal-coded shaft digitizer. *Electronic Computers, Transactions of the I.R.E. Professional Group on*, EC-2:1 – 4, 10 1953.

J. Lv, X. Shao, J. Xing, C. Cheng, and X. Zhou. A deep regression architecture with two-stage re-initialization for high performance facial landmark detection. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3691–3700, 2017. doi: 10.1109/CVPR.2017.393.

X. Miao, X. Zhen, X. Liu, C. Deng, V. Athitsos, and H. Huang. Direct shape regression networks for end-to-end face alignment. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5040–5049, 2018.

S. S. Mukherjee and N. M. Robertson. Deep head pose: Gaze-direction estimation in multimodal video. *IEEE Transactions on Multimedia*, 17(11):2094–2107, 2015.

Zhenxing Niu, Mo Zhou, Le Wang, Xinbo Gao, and Gang Hua. Ordinal regression with multiple output CNN for age estimation. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016-Decem:4920–4928, 2016. ISSN 10636919.

Hongyu Pan, Hu Han, Shiguang Shan, and Xilin Chen. Mean-variance loss for deep age estimation from a face. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5285–5294, 2018. doi: 10.1109/CVPR.2018.00554.

Sebastian Raschka. MLxtend: Providing machine learning and data science utilities and extensions to Python's scientific computing stack. *Journal of Open Source Software*, 3(24):638, April 2018. ISSN 2475-9066. doi: 10.21105/joss.00638. URL https://joss.theoj.org/papers/10.21105/joss.00638.

K. Ricanek and T. Tesafaye. Morph: a longitudinal image database of normal adult age-progression. In *7th International Conference on Automatic Face and Gesture Recognition (FGR06)*, pp. 341–345, 2006. doi: 10.1109/FGR.2006.78.

Rasmus Rothe, Radu Timofte, and Luc Van Gool. DEX: Deep EXpectation of Apparent Age from a Single Image. *Proceedings of the IEEE International Conference on Computer Vision*, 2015-Febru: 252–257, 2015. ISSN 15505499.

Nataniel Ruiz, Eunji Chong, and James M. Rehg. Fine-grained head pose estimation without keypoints. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2018.

C. Sagonas, G. Tzimiropoulos, S. Zafeiriou, and M. Pantic. 300 faces in-the-wild challenge: The first facial landmark localization challenge. In *2013 IEEE International Conference on Computer Vision Workshops*, pp. 397–403, 2013.

Shizhan Zhu, Cheng Li, C. C. Loy, and X. Tang. Face alignment by coarse-to-fine shape searching. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4998–5006, 2015.

Yang Song, Qiyu Kang, and Wee Peng Tay. Error-Correcting Output Codes with Ensemble Diversity for Robust Learning in Neural Networks. *Aaai*, 2021.

Y. Sun, X. Wang, and X. Tang. Deep convolutional network cascade for facial point detection. In *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3476–3483, 2013a.

Y. Sun, X. Wang, and X. Tang. Deep convolutional network cascade for facial point detection. In *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3476–3483, 2013b. doi: 10.1109/CVPR.2013.446.

G. Tzimiropoulos. Project-out cascaded regression with an application to face alignment. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3659–3667, 2015.

Gunjan Verma and Ananthram Swami. Error correcting output codes improve probability estimation and adversarial robustness of deep neural networks. *Advances in Neural Information Processing Systems*, 32(NeurIPS), 2019. ISSN 10495258.

Jingdong Wang, Ke Sun, Tianheng Cheng, Borui Jiang, Chaorui Deng, Yang Zhao, Dong Liu, Yadong Mu, Mingkui Tan, Xinggang Wang, Wenyu Liu, and Bin Xiao. Deep high-resolution representation learning for visual recognition. *IEEE transactions on pattern analysis and machine intelligence*, PP, April 2020. ISSN 0162-8828.

Xinyao Wang, Liefeng Bo, and Fuxin Li. Adaptive wing loss for robust face alignment via heatmap regression. In *2019 IEEE International Conference on Computer Vision (ICCV)*, pp. 6970–6980, 2019. doi: 10.1109/ICCV.2019.00707.

Wikipedia. Hadamard code. https://en.wikipedia.org/wiki/Hadamard_code, a.

Wikipedia. Unary coding. https://en.wikipedia.org/wiki/Unary_coding, b.

W. Wu and S. Yang. Leveraging intra and inter-dataset variations for robust face alignment. In *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 2096–2105, 2017.

W. Wu, Chen Qian, S. Yang, Q. Wang, Y. Cai, and Qiang Zhou. Look at boundary: A boundary-aware face alignment algorithm. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2129–2138, 2018.

X. Xiong and F. De la Torre. Supervised descent method and its applications to face alignment. In *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 532–539, 2013.

Zixuan Xu, Banghuai Li, Miao Geng, Ye Yuan, and Gang Yu. Anchorface: An anchor-based facial landmark detector across large poses. *ArXiv*, abs/2007.03221, 2020.

Tsun Yi Yang, Yi Ting Chen, Yen Yu Lin, and Yung Yu Chuang. Fsa-net: Learning fine-grained structure aggregation for head pose estimation from a single image. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2019-June:1087–1096, 2019. ISSN 10636919.

Jie Zhang, Shiguang Shan, Meina Kan, and Xilin Chen. Coarse-to-fine auto-encoder networks (cfan) for real-time face alignment. pp. 1–16, 09 2014. ISBN 978-3-319-10604-5. doi: 10.1007/978-3-319-10605-2_1.

S. Zhu, C. Li, C. C. Loy, and X. Tang. Unconstrained face alignment via cascaded compositional learning. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3409–3417, 2016. doi: 10.1109/CVPR.2016.371.

Xiangyu Zhu, Zhen Lei, Xiaoming Liu, Hailin Shi, and Stan Li. Face alignment across large poses: A 3d solution. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 146–155, 06 2016.

A ABLATION STUDY

Impact of quantization and decoding functions: Table 3 compares decoding functions equation 5 and equation 6 that predict quantized and smooth value respectively for different number of quantization intervals. This analysis aims to study the impact of quantization levels on accuracy and the use of expected value-based decoding ($\mathcal{D}^{GEN-EX}$) to reduce quantization error. As shown in Table 3, the use of ($\mathcal{D}^{GEN-EX}$) consistently results in lower error compared to ($\mathcal{D}^{GEN}$) for all quantization levels, where D^{GEN-EX} and $\mathcal{D}^{GEN}$ predict continuous and quantized value, respectively. Moreover, the error is lower for 128 quantization levels than 256 as the improvement resulting from fewer binary classifiers is higher than the increase in the quantization error. However, as quantization levels are decreased further, the advantage of using fewer classifiers is diminished by high quantization error. Thus there exists a trade-off between the number of binary classifiers and quantization error.

Table 3: Impact of the quantization and decoding functions on NME for facial landmark detection.

	COFW			300W		
Quantization levels	64	128	256	64	128	256
$\mathcal{E}^{\text{BEL-U}} + \mathcal{D}^{\text{GEN}}$	3.66	3.51	3.46	3.79	3.59	3.46
$\mathcal{E}^{\text{BEL-U}} + \mathcal{D}^{\text{GEN-EX}}$	3.46	3.41	3.44	3.54	3.47	3.44
$\mathcal{E}^{\text{BEL-J}} + \mathcal{D}^{\text{GEN}}$	3.65	3.49	3.43	3.76	3.58	3.46
$\mathcal{E}^{\text{BEL-J}} + \mathcal{D}^{\text{GEN-EX}}$	3.45	**3.40**	3.42	3.52	3.45	**3.43**

Effect of the number of training samples on BEL: We further analyze the effect of the number of training samples on the suitable encoding. The encoding functions explored in this work have varying complexity. Using a more complex encoding such as B2JDJ or HEXJ reduces the number of classifiers; however, it increases each classifier's complexity (i.e., the number of decision boundaries to be learned). Table 4 compares the error for different training dataset sizes and encodings for facial landmark detection benchmarks. Empirical results suggest that simpler encodings (e.g., J) are more suitable than complex encodings as the training dataset size decreases.

Table 4: Effect of training dataset size on optimal encoding function for facial landmark detection. BCE loss function and GEN-EX decoding function are used for the training and evaluation.

Encoding	#Classi-fiers/label	#Decision bound-aries/classifier	Reduction in the number training samples						
			0%	20%	40%	60%	80%	90%	95%
			COFW						
U	256	1	3.45	3.48	3.55	3.72	3.94	4.52	6.29
J	128	2	3.43	3.48	3.51	3.61	**3.88**	**4.32**	**5.39**
B1JDJ	65	4	3.42	**3.44**	3.52	**3.60**	4.11	4.50	5.68
B2JDJ	34	8	**3.41**	3.45	**3.48**	3.80	3.94	4.80	6.56
Hex	17	32	3.47	3.69	3.78	4.03	4.61	5.48	6.69
			300W						
U	256	1	3.48	3.55	3.58	3.64	3.89	4.26	5.66
J	128	2	3.46	3.56	3.52	3.58	**3.79**	**4.04**	**4.58**
B1JDJ	65	4	3.43	3.48	3.53	3.61	3.89	4.31	6.10
B2JDJ	34	8	3.42	**3.47**	**3.51**	**3.54**	3.88	4.50	5.80
Hex	17	32	**3.38**	3.64	3.73	3.97	4.41	5.38	6.60
			WFLW						
U	256	1	4.60	4.67	4.83	5.00	5.37	6.04	7.46
J	128	2	4.51	4.60	4.65	4.84	5.23	**5.64**	**6.39**
B1JDJ	65	4	4.43	**4.44**	4.52	4.66	5.08	5.90	8.39
B2JDJ	34	8	4.38	4.46	**4.49**	**4.61**	**5.02**	5.95	8.78
Hex	17	32	**4.37**	4.60	4.72	4.96	5.72	6.86	8.09
			AFLW						
U	256	1	1.50	1.53	1.53	1.56	1.61	1.68	1.83
J	128	2	1.50	1.51	1.52	1.54	1.60	1.68	1.79
B1JDJ	65	4	**1.47**	**1.50**	1.52	1.54	1.60	1.67	1.78
B2JDJ	34	8	**1.47**	**1.50**	**1.50**	**1.52**	**1.57**	**1.64**	**1.73**
Hex	17	32	1.49	1.54	1.54	1.55	1.59	1.71	1.89

Table 5: Effect of reflected binary conversion for B1JDJ encoding on facial landmark detection (BCE loss and GEN-EX decoding functions are used)

	COFW	300W	WFLW	AFLW
B1JDJ	3.43	3.46	4.43	1.47
B1JDJ-NRB	4.13	4.43	5.70	1.97

Table 6: Comparison of BEL with heatmaps for facial landmark detection (BCE loss and GEN-EX decoding functions are used)

	COFW	300W	WFLW	AFLW
BEL-U	3.45	3.46	4.60	1.50
BEL-J	3.48	3.46	4.51	1.50
BEL-H-U	4.13	4.43	5.70	1.97
BEL-H-J	10.17	33.02	22.50	2.99

Effect of reflected binary conversion: We use reflected binary to increase the distance between distant labels based on the design properties of suitable regression encodings we proposed. Table 5 shows the impact of using reflected binary conversion on error for facial landmark detection benchmarks. As shown in the table, the use of reflected binary significantly reduces the error.

Use of binary heamaps: Facial landmark detection approaches typically use heatmap regression. We also evaluate BEL-H-x, in which the real-valued heatmaps are converted to binary heatmaps with 8 quantization levels. Table 6 shows the impact of using reflected binary conversion on error for facial landmark detection benchmarks. In BEL-H-x, each real-valued pixel of the target heatmap is converted to a binary code using eight quantization levels. For unary code, a 64×64 real-valued heatmap of one facial landmark is converted to eight 64x64 binary heatmaps, resulting in $32,768$ $(8 \times 64 \times 64)$ binary classifiers compared to 512 for BEL-U. We believe that training a high number of binary classifiers results in poor accuracy for BEL-H-x.

Hyperparameter θ: We introduce a hyperparameter θ in the proposed learning approach, where θ denotes the size of the feature vector before the output layer. A lower value of θ can be used for encodings with a high correlation between the classifiers' output. Increasing the value of θ decreases the error but increases the number of network parameters. Thus there exists a trade-off between the number of model parameters and error. Figure 6 compares the decrease in the error for different encodings and θ values. We observe that more complex encodings that show lower correlation between the classifiers' outputs benefit more from an increase in the θ value.

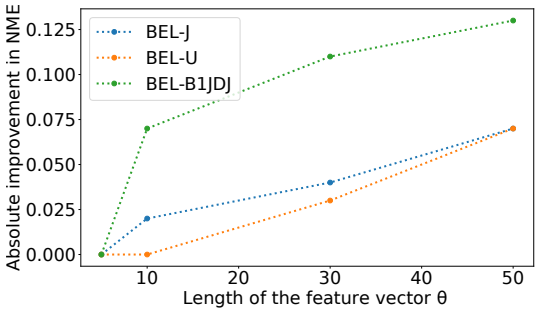


Figure 6: Effect of θ on error for different encodings (benchmark FDL1)

B Expected Error Derivation

This section explains the expected error equations used to compare BEL-U and BEL-J in Section 3.1. First we explain the decoding function used for BEL-U and derive the relation between the expected

error of labels and binary classifiers. Lastly, we explain the encoding/decoding functions and expected error relation for BEL-J.

B.1 PRELIMINARIES

Figure 7 represent the training and inference flow for binary-encoded labels. Here, $y_i \in \mathbb{R}$ represent the target label for sample i. Q_i represent the target quantized level, which is converted to the target binary code $B_i \in \{0,1\}^M$ using an encoding function $\mathcal{E}$. $C^1, ..., C^M$ binary classifiers are used in the proposed approach. $\hat{B}^i \in \{0,1\}^M$ is the predicted binary code, which is then decoded to $\hat{y}_i \in \mathbb{R}$ using a decoding function $\mathcal{D}$.

For the analysis in this section, we consider target label $y_i \in [1, N-1]$ and the quantized target label $Q_i \in \{1, 2, ..., N-1\}$. The decoding functions used in this analysis predict the quantized label, which introduces quantization error. We do not include quantization error in the expected label error for our analysis as it is constant for BEL-U and BEL-J. The expected value of absolute error between the target y and predicted labels $\hat{y}$ is used for the analysis as typically mean absolute error is used as the evaluation metric in regression problems.

$\hat{b}_i^k = T$ indicates correct binary classification by classifier C^k ($\hat{b}_i^k == b_i^k$) for sample i. $\hat{b}_i^k = F$ indicates incorrect binary classification by classifier C^k ($\hat{b}_i^k \neq b_i^k$) for sample i. Let $e_k(n)$ denote the error probability of a binary classifier C^k, when target quantized label $Q_i = n$ and $B_i = b_i^1, b_i^2, ..., b_i^M = \mathcal{E}(n)$.

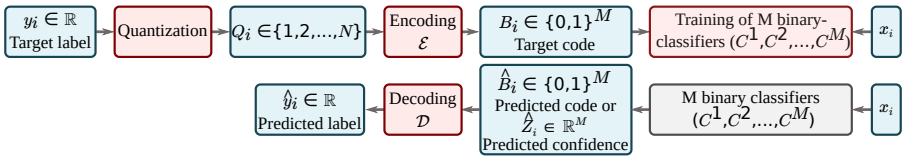
$$\begin{aligned} e_k(n) &= \mathbb{E}(|\hat{b}_i^k - b_i^k|) \\ &= Pr(\hat{b}_i^k = F) \text{ for target label } Q_i = n \text{ and target binary code } B_i = \mathcal{E}(Q_i) \end{aligned} \tag{7}$$

B.2 EXPECTED ERROR FOR BEL-U

Encoding and decoding functions The encoding and decoding functions for BEL-U are defined as:

$$\mathcal{E}^{\text{BEL-U}}(Q_i) = b_i^1, b_i^2, .., b_i^{N-2}, \text{ where } b_i^k = \begin{cases} 1 & k < Q_i \\ 0 & \text{Otherwise} \end{cases} \tag{8}$$

$$\mathcal{D}^{\text{BEL-U}}(\hat{b}_i^1, \hat{b}_i^2, ..., \hat{b}_i^{N-2}) = \sum_{k=1}^{N-2} \hat{b}_i^k + 1 \tag{9}$$



(a) Training and inference flow

Figure 7: The training (top) and inference (bottom) flow of binary-encoded labels for deep regression. Red colored blocks represent design aspects we focus on.

Expected error For target quantized label $Q_i = n$ ($n \in \{1, 2, ..., N-1\}$), ignoring the quantization error, the expected error between target y_i and predicted label $\hat{y}_i$ can be derived as:

$$
\begin{aligned}
\mathbb{E}(|\hat{y}_i - y_i|) &= \mathbb{E}\left(\Big|\big(\sum_{k=1}^{N-2} \hat{b}_i^k + 1\big) - \big(\sum_{k=1}^{N-2} b_i^k + 1\big)\Big|\right) \\
&= \mathbb{E}\left(\Big|\sum_{k=1}^{N-2} (\hat{b}_i^k - b_i^k)\Big|\right) \\
&\leqslant \mathbb{E}\left(\sum_{k=1}^{N-2} |\hat{b}_i^k - b_i^k|\right) \\
&= \sum_{k=1}^{N-2} \mathbb{E}\,|\hat{b}_i^k - b_i^k| \\
&= \sum_{k=1}^{N-2} e_k(n) \text{ (using equation 7)}
\end{aligned}
\tag{10}
$$

For a uniform distribution of target labels in the range $[1, N-1]$, the expected error can be derived as:

$$
\mathbb{E}(|\hat{y} - y|) \leqslant \frac{1}{N-1} \sum_{n=1}^{N-1} \sum_{k=1}^{N-2} e_k(n) \tag{11}
$$

B.3 Expected Error for BEL-J

Encoding and decoding functions For target quantized label $Q_i \in \{1, 2, ..., N-1\}$, BEL-J encoding requires $\frac{N}{2}$ bits/binary classifiers. The encoding for BEL-J can be defined as:

$$
\mathcal{E}^{\text{BEL-J}}(Q_i) = b_i^1, b_i^2, .., b_i^{\frac{N}{2}} \text{,where } b_i^k = \begin{cases} 1 & \frac{N}{2} - Q_i < k \leqslant N - Q_i \\ 0 & \text{Otherwise} \end{cases} \tag{12}
$$

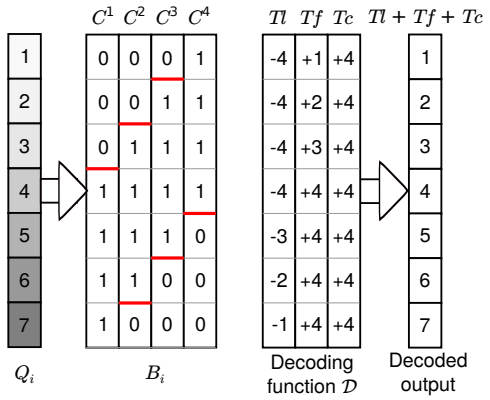


Figure 8: Encoding and Decoding functions' output for BEL-J approach and label $y \in [1, 7], N = 8$. For decoding function's output $y' = Tl + Tf + Tc$, $Tl = -\max_{k \in \{1 \ldots \frac{N}{2}\}} k\hat{b}_i^k$, $Tf = \max_{k \in \{1 \ldots \frac{N}{2}\}} \left(\frac{N}{2} - k + 1\right)\hat{b}_i^k$, and $Tc = \frac{N}{2}$

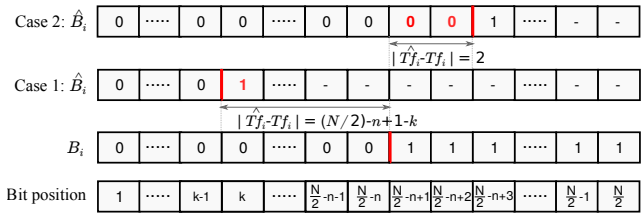


Figure 9: Effect of classifier error on $\hat{Tf}_i - Tf_i$ for label $Q_i = n$. Case 1 and case 2 represent erroneous outputs. $0/1$ highlighted in red color represents an error in the classifier's output. "-" represents error/no error in both cases.

Similarly, the decoding functions for BEL-J can be defined as:

$$\begin{aligned} \mathcal{D}^{\text{BEL-J}}(\hat{B}_i) &= Tl(\hat{B}_i) + Tf(\hat{B}_i) + Tc \\ \text{where, } Tl(\hat{B}_i) &= -\max_{k \in \{1 \ldots \frac{N}{2}\}} k\hat{b}_i^k \\ Tf(\hat{B}_i) &= \max_{k \in \{1 \ldots \frac{N}{2}\}} \left(\frac{N}{2} - k + 1\right)\hat{b}_i^k, Tc = \frac{N}{2} \end{aligned} \tag{13}$$

In equation 13, $Tl()$ finds the location of the last occurrence of "1" in the predicted binary code $\hat{B}_i$. Similarly, $Tf()$ finds the location of the first occurrence of "1" in the binary code $\hat{B}_i$. Figure 8 gives examples of binary codes for label $Q_i \in \{1, 2, ..., 7\}$ and the corresponding values of the different terms in equation 13. For example, for label $Q_i = 3$, the binary code is "0111". Here, the last occurrence of "1" is at position 4, and $Tl = -4$. Similarly, the first occurrence of "1" is at position 2, and $Tf = (4 + 1) - 2 = 3$.

Expected error For BEL-J code, binary classifiers $(C^1, C^2, ..., C^{\frac{N}{2}})$ are used. For a given input sample i, an error in any of the binary classifiers' outputs $(\hat{b}_i^1, \hat{b}_i^2, ..., \hat{b}^{\frac{N}{2}})$ will result in an error between $Tf(\hat{B}_i)/Tl(\hat{B}_i)$ and $Tf(B_i)/Tl(B_i)$ in equation 13. We refer to $Tf(\hat{B}_i)$ and $Tl(\hat{B}_i)$ as $\hat{Tf}_i$ and $\hat{Tl}_i$ (predicted binary code), and $Tf(B_i)$ and $Tl(B_i)$ as Tf_i and Tl_i (target binary code) for brevity. Expected value of the absolute error can be further expanded as:

$$\begin{aligned} \mathbb{E}(|\hat{y}_i - y_i|) &= \mathbb{E}(|\hat{Tf}_i + \hat{Tl}_i + Tc - (Tf_i + Tl_i + Tc)|) \\ &= \mathbb{E}(|(\hat{Tf}_i - Tf_i) + (\hat{Tl}_i - Tl_i)|) \\ &\leqslant \mathbb{E}(|\hat{Tf}_i - Tf_i| + |\hat{Tl}_i - Tl_i|) \\ &= \mathbb{E}(|\hat{Tf}_i - Tf_i|) + \mathbb{E}(|\hat{Tl}_i - Tl_i|) \end{aligned} \tag{14}$$

Thus, the sum of the expected error of $Tf()$ and $Tl()$ is the upper bound of the label's expected error. Further, we derive the relation between binary classifiers' error probabilities and $\mathbb{E}(|\hat{Tf}_i - Tf_i|)$ and $\mathbb{E}(|\hat{Tl}_i - Tl_i|)$.

We consider $Q_i = n$, where$1 \leqslant n \leqslant \frac{N}{2}$ for our derivation. In such a case, $Tf_i = n$ and $Tl_i = -\frac{N}{2}$. However, as the code is symmetric around $Q_i = \frac{N}{2}$, it can be shown that the derived equation for $\mathbb{E}\,|\hat{y}_i - y_i|$ can be used for $1 \leqslant Q_i \leqslant N - 1$.

1. Derivation of $\mathbb{E}\,|\hat{Tf}_i - Tf_i|$: As shown in equation 13, $Tf()$ finds the location k of the first occurrence of "1" in the binary sequence. In the case of an erroneous binary sequence, the position of the first occurrence of "1" might shift, which results in an error between $\hat{Tf}_i$ and Tf_i. Figure 9 shows examples of the correct and erroneous outputs of classifiers for label $Q_i = n$. For label $Q_i = n$, $b_i^k = 0$ for $k \in \{1, 2, ..., \frac{N}{2} - n\}$ and $b_i^k = 1$ for $k \in \{\frac{N}{2} - n + 1, \frac{N}{2} - n + 2, ..., \frac{N}{2}\}$.

For case 1, error in a classifier $C^k, k \in \{1, 2, ..., \frac{N}{2} - n\}$ is considered, where $b_i^k = 0$ and $\hat{b}_i^k = 1$. For $k \in \{1, 2, ..., \frac{N}{2} - n\}$, an error at classifier C^k will result in erroneous $\hat{Tf}_i$ only if all proceeding classifiers are correct, since if any of the proceeding classifier z is incorrect, i.e. $\hat{b}_i^z = 1$, then the

location of the first occurrence of "1" will be shifted to z, and any error in the following classifiers will not affect the value of $\hat{Tf}$. Such a case ($\hat{b}_i^1 = T, \hat{b}_i^2 = T, ..., \hat{b}_i^{k-1} = T, \hat{b}_i^k = F, \hat{b}_i^{k+1} = T/F, ..., \hat{b}_i^{\frac{N}{2}} = T/F$) considers a total of $2^{\frac{N}{2}-k}$ combinations out of $2^{\frac{N}{2}}$ for $k \in \{1, 2, ..., \frac{N}{2} - n\}$. Assuming that the binary classifiers are mutually independent, the error value and the probability of this combination can be shown to be:

$$|\hat{Tf}_i - Tf_i| = \Big(\frac{N}{2} - n + 1 - k\Big) \tag{15}$$

$$\begin{aligned} Pr(\hat{b}_i^1 = T, \hat{b}_i^2 = T, ..., \hat{b}_i^{k-1} = T, \hat{b}_i^k = F) &= Pr(\hat{b}^1 = T)Pr(\hat{b}_i^2 = T)...Pr(\hat{b}_i^{k-1} = T)Pr(\hat{b}^k = F) \\ &= \Big(\prod_{j=1}^{k-1}(1 - e_j(n))\Big) \cdot e_k(n) \end{aligned} \tag{16}$$

The above term considers combinations ($b'^1 = T, b'^2 = T, ..., b'^{k-1} = T, b'^k = F, b'^{k+1} = T/F, ..., b'^{\frac{N}{2}=T/F}$) for $k \in \{1, 2, ..., \frac{N}{2} - n\}$, which constitutes to a total of $\sum_{k=1}^{\frac{N}{2}-n} 2^{\frac{N}{2}-k}$ combinations out of $2^{\frac{N}{2}}$.

For case 2, error in a classifier $C^k, k \in \{\frac{N}{2} - n + 1, \frac{N}{2} - n + 2, ..., \frac{N}{2}\}$ is considered, where $b_i^k = 1$ and $\hat{b}_i^k = 0$. We consider a combination ($\hat{b}_i^1 = T, \hat{b}_i^2 = T, ..., \hat{b}_i^{\frac{N}{2}-n} = T, \hat{b}_i^{\frac{N}{2}-n+1} = F, ..., \hat{b}_i^{k-1} = F, \hat{b}_i^k = T, \hat{b}_i^{k+1} = T/F, ..., \hat{b}_i^{\frac{N}{2}=T/F}$). For this case, the position of the first occurrence of "1" will be moved to k, which will result in erroneous $\hat{Tf}_i$. Such a case would cover $2^{\frac{N}{2}-k}$ combinations out of $2^{\frac{N}{2}}$ for $k \in \{\frac{N}{2} - n + 1, \frac{N}{2} - n + 2, ..., \frac{N}{2}\}$. The error value and the probability of this combination can be shown to be:

$$|\hat{Tf}_i - Tf_i| = \Big(k - (\frac{N}{2} - n + 1)\Big) \tag{17}$$

$$\begin{aligned} &Pr(\hat{b}_i^1 = T, \hat{b}_i^2 = T, ..., \hat{b}_i^{\frac{N}{2}-n} = T, \hat{b}_i^{\frac{N}{2}-n+1} = F, ..., \hat{b}_i^{k-1} = F, \hat{b}_i^k = T) = \\ &\Big(\prod_{j=1}^{\frac{N}{2}-n}(1 - e_j(n))\Big) \cdot \Big(\prod_{j=\frac{N}{2}-n+1}^{k-1} e_j(n)\Big) \cdot \Big(1 - e_k(n)\Big) \end{aligned} \tag{18}$$

The above term considers combinations ($\hat{b}_i^1 = T, \hat{b}_i^2 = T, ..., \hat{b}_i^{\frac{N}{2}-n} = T, \hat{b}_i^{\frac{N}{2}-n+1} = F, ..., \hat{b}_i^{k-1} = F, \hat{b}_i^k = T, \hat{b}_i^{k+1} = T/F, ..., \hat{b}_i^{\frac{N}{2}=T/F}$), which constitutes to a total of $\sum_{k=\frac{N}{2}-n+1}^{\frac{N}{2}} 2^{\frac{N}{2}-k}$ combinations out of $2^{\frac{N}{2}}$ for $k \in \{\frac{N}{2} - n + 1, \frac{N}{2} - n + 2, ..., \frac{N}{2}\}$.

Combining equation 15-equation 18, the expected value of $|\hat{Tf}_i - Tf_i|$ can be derived as:

$$\begin{aligned} \mathbb{E}(|\hat{Tf}_i - Tf_i|) = &\sum_{k=1}^{\frac{N}{2}-n} \Big(\frac{N}{2} - n + 1 - k\Big) \cdot \Big(\prod_{j=1}^{k-1}(1 - e_j(n))\Big) \cdot e_k(n) \\ &+ \sum_{k=\frac{N}{2}-n+1}^{\frac{N}{2}} \Big(k - (\frac{N}{2} - n + 1)\Big) \cdot \Big(\prod_{j=1}^{\frac{N}{2}-n}(1 - e_j(n))\Big) \cdot \Big(\prod_{j=\frac{N}{2}-n+1}^{k-1}(e_j(n))\Big) \cdot \Big(1 - e_k(n)\Big) \\ = &\sum_{k=1}^{\frac{N}{2}-n} \Big(\frac{N}{2} - n + 1 - k\Big) \cdot e_k(n) \cdot \Big(\prod_{j=1}^{k-1}(1 - e_j(n))\Big) + \sum_{k=\frac{N}{2}-n+1}^{\frac{N}{2}} \Big(\prod_{j=\frac{N}{2}-n+1}^{k} e_j(n)\Big) \end{aligned} \tag{19}$$

The first term in equation 19 covers $\sum_{k=1}^{\frac{N}{2}-n} 2^{\frac{N}{2}-k}$ combinations and the second term considers $\sum_{k=\frac{N}{2}-n+1}^{\frac{N}{2}} 2^{\frac{N}{2}-k}$ combinations. Adding one combination where all the classifiers are correct, equation 19 considers all of the possible combinations $2^{\frac{N}{2}}$ to find expected value of $|\hat{Tf}_i - Tf_i|$.

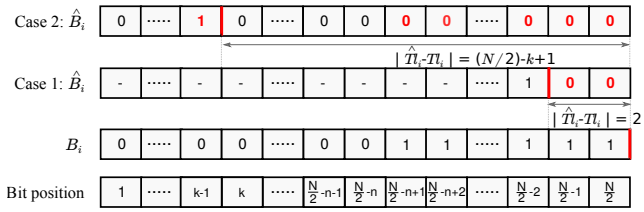


Figure 10: Effect of classifier error on $\hat{T}l_i - Tl_i$ for label $Q_i = n$. Case 1 and case 2 represent erroneous outputs. $0/1$ highlighted in red color represents an error in the classifier's output. "-" represents error/no error in both cases.

2. Derivation of $\mathbb{E}\,|\hat{T}l_i - Tl_i|$: As shown in equation 13, $Tl()$ finds the location k of the last occurrence of "1" in the binary sequence. In the case of an erroneous binary sequence, the position of the last occurrence of "1" might shift, which results in an erroneous value of $\hat{T}l_i$. Figure 10 shows examples of correct and erroneous outputs of classifiers for label $Q_i = n$.

For case 1, an error in a classifier $C^k, k \in \{\frac{N}{2} - n + 1, \frac{N}{2} - n + 2, ..., \frac{N}{2}\}$ is considered, where $b_i^k = 1$ and $\hat{b}_i^k = 0$. We consider a combination $(\hat{b}_i^{\frac{N}{2}} = F, \hat{b}_i^{\frac{N}{2}-1} = F, ..., \hat{b}_i^{k+1} = F, \hat{b}_i^k = T, \hat{b}_i^{k-1} = T/F, ..., \hat{b}_i^1 = T/F)$. For this case, position of the last occurrence of "1" will be moved to k, which will result in erroneous $\hat{T}l_i$. Such a case would cover 2^{k-1} combinations out of $2^{\frac{N}{2}}$. The error value and the probability of this combination can be shown to be:

$$|\hat{T}l_i - Tl_i| = \Big(\frac{N}{2} - k\Big) \tag{20}$$

$$Pr(\hat{b}_i^{\frac{N}{2}} = F, \hat{b}_i^{\frac{N}{2}-1} = F, ..., \hat{b}_i^{k+1} = F, \hat{b}_i^k = T) = \Big(\prod_{j=k+1}^{\frac{N}{2}} e_j(n)\Big) \cdot (1 - e_k(n)) \tag{21}$$

The above term considers combinations $(\hat{b}_i^{\frac{N}{2}} = F, \hat{b}_i^{\frac{N}{2}-1} = F, ..., \hat{b}_i^{k+1} = F, \hat{b}_i^k = T, \hat{b}_i^{k-1} = T/F, ..., \hat{b}_i^1 = T/F)$ for $k \in \{\frac{N}{2} - n + 1, \frac{N}{2} - n + 2, ..., \frac{N}{2}\}$, which constitutes to a total of $\sum_{k=\frac{N}{2}-n+1}^{\frac{N}{2}} 2^{k-1}$ combinations out of $2^{\frac{N}{2}}$.

For case 2, an error in a classifier $C^k, k \in \{1, 2, ..., \frac{N}{2} - n\}$ is considered, where $b_i^k = 0$ and $\hat{b}_i^k = 1$. We consider a combination $(\hat{b}_i^{\frac{N}{2}} = F, ..., \hat{b}_i^{\frac{N}{2}-n+1} = F, \hat{b}_i^{\frac{N}{2}-n} = T, ..., \hat{b}_i^{k+1} = T, \hat{b}_i^k = F, \hat{b}_i^{k-1} = T/F, ..., \hat{b}_i^1 = T/F)$. For this case, position of the last occurrence of "1" will be moved to k, which will result in erroneous $\hat{T}l_i$. Such a case would cover 2^{k-1} combinations out of $2^{\frac{N}{2}}$. The error value and the probability of this combination can be shown to be:

$$|\hat{T}l_i - Tl_i| = \Big(\frac{N}{2} - k\Big) \tag{22}$$

$$\begin{aligned} Pr(\hat{b}_i^{\frac{N}{2}} = F, ..., \hat{b}_i^{\frac{N}{2}-n+1} = F, \hat{b}_i^{\frac{N}{2}-n} = T, ..., \hat{b}_i^{k+1} = T, \hat{b}_i^k = F) = \\ \Big(\prod_{j=\frac{N}{2}-n+1}^{\frac{N}{2}} e_j(n)\Big) \cdot \Big(\prod_{j=k+1}^{\frac{N}{2}-n} (1 - e_j(n))\Big) \cdot (e_k(n)) \end{aligned} \tag{23}$$

The above term considers combinations $(\hat{b}_i^{\frac{N}{2}} = F, ..., \hat{b}_i^{\frac{N}{2}-n+1} = F, \hat{b}_i^{\frac{N}{2}-n} = T, ..., \hat{b}_i^{k+1} = T, \hat{b}_i^k = F, \hat{b}_i^{k-1} = T/F, ..., \hat{b}_i^1 = T/F)$ for $k \in \{1, 2, ..., \frac{N}{2} - n\}$, which constitutes to a total of $\sum_{k=1}^{\frac{N}{2}-n} 2^{k-1}$ combinations out of $2^{\frac{N}{2}}$.

Combining equation 20-equation 23, the expected value of $|\hat{T}l_i - Tl_i|$ can be derived as:

$$\begin{aligned}\mathbb{E}(|\hat{T}l_i - Tl_i|) = &\sum_{k=\frac{N}{2}-n+1}^{\frac{N}{2}} \Big(\frac{N}{2}-k\Big)\cdot\Big(\prod_{j=k+1}^{\frac{N}{2}} e_j(n)\Big)\cdot(1-e_k(n)) \\ &+ \sum_{k=1}^{\frac{N}{2}-n} \Big(\frac{N}{2}-k\Big)\cdot\Big(\prod_{j=\frac{N}{2}-n+1}^{\frac{N}{2}} e_j(n)\Big)\cdot\Big(\prod_{j=k+1}^{\frac{N}{2}-n}(1-e_j(n))\Big)\cdot(e_k(n)) \quad (24) \\ &= \sum_{k=\frac{N}{2}-n+1}^{\frac{N}{2}} \Big(\prod_{j=k}^{\frac{N}{2}} e_j(n)\Big) + \Big(\prod_{j=\frac{N}{2}-n+1}^{\frac{N}{2}} e_j(n)\Big)\cdot\sum_{k=1}^{\frac{N}{2}-n}\Big(\prod_{j=k}^{\frac{N}{2}-n}(1-e_j(n))\Big)\end{aligned}$$

The first term in equation 24 covers $\sum_{k=\frac{N}{2}-n+1}^{\frac{N}{2}} 2^{k-1}$ combinations and the second term considers $\sum_{k=1}^{\frac{N}{2}-n} 2^{k-1}$ combinations. Adding one combination where all the classifiers are correct, equation 24 considers all of the possible combinations $2^{\frac{N}{2}}$ to find expected value of $|\hat{T}l_i - Tl_i|$.

Combining equation 14, equation 19, and equation 24, the expected value of average error in terms of classifiers' error probabilities can de derived as:

$$\begin{aligned}\mathbb{E}(\hat{y}_i - y_i) \leqslant &\sum_{k=1}^{\frac{N}{2}-n} \Big(\frac{N}{2}-n+1-k\Big)\cdot e_k(n)\cdot\Big(\prod_{j=1}^{k-1}(1-e_j(n))\Big) + \sum_{k=\frac{N}{2}-n+1}^{\frac{N}{2}}\Big(\prod_{j=\frac{N}{2}-n+1}^{k} e_j(n)\Big) \\ &+ \sum_{k=\frac{N}{2}-n+1}^{\frac{N}{2}} \Big(\prod_{j=k}^{\frac{N}{2}} e_j(n)\Big) + \Big(\prod_{j=\frac{N}{2}-n+1}^{\frac{N}{2}} e_j(n)\Big)\cdot\sum_{k=1}^{\frac{N}{2}-n}\Big(\prod_{j=k}^{\frac{N}{2}-n}(1-e_j(n))\Big) \quad (25)\end{aligned}$$

As the binary code is symmetric around $\frac{N}{2}$ as shown in Figure 8, the expected errors for label $y_i \in [1, \frac{N}{2}]$ can be mirrored to find expected errors for label $y_i \in [\frac{N}{2}, N-1]$. For a uniform distribution of target labels in the range $[1, N-1]$, the expected error can be derived as:

$$\begin{aligned}\mathbb{E}(\hat{y}-y) \leqslant &\frac{1}{N-1}\sum_{n=1}^{N-1}\Big[\sum_{k=1}^{\frac{N}{2}-n}\Big(\frac{N}{2}-n+1-k\Big)\cdot e_k(n)\cdot\Big(\prod_{j=1}^{k-1}(1-e_j(n))\Big) + \sum_{k=\frac{N}{2}-n+1}^{\frac{N}{2}}\Big(\prod_{j=\frac{N}{2}-n+1}^{k} e_j(n)\Big) \\ &+ \sum_{k=\frac{N}{2}-n+1}^{\frac{N}{2}} \Big(\prod_{j=k}^{\frac{N}{2}} e_j(n)\Big) + \Big(\prod_{j=\frac{N}{2}-n+1}^{\frac{N}{2}} e_j(n)\Big)\cdot\sum_{k=1}^{\frac{N}{2}-n}\Big(\prod_{j=k}^{\frac{N}{2}-n}(1-e_j(n))\Big)\Big] \quad (26)\end{aligned}$$

We also verify the equation by comparing the expected value of error based on equation 25 for $Q_i \in 1, 2, ..., N-1$ with the expected error calculated by $100,000$ random samples of binary sequences for the same error probabilities $e_k(n)$. Figure 11 compares the expected error from equation 25 and measured from statistical samples, and validates error upper bounds calculated using equation 25 and equation 26.

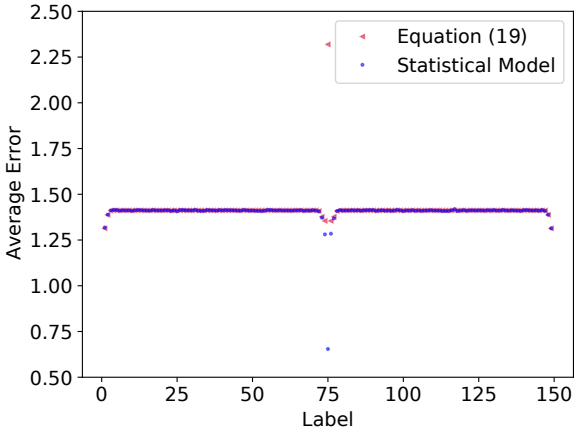
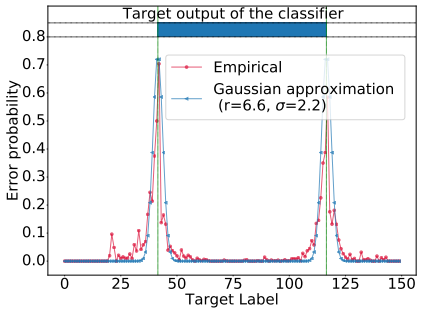


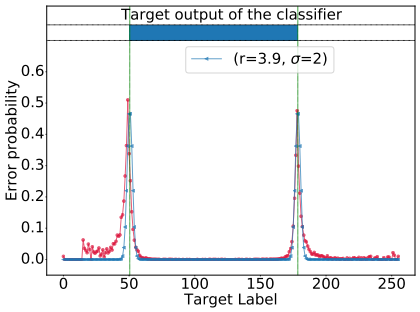
Figure 11: Comparison of expected value of error from equation 25 and random samples for given error probabilities of the classifiers.

C Error Probability of Classifiers

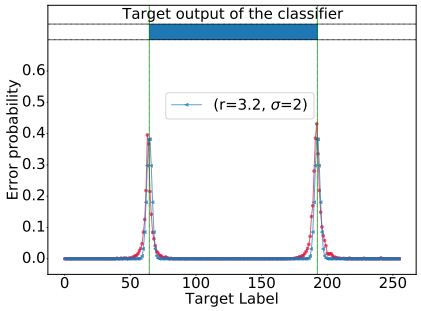
It is known that the error/misclassification probability $e_k(n)$ of a classifier tends to increase as the target label value n is closer to the classifier's decision boundaries Cardoso & Pinto da Costa (2007). We approximate $e_k(y)$ for a classifier C^k with t decision boundaries as a linear combination of t Gaussian distributions. Here, each Gaussian term is centered around a decision boundary. Figure 12 shows the empirically observed error probability distributions for different classifiers trained for different combinations of network and dataset. We also show the approximate error probability distribution using a linear combination of Gaussian distributions. Here r is a scalar multiplied with probability density of gaussian distribution and σ^2 is the variance (equation 3 and equation 4).



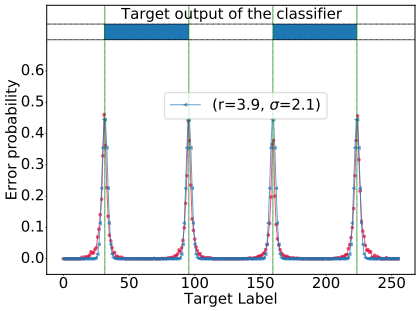
(a) ResNet50 (BIWI head pose estimation)



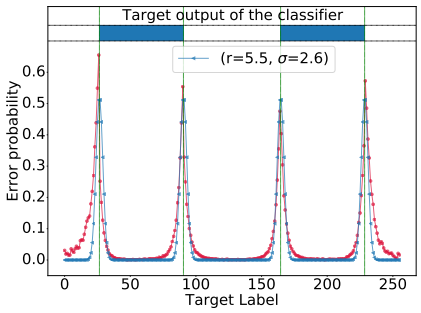
(b) HRNetV2-W18 (AFLW facial landmark detection)



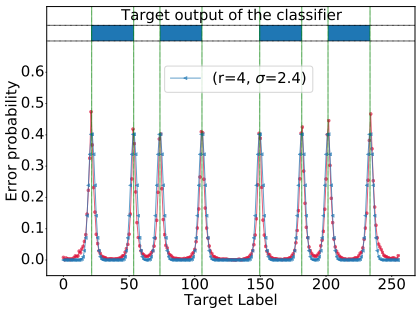
(c) HRNetV2-W18 (300W facial landmark detection)



(d) HRNetV2-W18 (300W facial landmark detection)



(e) HRNetV2-W18 (WFLW facial landmark detection)



(f) HRNetV2-W18 (WFLW facial landmark detection)

Figure 12: Classification error probability versus target label y for different classifiers. The top horizontal bar represent target output of the classifier. Blue color represents output 1.

D HEAD POSE ESTIMATION

Head pose estimation aims to find a human head's pose in terms of three angles: yaw, pitch, and roll. In this work, we consider landmark-free 2D head pose estimation. We use two evaluation protocols, Protocol-1 (Train dataset: 300W-LP (Zhu et al., 2016), Test dataset: AFLW2000 (Köstinger et al., 2011)) and Protocol-2 (70-30% train-test split of BIWI dataset (Fanelli et al., 2013)) with mean absolute error (MAE) as the evaluation metric.

Head pose estimation aims to find a human head's pose in terms of three angles: yaw, pitch, and roll. This work considers landmark-free 2D head pose estimation, which performs the task based on RGB image information without using facial landmark information.

D.1 EXPERIMENTAL SETUP

We use two evaluation protocols with three widely used datasets: 300W-LP (Zhu et al., 2016), BIWI (Fanelli et al., 2013), AFLW2000 (Zhu et al., 2016) and follow the evaluation setting of Hopenet (Ruiz et al., 2018) and FSA-Net (Yang et al., 2019). Mean absolute error (MAE) is used as the evaluation metric and the individual yaw, pitch, and roll and overall MAE are evaluated. NVIDIA 2080Ti GPU is used for all the experiments. Bold numbers represent the lowest error for the same feature extractor and data augmentation, and the underlined numbers represent the lowest error overall. Evaluation metrics for BEL and direct regression are averaged over five training runs with random seeds. We further explain two protocols in detail.

Bold numbers represent the lowest error for the same feature extractor and data augmentation, and the underlined numbers represent the lowest error overall. Evaluation metrics for BEL and direct regression are averaged over five training runs with random seeds.

Protocol 1: In this setting, the synthetic 300W-LP dataset is used for training, consisting of $122,450$ samples. The trained network is tested on a real-world AFLW2000 dataset. Yaw, pitch, and roll angles are in the range $[-99, 99]$ for both datasets. Table 7 provides a summary of the training parameters. For head pose estimation with ResNet50 architecture (Direct regression and BEL), two runs with

Table 7: Training parameters for head pose estimation with protocol 1.

Approach	Optimizer	Epochs	Batch size	Learning rate	Learning rate schedule	Training time (GPU hours)
ResNet50 with direct regression and BEL	Adam, weight decay=0, momentum = 0	20	16	0.00001	1/10 after 10 Epochs	4
RAFA-Net with direct regression (Behera et al., 2021) and BEL	RMSProp, momentum=0, rho = 0.9	100	16	0.001	-	48

different random seeds for each combination of learning rate $\{0.001, 0.0001, 0.00001\}$ and batch size $\{8, 16\}$ are used for hyperparameter tuning. For data augmentation, images are loosely cropped around the center in the training and testing datasets with random flipping and blurring. Samples with angles in the range [$-99°$,$99°$] are considered for training and testing. 200 quantization levels for binary-encoded labels.

For head pose estimation with RAFA-Net, we use the training parameters used in (Behera et al., 2021). For data augmentation, we use the approach used in the previous work on RAFA-Net for head pose estimation using direct regression. Angles in the range [$-180°$,$180°$] with 360 quantization levels are considered for BEL.

Protocol 2: BIWI dataset is used for training and evaluation in this protocol. BIWI dataset consists of 24 videos of 20 subjects with total $15,128$ frames. Three random splits of 70%-30% images are used for training and evaluation. Table 8 provides a summary of the training parameters.

For head pose estimation with ResNet50 architecture (Direct regression and BEL), two runs with different random seeds for each combination of learning rate $\{0.001, 0.0001, 0.00001\}$ and batch size $\{8, 16\}$ are used for hyperparameter tuning. For data augmentation, images are loosely cropped around the center in the training and testing datasets with random flipping. For the BIWI dataset, the yaw angle is in the range $[-75, 75]$, the pitch is in the range $[-65, 85]$, and the roll angle is in the range $[-55, 45]$. Total 150 quantization levels are used for yaw and pitch, and 100 quantization levels are used for roll.

For head pose estimation with RAFA-Net, we use the training parameters used in (Behera et al., 2021). For data augmentation, we use the approach used in the previous work on RAFA-Net for head pose estimation using direct regression. Angles in range [$-180°$,$180°$] with 360 quantization levels are considered for BEL.

Table 8: Training parameters for head pose estimation with protocol 1.

Approach	Optimizer	Epochs	Batch size	Learning rate	Learning rate schedule	Training time (GPU hours)
ResNet50 with direct regression and BEL	Adam, weight decay=0, momentum = 0	50	8	0.0001	1/10 after 30 Epochs	2
RAFA-Net with direct regression (Behera et al., 2021) and BEL	RMSProp, momentum=0, rho = 0.9	100	16	0.001	-	6

Table 9: Comparison of BEL design parameters on MAE for head pose estimation with ResNet50 feature extractor.

Decoding function	Loss function	Encoding function					
		U	J	B1JDJ	B2JDJ	HEXJ	HAD
BEL-J/BEL-U	BCE	4.78	4.84	-	-	-	-
GEN-EX	BCE	**4.77**	4.84	5.43	5.09	4.94	7.84
GEN	BCE	4.78	4.87	5.11	5.05	5.15	8.54
GEN-EX	CE	4.93	5.04	5.04	4.97	4.79	5.64
GEN	CE	5.07	5.17	5.13	5.10	4.99	5.62
GEN-EX	L1	5.05	5.18	5.19	5.09	5.17	5.07

D.2 Related work

Existing approaches for head pose estimation includes stage-wise soft regression (Rothe et al., 2015; Yang et al., 2019), combination of classification and regression (Mukherjee & Robertson, 2015; Ruiz et al., 2018), and ordinal regression (Hsu et al., 2019). SSR-Net (Rothe et al., 2015) proposed the use of stage-wise soft regression that uses softmax values of classification output to refine the label. FSA-Net (Yang et al., 2019) proposes extending stage-wise estimation to head pose estimation using feature aggregation. HopeNet (Ruiz et al., 2018) uses a combination of classification and regression loss to train a model for head pose estimation. Whereas QuatNet (Hsu et al., 2019) solves head pose estimation using a combination of regression L2 loss and ordinal regression loss. RAFA-Net (Behera et al., 2021) uses an attention based approach for feature extraction with direct regression.

D.3 Evaluation

Protocol 1 (HPE2 and HPE4) Table 9 compares different combinations of encoding, deocding, and loss functions for BEL approach on protocol 1 and ResNet50 feature extractor. Table 10 compares different combinations of encoding, deocding, and loss functions for BEL approach on protocol 1 and RAFA-Net feature extractor. Table 11 shows results for protocol 1 with error margin for 95%

Table 10: Comparison of BEL design parameters on MAE for head pose estimation with RAFA-Net feature extractor.

Decoding function	Loss function	Encoding function					
		U	J	B1JDJ	B2JDJ	HEXJ	HAD
BEL-J/BEL-U	BCE	3.94	4.00	-	-	-	-
GEN-EX	BCE	**3.90**	3.93	10.42	15.40	29.11	9.17
GEN	BCE	3.93	3.94	4.21	4.25	4.53	9.21
GEN-EX	CE	4.55	4.62	4.34	4.53	4.45	5.12
GEN	CE	4.68	4.75	4.46	4.61	4.63	5.29
GEN-EX	L1	4.45	5.87	5.11	9.34	10.43	17.89

confidence interval. For ResNet50 feature extractor, BEL-U/BCE/GEN-EX provides the lowest MAE compared to direct regression and HopeNet (Ruiz et al., 2018). Whereas, for RAFA-Net feature extractor, BEL-U/BCE/GEN-EX provides the lowest MAE compared to direction regression (L2 loss).

Table 11: Landmark-free 2D Head poses estimation evaluation for protocol 1. 200 and 360 quantization levels are used for BEL with ResNet50 and RAFA-Net.

Approach	Feature Extractor	#Params (M)	Yaw	Pitch	Roll	MAE
SSR-Net-MD (Rothe et al., 2015) (Soft regression)	SSR-Net	1.1	5.14	7.09	5.89	6.01
FSA-Caps-Fusion (Yang et al., 2019) (Soft regression)	FSA-Net	5.1	4.50	6.08	4.64	5.07
HopeNet* (α = 2) (Ruiz et al., 2018) (classification + regression loss)	ResNet50	23.9	6.47	6.56	5.44	6.16
Direct regression (L2 loss)	ResNet50	23.5	6.34	6.90	6.87	6.71 $\pm$ 0.13
BEL-U/BCE/GEN-EX	ResNet50	23.6	**4.54**	**5.76**	**3.96**	**4.77** $\pm$ 0.05
RAFA-Net (Behera et al., 2021) (Direct Regression)	RAFA-Net	69.8	3.60	4.92	3.88	4.13
BEL-U/BCE/GEN-EX	RAFA-Net	69.8	**3.28**	**4.78**	**3.55**	**3.90** $\pm$ 0.03

$*$uses different data preprocessing

Protocol 2 (HPE1 and HPE3) Table 12 compares different combinations of encoding, deocding, and loss functions for BEL approach on protocol 2 and ResNet50 feature extractor. Table 13 compares different combinations of encoding, deocding, and loss functions for BEL approach on protocol 2 and RAFA-Net feature extractor. Table 14 shows results for protocol 2 with error margin for 95% confidence interval. For ResNet50 feature extractor, BEL-U/BCE/GEN-EX provides the lowest MAE compared to direct regression (L2 loss). Whereas, for RAFA-Net feature extractor, BEL-J/BCE/GEN-EX provides the lowest MAE compared to direction regression (L2 loss). For both the feature extractors, BEL results in significant decrease in the MAE for pitch compared to direct regression.

Table 12: Comparison of BEL design parameters on MAE for head pose estimation with ResNet50 feature extractor.

Decoding function	Loss function	Encoding function					
		U	J	B1JDJ	B2JDJ	HEXJ	HAD
BEL-J/BEL-U	BCE	3.38	3.65	-	-	-	-
GEN-EX	BCE	**3.37**	3.64	5.11	8.02	4.76	7.53
GEN	BCE	3.38	3.65	5.16	8.16	4.99	7.73
GEN-EX	CE	4.22	3.55	3.88	4.08	4.09	5.50
GEN	CE	4.25	3.62	3.93	4.06	4.39	5.48
GEN-EX	L1	3.81	3.93	3.66	3.59	5.99	4.21

Table 13: Comparison of BEL design parameters on MAE for head pose estimation with RAFA-Net feature extractor.

Decoding function	Loss function	Encoding function					
		U	J	B1JDJ	B2JDJ	HEXJ	HAD
BEL-J/BEL-U	BCE	3.47	3.16	-	-	-	-
GEN-EX	BCE	3.46	**3.12**	11.67	13.29	25.42	5.75
GEN	BCE	3.49	3.14	3.62	3.78	4.44	5.83
GEN-EX	CE	3.82	3.91	3.52	3.49	3.98	3.98
GEN	CE	3.92	4.09	3.62	3.65	4.35	4.28
GEN-EX	L1	3.72	3.60	4.31	4.29	6.61	18.69

Table 14: Landmark-free 2D Head poses estimation evaluation for protocol 2. 150 and 360 quantization levels are used for BEL with ResNet50 and RAFA-Net.

Approach	Feature Extractor	#Params (M)	Yaw	Pitch	Roll	MAE
SSR-Net-MD (Rothe et al., 2015) (Soft regression)	SSR-Net	1.1	4.24	4.35	4.19	4.26
FSA-Caps-Fusion (Yang et al., 2019) (Soft regression)	FSA-Net	5.1	2.89	4.29	3.60	3.60
Direct regression (L2 loss)	ResNet50	23.5	4.62	5.24	4.43	4.76 ± 0.35
BEL-U/BCE/GEN-EX	ResNet50	23.6	**3.28**	**3.36**	**3.44**	$\mathbf{3.37} \pm 0.09$
RAFA-Net (Behera et al., 2021) (Direct Regression)	RAFA-Net	69.8	3.07	4.30	2.82	3.40
BEL-J/BCE/GEN-EX	RAFA-Net	69.8	**2.83**	**3.82**	**2.70**	$\mathbf{3.12} \pm 0.12$

E FACIAL LANDMARK DETECTION

Facial landmark detection is a problem of detecting the (x, y) coordinates of keypoints in a given face image. Normalized mean error (NME) and failure rate (f@8% and f@10%) are used as evaluation metrics. The state-of-the-art approach with HRNetV2-W18 (Wang et al., 2020) uses heatmap regression, which minimizes the pixel-level loss between the predicted and target heatmaps. The target heatmap is generated by Gaussian distribution around the landmark location. BEL-x predicts of (x, y) coordinates directly with 256 quantization levels.

E.1 EXPERIMENTAL SETUP

We use the COFW (Burgos-Artizzu et al., 2013), 300W (Sagonas et al., 2013), WFLW (Wu et al., 2018), and AFLW (Köstinger et al., 2011) datasets for the evaluation with data augmentation and evaluation protocol used in (Wang et al., 2020). Normalized mean error (NME) and failure rate (f@10%) are used as evaluation metrics. Inter-ocular distance for normalization is used for the COFW, 300W, and WFLW datasets and the bounding box-based normalization is used for the AFLW dataset. Further, 256 quantization levels are used for binary-encoded labels. Data augmentation is performed by random flipping, $0.75 - 1.25$ scaling, and ± 30 degrees in-plane rotation for all the datasets.

COFW The COFW dataset (Burgos-Artizzu et al., 2013) consists of $1,345$ training and 507 testing images. Each image is annotated with 29 facial landmarks.

300W This dataset is a combination of HELEN, LFPW, AFW, XM2VTS, and IBUG datasets. Each image is annotated with 68 facial landmarks. The training dataset consists of $3,148$ images. We

evaluate the trained model on four test sets: full test set with 689 images, common subset with 554 images from HELEN and LFPW, challenging subset with 135 images from IBUG, the official test set with 300 indoor and 300 outdoor images.

WFLW WFLW dataset consists of $7,500$ training images where each image is annotated with 98 facial landmarks. Full test dataset consists of $2,500$ images. We use test subsets: large pose (326 images), expression (314 images), illumination (698 images), make-up (206 images), occlusion (736 images) and blur (773 images).

AFLW Each image has 19 annotated facial key points in this dataset. AFLW dataset consists of $20,000$ training images where each image is annotated with 19 facial landmarks. The full test dataset consists of $4,836$ images, and the frontal test set consists of $1,314$ images.

Training parameters Two runs with different random seeds for a learning rate to decide the learning rate. We considered learning rate $\{0.0003, 0.0005, 0.0007\}$. $\theta \in 10, 30$ is considered for all the codes, and the value is selected for a trade-off between the NME and the model size as a higher value of θ increases the model size. Table 15 provides a summary of all the training parameters.

Table 15: Training parameters for facial landmark detection for HRNetV2-W18 feature extractor and direct regression and BEL regression.

Dataset	Optimizer	Epochs	Batch size	Learning rate (BEL/Direct regression)	Learning rate schedule	Training time (GPU hours)
COFW	Adam, weight decay=0, momentum = 0	60	8	0.0005/0.0003	1/10 after 30 and 50 Epochs	$\frac{1}{2}$
300W	Adam, weight decay=0, momentum = 0	60	8	0.0007/0.0003	1/10 after 30 and 50 Epochs	3
WFLW	Adam, weight decay=0, momentum = 0	60	8	0.0003/0.0003	1/10 after 30 and 50 Epochs	5
AFLW	Adam, weight decay=0, momentum = 0	60	8	0.0005/0.0005	1/10 after 30 and 50 Epochs	8

Network Architecture HRNetV2-W18 feature extractor's output is 240 channels of size 64×64. For heatmap regression, a 1×1 convolution is used to get P heatmaps of size 64×64, where P is the number of landmarks. BEL-x predicts of (x, y) coordinates directly with 256 quantization levels. Figure 13 shows the modified architecture of HRNetV2-W18 for BEL-x.

E.2 Related work

Facial landmark detection is an extensively studied problem used for facial analysis and modeling. Common regression approaches for this tasks includes regression using MSE loss (Xiong & De la Torre, 2013; Lv et al., 2017), cascaded regression (Miao et al., 2018; Tzimiropoulos, 2015; Zhu et al., 2016; Sun et al., 2013b), and coarse to fine regression (Sun et al., 2013a; Shizhan Zhu et al., 2015; Zhang et al., 2014). State-of-the-art methods for this task learns heatmaps by regression to find facial landmarks. SAN (Dong et al., 2018) augments training data using temporal information and GAN-generated faces. DVLN (Wu & Yang, 2017), CFSS (Shizhan Zhu et al., 2015), LAB (Wu et al., 2018), DSRN (Miao et al., 2018) take advantage of correlations between facial landmarks. DAN (Kowalski et al., 2017) introduces a progressive refinement approach using predicted landmark heatmaps. LAB (Wu et al., 2018) also exploits extra boundary information to improve the accuracy. LUVLi (Kumar et al., 2020) proposes a landmark's location, uncertainty, and visibility likelihood-based loss. Bulat & Tzimiropoulos (2016) proposes the use of binary heatmaps with pixel-wise

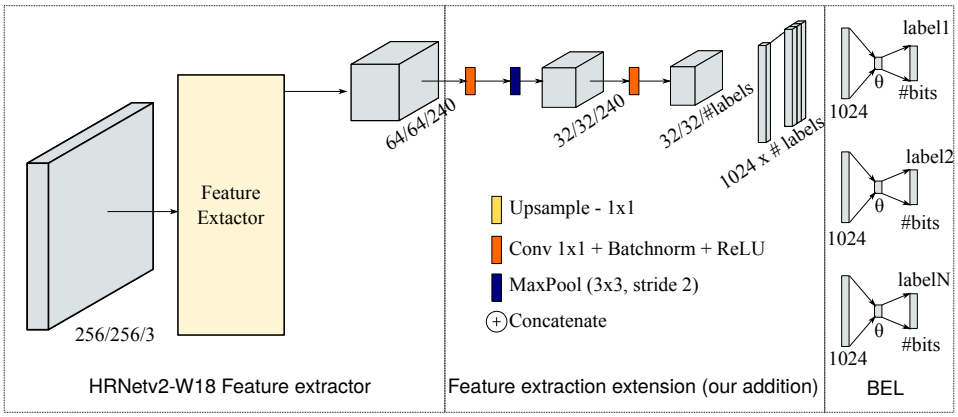


Figure 13: HRNetV2-W18 feature extractor combined with BEL regressor for (x,y) coordinates

Table 16: Comparison of BEL design parameters on NME for facial landmark detection COFW datasets.

Decoding function	Loss function	Encoding function					
		U	J	B1JDJ	B2JDJ	HEXJ	HAD
BEL-J/BEL-U	BCE	3.47	3.45	-	-	-	-
GEN-EX	BCE	3.45	3.43	3.42	3.41	3.47	4.28
GEN	BCE	3.46	3.45	3.43	3.47	3.66	4.43
GEN-EX	CE	3.37	3.37	3.38	3.41	**3.34**	3.69
GEN	CE	3.44	3.44	3.44	3.49	3.57	3.69
GEN-EX	L1	3.44	3.41	3.45	3.47	3.41	4.52

binary cross-entropy loss. AWing (Wang et al., 2019) proposes adapted wing loss to improve the accuracy of heatmap regression. AnchorFace (Xu et al., 2020) demonstrates that anchoring facial landmarks on templates improves regression performance for large poses. HRNet (Wang et al., 2020) proposes a CNN architecture to maintain high-resolution representations across the network, and uses heamap regression.

E.3 Evaluation

COFW (FDL1) Table 16 compares different combinations of encoding, deocding, and loss functions for BEL approach on COFW dataset. Table 17 compares different approaches for COFW dataset. BEL-HEXJ/CE/GEN-EX results in the lowest error.

Table 17: Facial landmark detection results on COFW dataset. The failure rate is measured at the threshold 0.1. $\theta = 30$ is used for BEL.

Approach	Feature Extractor	#Params/ GFlops	Test NME	$FR_{0.1}$
LAB (w B) (Wu et al., 2018)	Hourglass	25.1/19.1	3.92	0.39
AWing (Wang et al., 2019)*	Hourglass	25.1/19.1	4.94	-
HRNetV2-W18 (Wang et al., 2020) (Heatmap regression)	HRNetV2-W18	9.6/4.6	3.45	0.19
Direct regression (L2 loss)	HRNetV2-W18	10.2/4.7	3.96 ± 0.02	0.29
Direct regression (L1 loss)	HRNetV2-W18	10.2/4.7	3.60 ± 0.02	0.29
BEL-HEXJ/CE/GEN-EX	HRNetV2-W18	10.6/4.6	$\underline{\mathbf{3.34}} \pm 0.02$	0.40

$*$Uses different data augmentation for the training

Table 18: Comparison of BEL design parameters on NME for facial landmark detection 300W datasets.

Decoding function	Loss function	Encoding function					
		U	J	B1JDJ	B2JDJ	HEXJ	HAD
BEL-J/BEL-U	BCE	3.5	3.49	-	-	-	-
GEN-EX	BCE	3.48	3.46	3.43	3.42	3.38	4.71
GEN	BCE	3.50	3.49	3.45	3.45	3.55	4.78
GEN-EX	CE	3.40	**3.36**	3.37	3.41	3.37	3.62
GEN	CE	3.50	3.45	3.45	3.51	3.59	3.65
GEN-EX	L1	3.41	3.39	3.49	3.67	3.43	4.04

300W (FDL2) Table 18 compares different combinations of encoding, deocding, and loss functions for BEL approach on COFW dataset. Table 19 compares different approaches for 300W dataset.approaches.

Table 19: Facial landmark detection results on 300W dataset. $\theta = 10$ is used for BEL.

Approach	Feature Extractor	#Params/ GFlops	Test	Common	Challenging	Full
DAN (Kowalski et al., 2017)	-	-	-	3.19	5.24	3.59
LAB (w B) (Wu et al., 2018)	Hourglass	25.1/19.1	-	2.98	5.19	3.49
AnchorFace (Xu et al., 2020)	ShuffleNet-V2	-	-	3.12	6.19	3.72
AWing (Wang et al., 2019)*	Hourglass	25.1/19.1	-	2.72	4.52	3.07
LUVLi (Kumar et al., 2020)	CU-Net	-	-	2.76	5.16	3.23
HRNetV2-W18 (Wang et al., 2020) (Heatmap regression)	HRNetV2-W18	9.6/4.6	-	2.87	**5.15**	**3.32**
Direct regression (L2 loss)	HRNetV2-W18	10.2/4.7	4.40	3.25	5.65	3.71 ± 0.05
Direct regression (L1 loss)	HRNetV2-W18	10.2/4.7	4.26	3.10	5.42	3.54 ± 0.03
BEL-J/CE/GEN-EX	HRNetV2-W18	11.2/4.6	**3.94**	**2.87**	5.35	3.36 ± 0.02

$*$Uses different data augmentation for the training

WFLW (FDL3) Table 20 compares different combinations of encoding, deocding, and loss functions for BEL approach on COFW dataset. Table 21 compares different approaches for WFLW dataset.

Table 20: Comparison of BEL design parameters on NME for facial landmark detection WFLW datasets.

Decoding function	Loss function	Encoding function					
		U	J	B1JDJ	B2JDJ	HEXJ	HAD
BEL-J/BEL-U	BCE	4.62	4.54	-	-	-	-
GEN-EX	BCE	4.6	4.51	4.43	4.38	4.37	7.18
GEN	BCE	4.62	4.53	4.44	4.42	4.55	7.14
GEN-EX	CE	4.36	4.34	4.36	**4.33**	4.34	5.15
GEN	CE	4.46	4.44	4.47	4.47	4.56	4.83
GEN-EX	L1	4.39	4.42	4.47	4.47	4.45	4.74

Table 21: Facial landmark detection results (NME) on WFLW test and 6 subsets: pose, expression (expr.), illumination (illu.), make-up (mu.), occlusion (occu.) and blur. $\theta = 10$ is used for BEL.

Approach	Feature Extractor	#Params/ GFlops	Test	Pose	Expr.	Illu.	MU	Occu.	Blur
LAB (w B) (Wu et al., 2018)	Hourglass	25.1/19.1	5.27	10.24	5.51	5.23	5.15	6.79	6.32
AnchorFace (Xu et al., 2020)*	HRNetV2-W18	-/5.3	4.32	7.51	4.69	4.20	4.11	4.98	4.82
AWing (Wang et al., 2019)*	Hourglass	25.1/19.1	4.36	7.38	4.58	4.32	4.27	5.19	4.96
LUVLi (Kumar et al., 2020)	CU-Net	-	4.37	-	-	-	-	-	-
HRNetV2-W18 (Wang et al., 2020) (Heatmap regression)	HRNetV2-W18	9.6/4.6	4.60	7.94	4.85	4.55	4.29	5.44	5.42
Direct regression (L2 loss)	HRNetV2-W18	10.2/4.7	5.56 ± 0.05	10.17	6.13	5.49	5.29	6.83	6.52
Direct regression (L1 loss)	HRNetV2-W18	10.2/4.7	4.64 ± 0.03	8.13	4.96	4.49	4.45	5.41	5.25
BEL-B2JDJ/CE/GEN-EX	HRNetV2-W18	11.7/4.6	$\mathbf{4.33} \pm 0.03$	**7.38**	**4.60**	**4.25**	**4.10**	**5.10**	**5.01**

$*$Uses different data augmentation for the training

Table 22: Comparison of BEL design parameters on NME for facial landmark detection AFLW datasets.

Decoding function	Loss function	Encoding function					
		U	J	B1JDJ	B2JDJ	HEXJ	HAD
BEL-J/BEL-U	BCE	1.51	1.52	-	-	-	-
GEN-EX	BCE	1.50	1.50	**1.47**	**1.47**	1.49	1.52
GEN	BCE	1.51	1.52	1.50	1.49	1.54	1.55
GEN-EX	CE	1.48	**1.47**	**1.47**	**1.47**	**1.47**	**1.47**
GEN	CE	1.52	1.51	1.51	1.51	1.52	1.52
GEN-EX	L1	**1.47**	**1.47**	1.48	1.48	1.48	1.59

AFLW (FDL4) Table 22 compares different combinations of encoding, deocding, and loss functions for BEL approach on COFW dataset. Table 23 compares different approaches for AFLW dataset.

Table 23: Facial landmark detection results on AFLW dataset. $\theta = 30$ is used for BEL.

Approach	Feature Extractor	#Params/ GFlops	Full	Frontal
LAB (w/o B) (Wu et al., 2018)	Hourglass	25.1/19.1	1.85	1.62
AnchorFace (Xu et al., 2020)	ShuffleNet-V2	-	1.56	
LUVLi (Kumar et al., 2020)	CU-Net	-	1.39	1.19
HRNetV2-W18 (Wang et al., 2020) (Heatmap regression)	HRNetV2-W18	9.6/4.6	1.57	1.46
Direct regression (L2 loss)	HRNetV2-W18	10.2/4.7	2.10 ± 0.02	1.71
Direct regression (L1 loss)	HRNetV2-W18	10.2/4.7	1.51 ± 0.01	1.34
BEL-J/CE/GEN-EX	HRNetV2-W18	10.8/4.6	$\mathbf{1.47} \pm 0.00$	**1.30**

$*$Uses different data augmentation for the training

F AGE ESTIMATION

This task aims to predict the age of a person from an image. We use MORPH-II (Ricanek & Tesafaye, 2006) an AFAD (Niu et al., 2016) datasets with data augmentation and evaluation protocols used in (Pan et al., 2018).

F.1 EXPERIMENTAL SETUP

We use the MORPH-2 (Ricanek & Tesafaye, 2006) and AFAD (Niu et al., 2016) datasets for our evaluation. Cumulative Score (CS) and MAE are used as evaluation metrics. We preprocess the MORPH-2 dataset by aligning images first along the average eye position (Raschka, 2018), then by re-aligning so that the tip of the nose is in the center of each image. We do not preprocess the AFAD dataset as faces are already centered. Afterwards, face images are resized into $256 \times 256 \times 3$ and randomly cropped to $224 \times 224 \times 3$ for training. For testing, a center crop of $224 \times 224 \times 3$ is taken.

MORPH-2 This dataset consists of 55,608 face images with age labels between 16 and 70. The dataset is randomly divided into 39,617 training, 4,398 validation, and 11,001 testing images.

AFAD This dataset consists of 164,432 Asian facial images and age labels between 15 and 40. The dataset is randomly divided into 118,492 training, 13,166 validation, and 32763 testing images.

Training parameters We performed two runs with different random seeds to determine the learning rate between $[0.00001, 0.0001, 0.001]$ and use a batch size of 64 for all experiments. We use ImageNet pretrained weights to initialize the network.

Table 24: Training parameters for age estimation using MORPH=II and AFAD dataset

Optimizer	Epochs	Batch size	Learning rate	Learning rate schedule
Adam, weight decay=0, momentum=0	50	64	0.0001	-

F.2 RELATED WORK

Existing approaches for age estimation include ordinal regression (Niu et al., 2016; Cao et al., 2020), soft regression (Rothe et al., 2015), and expected value ordinal regression (Pan et al., 2018; Gao et al., 2018). OR-CNN (Niu et al., 2016) proposed the use of ordinal regression via binary classification to predict the label. CORAL-CNN (Cao et al., 2020) refined this approach by enforcing the ordinality of the model output. SSR-Net (Rothe et al., 2015) proposed the use of stage-wise soft regression using the softmax of the classification output to refine the predicted label. MV-Loss (Pan et al., 2018) extended the soft regression approach by penalizing the output of the model based on the variance of the age distribution, while DLDL (Gao et al., 2018) proposed to use the KL-divergence between the softmax output and a generated label distribution to train a model.

F.3 EVALUATION

Table 25 compares different combinations of encoding, deocding, and loss functions for BEL approach on MORPH-II dataset and ResNet50 feature extractor. Table 27 compares different combinations of encoding, deocding, and loss functions for BEL approach on AFAD dataset and ResNet50 feature extractor. Table 26 and Table 28 compare BEL with other regression approaches for MORPH-II and AFAD datasets. BEL-x provides lower MAE than the direct regression for both datasets, and BEL-B1JDJ/MAE/GEN-EX results in the lowest MAE for the AFAD dataset compared to the state-of-the-art approaches. OR-CNN (Niu et al., 2016) and CORAL-CNN (Cao et al., 2020) use ordinal regression by binary classification approach. We achieve significant improvement over these approaches due to the differences in encoding and decoding function, regressor architecture,

Table 25: Comparison of BEL design parameters on MAE for age estimation for MORPH-II dataset and ResNet50 feature extractor.

Decoding function	Loss function	Encoding function					
		U	J	B1JDJ	B2JDJ	HEXJ	HAD
BEL-J/BEL-U	BCE	2.32	**2.27**	-	-	-	-
GEN-EX	BCE	2.30	2.29	2.35	2.49	2.45	2.99
GEN	BCE	2.28	2.28	2.34	2.51	2.54	3.07
GEN-EX	CE	2.55	2.54	2.75	2.65	2.63	12.33
GEN	CE	2.60	2.58	2.61	2.66	2.61	3.10
GEN-EX	L1	2.30	2.30	2.32	2.30	2.32	2.29

Table 26: Age estimation results on MORPH-II dataset. $\theta = 10$ is used for BEL.

Approach	Feature extractor	#Parameters (M)	MORPH-II (MAE)	MORPH-II (CS$\theta = 5$)
OR-CNN (Niu et al., 2016) (Ordinal regression by binary classification)	-	1.0	2.58	0.71
MV Loss (Pan et al., 2018) (Direct regression)	VGG-16	138.4	2.41	0.889
DLDL-v2 (Gao et al., 2018) (Ordinal regression with multi-class classification)	ThinAgeNet	3.7	<u>1.96</u>*	-
CORAL-CNN (Cao et al., 2020) (Ordinal regression by binary classification)	ResNet34	21.3	2.49	-
Direct Regression (L2 Loss)	ResNet50	23.1	2.44 ± 0.01	0.903 ± 0.002
BEL-J/BCE/BEL-J	ResNet50	23.1	**2.27** ± 0.01	**<u>0.928</u>** ± 0.001

Uses custom data augmentation for the training

and training framework. This improvement shows the potential of these design aspects to improve regression accuracy.

Table 27: Comparison of BEL design parameters on MAE for age estimation for AFAD dataset and ResNet50 feature extractor.

Decoding function	Loss function	Encoding function					
		U	J	B1JDJ	B2JDJ	HEXJ	HAD
BEL-J/BEL-U	BCE	3.13	3.15	-	-	-	-
GEN-EX	BCE	3.14	3.16	3.32	3.35	3.28	3.34
GEN	BCE	3.13	3.19	3.41	3.44	3.41	3.52
GEN-EX	CE	3.26	3.29	3.38	3.44	3.40	3.30
GEN	CE	3.36	3.34	3.42	3.47	3.40	3.45
GEN-EX	L1	3.13	3.12	**3.11**	3.12	3.13	3.13

Table 28: Age estimation results on AFAD dataset. $\theta = 10$ is used for BEL.

Approach	Feature extractor	#Parameters (M)	AFAD (MAE)	AFAD (CS$\theta = 5$)
OR-CNN (Niu et al., 2016) (Ordinal regression by binary classification)	-	1.0	3.51	0.74
CORAL-CNN (Cao et al., 2020) (Ordinal regression by binary classification)	ResNet34	21.3	3.47	-
Direct Regression (L2 Loss)	ResNet50	23.1	3.21 ± 0.02	0.810 ± 0.02
BEL-B1JDJ/MAE/GEN-EX	ResNet50	23.1	**3.11** ± 0.01	**0.823** ± 0.001

G END-TO-END AUTONOMOUS DRIVING

We evaluate our approach on the NVIDIA PilotNet dataset for end-to-end autonomous driving (Bojarski et al., 2016). In this task, the steering wheel's next angle is predicted from an image of the road.

G.1 EXPERIMENTAL SETUP

End-to-end autonomous driving is the problem of predicting the target wheel state directly from an image representing the view in front of a car. We use a driving dataset consisting of 45,500 images taken around Rancho Palos Verdes and San Pedro, California (Chen). We crop images to $256 \times 70 \times 3$ then resized them to $200 \times 66 \times 3$. We randomly vary the brightness of the image between $[0.2\times, 1.5\times]$ and randomly perturb the steering direction. We use PilotNet architecture (Bojarski et al., 2017) and use $\theta = 10$ with 670 quantization levels for BEL.

Training parameters We performed two runs with different random seeds to determine the learning rate between $[0.00001, 0.0001, 0.001]$ and use a batch size of 64 for all experiments.

Table 29: Training parameters for end-to-end autonomous driving using PilotNet.

Optimizer	Epochs	Batch size	Learning rate	Learning rate schedule
SGD with weight decay=1e-5, momentum=0	50	64	0.1	1/10 at 10, 30 epochs

G.2 RELATED WORK

End-to-end autonomous driving is a novel task that has become increasingly relevant due to the rise of self-driving vehicles. The autonomous driving model's task is to predict the future driving angle based on a forward-facing image from the perspective of the vehicle. PilotNet (Bojarski et al., 2017) used a small, application-specific network to succeed within the time constraints of autonomous driving.

G.3 EVALUATION

Table 30 compares different combinations of encoding, deocding, and loss functions for BEL approach on PilotNet dataset and feature extractor. Table 31 compares binary encoded labels with the PilotNet which uses direct regression (Bojarski et al., 2017). BEL-x results in a significant reduction in the MAE compared to the direct regression approach.

Table 30: Comparison of BEL design parameters on MAE for end-to-end autonomous driving for PilotNet dataset and feature extractor.

Decoding function	Loss function	Encoding function					
		U	J	B1JDJ	B2JDJ	HEXJ	HAD
BEL-J/BEL-U	BCE	4.34	3.91	-	-	-	-
GEN-EX	BCE	4.57	4.20	4.83	4.96	5.29	10.12
GEN	BCE	4.37	3.95	3.51	3.61	4.01	10.00
GEN-EX	CE	4.30	4.16	4.99	5.87	5.39	87.17
GEN	CE	3.15	**3.11**	3.14	3.21	3.64	6.20
GEN-EX	L1	4.10	4.11	4.34	4.34	4.11	5.09

Table 31: End-to-end autonomous driving results on PilotNet dataset and architecture (Bojarski et al., 2017; 2016).

Approach	Feature extractor	#Parameters (M)	MAE
PilotNet (Bojarski et al., 2017)	PilotNet	1.8	4.24 ± 0.45
BEL-J/GEN/CE	PilotNet	1.8	$\underline{\mathbf{3.11}} \pm 0.01$