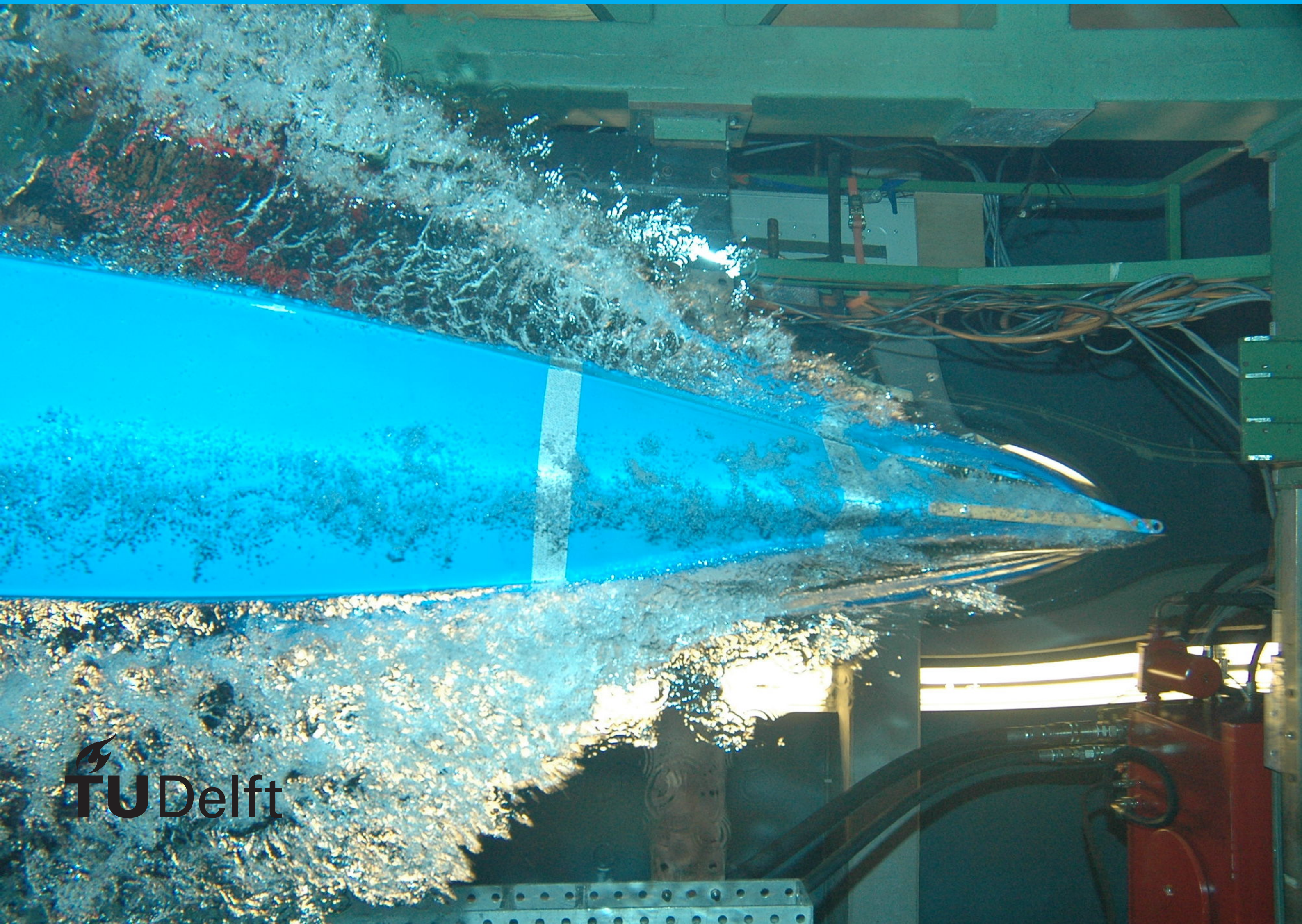
Classifying Continuous Labels: A Simple Tweak to Make Regression Robust

Ziyu Bao



Classifying Continuous Labels: A Simple Tweak to Make Regression Robust

by

Ziyu Bao

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Monday July 4, 2022 at 1:00 PM.

Student number:	4436113	
Project duration:	September 1, 2019 – July 4, 2022	
Thesis committee:	Dr. Jan van Gemert,	TU Delft, Supervisor, Committee Chair
	Dr. Silvia-Laura Pintea,	TU Delft, Daily Supervisor, Committee Member
	Dr. Cynthia Liem,	TU Delft, Committee Member

Preface

This report presents the work of my master's thesis project on the topic of "Classifying Regression Labels: A Simple Tweak to Make Regression Robust". This research was conducted at Computer Vision Lab of Pattern Recognition and Bioinformatics Group in TU Delft under the supervision of Dr. Jan van Gemert.

First and foremost, I would like to express my deepest appreciation for my supervisor, Dr. Jan van Gemert, for guiding me through the thesis process. I would like to express my deepest gratitude to Dr. Silvia-Laura Pintea for being my daily supervisor during my thesis and in charge of most of my questions and being always patient. This work cannot be done without her supervision. I would also like to thank Dr. Cynthia Liem for her interest in my thesis and for evaluation of my work.

During the project, I have learned a lot. I have learned how to deal with outside environmental changes because the project begins when COVID-19 breaks out. I have learned how to prioritize things and make a wise plan.

Last but not least, I would like to thank my parents and friends for helping me to overcome all obstacles during the project and for supporting me emotionally along the whole path.

Ziyu Bao
Delft, July 2022

Contents

1
Scientific Paper

Classifying Continuous Labels: A Simple Tweak to Make Regression Robust

Ziyu Bao
b13706948771@icloud.com

Dr. S. L. Pintea
silvia.laura.pintea@gmail.com

Dr. J. Gemert
j.c.vangemert@tudelft.nl

Delft University of Technology
Mekelweg 5, 2628 CD Delft, Netherlands

Abstract

Regression is difficult because of noise, imbalanced data sampling, missing data, etc. We propose a method by classifying the continuous regression labels to tackle regression robustness problems. We analyze if our method can help regression, given that the class information is already included in the regression labels. We start by extensively experimenting on 1D synthetic datasets and find out that classification can help regression when the data sampling is imbalanced. This happens when the data are clean, noisy in inputs and noisy in outputs, but not when they are partially missing. We then validate our conclusion on the KITTI dataset by estimating 3D object orientation. We conclude that our method can help regression in real-world.

Keywords— *regression, robustness, imbalance*

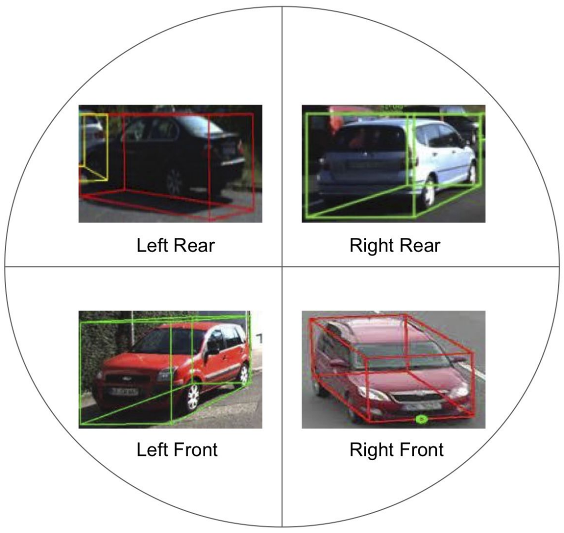


Figure 1. Overview of orientation estimation assisted by binning. Instead of regressing to the ray orientation $\in \mathbb{R}$ spanning 360 degrees, multi-bin regression first divides the orientation into more than one bin and learns in which bin the true orientation lies and how to regress to it from the center orientation of that bin. The best number of bins is task-dependent.

1. Introduction

Regression, where the goal is to make models learn to predict continuous labels, is one of the most fundamental tasks in machine learning. Applications of regression techniques exist ubiquitously in real world, e.g., facial landmark detection [20, 33], head-pose estimation [16, 31], depth estimation [14, 15], age estimation [27, 28] and cell counting [9, 29]. Although regression is fundamental and ubiquitous, its performance is highly sensitive to noise, imbalanced sampling, missing data, etc. In this paper, we aim to tackle this issue by investigating whether an auxiliary classification task, which learns to predict binned continuous labels, can help regression to be robust and under which circumstances can it help.

Classification in combination with regression for better regression performance yields promising results. In [33], Zhang et al. optimize facial landmark detection with heterogeneous classification tasks including gender estimation, smile detection and head-pose classification. Traditionally, researchers approach facial landmark detection as a standalone problem. However, Zhang et al. demonstrate that correlated classification tasks are helpful. Effectively exploiting the intrinsic correlation can help detection to be more accurate and constrain the solution space. Their method achieves robust detection, especially of faces with severe occlusion and pose variation. [30] analyzes the review rating task and proves that using both classification and regression losses to model the same target labels is better than using either loss. They point out that both regression and classification have their own advantages. Although classification achieves better results when the number of target labels is small, regression utilizes ordinal information and is suitable for a big range of target labels. [9] transforms the traditionally regression-based cell counting task into an image classification task because regression-based predictions often deviate from the ground truth. However, they realize that a model trained only with a classification method lacks the generalizability to unseen cell counts. They demonstrate that regression has a better generalizability and stability on unseen test images and propose an ensemble scheme which combines the precision of classification and the generalizability of regression. We build on this trend and pay extra attention to the unique characteristics of classification.

Our research question is: *Can an auxiliary classification task which learns binned regression labels help the main regression task to be robust*? Our hypothesis is yes. To answer our research question, we carefully design our data. From imbalanced regression [6], robust regression [24] and zero-shot learning [21], we know that a complete list of data settings should be imbalanced, noisy, and partially missing. The meaning that the data is noisy is two-fold: noisy in inputs and noisy in labels. Different levels of binning are important. After validating our hypothesis on 1D synthetic data, we validate it on the real-world KITTI [11] dataset with the task of 3D object orientation estimation [18]. We show in Figure 1 how binning in combination with regression solves a regression task in the real world.

The main contributions of this work are:

- We propose learning binned regression labels as an auxiliary task for more robust regression. We show that there is no additional information or target labels needed.
- We analyze carefully and thoroughly, using controlled data settings on synthetic data, to understand whether and how the auxiliary classification task can help regression and demonstrate that it helps when the data sampling is imbalanced.
- We prove that our finding generalizes well to the real world by validating it on a 3D orientation estimation task using the KITTI dataset.

2. Related Work

2.1. Robust Regression

Robust regression assumes that the data distribution violates assumptions of the regression method, and aims to improve the regression method such that it is not affected even when violations occur. The M-estimator [13] and its alternatives [2, 22, 23] optimize the loss method design and are resistant to outliers in response variables. However, they are less robust to heteroscedastic errors which depend on the input. What's more, better loss designs can always be combined with our auxiliary classification task to achieve further improvement. Other robust regression methods use deep learning to learn data distribution automatically [3, 19]. One drawback is that they have to design correct hyperparameters and define correct dependence relations for the model to learn. They are interesting works that go in another direction from our approach. Data augmentation and re-sampling techniques [7, 8] are other approaches which focus on mitigating violations of assumptions of the data distribution. They are effective methods but are not efficient in time. All the methods are great, but our approach is something different.

2.2. Regression with Classification Loss

Classification has been shown to improve regression. Pose classification helps landmark localization to be more accurate [33]. Ordinal class information of the depth helps the exact depth to regress better [10]. However, there are also papers which do not report a better performance, e.g. [12], and there are conditions for classification to help regression in [33] and [10]. In [33], task-wise early stopping is used. [10] uses the ordinal information instead of treating the classes as irrelevant “buckets”. In this paper, we do not only validate that classification can improve regression, but we take this one step further and analyze in which conditions this happens.

2.3. Curriculum Learning

Curriculum learning follows a coarse-to-fine scheme and achieves loss function robustness and faster convergence [26]. Since Bengio et al. [5] first introduce the concept of curriculum learning in the context of classification, the coarse-to-fine approach has been used in different components of the learning process. Belagiannis et al. [4] typically use progressively higher resolution images to achieve state-of-the-art pose regression performance. The coarse-to-fine scheme here is serial. However, our approach follows a parallel coarse-to-fine scheme: Classification can be seen as a coarse learning task because the binned labels are less precise than the continuous ones, and we learn regression and classification at the same time. We prove that parallel coarse-to-fine scheme also achieves loss function robustness under certain circumstances.

3. Method

3.1. Classification-Assisted Regression (CAR) Loss

Figure 2 explains our proposed method. The goal is to learn a continuous label. We calculate the minimum and maximum of all labels of the training dataset. We then discretize the range into a number of bins. The number of bins is a hyperparameter, and needs hyperparameter searching

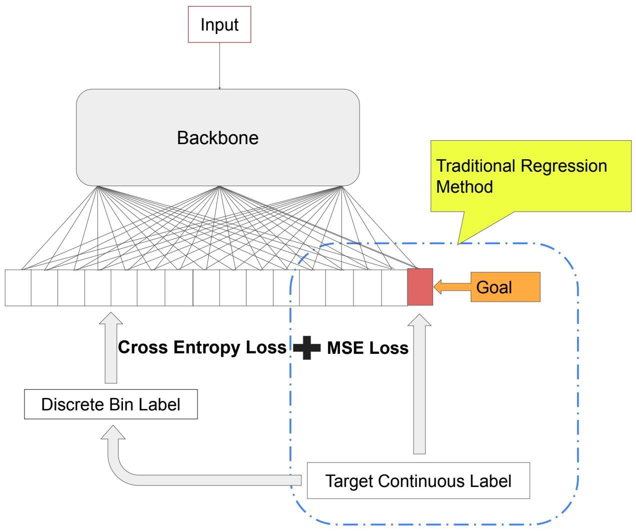


Figure 2. Overview of our proposed CAR method. The goal is to learn the continuous regression label. The continuous label is transformed into the discrete bin label after doing statistics on the range of the training target labels. Learning the discrete label is the auxiliary task. Learning the regression label is the main task. Both tasks are learned at the same time. We learn the main regression task by the MSE loss and the auxiliary classification task by the CE loss of target classification labels from the results of an indicator function of the target continuous labels. There is no additional information needed. The traditional regression method only learns the former. The last layer of the model architecture is usually an FC layer or an ensemble of FC layers. This module can be attached to any backbone architectures.

for the best performance. We perform our 1D experiments with the following numbers of bins: 2, 4, 8, 16, 32, 64, 128, 256, 512. In our 3D object orientation estimation experiments, we use 2 and 4 bins. After dividing the range into bins, the target continuous label has its corresponding discrete classification label. We use non-overlapping bins in 1D experiments. We use overlapping bins in 3D experiments. In 3D experiments, one continuous label can have two bin labels. We choose the bin that has a lower center value. The overlap is one-sixth of the size of a bin. After defining both labels, two predictions are calculated from two heads of any sort of backbone models. In 1D experiments, we use a Multi-Layer Perceptron (MLP) with a single hidden layer of 10 nodes. The regression prediction is from a matrix multiplication of the hidden-layer results with a Fully-Connected (FC) layer of 1-by-10. The classification prediction is a vector whose size is equal to the number of bins, and it is obtained from the result of a Soft-max function of the result of another matrix multiplication of the hidden-layer results with another FC layer of bin-number-by-10. In 3D experiments, the backbone model is VGG19 [25]. The regression label is transformed into two regression labels: sine and cosine of the angle, which co-define a unique angle. The regression branch contains 2 FC layers, instead of 1. The first FC layer is of 256-by-512 $\times$ 7 $\times$ 7 and the second FC layer is of 256-by-256. The ReLU [1] function is used as the activation function between the FC layers. The classification branch has the same architecture. The 3D object orientation estimation task first requires object detection. We train the MS-CNN [18] as the object detector and the result of it is provided to the VGG19. After obtaining the regression and classification predictions, we use the Mean Squared Error (MSE) loss between the regression predictions and the target continuous labels to calculate the loss of the regression head and the Cross Entropy (CE) loss between the classification probability predictions and the results of the indicator function of the target continuous labels to calculate the loss of the classification head, and then sum up the two, and use gradient descent to jointly update the backbone. In 3D experiments, the loss is summed up with the object detection losses and is used to jointly update the MS-CNN and the VGG19. Our loss is defined as:

$$L = \frac{1}{n}\Sigma(y-\hat{y})^2 - \frac{1}{m}\Sigma_{i=1}^{m}\mathbb{1}_{bin_i}(y)\cdot log(\hat{y}_i) \qquad (1)$$

"n" is the dimensionality of the target continuous label. If the regression label is uni-dimensional, "n" is 1. "y" is the target regression label and "$\hat{y}$" is the predicted regression value. "m" is the number of bins. "$\mathbb{1}_{bin_i}(y)$" is the indicator function of "y". It is 1 if bin_i is where the target lies, and it is 0 if not. "$\hat{y}_i$" is the predicted classification probability for that bin to be where the target lies.

3.2. Controlled Settings

In 1D experiments, we use 4 different settings of data. They are "Clean Inputs and Outputs", "Noisy Inputs", "Noisy Outputs" and "Out-of-Distribution Testing". At each setting, we use 4 different levels of imbalance of the distribution of the target continuous labels. In 3D experiments, there is only one data setting, which is the inputs and outputs of the KITTI dataset. We manually construct a subset with "Mildly Imbalanced" label distribution, so that we compare two levels of target label distribution, i.e. "Mildly Imbalanced" and "Severely Imbalanced".

3.3. Training and Evaluation

We have training, validation and testing splits of datasets. We draw the validation sets from a joint distribution as the training sets. We respectively use the CAR loss and the MSE loss to train and compare their performances in testing. After at least 100 epochs of convergence in 1D experiments and 5 epochs of convergence in 3D experiments, we take the weights of the model whose performance measure on the validation set is the lowest as the weights of the model for testing. In 1D experiments, the performance measure for the validation set and the testing set is the MSE. In 3D experiments, the performance measure is the official metric of KITTI, i.e., Average Orientation Similarity (AOS) [11], for measuring orientation estimation performance with object detection. AOS is defined as follows:

$$AOS = \frac{1}{11}\Sigma_{r\in\{0,0.1,\dots,1\}} \max_{\tilde{r}:\tilde{r}\geq r} s(\tilde{r}) \tag{2}$$

"r" is the recall threshold of the object detection. "$s(\tilde{r})$" is the cosine similarity of all objects detected with a recall higher than "r". $s(\tilde{r}) \in [0,1]$. Its normalized value over 11 recall thresholds, i.e. the AOS, is also in [0, 1]. "$s(r)$" is defined as:

$$s(r) = \frac{1}{|D(r)|}\Sigma_{i\in D(r)}\frac{1+cos\Delta_{\Theta}^{(i)}}{2}\delta_i \tag{3}$$

"D(r)" is the set of all positively detected objects above the recall threshold "r". "$\Delta_{\Theta}^{(i)}$" is the difference between the predicted and the target orientations. "δ_i" is to record if "i"-th object has been calculated with the ground truth, and it avoids multiple predictions from matching to the same ground truth.

We ignore the classification performance in validation and testing. We repeat 16 times per setting of experiments. If "best" models using CAR loss to train has lower performance measures than "best" models using MSE loss to train, and the standard deviations of the performance measures are mutually exclusive, we conclude in that experiment a confirmation to our research question.

4. Experiments

4.1. 1D Synthetic Sinusoidal Curves Fitting

4.1.1 Dataset

The synthetic dataset contains 2048 data points. The regression labels are the outputs of Equation 4.

$$y = 2\sin(2x), x \in [0, 2\pi] \tag{4}$$

"x" are the inputs. "y" are the outputs. "y" belongs to $[-2, 2]$. Figure 3 explains different settings of our data. The left-most "Target Label Distribution" column depicts the distribution of "y". There are 4 levels of severity of imbalance of "y", depicted from the first row to the 4th row. The last row depicts the uniform distribution of "y" in testing. Their corresponding classification labels are equally-spaced labels. In the right 4 columns, the classification labels are depicted as 4 different levels of shading in the background when the number of bins is equal to 4. Each row of the right 4 columns has the distribution of "y" the same as the "Target Label Distribution" column. The right 4 columns are 4 different settings of our experiments, namely, from left to right, "Clean Inputs and Outputs", "Noisy Inputs", "Noisy Outputs" and "Out-of-Distribution Testing". The details of the four settings are:

- The "Clean Inputs and Outputs" setting. The inputs are uniformly distributed in [0, 2π] and the regression labels follow Equation 4.
- The 'Noisy Inputs" setting. The inputs are shifted uniformly at a distance of [-0.4π, 0.4π] away from their corresponding inputs in the clean setting. The regression labels follow Equation 4 with shifted "x"'s.
- The 'Noisy Outputs" setting. The inputs are uniformly distributed in [0, 2π]. The regression labels are the results of Equation 4 and then are shifted uniformly at a distance of [-2, 2] away from their initial results. One-sixteenth of the regression labels are forced to change into outliers, i.e. their values are either ± 4 of the initial results of Equation 4.
- The "Out-of-Distribution Testing" setting. We divide the inputs in the clean setting into 8 equally-spaced intervals, and then divide each interval into halves. We use the left halves in training and the right halves in testing. The regression outputs follow Equation 4. The training set and the testing set are mutually exclusive.

We repeat the experiment of one setting 16 times. By one setting, we mean the same level of imbalance, the same number of bins and the same inputs and outputs setting. At each repetition, we randomly select a mean of the Gaussian distribution of the regression labels. We experiment with $\{$2, 4, 8, 16, 32, 64, 128, 256, 512$\}$ bins. To make sure all classification labels exist, we enforce that every bin in 512 bins contains at least 1 data point. This is why the "Moderately Imbalanced" row of the "Target Label Distribution" column has a straight blue line at the bottom. In total, we perform 4 levels of imbalance $\times$ 9 numbers of bins $\times$ 4 inputs and outputs settings $\times$ 16 repetitions = 2304 repetitions.

4.1.2 Experiment 1: Can classification help regression, and under which circumstances?

Figure 4 shows the performances on 1D synthetic testing sets of training using the MSE loss compared with training using the CAR loss. The columns are the 4 data settings and the rows are the 4 levels of imbalance. The y-axis of each grid cell represents the testing MSE values. The range of each y-axis is $[0, 1.15]$ for consistency across cells. Each solid circle of the CAR method depicts the means of the MSE's of 16 repetitions, and the error bars depict the standard deviations. The upper error bar has a length of a standard deviation. So is the lower error bar. The x-axis of each grid cell is the numbers of bins, from 2 to 512. The MSE method is not relevant to the numbers of bins. We draw it

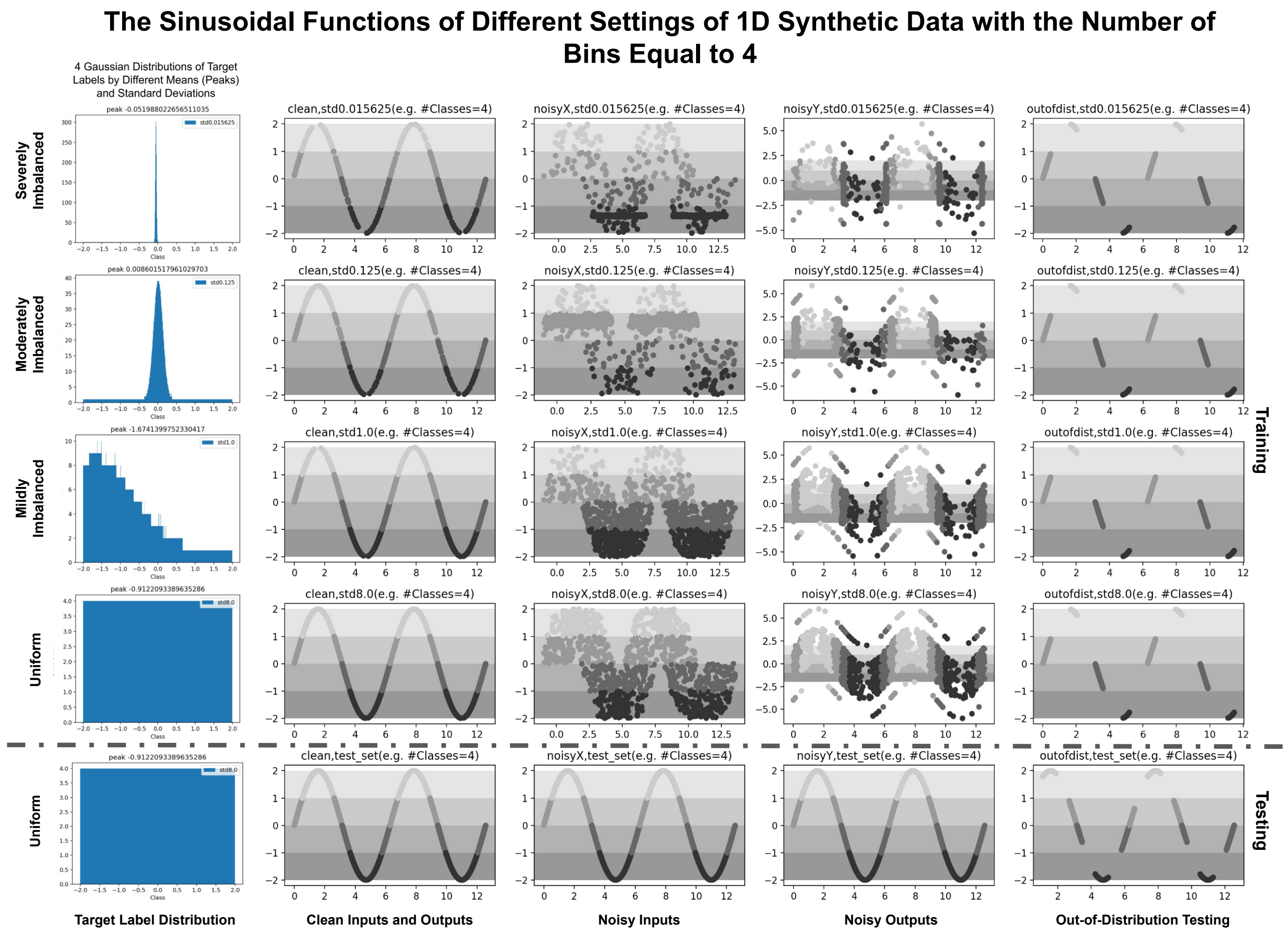


Figure 3. Overview of different data settings of 1D experiments with the number of bins equal to 4. The left-most column depicts different levels of severity of imbalance of the Gaussian distribution of the regression labels. The right 4 columns depict the sinusoidal curves of four settings of inputs and outputs. In the right 4 columns, the x-axis of each grid cell represents the inputs. The y-axis of each grid cell represents the targets. The upper 4 rows depict the data in training. The bottom row depicts the data in testing. The rows in the right 4 columns have the same data distributions as the rows in the left-most column. This grid provides an intuitive understanding of different settings of the data in 1D sinusoidal curve fitting experiments.

as a line instead of a poly-line. The mean is depicted as the solid straight line. The standard deviation is depicted as the shaded stripes above and below the straight line. We draw error bars and shaded stripes to show if they are disconnected or not. If the error bars and the shaded stripes are not connected, the difference is statistically significant.

Conclusion 1: Classification can help regression. The "Severely Imbalanced" row of the "Clean Inputs and Outputs" column in Figure 4 is one obvious example from which we can conclude that classification can help regression. From 2 bins to 512 bins, the CAR method has lower MSE means than the MSE method. From 4 bins to 512 bins, the CAR method has lower MSE means than the MSE method with disconnected standard deviations. Someone may expect the margin between the MSE line and CAR poly-line becomes larger as the number of bins increases. In fact, we would expect a threshold number of bins, under which, the CAR method performs increasingly better than the MSE method as the number of bins increases, and above which, the two methods show an equal margin between their performances across all numbers of bins. This is in alignment with the behavior in [18]. The best number of bins is 8 in this setting. Increasing the number of bins even decreases the performance of our CAR method. This trend also exists in the "Noisy Inputs" and "Noisy Outputs" columns. The reason may be that increasing the number of bins eventually decreases the amount of training data within each bin, and classification becomes so similar with regression that no "coarse-to-fine" effect is possible.

Conclusion 2: The severity of imbalance is the rea-

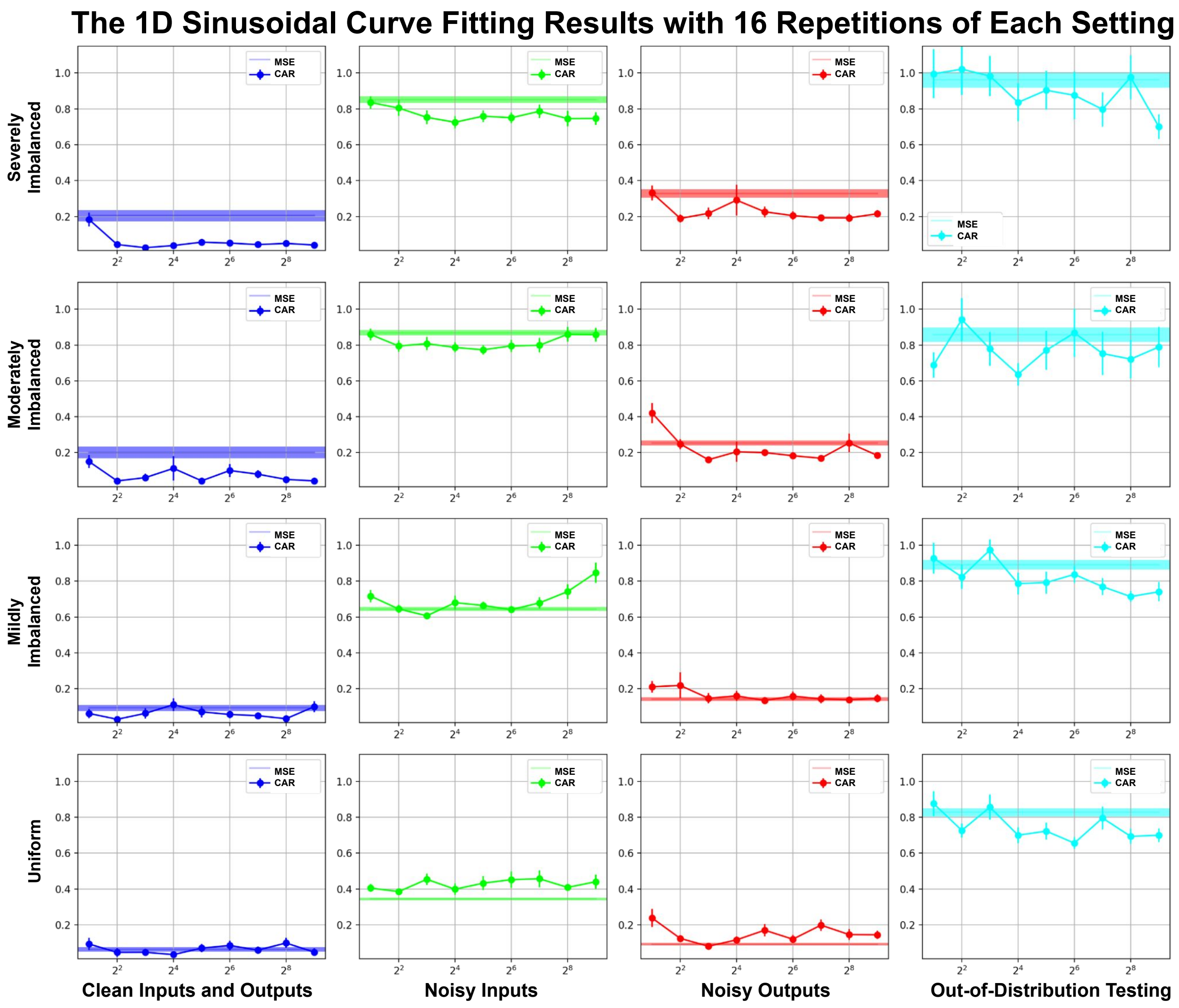


Figure 4. This is the performance, measured by the MSE on the testing sets, between using the MSE loss and the CAR loss to train. The rows are 4 levels of imbalance of the training label distributions. The columns are 4 data settings. In each grid cell, the x-axis is the numbers of bins, and the y-axis is the MSE values. The range of the y-axis is the same across cells, and it is $[0, 1.15]$. Because using the MSE loss to train is not affected by the numbers of bins, the MSE method is drawn as a straight line. If the circles of the poly-line of the CAR method are lower than the straight line of the MSE method, and the error bars are disconnected, the CAR method outperforms the MSE method. This figure shows in each setting whether the CAR method outperforms the MSE method.

son that classification can help regression. Along the "Clean Inputs and Outputs" column in Figure 4, we compare the levels of imbalance. We see that, as the imbalance decreases, the margin between CAR and MSE becomes smaller and eventually becomes insignificant when the label distribution is "Mildly Imbalanced" and "Uniform". This trend also exists in the "Noisy Inputs" and "Noisy Outputs" columns. In the "Noisy Inputs" and "Noisy Outputs" columns, the CAR poly-line even goes higher than the MSE line in the "Uniform" row. We conclude that the severity of imbalance is the cause for classification to help regression.

Conclusion 3: The means of the Gaussian distributions of the regression labels is irrelevant to the winning of the CAR method. In one setting, the label distribution is the only changing factor. The means of the label distributions is relevant if the error bars are long.

Conclusion 4: Classification cannot help regression when the testing data are out of distribution. The "Out-of-Distribution Testing" column has a unique behavior. The difference between the CAR poly-line and the MSE line does not become smaller as the training label distribution becomes less imbalanced. There is hardly any number of bins, with which the error bars of the CAR circle are disconnected with the shaded stripes of the MSE line. This shows that the CAR method does not help regression under all tested circumstances when the training data are partially missing or corrupted.

Conclusion 5: Noise makes classification harder to help regression. We look at the columns of "Noisy Inputs" and "Noisy Outputs" and can see that at the "Mildly Imbalanced" row, more than half of the circle points are higher than the MSE line. However, in the "Clean Inputs and Outputs" column, only at the "Uniform" row, it happens that more than half of circle points are higher than the MSE line. This shows that noise can degrade the "helping" effect.

There are two other phenomena that are worth noting. The first exists ubiquitously. The margin between the MSE and CAR methods is not the same after a threshold number of bins. There are more than one "jumps" in the CAR poly-lines. What causes the "jumps"? The second is the tendency, in the "Out-of-Distribution Testing" column, that the margin between the MSE line and the CAR poly-line increases as the number of bins increases. They are interesting phenomena which need further confirmation and examination.

4.2. 3D Real-World Object Orientation Estimation

4.2.1 Dataset

The KITTI dataset has 6732 training images and 749 testing images. There are 32456 objects in the training images and 4057 objects in the testing images. The objects include cars, pedestrians and cyclists. Figure 5 shows the orientation label distributions of the training, validation and testing sets. The orientation label distributions of the "Training" and "Validation" rows of the "Severely Imbalanced" column is the same as those in the KITTI dataset. We randomly select 10% of the original training images to form the validation set, and the rest of them to form the training set. The "Mildly Imbalanced" column shows label distributions of manually-selected images. We take a subset of the images of the "Training" and "Severely Imbalanced" cell to form the images of the "Training" and "Mildly Imbalanced" cell, and make sure that the subset contains objects whose orientations follow a uniform distribution. The labels of the objects in the images end up with mildly imbalanced distribution. We do the same for the validation set. The testing sets of the "Severely Imbalanced" and "Mildly Imbalanced" settings are the same subset of the original testing images. In the "Severely Imbalanced" setting, we use 6059 images to train, 673 images to validate and 80 images to test. In the "Mildly Imbalanced" setting, we use 836 images to train, 72 images to validate and 80 images to test.

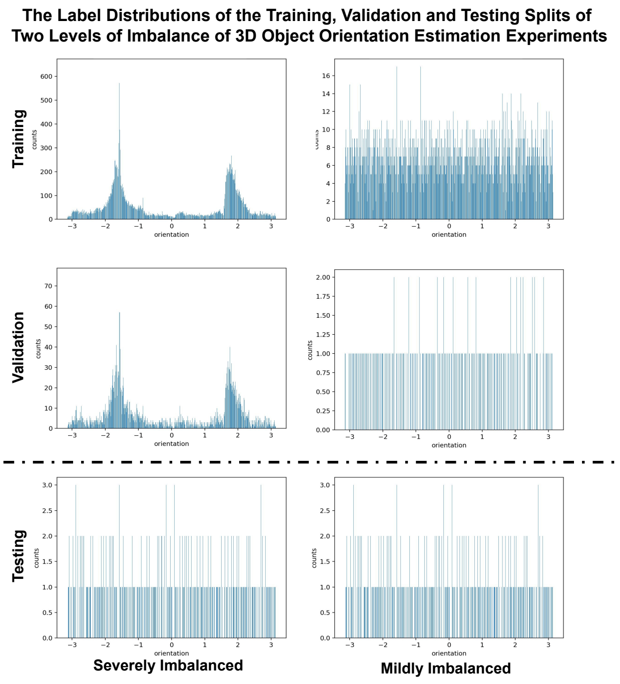


Figure 5. Overview of the label distributions of orientation estimation experiments. The "Training" and "Testing" of the "Severely Imbalanced" column shows the imbalance of data in real-world. It has two peaks instead of one. The "Mildly Imbalanced" column is more uniform. It is from manual selection of the original KITTI images. We show the difference between the "Severely Imbalanced" and "Mildly Imbalanced" columns. They are essential control factors to demonstrate whether classification can help regression in real-world and whether the imbalance is still the reason.

4.2.2 Experiment 2: Can classification help regression in real-world, and is the imbalance still the reason?

Figure 6 shows the AOS performances of orientation estimation between models trained using the CAR method and the MSE method, under "Severely Imbalanced" and "Mildly Imbalanced" settings.

Conclusion 1: Classification can help regression in real-world. We see in Figure 6 that "The CAR method: multibin of 4 bins" has higher AOS values than "The MSE method: single bin with sine and cosine" for both "Severely Imbalanced" and "Mildly Imbalanced" settings, with disconnected error bars. We conclude that classification can help regression in real-world.

Conclusion 2: The severity of imbalance is the reason that classification can help regression in real-world.

We observe a threshold number of bins equal to 2 of the "Severely Imbalanced" setting, and a threshold number of bins equal to 4 of the "Mildly Imbalanced" setting. We see

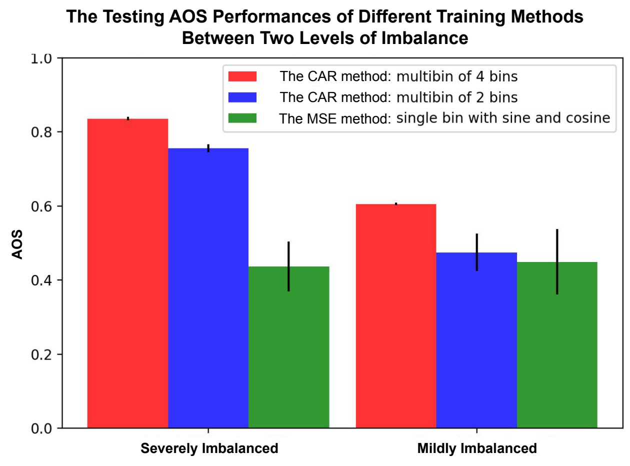


Figure 6. This is the AOS results of the orientation estimation experiments. We use the CAR method with the numbers of bins equal to 2 and 4 for training, and we compare them with using the MSE method for training. The CAR methods mostly outperform the MSE method with a large margin, especially under the “Severely Imbalanced” setting.

that the CAR methods outperform the MSE method with a large margin under the “Severely Imbalanced” setting, and the CAR methods do not outperform the MSE method with a large margin under the “Mildly Imbalanced” setting. When the number of bins is equal to 2 under the “Mildly Imbalanced” setting, the AOS value of the CAR method is not higher than that of the MSE method with disconnected error bars. We can conclude that the severity of imbalance is the reason that classification can help regression in real-world.

5. Conclusion

We propose a method to improve regression robustness without using extra information. We call the method CAR. We demonstrate empirically that the CAR method outperforms the traditional MSE method on both synthetic data and real-world data. We investigate the unique properties of the CAR method and find out the method works the best when the data label distribution is highly imbalanced. We demonstrate that this property holds for both synthetic and real-world data.

6. Discussion

Analysis limitations. We use uni-dimensional labels, both in synthetic and real-world datasets. This constrains the domain of our method. Multidimensional regression problems are popular nowadays, e.g. 3D human mesh reconstruction [32]. We do not perform the training correctly. The CAR method has a higher scale than the MSE method, but we do not use different learning rates. This can result in faster convergence of the CAR method. We train exhaustively. This can lower the probability for us to compare a converged CAR method with a non-converged MSE method.

Method limitations. The number of bins is a hyperparameter and is searched by trial-and-error. In [3], Barron treats the hyperparameters as adaptive variables, and he updates the hyperparameters using gradient descent. We can do the same with the number of bins. After registering the number of bins as an adaptive variable, the indicator function to make the classification labels should be made continuous so that the number of bins can be updated automatically. Our method assumes regression labels with upper and lower boundaries, so that we can discretize the regression labels into bins. [17] transforms infinite regression problems into finite regression problems using hypersphere. It is an interesting further direction.

References

[1] A. F. Agarap. Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375*, 2018.

[2] M. G. Akritas, S. A. Murphy, and M. P. Lavalley. The theil-sen estimator with doubly censored data and applications to astronomy. *Journal of the American Statistical Association*, 90(429):170–177, 1995.

[3] J. T. Barron. A general and adaptive robust loss function. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4331–4339, 2019.

[4] V. Belagiannis, C. Rupprecht, G. Carneiro, and N. Navab. Robust optimization for deep regression. In *Proceedings of the IEEE international conference on computer vision*, pages 2830–2838, 2015.

[5] Y. Bengio, J. Louradour, R. Collobert, and J. Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48, 2009.

[6] P. Branco, L. Torgo, and R. P. Ribeiro. Smogn: a pre-processing approach for imbalanced regression. In *First international workshop on learning with imbalanced domains: Theory and applications*, pages 36–50. PMLR, 2017.

[7] P. Branco, L. Torgo, and R. P. Ribeiro. Pre-processing approaches for imbalanced distributions in regression. *Neurocomputing*, 343:76–99, 2019.

[8] L. Camacho, G. Douzas, and F. Bacao. Geometric smote for regression. *Expert Systems with Applications*, page 116387, 2022.

[9] X. Ding, Q. Zhang, and W. J. Welch. Classification beats regression: Counting of cells from greyscale microscopic images based on annotation-free training samples. In *CAAI International Conference on Artificial Intelligence*, pages 662–673. Springer, 2021.

[10] H. Fu, M. Gong, C. Wang, K. Batmanghelich, and D. Tao. Deep ordinal regression network for monocular depth estimation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2002–2011, 2018.

[11] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE conference on computer vision and pattern recognition*, pages 3354–3361. IEEE, 2012.

[12] G. Gkioxari, B. Hariharan, R. Girshick, and J. Malik. R-cnns for pose estimation and action detection. *arXiv preprint arXiv:1406.5212*, 2014.

[13] P. J. Huber. Robust estimation of a location parameter. In *Breakthroughs in statistics*, pages 492–518. Springer, 1992.

[14] M. Klingner and T. Fingscheidt. Online performance prediction of perception dnns by multi-task learning with depth estimation. *IEEE Transactions on Intelligent Transportation Systems*, 22(7):4670–4683, 2021.

[15] L. Liebel and M. Körner. Multidepth: Single-image depth estimation via multi-task regression and classification. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 1440–1447. IEEE, 2019.

[16] H. Liu, S. Fang, Z. Zhang, D. Li, K. Lin, and J. Wang. Mfdnet: Collaborative poses perception and matrix fisher distribution for head pose estimation. *IEEE Transactions on Multimedia*, 24:2449–2460, 2021.

[17] P. Mettes, E. van der Pol, and C. Snoek. Hyperspherical prototype networks. *Advances in neural information processing systems*, 32, 2019.

[18] A. Mousavian, D. Anguelov, J. Flynn, and J. Kosecka. 3d bounding box estimation using deep learning and geometry. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 7074–7082, 2017.

[19] D. A. Nix and A. S. Weigend. Estimating the mean and variance of the target probability distribution. In *Proceedings of 1994 ieee international conference on neural networks (ICNN'94)*, volume 1, pages 55–60. IEEE, 1994.

[20] R. Ranjan, V. M. Patel, and R. Chellappa. Hyperface: A deep multi-task learning framework for face detection, landmark localization, pose estimation, and gender recognition. *IEEE transactions on pattern analysis and machine intelligence*, 41(1):121–135, 2017.

[21] B. Romera-Paredes and P. Torr. An embarrassingly simple approach to zero-shot learning. In *International conference on machine learning*, pages 2152–2161. PMLR, 2015.

[22] P. Rousseeuw and V. Yohai. Robust regression by means of s-estimators. In *Robust and nonlinear time series analysis*, pages 256–272. Springer, 1984.

[23] P. J. Rousseeuw. Least median of squares regression. *Journal of the American statistical association*, 79(388):871–880, 1984.

[24] P. J. Rousseeuw and M. Hubert. Robust statistics for outlier detection. *Wiley interdisciplinary reviews: Data mining and knowledge discovery*, 1(1):73–79, 2011.

[25] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[26] P. Soviany, R. T. Ionescu, P. Rota, and N. Sebe. Curriculum learning: A survey. *arXiv preprint arXiv:2101.10382*, 2021.

[27] Y. Tingting, W. Junqian, W. Lintai, and X. Yong. Three-stage network for age estimation. *CAAI Transactions on Intelligence Technology*, 4(2):122–126, 2019.

[28] M. Xia, X. Zhang, L. Weng, Y. Xu, et al. Multi-stage feature constraints learning for age estimation. *IEEE Transactions on Information Forensics and Security*, 15:2417–2428, 2020.

[29] W. Xie, J. A. Noble, and A. Zisserman. Microscopy cell counting and detection with fully convolutional regression networks. *Computer methods in biomechanics and biomedical engineering: Imaging & Visualization*, 6(3):283–292, 2018.

[30] J. Xu, H. Yin, L. Zhang, S. Li, and G. Zhou. Review rating with joint classification and regression model. In *National CCF Conference on Natural Language Processing and Chinese Computing*, pages 529–540. Springer, 2017.

[31] T.-Y. Yang, Y.-T. Chen, Y.-Y. Lin, and Y.-Y. Chuang. Fsanet: Learning fine-grained structure aggregation for head pose estimation from a single image. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1087–1096, 2019.

[32] W. Zeng, W. Ouyang, P. Luo, W. Liu, and X. Wang. 3d human mesh regression with dense correspondence. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7054–7063, 2020.

[33] Z. Zhang, P. Luo, C. C. Loy, and X. Tang. Facial landmark detection by deep multi-task learning. In *European conference on computer vision*, pages 94–108. Springer, 2014.

2 Background on Deep Learning

Deep learning includes statistics and predictive modeling. It is a subdivision of machine learning and artificial intelligence (AI). Compared with machine learning, it is more complex and has a higher need of data. It has a wide range of applications, such as object detection [29], object orientation estimation [12], facial landmark detection [28], human pose estimation [27] and semantic segmentation [11].

2.1. Development

2.1.1. Perceptron

Deep learning is also called deep neural networks (NN) learning. The adjective "neural" refers to neuroscience [10], which is the inspiration of neural networks and the starting point of current success of deep learning. A perceptron is an "artificial" neuron, and it is the most basic unit of an NN. Figure 2.1 shows the architecture of a biological neuron and an artificial neuron. An artificial neuron is composed of 3 key units. They are a weight matrix, a summing unit and an activation function. The weight matrix closely resembles the dendrites of a biological neuron. The summation and activation are done in the unit closely resembling the cell nucleus. The output of the activation function is passed to the second neuron. The process closely resembles a signal being passed through by the axon. We will cover activation functions in details later.

2.1.2. Multi-Layer Perceptron (MLP)

MLP builds on the idea of a single perceptron and combines multiple perceptrons to form an artificial neural network. Many people call MLP as a "vanilla" artificial neural network [13] because it often has only one hidden layer. Figure 2.2 shows the architecture of an MLP with a single hidden layer of 3 neurons. The hidden layer result $\hat{h}$ is computed by a matrix multiplication with the inputs $\hat{x}$ as in Equation 2.1, and the output layer result $\hat{y}$ is computed by a matrix multiplication with the hidden layer result $\hat{h}$ as in Equation 2.2. w_h and w_o are respectively the weights between the input layer and the hidden layer, and the weights between the hidden layer and the output layer. "$a()$" refers to an activation function.

$$\hat{h} = a(w_h \cdot \hat{x} + b_h) \tag{2.1}$$

$$\hat{y} = a(w_o \cdot \hat{h} + b_o) \tag{2.2}$$

The process of obtaining $\hat{y}$ is called feed-forwarding. The MLP is sometimes also called a feedforward artificial neural network. Its layers are sometimes called fully-connected (FC) layers because all nodes are connected to one another.

2.1.3. Convolutional Neural Network (CNN)

CNNs [17] are one of the most important factors of today's success of deep learning. CNNs build on the idea of MLP, and introduce convolutional layers instead of FC layers to solve computer vision problems. CNNs are specially designed for grid inputs such as an image and a video. They are common inputs in

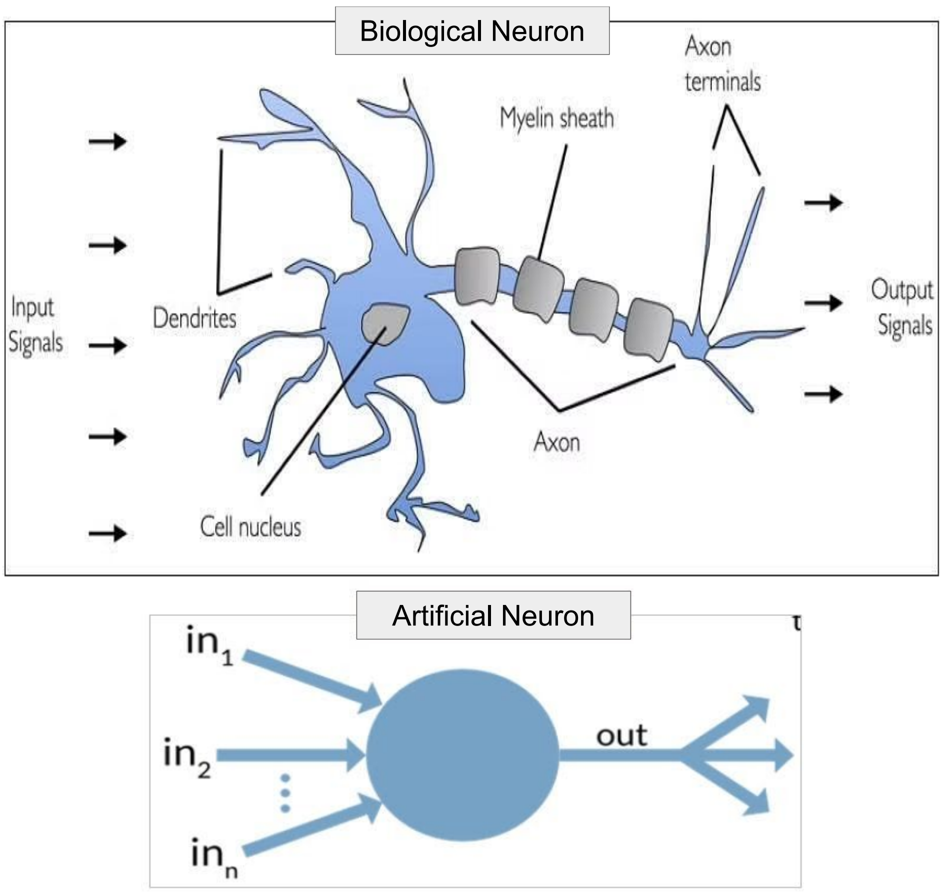


Figure 2.1: Overview of a biological neuron and an artificial neuron. It is interesting to note how similar they are. The figure is composed of two figures thanks to `https://www.simplilearn.com/tutorials/deep-learning-tutorial/perceptron`

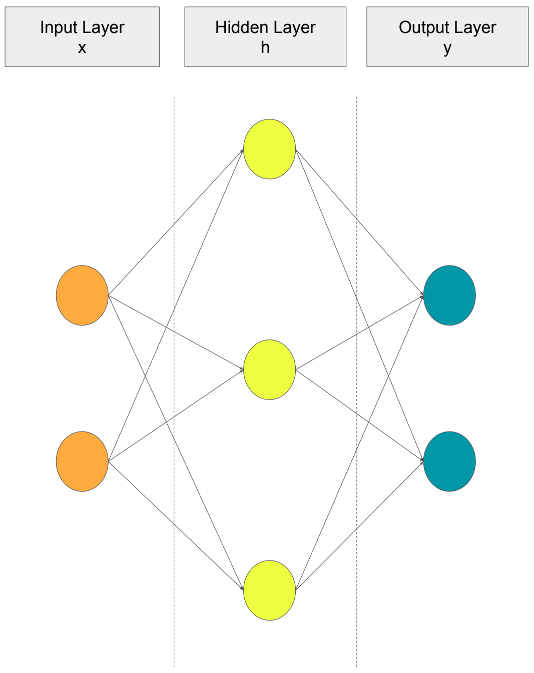


Figure 2.2: Overview of a single hidden layer MLP with 3 hidden neurons. It demonstrates the architecture of the MLP

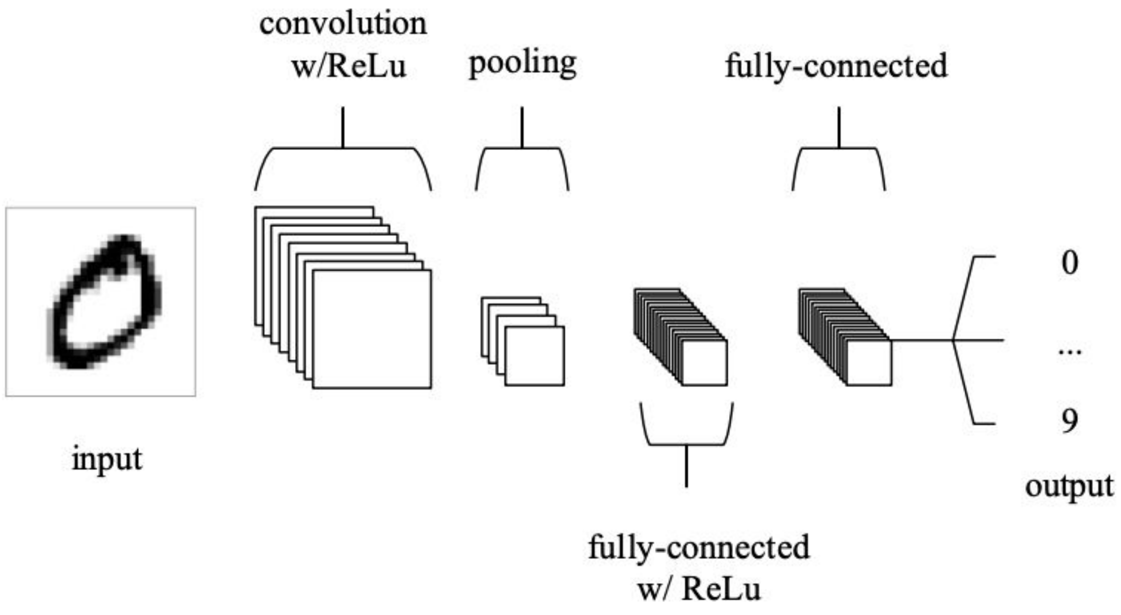


Figure 2.3: Overview of a simple CNN architecture, comprised of a convolutional layer, activation functions, a pooling layer and 2 FC layers [17].

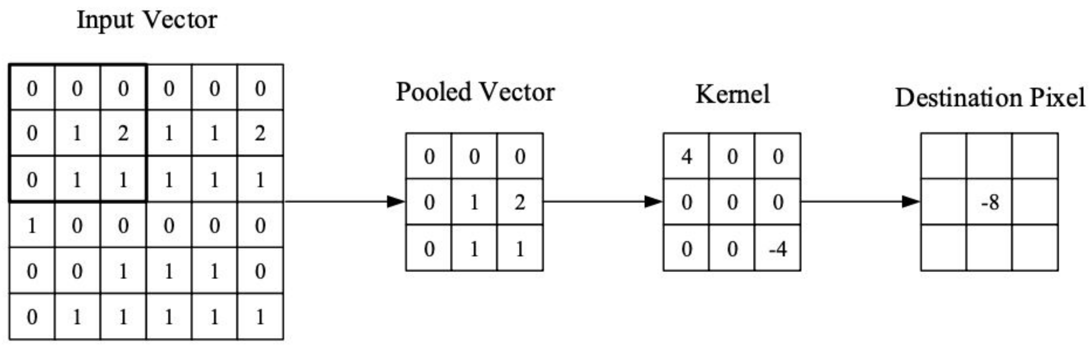


Figure 2.4: A simple calculation happened in a convolutional layer [17]. The destination pixel is calculated by a weighted sum of the input vector.

computer vision tasks. With CNNs, features of images can be extracted by machines instead of hand-crafting. The depth of the model is often related to the number of convolutional layers applied. The depth hence becomes important. It helps in extracting and recombining features. Figure 2.3 depicts a simple CNN architecture. The layers to the outputs are usually comprised of FC layers. We first process the input through a convolutional layer with a non-linear activation function, and then down-sample it by a pooling layer. The convolutional layer, pooling layer and activation functions are explained in details below.

Convolutional Layer

A convolutional layer is the main building block of an CNN. Figure 2.4 explains a simple calculation happened in the convolutional layer. The “Pooled Vector” is what we call a convolution. The center of it is placed on the input vector. The values of it are used as weights of the input vector pixels, and the result of an activation function of the weighed sum of the input vector pixels is the destination pixel. An RGB image contains “depth” dimension. The convolutions can also be 3-dimensional.

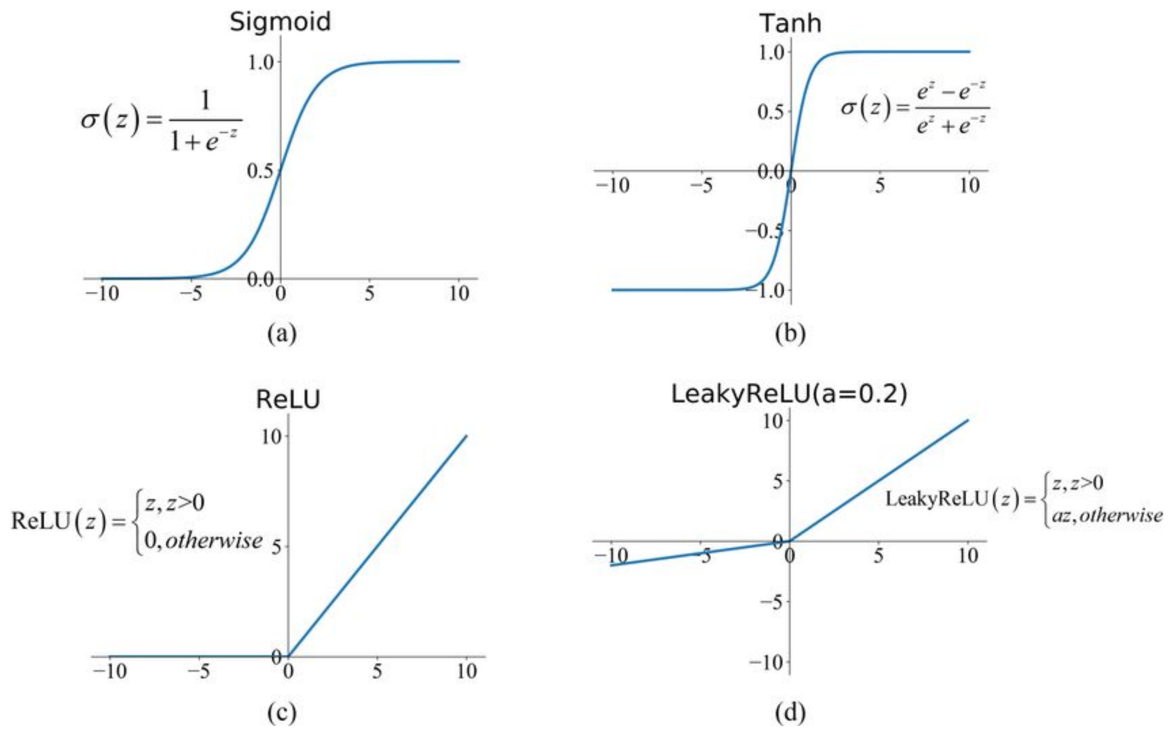


Figure 2.5: Common activation functions used in deep learning [7]. (a) Sigmoid, (b) Tanh, (c) ReLU, and (d) LReLU

Pooling Layer
The pooling layer is used to reduce the dimensionality of the representation of the result of the convolutional layer. With multiple convolutional layers and pooling layers, the complexity of the model is reduced. In the case of a max-pooling layer, the pooling layer kernels are applied to the spatial dimension of the activated results of the convolutional layer, and find the pixel whose value is the highest in its neighborhood. In the case of a 2-by-2 max-pooling layer applied with a stride of 2, the dimensionality of the representation is 4 times smaller each time it passes through a pooling layer.

Activation Function
In Equation 2.1, $w_h \cdot \hat{x} + b_h$ is only a linear operation. It can be used to solve linear problems, but not non-linear problems. Activation functions are non-linear functions, which brings non-linearity to the model predictive ability. Common activation functions are depicted in Figure 2.5.

2.2. Optimization

2.2.1. Loss Function

Optimizing the artificial NNs is to find the optimum parameters of the layers by minimizing a metric function, which we call a loss function. The loss function is continuous and end-to-end differentiable. We build on top of Equation 2.2 and calculate the common loss function for regression, i.e. the Mean Squared Error loss, as follows:

$$L = \frac{1}{n}\Sigma(y - \hat{y})^2 \tag{2.3}$$

"n" is the dimensionality of the target continuous label. If the regression label is uni-dimensional, "n" is 1. "y" is the target regression label and "$\hat{y}$" is the predicted regression value.

2.2.2. Gradient Descent

The reason that machine learning has "machine" in it is that the minimization of the loss is fully automatic. This is done by gradient descent. After obtaining the loss through the feedforward process, we need to update all parameters, Θ, of the model in the direction of the negative gradient of the loss

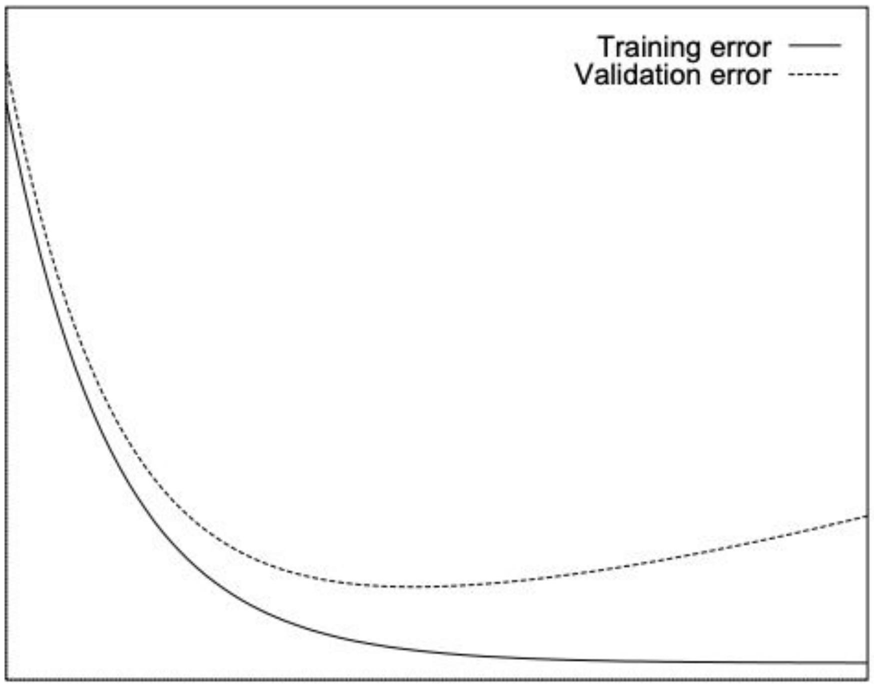


Figure 2.6: A scenario to decide when to stop training early [18]. We should stop training as the validation error curve starts to rise.

function, $-\frac{\delta L}{\delta \Theta}$. Remember that the loss function is differentiable. We use the chain rule of calculus to propagate back how much each parameter of the model should change according to the negative gradient of the loss.

2.2.3. Learning Rate

How much each parameter of the model should change according to the gradient descent is called a “step size”. Learning rate is a hyperparameter used to define an optimal “step size” for the training. If the learning rate is too big, the model can never converge to an optimum. If the learning rate is too small, the training can be very slow and there is a risk for the model to converge to a local optimum. Using right hyperparameters is an art in training deep learning models.

2.3. Regularization

Regularization is the method to avoid the model from being overly trained. An overly-trained model can perform poorly on unseen testing data. This phenomenon is called overfitting. The easiest solution to overfitting is to provide more than enough training data, so that perfectly fitting to the training data is also perfectly fitting to the testing data. However, data are expensive and the testing domain in real-world is large. Therefore, we use regularization methods to avoid overfitting and obtain the best model using limited training data.

2.3.1. Early Stopping

Early stopping is one popular regularization technique. It is easy to use. We take a subset of our training data to form a validation set. This set is not used to update the model. Instead, after each time the model is updated using the rest of the training data, the performance is measured by the validation set. The performance measure can be the loss function used in training, or a different metric. Figure 2.6 shows when we should stop training.

3 Robust Regression

3.1. Definition

Robust regression is about training regression models with noisy, corrupted or imbalanced data and still being able to find the right relationship between the dependent and independent variables and being able to make correct predictions on testing data. Take the MSE method for example, it is highly affected by outliers in the dependent variables. Figure 3.1 shows, with outliers in the data, the degree of the deviation of the MSE method from the true relationship, and how negligible the deviation is when training with a robust regression method [1]. Because noisy and corrupted data are almost unavoidable in the real-world, robust regression becomes important.

3.2. Method

3.2.1. M-Estimator

Huber [14] introduces the M-estimator. The “M” here represents “maximum likelihood”. The M-estimator assumes the true relationship to be the sample average. It achieves robustness both in mean and median estimation. It is robust to outliers in the independent variables. However, it is not better than the MSE method when outliers exist in the dependent variables.

3.2.2. Least Trimmed Squares (LTS)

[22] introduces the LTS. It is better than the M-estimator because it is also robust to outliers in the dependent variables. Instead of minimizing the sum of squares of residuals, as in the MSE method, it minimizes a subset of samples. The rest samples remain unused. In this way, it possibly leaves out outliers in computation.

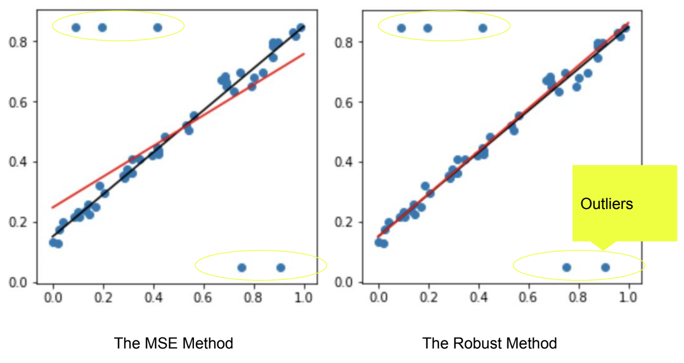


Figure 3.1: A comparison between the MSE method and a robust regression method [1]. Outliers affect the MSE method and not affect the robust method.

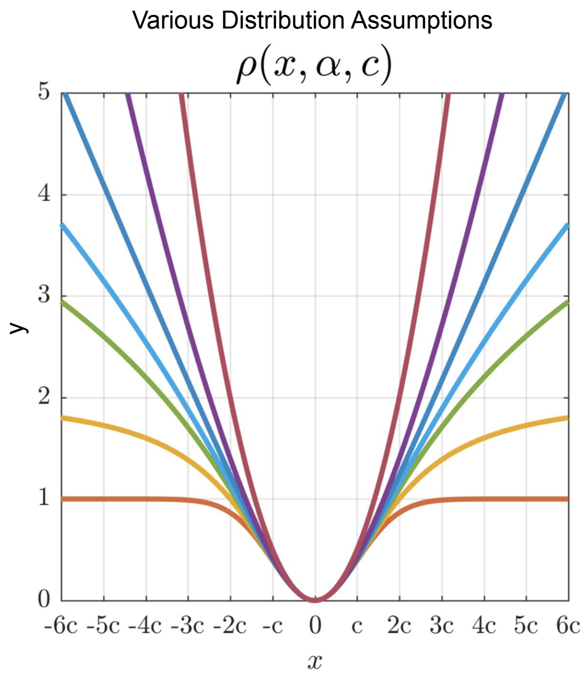


Figure 3.2: A comparison between various distribution assumptions [1]. Different values of α correspond to different loss functions: L2 loss ($\alpha = 2$), Charbonnier loss ($\alpha = 1$), Cauchy loss ($\alpha = 0$), Geman-McClure loss ($\alpha = -2$), and Welsch loss ($\alpha = -\infty$).

3.3. Deep Robust Regression

3.3.1. Noisy or Corrupted Data

[16] introduces a Gaussian negative log likelihood estimator. They assume that the training data is noisy with a Gaussian distribution. If they model the standard deviation of the noise correctly, they can learn the true relationship. They learn the means of the Gaussian model as the true relationship. Their method is only tested on synthetic data. Barron [1] uses a general form of the Gaussian negative likelihood loss. He defines a formula which extends from the Gaussian distribution and can reproduce multiple existing loss functions as shown in Figure 3.2. He finds a way to update the standard deviation parameter, α, and the scale parameter, c, using back propagation, and hence extends manual training to adaptive deep learning.

3.3.2. Imbalanced Data

Unlike the case of noisy or corrupted data where we do not know the true distribution, in the case of imbalanced data, we know the true distribution. The true distribution is to be balanced. [2, 26] use resampling and synthetic data augmentation to balance out the data. [24] uses reweighting. Both of them achieve convincing results. A result of reweighting is shown in Figure 3.3. Reweighting models the testing distribution better than the least squares method, even when both were trained with the same imbalanced distribution.

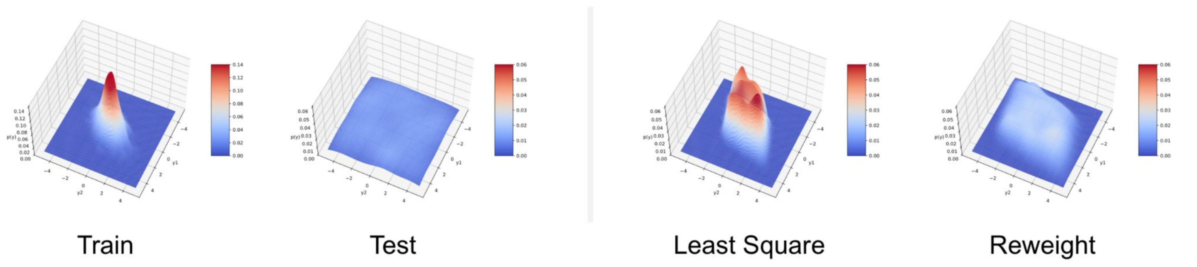


Figure 3.3: Reweighting compared with least squares method [20]. It clearly shows that reweighting models the testing distribution better. Both methods were trained under the training distribution.

4 Object Detection

4.1. Problem Definition

Object detection is the task to automatically detect instances in the images. The instances can be cars, humans, animals, registration plates, etc. Sometimes the scope of an object detection task is broad and requires detection of multiple modalities [6, 25], and the other times the scope of an object detection task can be narrow and only requires one modality, e.g. face detection used in mobile phones [15]. Constructing datasets should also consider negative input samples. Figure 4.1 shows the object detection task in autonomous driving. The model not only detects various instances successfully, classifies them correctly, but also detects remote small instances successfully. In real-world, the task often requires the model to process the input in real-time, which means e.g. 30 frames per second.

4.2. Milestone Detector

Figure 4.2 shows the development of object detectors over the past two decades. Before 2012, deep learning was not used in object detection. Object detectors have to use sophisticated handcrafted features. At that time, interesting feature descriptors were investigated and some of them are still highly useful today, e.g. Histogram of Oriented Gradients (HOG) feature descriptor [4]. RCNN [9] leads the era of two-stage detectors. The two-stage detectors follow a “coarse-to-fine” scheme, and first detect the bounding boxes of any instances and then define what classes these instances belong. YOLO [19] leads the era of one-stage detectors. Instead of first proposing detection and then refining the proposal, YOLO divides the image into subdivisions and predicts bounding boxes and their probabilities at the same time. One-stage detectors are faster in speed, but lower in localization accuracy, than two-stage detectors. MS-CNN [3] is a two-stage detector. It can solve multiscale problems. For example, in the KITTI [8] dataset, which is a dataset about objects on the road, pedestrians, cyclists and cars are objects of different scales and all need detecting. MS-CNN is suitable for this problem.

4.3. Application and Difficulty

Object detection is the basic task for many computer vision tasks. For example, the object orientation estimation task first requires correct pedestrian, cyclist and car detection. Facial Landmark localization first requires correct face detection. Human pose estimation first requires correct human detection. Here, we introduce the pedestrian detection task and its difficulties and challenges.

4.3.1. Pedestrian Detection

Pedestrian detection is an important task in autonomous driving. The success of Faster RCNN [21] has promoted the progress of this area. The difficulties of pedestrian detection are:

- In real-world, the pedestrians can be captured from remote distances by cameras. The number of pixels that comprise such pedestrians may be very few. In a benchmark dataset for pedestrian detection [5], 15% of pedestrian objects are less than 30 pixels in height. Such data are hard for correct detection and even harder for further tasks.

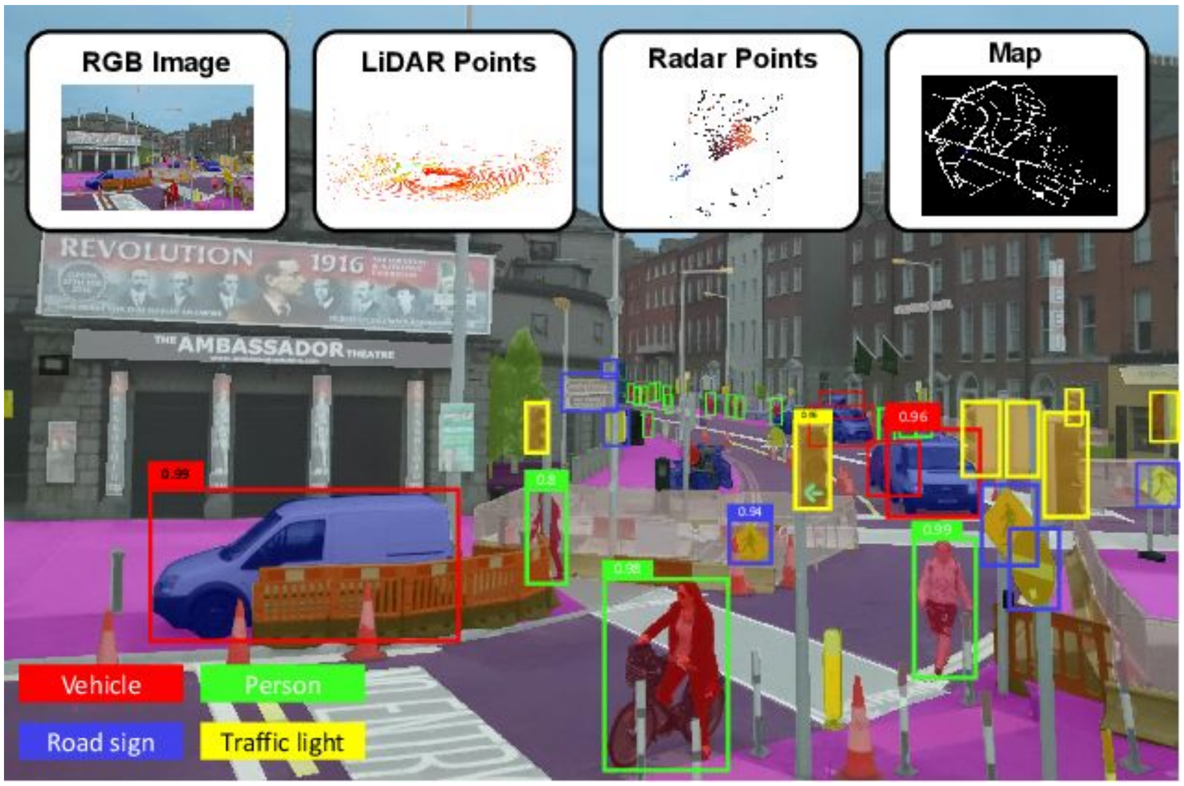


Figure 4.1: A multi-modality object detection task used in autonomous driving of vehicles [7]. It shows the difficulty of object detection and the success of it. It successfully detects various instances and small instances.

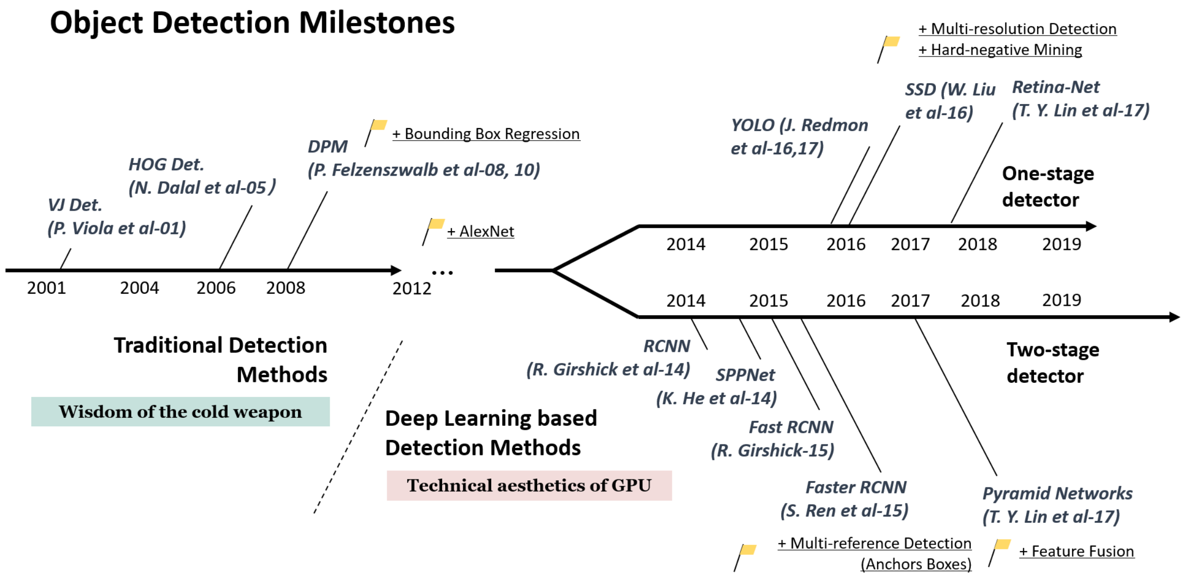

Object Detection Milestones
VJ Det. (P. Viola et al-01)
HOG Det. (N. Dalal et al-05)
DPM (P. Felzenszwalb et al-08, 10)
+ Bounding Box Regression
+ AlexNet
2001
2004
2006
2008
2012
Traditional Detection Methods
Wisdom of the cold weapon
Deep Learning based Detection Methods
Technical aesthetics of GPU
YOLO (J. Redmon et al-16,17)
SSD (W. Liu et al-16)
Retina-Net (T. Y. Lin et al-17)
+ Multi-resolution Detection
+ Hard-negative Mining
One-stage detector
2014
2015
2016
2017
2018
2019
Two-stage detector
RCNN (R. Girshick et al-14)
SPPNet (K. He et al-14)
Fast RCNN (R. Girshick-15)
Faster RCNN (S. Ren et al-15)
Pyramid Networks (T. Y. Lin et al-17)
+ Multi-reference Detection (Anchors Boxes)
+ Feature Fusion

Figure 4.2: An overview of the milestone object detectors over the past two decades [29]. Object detectors are faster, more robust, and suitable for multiscale.

- In images on the road, there are many hard negative samples, e.g., the standing traffic signs and the vertical water pipes. They look very similar to pedestrians and are in fact often detected by our trained MS-CNN model.
- Occlusion is a common problem in object detection. Pedestrians are often occluded by other pedestrians and objects on the road. Such data make both training and detection difficult.

5

Object Orientation Estimation

5.1. Problem Definition

Object orientation estimation is about predicting the orientation where the object faces. The target facing orientation is a continuous scalar. It spans from 0 to 360 degrees. Often, we care about objects that can move, e.g. cyclists, cars and pedestrians. We do not care about e.g. traffic signs and lights. Estimating the orientation can help in predicting the situation that is about to happen, and prevent e.g. collisions on the road. Figure 5.1 shows an example scenario of object orientation estimation on the road.

5.2. Methods

People use mainly 3 methods in object orientation estimation. They are:

- representing the orientation as a point on a unit circle and minimizing the MSE loss,
- representing the orientation as an angle scalar and minimizing the angular difference by the cosine similarity,
- representing the orientation as a pair of a sine and a cosine and minimizing two MSE losses.

[12] reports that the third method is the best.

5.3. Quantitative Evaluation

Evaluation of object orientation estimation requires both evaluation of object detection and evaluation of object orientation estimation within the bounding box. To evaluate object detection, we use the precision and recall. Precision is the ratio of predicted detections that are correct. Recall is the ratio of the number of correctly detected objects divided by all positive objects that need to be detected.

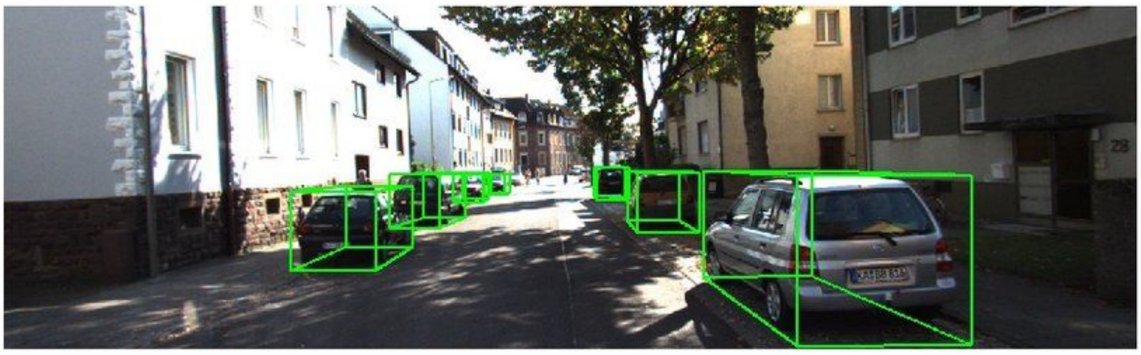


Figure 5.1: An example orientation estimation scenario [23] in real-world. Objects can be occluded. They have different lighting conditions. Their scales are different.

Using the recall of the object detection and cosine similarity between the predicted angle and the target angle, we evaluate the object orientation estimation using the Average Orientation Similarity (AOS) as follows:

$$AOS = \frac{1}{11} \Sigma_{r \in \{0,0.1,\dots,1\}} \max_{\tilde{r}:\tilde{r} \geq r} s(\tilde{r}) \tag{5.1}$$

"r" is the recall threshold of the object detection. "$s(\tilde{r})$" is the cosine similarity of all objects detected with a recall higher than "r". $s(\tilde{r}) \in [0, 1]$. Its normalized value over 11 recall thresholds, i.e. the AOS, is also in [0, 1]. "$s(r)$" is defined as:

$$s(r) = \frac{1}{|D(r)|} \Sigma_{i \in D(r)} \frac{1 + cos\Delta_{\Theta}^{(i)}}{2} \delta_i \tag{5.2}$$

"D(r)" is the set of all positively detected objects above the recall threshold "r". "$\Delta_{\Theta}^{(i)}$" is the difference between the predicted and the target orientations. "δ_i" is to record if "i"-th object has been calculated with the ground truth, and it avoids multiple predictions from matching to the same ground truth.

Bibliography

[1] Jonathan T Barron. A general and adaptive robust loss function. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4331–4339, 2019.

[2] Paula Branco, Luís Torgo, and Rita P Ribeiro. Smogn: a pre-processing approach for imbalanced regression. In *First international workshop on learning with imbalanced domains: Theory and applications*, pages 36–50. PMLR, 2017.

[3] Zhaowei Cai, Quanfu Fan, Rogerio S Feris, and Nuno Vasconcelos. A unified multi-scale deep convolutional neural network for fast object detection. In *European conference on computer vision*, pages 354–370. Springer, 2016.

[4] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, volume 1, pages 886–893. Ieee, 2005.

[5] Piotr Dollár, Christian Wojek, Bernt Schiele, and Pietro Perona. Pedestrian detection: A benchmark. In *2009 IEEE conference on computer vision and pattern recognition*, pages 304–311. IEEE, 2009.

[6] Di Feng, Christian Haase-Schütz, Lars Rosenbaum, Heinz Hertlein, Claudius Glaeser, Fabian Timm, Werner Wiesbeck, and Klaus Dietmayer. Deep multi-modal object detection and semantic segmentation for autonomous driving: Datasets, methods, and challenges. *IEEE Transactions on Intelligent Transportation Systems*, 22(3):1341–1360, 2020.

[7] Junxi Feng, Xiaohai He, Qizhi Teng, Chao Ren, Honggang Chen, and Yang Li. Reconstruction of porous media from extremely limited information using conditional generative adversarial networks. *Physical Review E*, 100(3):033308, 2019.

[8] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE conference on computer vision and pattern recognition*, pages 3354–3361. IEEE, 2012.

[9] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[10] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[11] Shijie Hao, Yuan Zhou, and Yanrong Guo. A brief survey on semantic segmentation with deep learning. *Neurocomputing*, 406:302–321, 2020.

[12] Kota Hara, Raviteja Vemulapalli, and Rama Chellappa. Designing deep convolutional neural networks for continuous object orientation estimation. *arXiv preprint arXiv:1702.01499*, 2017.

[13] Trevor Hastie, Robert Tibshirani, Jerome H Friedman, and Jerome H Friedman. *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer, 2009.

[14] Peter J Huber. Robust estimation of a location parameter. In *Breakthroughs in statistics*, pages 492–518. Springer, 1992.

[15] Ashu Kumar, Amandeep Kaur, and Munish Kumar. Face detection techniques: a review. *Artificial Intelligence Review*, 52(2):927–948, 2019.

[16] David A Nix and Andreas S Weigend. Estimating the mean and variance of the target probability distribution. In *Proceedings of 1994 ieee international conference on neural networks (ICNN'94)*, volume 1, pages 55–60. IEEE, 1994.

[17] Keiron O'Shea and Ryan Nash. An introduction to convolutional neural networks. *arXiv preprint arXiv:1511.08458*, 2015.

[18] Lutz Prechelt. Early stopping-but when? In *Neural Networks: Tricks of the trade*, pages 55–69. Springer, 1998.

[19] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[20] Jiawei Ren, Mingyuan Zhang, Cunjun Yu, and Ziwei Liu. Balanced mse for imbalanced visual regression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7926–7935, 2022.

[21] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015.

[22] Peter J Rousseeuw. Least median of squares regression. *Journal of the American statistical association*, 79(388):871–880, 1984.

[23] Vishwanath A Sindagi, Yin Zhou, and Oncel Tuzel. Mvx-net: Multimodal voxelnet for 3d object detection. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 7276–7282. IEEE, 2019.

[24] Michael Steininger, Konstantin Kobs, Padraig Davidson, Anna Krause, and Andreas Hotho. Density-based weighting for imbalanced regression. *Machine Learning*, 110(8):2187–2211, 2021.

[25] Peng Tang, Xinggang Wang, Xiang Bai, and Wenyu Liu. Multiple instance detection network with online instance classifier refinement. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2843–2851, 2017.

[26] Luís Torgo, Rita P Ribeiro, Bernhard Pfahringer, and Paula Branco. Smote for regression. In *Portuguese conference on artificial intelligence*, pages 378–389. Springer, 2013.

[27] Alexander Toshev and Christian Szegedy. Deeppose: Human pose estimation via deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1653–1660, 2014.

[28] Yue Wu and Qiang Ji. Facial landmark detection: A literature survey. *International Journal of Computer Vision*, 127(2):115–142, 2019.

[29] Zhengxia Zou, Zhenwei Shi, Yuhong Guo, and Jieping Ye. Object detection in 20 years: A survey. *arXiv preprint arXiv:1905.05055*, 2019.