



UNIVERSIDADE FEDERAL DO ESTADO DO RIO DE JANEIRO
CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

Privacy-Preserving Training of Monocular Depth Estimators for Autonomous Vehicles
with Federated Self-Supervision, Bayesian Optimization, and Continual Learning

Elton Figueiredo de Souza Soares

Orientador

Dr. Carlos Alberto Vieira Campos

RIO DE JANEIRO, RJ - BRASIL
FEVEREIRO DE 2025

Privacy-Preserving Training of Monocular Depth Estimators for Autonomous Vehicles
with Federated Self-Supervision, Bayesian Optimization, and Continual Learning

ELTON FIGUEIREDO DE SOUZA SOARES

TESE APRESENTADA COMO REQUISITO PARCIAL PARA OBTENÇÃO DO TÍTULO DE DOUTOR PELO PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA DA UNIVERSIDADE FEDERAL DO ESTADO DO RIO DE JANEIRO (UNIRIO).

APROVADA PELA COMISSÃO EXAMINADORA ABAIXO ASSINADA.

Aprovada por:

Documento assinado digitalmente
CARLOS ALBERTO VIEIRA CAMPOS
Data: 01/04/2025 09:38:54-0300
Verifique em <https://validar.iti.gov.br>

D.Sc. Carlos Alberto Vieira Campos — UNIRIO

Documento assinado digitalmente
SIDNEY CUNHA DE LUCENA
Data: 29/03/2025 10:10:10-0300
Verifique em <https://validar.iti.gov.br>

D.Sc. Sidney Cunha de Lucena — UNIRIO

Documento assinado digitalmente
CARLOS EDUARDO RIBEIRO DE MELLO
Data: 28/03/2025 10:34:07-0300
Verifique em <https://validar.iti.gov.br>

D.Sc. Carlos Eduardo Ribeiro de Mello — UNIRIO

Documento assinado digitalmente
MIGUEL ELIAS MITRE CAMPISTA
Data: 25/03/2025 14:35:34-0300
Verifique em <https://validar.iti.gov.br>

D.Sc. Miguel Elias Mitre Campista — UFRJ

Documento assinado digitalmente
JOEL JOSE PUGA COELHO RODRIGUES
Data: 24/03/2025 21:56:45-0300
Verifique em <https://validar.iti.gov.br>

Ph.D. Joel José Puga Coelho Rodrigues — UFPI

RIO DE JANEIRO, RJ - BRASIL
FEVEREIRO DE 2025.

Catálogo informatizado pelo(a) autor(a)

F676	Figueiredo de Souza Soares, Elton Privacy-Preserving Training of Monocular Depth Estimators for Autonomous Vehicles with Federated Self- Supervision, Bayesian Optimization, and Continual Learning / Elton Figueiredo de Souza Soares. -- Rio de Janeiro : UNIRIO, 2025. 155 Orientador: Carlos Alberto Vieira Campos. Tese (Doutorado) - Universidade Federal do Estado do Rio de Janeiro, Programa de Pós-Graduação em Informática, 2025. 1. Aprendizado Federado. 2. Veículos Autônomos. 3. Sistemas de Transporte Inteligentes. I. Vieira Campos, Carlos Alberto, orient. II. Título.
------	---

Dedico esta tese a todos que de alguma forma colaboraram para que eu concluísse este trabalho. Seja por colaborarem diretamente com a realização da pesquisa, seja por colaborarem indiretamente me ensinando, orientando, apoiando, incentivando e/ou inspirando com seus exemplos de dedicação, perseverança e sabedoria.

Agradecimentos

Primeiramente, agradeço à minha excelentíssima esposa, Carla Santos Franco Soares, por estar ao meu lado durante mais esse desafio e por sempre me apoiar e incentivar a seguir em frente. Quando a fé e a paciência me faltaram, você sempre teve de sobra para me convencer de que ainda era possível e não me deixou desistir deste objetivo.

Agradeço à minha mãe Jussara e ao meu pai Ecio, por me concederem a honra de ser filho de pessoas tão incríveis, e por me darem as condições necessárias para estar nos lugares certos, nos momentos oportunos, com a mentalidade adequada para atingir o nível de realização acadêmica, profissional e pessoal que me permita honrar e orgulhar vocês e vossos pais, meus eternamente amados avós (Marleni, João, Alaide e Joaquim) assim como tantos outros familiares e entes queridos que participaram de minha formação como ser humano, em especial aqueles com quem já não podemos confraternizar e compartilhar bons momentos (tia Lena, tio Jorge, tias-avós Penha e Lélia, e tios-avós Luis e Lucas).

Agradeço ao meu orientador, Prof. Carlos Alberto Vieira Campos, pela paciência na orientação, dedicação e incentivo prestados durante mais esta etapa de minha jornada acadêmica, assim como por sua participação em meu desenvolvimento profissional e pessoal, seja pelos seus conselhos ou pelo seu exemplo. Sua experiência e sabedoria foram fundamentais para que eu pudesse dar o meu melhor como aluno e não deixasse de acreditar em minha capacidade de superar os desafios que se apresentaram durante esta jornada. Sou muito grato pelos mais de 10 anos desta amizade e mentoria que mudou minha vida.

Agradeço ao Pró-Reitor da UNIRIO, Prof. Sidney Cunha de Lucena, pela disponibilidade em participar desta banca e contribuir diretamente com mais esta etapa de minha jornada acadêmica. Também gostaria de registrar minha gratidão pela enorme serenidade e seriedade com que vem desempenhando um papel fundamental na gestão de nossa universidade, de modo a garantir que a UNIRIO seja cada vez mais reconhecida como uma universidade de excelência, e que transforme a vida de cada vez mais alunos.

Também gostaria de deixar um agradecimento especial a um amigo, Prof. Leonardo Guerreiro Azevedo, pela generosidade e empatia que sempre dispensou a mim, a todos os seus alunos, estagiários e colegas, e por estar presente e contribuir em tantos momentos importantes de minha formação profissional e acadêmica. É motivo de grande orgulho para mim ter tido a honra de trabalhar com você. Seus conselhos, incentivo e apoio também foram fundamentais para que eu conseguisse atingir este objetivo.

Agradeço profundamente aos demais membros (titulares e suplentes) da banca examinadora Prof. Joel José Puga Coelho Rodrigues, Prof. Miguel Elias Mitre Campista, Prof. Carlos Eduardo Ribeiro de Mello e Prof. Pedro Nuno de Souza Moura por aceitarem o convite e pela disposição em avaliar e contribuir com este trabalho. Me sinto extremamente honrado e grato pelas suas considerações, questionamentos e aprovação deste trabalho. É um privilégio poder contar com uma banca examinadora tão qualificada e comprometida com a produção de pesquisa científica de alta qualidade em nosso país.

Agradeço, também, aos co-autores dos diversos artigos produzidos ao longo deste curso, incluindo aqueles que não foram diretamente relacionados a esta tese, uma vez que também contribuíram significativamente para minha formação como pesquisador. Em especial, agradeço ao Dr. Renan Francisco Santos Souza, com quem tive a oportunidade de colaborar em publicações de alto impacto e ao Dr. Emilio Ashton Vital Brazil pela colaboração que produziu uma das principais contribuições desta tese, cuja publicação me permitiu participar pela primeira vez de uma conferência internacional, de maneira presencial, complementando ainda mais as experiências adquiridas durante este curso.

Agradeço também a instituição UNIRIO, ao corpo docente e servidores pela excelência e qualidade no ensino e serviços prestados. Em especial, agradeço aos servidores Ives, Leandro, Douglas e Alessandra, da secretaria do BSI/PPGI, que desde o meu primeiro dia como aluno foram extremamente solícitos e eficientes no desempenho de suas funções.

Por fim, gostaria de agradecer a IBM Research Brazil, em especial aos meus gerentes (Marcio Moreno, Renato Cerqueira e Viviane Silva), líderes (Vinícius Segura e Guilherme Lima), diretores (Bruno Flach e Ulisses Mello), e demais colegas com quem colaborei e convivi desde que iniciei esta jornada (Raphael Thiago, Marcelo Nery, Max de Bayser, Wallas Santos, Sandro Fiorini, Marcelo Machado, Alan Braz, entre muitos outros), por todo apoio, parceria e inspiração que, certamente, foram fundamentais para que este objetivo fosse alcançado da melhor maneira possível. Trabalhar nesta organização transformou positivamente a minha vida, e também foi determinante para que eu conseguisse concluir o grande desafio que foi cursar este doutorado em um período tão adverso para a sociedade brasileira, e de muitas mudanças no mundo, na IBM e em minha vida pessoal.

Soares, Elton Figueiredo de Souza **Privacy-Preserving Training of Monocular Depth Estimators for Autonomous Vehicles with Federated Self-Supervision, Bayesian Optimization, and Continual Learning**. UNIRIO, 2025. 155 páginas. Tese de Doutorado. Departamento de Informática Aplicada, UNIRIO.

RESUMO

A inteligência artificial tem sido usada para aprimorar sistemas de transporte inteligentes, com ênfase particular em veículos autônomos. A visão computacional é uma área crucial de pesquisa para a implantação bem-sucedida desses veículos. A tarefa de estimativa de profundidade baseada em imagem ganhou atenção significativa nessa área devido à sua relação custo-benefício e ampla gama de aplicações. Sua variação monocular tem sido o foco principal da pesquisa devido à sua versatilidade, uma vez que requer apenas uma câmera. Embora os métodos mais recentes tenham utilizado o aprendizado autossupervisionado com sucesso, eles não consideraram os principais requisitos para uso em veículos autônomos, como preservação da privacidade, consumo de rede reduzido durante o treinamento e distribuição otimizada de custos de computação. Enquanto isso, o aprendizado federado demonstrou ajudar a atender a esses requisitos em vários domínios.

Assim, propomos três métodos que combinam aprendizado federado e autossupervisão para permitir o treinamento de estimadores de profundidade monoculares com eficácia equivalente e eficiência superior em comparação aos métodos atuais. O primeiro método proposto usa a média federada padrão como função de agregação, enquanto o segundo aplica a otimização bayesiana para aprimorar a agregação. No terceiro método, aplicamos média federada padrão combinada com regularização e repetição de experiência para abordar o aprendizado contínuo em cenários de veículos autônomos. Nossos experimentos, conduzidos usando benchmarks públicos, demonstram que nossas soluções alcançam desempenho do estado da arte, com maior eficiência do que soluções anteriores.

Palavras-chave: Estimativa de profundidade monocular, aprendizagem federada, sistemas de transporte inteligentes, veículos autônomos, visão computacional, inteligência artificial.

ABSTRACT

Artificial intelligence has been used to enhance intelligent transportation systems, with a particular emphasis on autonomous vehicles. Computer vision is a crucial area of research for successful deployment of those vehicles. Image-based depth estimation task has gained significant attention in this area due to its cost-effectiveness and wide range of applications. Its monocular variant has been the main focus of research due to its versatility, since it requires only one camera. Although latest methods leveraged self-supervised learning successfully, they did not consider key requirements for use in autonomous vehicles, such as privacy preservation, reduced network consumption during training, and optimized computation cost distribution. Meanwhile, federated learning has been shown to help address these requirements in multiple domains.

Thus, we propose three incremental methods that combine federated learning and self-supervision to enable the training of monocular depth estimators with equivalent efficacy and superior efficiency compared to current methods. The first method proposed uses standard federated averaging as aggregation function, while the second one applied Bayesian optimization to enhance the aggregation. On the third method, we applied standard federated averaging combined with regularization and experience replay to tackle the continual learning in autonomous vehicle scenarios. Our experiments, conducted using public benchmarks, demonstrate that our solutions achieve state-of-the-art performance, with greater efficiency than prior solutions.

Keywords: Monocular Depth Estimation, Federated Learning, Intelligent Transportation Systems, Autonomous Vehicles, Computer Vision, Artificial Intelligence.

Contents

1	Introduction	1
1.1	Motivation and Justification	2
1.1.1	Federated Learning, Privacy Preservation, Regulations, and Policies	4
1.2	Problem	5
1.3	Relationship with Information Systems Research	5
1.4	Objectives	6
1.4.1	Main Objective	6
1.4.2	Partial Objectives	6
1.5	Research Questions	6
1.6	Research Contributions	7
1.7	Underlying Scientific Theories	9
1.8	Research Methods	9
1.9	Publications during the doctoral program	10
1.10	Structure	11
2	Theoretical Background	12
2.1	Smart Mobility and Intelligent Transportation Systems	12

2.1.1	Artificial Intelligence Applications in Intelligent Transportation Systems	13
2.1.2	Computer Vision for Autonomous Vehicles	14
2.2	Monocular Depth Estimation Problem	15
2.2.1	Evolution of Monocular Depth Estimation Solutions	16
2.2.2	Deep Learning for Monocular Depth Estimation	18
2.3	Self-Supervised (or Unsupervised) Federated Learning	20
2.3.1	Self-Supervised Learning	20
2.4	Federated Learning	21
2.4.1	Self-Supervised Federated Learning History	23
2.5	Bayesian Optimization	24
2.5.1	Probabilistic Surrogate Model	25
2.5.2	Acquisition Function	26
2.5.2.1	Probability of Improvement	27
2.5.2.2	Expected Improvement	28
2.5.2.3	Upper/Lower Confidence Bound	29
2.5.2.4	Gaussian Process Hedge	29
2.6	Continual, Incremental, and Lifelong Learning	30
2.6.1	Types of continual learning scenarios	31
2.6.2	Classes of continual learning methods	33
2.6.2.1	Regularization-Based Methods	33
2.6.2.2	Replay-Based or Rehearsal-Based Methods	34
3	Privacy-Preserving Training of Monocular Depth Estimators for Computer Vision in Autonomous Vehicles via Self-Supervised Federated Learning	36
3.1	Introduction	36

3.2	Related Work	37
3.2.1	Self-Supervised Learning for Monocular Depth Estimation . . .	37
3.2.2	Self-Supervised Federated Learning	38
3.3	Proposed Method	40
3.3.1	Self-Supervised Monocular Depth Estimation	40
3.3.1.1	Model Architecture	40
3.3.1.2	Frame Warping	43
3.3.1.3	Photometric Loss	43
3.3.1.4	Mask-Weighted Photometric Loss	44
3.3.1.5	Combined Self-Supervision Loss Function	44
3.3.2	Federated Learning	45
3.4	Evaluation Experiments	47
3.4.1	Datasets and Scenarios	47
3.4.1.1	Centralized Training	48
3.4.1.2	Federated Training with IID samples	48
3.4.1.3	Federated Training with non-IID samples	48
3.4.2	Metrics	49
3.4.3	Implementation Details	51
3.4.4	Ablation Studies	52
3.4.4.1	Impact of number of participants per round	52
3.4.4.2	Impact of number of local epochs	52
3.4.4.3	Impact of number of federation rounds	55
3.4.4.4	Impact of data heterogeneity	56
3.4.4.5	Final Recommendation	57

3.4.5	Comparison with Centralized Training	57
3.4.6	Comparison with State-of-the-Art on SSL-based monocular depth estimation	60
3.5	Discussion	61
3.6	Conclusion	62
4	Enhancing Federated Averaging of Self-Supervised Monocular Depth Estimators for Autonomous Vehicles with Bayesian Optimization	63
4.1	Introduction	63
4.2	Related Work	64
4.2.1	Self-Supervised Federated Learning	64
4.2.2	Self-Supervised Federated Learning for Monocular Depth Estimation in Autonomous Vehicles	65
4.2.3	Bayesian Optimization for Federated Learning	66
4.3	Proposed Method	67
4.3.1	Bayes Optimized FedAvg (BOFedAvg)	68
4.3.2	Bayes Optimized FedSCDepth (BOFedSCDepth)	71
4.4	Evaluation Experiments	72
4.4.1	Datasets	72
4.4.2	Scenarios	72
4.4.3	Metrics	72
4.4.4	Implementation Details	73
4.4.5	Ablation Studies	74
4.4.5.1	Impact of number of participants per round	75
4.4.5.2	Impact of number of local epochs	75
4.4.5.3	Impact of number of federation rounds	76

4.4.5.4	Impact of data heterogeneity	77
4.4.5.5	Impact of Search Range Hyperparameter	77
4.4.5.6	Impact of Acquisition Function Choice	79
4.4.6	Comparison with State-of-the-Art on Self-Supervised FL of Monocular Depth Estimators	79
4.5	Discussion	83
4.6	Limitations and Threats to Validity	83
4.7	Conclusion	84
5	Continual Federated Learning of Monocular Depth for Autonomous Vehicles	85
5.1	Introduction	85
5.2	Related Work	86
5.3	Proposed Methods	87
5.3.1	Experience Replay FedSCDepth	87
5.3.2	Regularized FedSCDepth	89
5.3.3	Experience Replay Regularized FedSCDepth	90
5.4	Evaluation Experiments	92
5.4.1	Datasets and Scenarios	92
5.4.2	Metrics	93
5.4.3	Implementation Details	93
5.4.4	Results Analysis	94
5.4.4.1	Difference Proportions Analysis	94
5.4.4.2	Stability-Plasticity Trade-Off Analysis	96
5.4.4.3	Final Efficacy Metrics Analysis	98
5.4.5	Combination with Bayesian Optimization	99

5.4.5.1	Difference Proportions Analysis	99
5.4.5.2	Stability-Plasticity Trade-Off Analysis	101
5.4.5.3	Final Efficacy Metrics Analysis	102
5.5	Discussion	103
5.6	Conclusion	104
6	Conclusions and Future Work	105
6.1	Key Contributions and Results	105
6.2	Future Research Directions	107
6.2.1	Privacy-Preservation Gaps in Federated Learning	107
6.2.2	Efficiency & Efficacy Gaps in Federated Learning	107
6.2.3	Extended Continual Federated Learning Studies	107
6.2.4	Foundation Models for Monocular Depth Estimation	108
6.2.5	Applications of Federated Learning in Intelligent Transportation Systems	108
	Bibliography	110

List of Figures

2.1	Illustration of Smart Mobility and other Smart City components.	12
2.2	Illustration of the “Expanded Tunnel of Reality” and an example of RGB Image and colorized depth mask.	16
2.3	Evolution history of monocular depth estimation.	17
2.4	Generic model of supervised learning for monocular depth estimation, whose inputs are the RGB image at instant t and the ground truth depth map. The output is the estimated depth map. The loss is computed comparing the ground truth and output maps.	18
2.5	Generic model of semi-supervised learning for monocular depth estimation, whose inputs are the RGB image at instant t and the alternative ground truth signal (Sparse depth map obtained from a LIDAR sensor). The output is the estimated depth map.	19
2.6	Generic model of self-supervised learning for depth estimation based on monocular sequences of RGB images, collected at instants $t - 1$, t and $t + 1$	19
2.7	Example of FL architecture with AVs as the federated nodes.	22
2.8	Visualization of BO with Gaussian Process for 10 iterations using Expectation of Improvement as main acquisition function. The lower chart shows alternative acquisition function results using the Probability of Improvement and Upper Confidence Interval.	27
2.9	Decision tree for the three continual learning scenarios.	31
2.10	Taxonomy of most common categories continual learning methods.	33

3.1	Depth estimation neural network architecture.	41
3.2	Pose estimation neural network architecture.	41
3.3	Self-supervised learning component overview.	43
3.4	Federated self-supervised learning of a monocular depth estimator for AVs.	45
3.5	Sample count by participant in FT-IID and FT-NIID scenarios, with re-distribution of remaining samples, in KITTI (a) and DDAD (b) datasets.	48
3.6	Lowest Λ (a), its $\bar{\mathcal{O}}_{max}$ (c), $\bar{\mathcal{O}}_{min}$ (e), and $\hat{\tau}_{FL}$ to obtain it (g), by number of AVs (ϕ). Lowest Λ (b), its $\bar{\mathcal{O}}_{max}$ (d), $\bar{\mathcal{O}}_{min}$ (f), and $\hat{\tau}_{FL}$ to obtain it (h), by number of epochs (e), in KITTI.	53
3.7	Lowest Λ (a), its $\bar{\mathcal{O}}_{max}$ (c), $\bar{\mathcal{O}}_{min}$ (e), and $\hat{\tau}_{FL}$ to obtain it (g), by number of AVs (ϕ). Lowest Λ (b), its $\bar{\mathcal{O}}_{max}$ (d), $\bar{\mathcal{O}}_{min}$ (f), and $\hat{\tau}_{FL}$ to obtain it (h), by number of epochs (e), in DDAD.	54
3.8	Lowest Λ (a), its $\bar{\mathcal{O}}_{max}$ (b), $\bar{\mathcal{O}}_{min}$ (c), and $\hat{\tau}_{FL}$ (d), by FL rounds elapsed ($ \hat{\mathcal{R}} $), in KITTI.	55
3.9	Lowest Λ (a), its $\bar{\mathcal{O}}_{max}$ (b), $\bar{\mathcal{O}}_{min}$ (c), and $\hat{\tau}_{FL}$ (d), by FL rounds elapsed ($ \hat{\mathcal{R}} $), in DDAD.	56
3.10	Lowest Λ by training steps (τ and $\hat{\tau}_{FL}$), in KITTI (a) and DDAD (b).	57
3.11	$\bar{\mathcal{O}}_{max}$ (a) and $\bar{\mathcal{O}}_{min}$ (b) required to reach lowest Λ , by number of training steps (τ and $\hat{\tau}_{FL}$), in KITTI.	58
3.12	$\bar{\mathcal{O}}_{max}$ (a) and $\bar{\mathcal{O}}_{min}$ (b) required to reach lowest Λ , by number of training steps (τ and $\hat{\tau}_{FL}$), in DDAD.	58
4.1	BOFedAvg overview.	67
4.2	Bayes optimized self-supervised FL of monocular depth estimators for AVs.	67
4.3	Lowest global validation loss (a), its communication cost upper bound (b), and lower bound (c), and number of train steps (d), validation steps (e), and averaging operations (f) to obtain it, by the number of rounds.	76
4.4	Lowest validation loss (a) and number of averaging operations required for obtaining it (b), by value of Ψ for KITTI and DDAD datasets.	77

4.5	Lowest validation loss obtained with varying values of Ψ (a, b), and its required number of validation steps (c, d), and averaging operations (e, f), by the number of training steps, for KITTI and DDAD datasets.	78
4.6	Lowest validation loss (a) and number of averaging operations for obtaining it (b), per acquisition function for KITTI and DDAD datasets.	79
4.7	Lowest validation loss obtained with BOFSCD and FSCD (a, b), and its estimated communication cost upper and lower bounds (c, d), by the number of training steps, for KITTI and DDAD datasets.	80
4.8	The number of validation steps (a, b) and the number of averaging operations (c, d), by the number of training steps, for KITTI and DDAD datasets.	81
5.1	ERFedSCDepth overview.	87
5.2	RFedSCDepth overview.	90
5.3	ERRFedSCDepth overview.	91
5.4	Difference proportion results obtained when retraining a model from DDAD on KITTI. According to Equation 5.2, positive values in (a) and (b) indicate a reduction in error metrics, while negative values in (c) and (d) indicate an increase in accuracy metrics.	95
5.5	Difference proportion results obtained when retraining a model from KITTI on DDAD. According to Equation 5.2, positive values in (a) and (b) indicate a reduction in error metrics, while negative values in (c) and (d) indicate an increase in accuracy metrics.	96
5.6	Normalized SPTO error (err) metrics (a, c) and accuracy (acc) metrics (b, d) obtained with test dataset, when retraining a model from DDAD to KITTI, and vice versa.	97
5.7	Difference proportion results when retraining a model from DDAD on KITTI.	100
5.8	Difference proportion results when retraining a model from KITTI on DDAD.	101

5.9	Normalized SPTO error (err) metrics (a, c) and accuracy (acc) metrics (b, d) obtained with test dataset, when retraining a model from DDAD to KITTI, and vice-versa.	102
-----	--	-----

List of Tables

3.1	Characterization of self-supervised FL methods, regarding their use on AVs (AV) based on the datasets in which they were tested, and if they were evaluated on computer vision (CV) tasks with IID and non-IID (NIID) data.	38
3.2	Efficacy comparison with CT on KITTI Dynamic Regions (Eigen Split).	59
3.3	Efficacy comparison with CT on KITTI Static Regions (Eigen Split).	59
3.4	Efficacy comparison with CT on DDAD Dynamic Regions.	60
3.5	Efficacy comparison with CT on DDAD Static Regions.	60
3.6	Efficacy Comparison with state-of-the-art on KITTI (Eigen Split).	60
3.7	Efficacy Comparison with state-of-the-art on DDAD.	61
4.1	Characterization of self-supervised FL works regarding their use of BO, applicability to AVs (AV), evaluation on computer vision (CV) tasks, experimentation with IID and NIID, and Datasets used.	64
4.2	Lowest global validation loss (Λ_{min}), its max and min communication costs ($\mathcal{O}_{max}/\mathcal{O}_{min}$), number of training steps (τ_{train}), validation steps (τ_{val}) and averaging operations (η), by number of active AVs per round ($ \mathbb{U} $), grouped by dataset (DS) and FT scenarios (Scenario). *Best results per group in bold.	74
4.3	Lowest global validation loss (Λ_{min}), its $\mathcal{O}_{max}/\mathcal{O}_{min}$, number of training steps (τ_{train}), validation steps (τ_{val}) and averaging operations (η), by number of active AVs per round ($ \mathbb{U} $), grouped by dataset (DS) and FT Scenarios. *Best results per group in bold.	74

4.4	Lowest global validation loss (Λ_{min}), its max and min communication costs ($\mathfrak{O}_{max}/\mathfrak{O}_{min}$), number of training steps (τ_{train}), validation steps (τ_{val}) and averaging operations (η), by number of epochs (e), grouped by dataset (DS) and FT scenarios (Scenario).	75
4.5	Lowest global validation loss (Λ_{min}), its max and min communication costs ($\mathfrak{O}_{max}/\mathfrak{O}_{min}$), number of training steps (τ_{train}), validation steps (τ_{val}) and averaging operations (η), by number of epochs (e), grouped by dataset (DS) and FT scenarios (Scenario).	75
4.6	Efficacy of BOFSCD (Ours) and FSCD (Baseline) methods on KITTI (Eigen Split) dataset (DS), at round (r), grouped by DS and r . *Best results per group in bold.	81
4.7	Efficacy of BOFSCD (Ours) and FSCD (Baseline) methods on DDAD dataset (DS), at round (r), grouped by DS and r . *Best results per group in bold.	82
5.1	Abbreviations of method names.	94
5.2	SPTO of (non-normalized) error and accuracy metrics obtained with test datasets, when retraining a model learned with Dataset 1 (DS1) on Dataset 2 (DS2).	97
5.3	Efficacy of models trained with DDAD and retrained on KITTI with all methods, evaluating on Test Dataset (TD). *Best of results of each scenario highlighted in bold.	98
5.4	Efficacy of models trained with KITTI and retrained on DDAD with all methods, evaluating on Test Dataset (TD). *Best of results of each scenario highlighted in bold.	98
5.5	Abbreviations of method names (with BO).	99
5.6	SPTO of (non-normalized) error and accuracy metrics obtained with test datasets, when retraining a model learned with Dataset 1 (DS1) on Dataset 2 (DS2).	101
5.7	Efficacy of models trained with DDAD and retrained on KITTI with all methods, evaluating on Test Dataset (TD). *Best of results of each scenario highlighted in bold.	103

5.8	Efficacy of models trained with KITTI and retrained on DDAD with all methods, evaluating on Test Dataset (TD). *Best of results of each scenario highlighted in bold.	103
-----	---	-----

1. Introduction

The detrimental effects of an inadequately managed transportation system on overall quality of life have led to Smart Mobility being seen as a key strategy to transition to more environmentally friendly and sustainable transportation systems [1]. Alternatively, it can be viewed as a comprehensive set of initiatives to improve urban efficiency, effectiveness, and environmental sustainability. One such initiative is the development of Intelligent Transportation Systems (ITS), which has been in progress since the 1970s and can be characterized by the integration of advanced technologies [2].

Substantial research efforts have been dedicated to the integration of Big Data analytics and cutting-edge Artificial Intelligence (AI) methodologies to improve ITS performance [3]. In contrast, relatively less attention has been devoted to developing intelligent agents to support ITS. The primary focus in this regard has been on the development of Autonomous Vehicles (AVs), which have emerged as a pivotal area of investigation within the ITS initiative [4]. AV research has also used advanced AI techniques to tackle its most critical tasks, including computer vision.

Scene depth estimation plays a crucial role in computer vision, as it allows the perception and comprehension of three-dimensional scenes [5]. Traditionally, active depth estimation methods have relied on lasers, structured light, and other reflections on the object surface [5]. However, these approaches often require substantial investments in human labor and computational resources to obtain dense and accurate depth maps [6]. To overcome these limitations, researchers have been exploring innovative solutions to enable efficient and cost-effective scene depth estimation, which is essential for the widespread adoption of AVs and ITS. Consequently, image-based depth estimation has emerged as a primary area of investigation in computer vision for AVs, due to its reduced deployment costs and the broader range of applicable scenarios [5].

Image-based depth estimation techniques typically rely on calculating the disparity

between two 2D images captured by a binocular camera to generate a depth map [7]. However, binocular depth estimation methods require the presence of at least two fixed cameras, which can be a limitation, particularly in scenarios where the scene lacks sufficient texture or has minimal features [8]. Provided that, research shifted its attention to monocular depth estimation [9]. The appeal of monocular depth estimation lies in its ability to utilize a single camera to capture an image or video sequence, thus eliminating the need for specialized equipment [5]. However, the regression of depth in 3D space from monocular images is an inherently ill-posed problem due to the lack of a reliable stereoscopic visual relationship [5].

In particular, monocular images are reduced to a 2D representation of the 3D world, which precludes the capture of depth information. As a result, it is impossible to infer the size and distance of an object in the scene or determine whether it is occluded by another object [5]. This limitation stems from the fact that the imaging process does not provide direct information about the depth of the scene, making it challenging to accurately perceive and understand the 3D environment. Therefore, it is essential to infer the depth of each pixel from the monocular image. Using the resulting pixel depth map, it is possible to determine the size and distance of the objects present in the scene. When the estimated depth accurately captures the 3D topology of the scene, the depth estimation method used can be deemed effective [5].

Finally, one important note is that today’s consumer vehicles are already equipped with several sensors, such as LIDAR and Birdseye (360 degrees) cameras. Although these advanced sensors can be used for depth estimation [10–12], we argue that monocular depth estimation represents a low-cost backup strategy that can be leveraged by AVs in scenarios in which other methods may fail, become unavailable, or unfeasible.

1.1 Motivation and Justification

Recent advances in monocular depth estimation have leveraged self-supervised learning of deep neural networks in conjunction with other computer vision tasks, such as ego-motion (pose estimation) and semantic segmentation [13, 14]. However, existing solutions for monocular depth estimation have yet to address the critical requirements of AV deployments, such as:

1. Preserving privacy when utilizing unlabeled private data captured by multiple AVs;
2. Minimizing network consumption, as the volume of data that needs to be shared

through ITS network infrastructure may render the learning process infeasible;

3. Efficient distribution of computational costs, as the concentration of all training processing on a central server may result in suboptimal utilization of the combined computing power of AVs;
4. Resilience to connectivity issues, as the transfer of all training data to the central server before training may render the system highly reliant on a stable connection, which is unlikely to occur in realistic AV deployments.

These limitations underscore the need for novel approaches that can effectively address these challenges and enable the development of robust and efficient monocular depth estimation methods for AVs.

Federated learning (FL) has been extensively explored in recent research on ITS and AVs [15, 16] to address the aforementioned requirements. The key benefits of FL [17] include [18]:

1. Enhanced privacy preservation, as the need to share raw data collected by individual vehicles with a central server or other vehicles is eliminated;
2. Reduced network consumption, as the size of model updates transmitted during the FL process is significantly smaller than the raw data;
3. Distributed computational cost allocation, which leverages the collective computational power of cooperating AVs rather than relying solely on a central server;
4. Increased resilience to connectivity disruptions when compared to centralized approaches.

Furthermore, Bayesian Optimization (BO) has been widely used in recent studies [19] for automated machine learning and has been used to improve local training in FL frameworks [20]. Unlike traditional optimization methods, such as grid or random search, BO takes advantage of past evaluations of an objective function and an acquisition function to select new inputs that maximize the expected improvement in outputs [21]. Consequently, BO typically identifies better inputs with fewer objective function evaluations [22].

Moreover, continual learning presents an additional challenge for both centralized and federated training, as updating models with novel datasets may induce the forgetting of previously acquired samples (catastrophic forgetting problem [23, 24]), thus diminishing the model’s versatility. Meanwhile, recent research indicates that experience replay and

regularization strategies can be used to alleviate forgetting in autonomous driving use cases [24].

1.1.1 Federated Learning, Privacy Preservation, Regulations, and Policies

FL has an intrinsic advantage in protecting privacy compared to “traditional” centralized machine learning approaches [25]. From a naive perspective, it could be considered sufficient to comply with recent privacy-related regulations put forth by nations across the world (and derived policies being adopted by public and private organizations as a result of their enforcement) such as the EU / UK General Data Protection and Regulation (GDPR) [26,27], Brazil’s *Lei Geral de Proteção de Dados* (LGPD) [28], and the California Consumer Privacy Act (CCPA) [29] from the USA, because private (and potentially personal) training data is stored and processed locally, and only model parameters are exchanged over the network with one or more aggregation servers [25]. However, several works have already demonstrated and discussed multiple ways through which sensitive information can be inferred from the model parameters that are shared in federated training, indicating that the use of FL alone is far from providing privacy guarantees and therefore cannot ensure full compliance with privacy-related regulations and policies [30].

Meanwhile, a substantial body of research is already being produced to combine traditional privacy preservation techniques into the FL framework [31] that demonstrates the potential of FL as the baseline for a new era of privacy-preserving AI. Therefore, in the present thesis, FL is leveraged as a Privacy Enhancing Technology (PET) [32–38], which means that it increases the level of privacy-preservation of the training of monocular depth estimators but, by no means, guarantees the privacy of all the personal information potentially present in the local training data, since the provision of such guarantee would require the combination of multiple PETs to an extent that is still challenging for most research and industry organizations across the world, and that could still be eventually breached with the advancement of the cyberattackers’ knowledge and skills, and computer science in general (*e.g.*, Quantum Computing [39,40]). Thus, when we refer to privacy preservation in the present work, it should only be interpreted as reducing the exposure of personal data or making it harder to access sensitive information without authorization, which can be true by leveraging only one, or many of the known PETs in the literature, such as FL.

1.2 Problem

This work focuses on the monocular depth estimation problem, focusing on instances of this problem associated with AV deployments.

In a supervised view, the monocular depth estimation problem can be framed as a continuous regression problem. Given a training set $\mathcal{T} = \{(I_i, D_i)\}_{i=1}^M$, where M is the number of samples, $I_i \in I$ is the set of input images and $D_i \in \mathcal{D}$ is the set of ground truth depth maps, the problem can be viewed as learning a nonlinear transformation $\Phi : I \rightarrow \mathcal{D}$, which takes an image as input and outputs a corresponding depth map [41]. However, in an unsupervised (or self-supervised) view, this problem can be formulated in several ways, where, instead of fitting the model based on ground truth depth maps, training will be performed using an alternative loss function based on a pretext task, such as photometric self-consistency loss [42]. Thus, in a more general sense, the monocular depth estimation problem can be defined as the problem of training a model to infer the correct depth of each pixel of a two-dimensional image captured by a monocular camera. In the AV use case, such a camera would be mounted on top of a moving vehicle, and this inference would need to be performed in a timely manner, to allow the vehicle to understand the three-dimensional world around it and make the correct decisions/actions based on it.

1.3 Relationship with Information Systems Research

An information system can be defined technically as a set of components that collect (or retrieve), process, store, and distribute information to support decision-making and control in an organization [43]. In addition to decision-making, coordination, and control, information systems can also help managers and workers analyze problems, visualize complex issues, and create new products [43].

Information systems contain information about people, places, and things within the organization or in the environment surrounding it [43]. By information, we mean data that have been shaped into a form that is meaningful and useful to human beings. Data, in contrast, are streams of facts that represent events that occur in organizations or the physical environment before they have been processed and organized in a form that people can understand and use [43].

In the context of ITS, we can consider governments as organizations. Vehicles, personal mobile devices, and sensors in road infrastructure can be seen as data sources. In this context, we can consider ITS as systems of systems whose main objective is to max-

imize the quality of life of a city's citizens through efficient and effective mobility policies. The establishment and adaptation of these policies become possible through the use of information systems specialized in extracting knowledge from data collected through physical and virtual sensors in cities, and their implementation can be supported by information systems that provide useful information for decision-making at different levels of government management, as well as triggering actuators in integrated infrastructure.

1.4 Objectives

This section presents the main and partial objectives of this work.

1.4.1 Main Objective

The main objective of this work is to develop solutions for the problem of monocular depth estimation for AVs. These solutions must enable the generation of depth maps of images captured by monocular cameras in moving AVs with high efficacy and efficiency, privacy preservation, reduced network consumption, improved computational cost distribution, and increased resiliency to connectivity loss.

1.4.2 Partial Objectives

The partial objectives of this work can be summarized as follows.

- Develop a solution to the problem of monocular depth estimation in AVs using self-supervised FL for collaboratively training a depth estimator using unlabeled Independently and Identically Distributed (IID) and non-IID data captured by vehicles with high efficacy, efficiency, and privacy-preservation;
- Leverage BO to increase the efficacy and efficiency of the solution;
- Leverage regularization and experience replay to reduce catastrophic forgetting in continual learning scenarios in which the solution developed can be deployed.

1.5 Research Questions

The present work will try to answer the following Research Questions (RQs):

1. **RQ1** - Is the *efficiency* of the self-supervised learning of monocular depth estimation models *greater* when applying FL (with IID and non-IID data) or a centralized approach ¹, in the AV use case?
2. **RQ2** - Is the *efficacy* of self-supervised learning monocular depth estimation models *equivalent* when applying FL (with IID and non-IID data) instead of a centralized approach, in the AVs use case?
3. **RQ3** — Is the *efficiency* of the self-supervised FL of monocular depth estimation models *improved* when combining BO with FL instead of using standard FL, with IID and non-IID data, in the AV use case?
4. **RQ4** - Is the *efficacy* of self-supervised FL of monocular depth estimation models *preserved* when combining BO with FL instead of using standard FL, with IID and non-IID data, in the AV use case?
5. **RQ5** - Does the integration of experience replay with FL *enhance* the *efficacy* of continual self-supervised FL for monocular depth estimation compared to naive continual FL, within the AV domain?
6. **RQ6** - Does the integration of regularization with FL *enhance* the *efficacy* of continual self-supervised FL for monocular depth estimation compared to naive continual FL, within the AV domain?
7. **RQ7** - Does the combined integration of experience replay and regularization with FL *enhance* the *efficacy* of continual self-supervised FL for monocular depth estimation compared to naive continual FL, within the AV domain?

1.6 Research Contributions

The main Research Contributions (RCs) of this work are as follows:

1. **RC1** - Proposal of FedSCDepth method to solve the problem of monocular depth estimation in AVs using self-supervised FL for collaboratively training a depth estimator using unlabeled data captured by vehicles with high efficacy, efficiency, and privacy;
2. **RC2** - Evaluation of FedSCDepth prototype implementation using a real monocular depth estimation dataset for AVs, demonstrating its state-of-the-art performance,

¹Training a model on a central server with data collected by all vehicles.

lower computation, and communication costs compared to centralized training, across IID and non-IID data scenarios.

3. **RC3** - Proposal of the Bayes Optimized FedAvg (BOFedAvg) aggregation function, as an extension of the proposal in [44], offering a flexible formalization that leverages the weights of FL aggregation and the associated global loss as a robust prior for BO.
4. **RC4** - Proposal of the BOFedSCDepth method, which combines FedSCDepth [45, 46] and BOFedAvg. This method addresses the monocular depth estimation problem in AVs, enabling collaborative training of a depth estimator with unlabeled private data.
5. **RC5** - Empirical evaluation of a prototype of the BOFedSCDepth using two state-of-the-art benchmark datasets (KITTI [47] and DDAD [48]) for monocular depth estimation in AVs.
6. **RC6** - We show that BOFedSCDepth achieves results in self-supervised FL that match the state-of-the-art for monocular depth estimation, with up to 40.1% lower test loss than our baseline and up to 33.3% reduction in communication cost, using IID and Non-IID data.
7. **RC7** - Proposal of Experience Replay FedAvg (ERFedAvg) aggregation function that leverages the shared validation sets of previously seen datasets during model aggregation to reduce catastrophic forgetting.
8. **RC8** - Proposal of the Regularized FedAvg (RFedAvg) aggregation function that takes advantage of the global model trained on previous datasets during the aggregation step of the training with a new dataset to regularize the weights of the new global model.
9. **RC9** - Proposal of Experience Replay Regularized FedAvg (ERRFedAvg) aggregation function that combines the aggregation strategies of ERFedAvg and RFedAvg.
10. **RC10** - ERFedSCDepth, RFedSCDepth, and ERRFedSCDepth federated training methods, which combine FedSCDepth [45] and ERFedAvg, RFedAvg, and ERRFedAvg, respectively.
11. **RC11** - Empirical comparison of FedSCDepth, ERFedSCDepth, RFedSCDepth, and ERRFedSCDepth continual learning effectiveness using two state-of-the-art benchmark datasets (KITTI [47] and DDAD [48]) for monocular depth estimation in AVs.

1.7 Underlying Scientific Theories

The present work, and much of the research in ITS, are based on the same theories that most AI methods and tools are based upon. For instance, connected AVs are essentially based on three theories of Information Systems and Computer Science: General Systems Theory, Information Processing Theory, and Computational Learning Theory.

General Systems Theory [49] defines a system as a cohesive conglomerate of inter-related and interdependent parts that is natural or man-made. In addition, it defines that some systems function to support other systems, such as the information systems that support ITS and the computer vision systems and subsystems that support AV operation.

Information Processing Theory [50] defines fundamental concepts for the information systems used in this research, such as the concept of sensory memory, instantiated by the sensors used for data collection, short-term memory, instantiated in the various computational components present in vehicles, mobile devices and infrastructure, and long-term memory, instantiated by the knowledge bases and datasets generated during the various stages of the knowledge discovery process in ITS.

Finally, the Computational Learning Theory [51] studies the design and analysis of machine learning algorithms. In particular, such algorithms aim to make accurate predictions or representations based on observations, a fundamental task in the extraction of knowledge necessary for the development of depth estimation solutions and the establishment and adaptation of more efficient mobility policies overall.

1.8 Research Methods

The research methods used in this work fall into the category of quantitative research methods. Quantitative methods encompass a set of techniques designed to answer research questions through the analysis of quantitative data [52]. These methods are grounded in two main principles: a focus on numerical data and a commitment to a positivist philosophy [53].

The first principle involves using numbers to represent theoretical constructs, and the interpretation of these numbers provides strong scientific evidence to understand phenomena. Although statistical tools are commonly associated with quantitative methods, they are not always required. The second principle reflects a positivist approach to research [54], which is underpinned by a realist ontology [55] (the belief in an objective,

fixed, and independent reality) and an objectivist ontology [56] (the notion that reality and truth can be objectively discovered and measured).

Furthermore, empiricist epistemology [57] of quantitative research emphasizes that knowledge is derived from sensory experiences and empirical observation. These fundamental principles guide the application of quantitative methods to explore real-world events, uncover relationships, and establish general laws that can be documented and explained [52].

1.9 Publications during the doctoral program

The following peer-reviewed research articles were published (or accepted for publication) as a direct result of this work, until the moment in which this thesis was published:

- [45]: **Soares, Elton F. de S.**, Campos, Carlos Alberto V. “*Privacy-Preserving Training of Monocular Depth Estimators via Self-Supervised Federated Learning*”. In **IEEE 100th Vehicular Technology Conference (VTC2024-Fall)**, Washington DC, United States, 2024, pp. 1-6. doi: 10.1109/VTC2024-Fall63153.2024.10757815.
- [58]: **Soares, Elton F. de S.**, Brazil, Emilio V., Campos, Carlos Alberto V. “*Leveraging Bayesian Optimization to Enhance Self-Supervised Federated Learning of Monocular Depth Estimators*”. In **IEEE 27th International Conference on Intelligent Transportation Systems (ITSC 2024)**, Edmonton, Canada, 2024.
- [59]: **Soares, Elton F. de S.**, Campos, Carlos Alberto V. “*Towards Continual Federated Learning of Monocular Depth for Autonomous Vehicles*”. In **IEEE 100th Vehicular Technology Conference (VTC2024-Fall)**, Washington DC, United States, 2024, pp. 1-5. doi: 10.1109/VTC2024-Fall63153.2024.10757948.
- [60]: **Soares, Elton F. de S.**, Campos, Carlos Alberto V. “*Self-Supervised Learning of Monocular Depth Estimators in Autonomous Vehicles with Federated Learning*”. In **Engineering Applications of Artificial Intelligence** vol 151, 110572, July. 2025, doi: 10.1016/j.engappai.2025.110572.
- [61]: **Soares, Elton F. de S.**, Brazil, Emilio V., Campos, Carlos Alberto V. “*Enhancing federated averaging of self-supervised monocular depth estimators for autonomous vehicles with Bayesian optimization*”. In **Future Generation Computer Systems** vol 167, 107752, June. 2025, doi: 10.1016/j.future.2025.107752.

1.10 Structure

This section aims to provide an overview of the structure and content of this doctoral thesis, presenting the logical sequence of the chapters and outlining the main topics and objectives addressed in each of them, divided as follows:

- Chapter 1: Provides a brief introduction to the subject matter, designed to contextualize the theme, justification, and motivation for carrying out this thesis, the main research problem and objective, as well as the partial objectives, research questions, and research contributions.
- Chapter 2: Presents the theoretical background that guided the development of this research, highlighting the main concepts and technologies involved in this work.
- Chapter 3: Addresses monocular depth estimation for AVs using a collaborative approach, employing federated and self-supervised learning with unlabeled vehicle data to train a depth estimator effectively, efficiently, and with privacy preservation. Moreover, in this chapter, we address research questions **RQ1** and **RQ2** and present the research contributions **RC1** and **RC2**.
- Chapter 4: Improves monocular depth estimation for AVs by leveraging a fusion of FL and BO. In this chapter, we address research questions **RQ3** and **RQ4** and present the research contributions **RC3** to **RC6**.
- Chapter 5: Addresses continual learning of monocular depth for AVs using FL, regularization, and experience replay, and answers research questions **RQ5** to **RQ7** and presents the research contributions **RC7** to **RC11**.
- Chapter 6: Presents the conclusion of the work, pointing out the main contributions of the thesis, and explores potential themes for future work.

2. Theoretical Background

In this section, we present the theoretical background on Smart Mobility, ITS, AI, AVs, computer vision, monocular depth estimation, self-supervised FL, BO, and continual learning.

2.1 Smart Mobility and Intelligent Transportation Systems

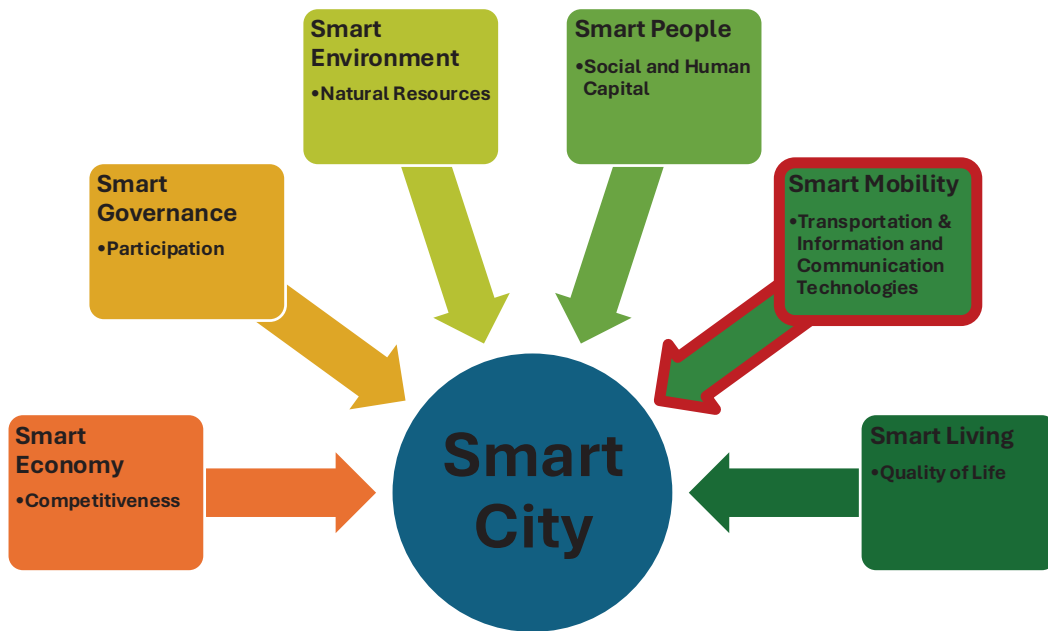


Figure 2.1: Illustration of Smart Mobility and other Smart City components.

Mobility is one of the most difficult topics to face in large metropolitan areas. It involves both environmental and economic aspects and needs both advanced technologies and virtuous people behaviors. Smart Mobility can be seen as one of the key components of a Smart City [62], as highlighted in Figure 2.1. It is largely permeated by Information

and Communication Technology, used in multiple applications, to support the optimization of traffic flows, but also to collect the opinions of citizens about livability in cities or the quality of local public transport services [63]. The initiatives on smart mobility can be grouped into four categories defined by [63]: public transport companies and organizations, private companies and citizens, public bodies and local governments, and the combination of all of them (ITS). Moreover, ITS can be seen as advanced applications for collecting, storing, and processing data, information, and knowledge to plan, implement, and evaluate integrated initiatives and policies of Smart Mobility [63]. They are a large and heterogeneous set of applications, including demand control systems for access to reserved areas (cordon pricing, congestion pricing, electronic tolling, with GPS, pay as you drive); integrated parking guidance systems; Variable Message Signs; Urban Traffic Control; Video surveillance systems for area and environment security; Integrated systems for mobility management; Traffic data collection systems; Expert systems for the correlation and filtering of events. In this set, the role of Information and Communication Technology is essential in supporting applications and systems of detection and processing of data and information. These systems can be very sophisticated and are designed to handle different types of information concerning various activities related to mobility, including systems designed to detect and drive traffic, video surveillance systems, parking systems, etc. [63].

2.1.1 Artificial Intelligence Applications in Intelligent Transportation Systems

Recently, AI-based techniques have been adopted in Smart Mobility and ITS. AI approaches for solving transportation problems frequently use machine learning algorithms, such as artificial neural networks, support vector machines, and Bayesian networks [64–67].

Artificial neural networks, with the ability to perform non-linear mapping between inputs and outputs through the consideration of hidden layers and sufficient training, are suitable for addressing transportation problems in which the parametric relationships among variables are not well understood. In the literature, artificial neural networks are commonly adopted in state estimation/forecasting [68, 69], incident detection [70], traffic/infrastructure control [71], and behavior analysis [72].

Similarly to artificial neural networks, support vector machines are supervised learning models that analyze input data, but are more focused on the classification of stages/scenarios. As a result, although support vector machines have been applied to other transport-related problems [73, 74], they are mainly used for problems such as incident

detection [75, 76] and accident prediction [77, 78] in the context of ITS.

Unlike artificial neural networks and support vector machines, which are solely data-driven, Bayesian networks are a type of statistical model that considers the probabilities and conditional dependencies of the control variables. In the ITS literature, Bayesian networks have been used for various transportation problems [79, 80], but are mainly used when the focus is on traffic forecasting [81, 82] and incident / accident-related problems [83, 84].

2.1.2 Computer Vision for Autonomous Vehicles

Significant research efforts have been invested in the integration of Big Data analytics and cutting-edge AI methodologies to improve the performance of ITS [3]. In contrast, relatively less attention has been devoted to the development of intelligent agents to support ITS, the main focus being the development of AVs [4]. AV research has also used advanced AI techniques to tackle its most critical tasks, including computer vision.

Scene depth estimation plays a pivotal role in computer vision, as it enables the perception and comprehension of three-dimensional scenes [5]. Traditionally, active depth estimation methods have relied on lasers, structured light, and other reflections on the object surface [5]. However, these approaches often require substantial investments in human labor and computational resources to obtain dense and accurate depth maps [6]. To overcome these limitations, researchers have been exploring innovative solutions to enable efficient and cost-effective scene depth estimation, which is essential for the widespread adoption of AVs and ITS. Consequently, image-based depth estimation has emerged as a primary area of investigation in computer vision for AVs, due to its reduced deployment costs and the broader range of applicable scenarios [5]. Image-based depth estimation techniques typically rely on calculating the disparity between two 2D images captured by a binocular camera to generate a depth map [7]. However, binocular depth estimation methods require the presence of at least two fixed cameras, which can be a limitation, particularly in scenarios where the scene lacks sufficient texture or has minimal features [8]. Provided that, research shifted its attention to monocular depth estimation [9]. The appeal of monocular depth estimation lies in its ability to utilize a single camera to capture an image or video sequence, thus eliminating the need for specialized equipment [5]. However, the regression of depth in 3D space from monocular images is an inherently ill-posed problem due to the lack of a reliable stereoscopic visual relationship [5]. To overcome this limitation, researchers have been exploring innovative solutions to enable accurate and efficient monocular depth estimation, which is essential

for the widespread adoption of AVs and ITS.

A crucial point to consider is that modern consumer vehicles are already equipped with an array of advanced sensing technologies, including Light Detection and Ranging (LIDAR) systems and Bird’s Eye View (360-degree) cameras. While these sophisticated sensors are commonly utilized for depth estimation purposes [10–12], we contend that monocular depth estimation serves as a cost-effective and complementary alternative. Specifically, this technique offers a viable redundancy mechanism that can be strategically employed in autonomous vehicle (AV) systems to ensure operational continuity in situations where primary depth-sensing methods may fail, become temporarily unavailable, or prove impractical due to environmental or technical constraints.

2.2 Monocular Depth Estimation Problem

The problem of estimating depth from a single RGB (Red, Green, Blue) image can be formulated as a continuous regression task. Given a large dataset of RGB-depth image pairs, the goal is to learn a mapping between the RGB image space I and the depth map domain $\mathcal{D}$ [41]. A common approach is to employ a mean squared error (MSE) loss function in logarithmic space or its variants, which is commonly used in supervised learning settings where pixel-level ground truth is available [41].

In Figure 2.2 we illustrate the problem of monocular depth estimation based on the abstraction of the “Expanded Tunnel of Reality” (ETOR) [85] which can be seen as a set of invisible planes aligned with the camera’s focal plane, each having a different depth. The points of real objects are associated with these planes according to their depth. Also in Figure 2.2, we present an example of an RGB image and its corresponding colored depth mask [86], also referred to as depth map. Moreover, given a training set $\mathcal{T} = \{(I_i, D_i)\}_{i=1}^M$, where $I_i \in I$ and $D_i \in \mathcal{D}$, the problem can be viewed as learning a nonlinear mapping $\Phi : I \rightarrow \mathcal{D}$. which takes an RGB image as input and outputs a corresponding depth map. This formulation is typically applicable to supervised learning algorithms, where pixel-level ground truth is available. However, some methods relax this constraint by introducing additional requirements and constraints [41]. In some sense, the main goal of a monocular depth estimator is to be able to infer the correct placement of each pixel of the RGB image in the ETOR, producing an accurate depth map as a result. The main challenge for this is the fact that the estimator must rely on the 2D information captured by the RGB image. Thus, the research community proposed several methods to achieve this task, leveraging advanced image processing and machine learning techniques.

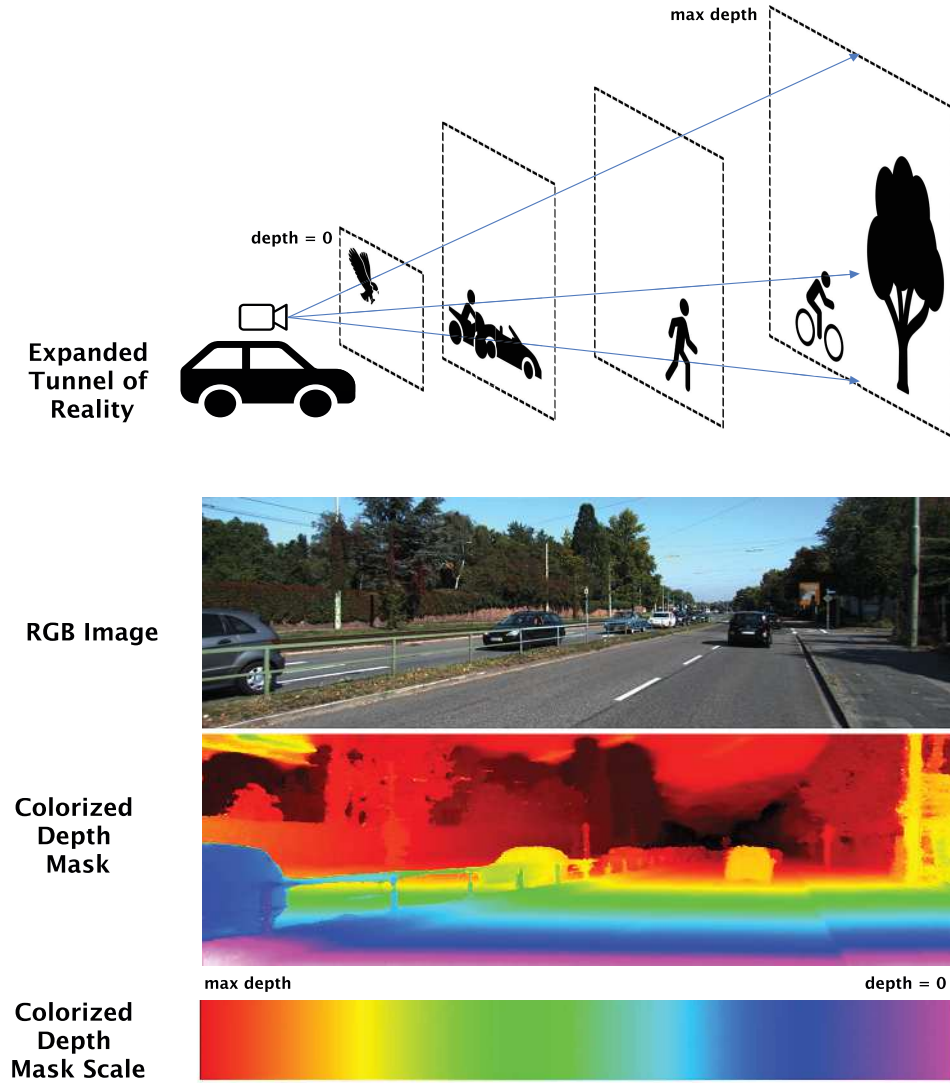


Figure 2.2: Illustration of the “Expanded Tunnel of Reality” and an example of RGB Image and colorized depth mask.

2.2.1 Evolution of Monocular Depth Estimation Solutions

As highlighted in Figure 2.3, the evolution history of monocular depth estimation solutions can be divided into three major periods: the early period, the machine learning period, and the deep learning period [5].

In the early period of depth estimation research (up to 2009 [5]), depth maps were estimated using depth cues, such as vanishing points [87], focus and defocus [88], and shadow [89]. However, most of these methods were applied in constrained scenes [5].

In the machine learning period of depth estimation (2009 to 2014 [5]), several hand-crafted features and probabilistic graph models have been proposed, such as scale-invariant feature transform (SIFT) [90], speeded up robust features (SURF) [91], pyramid his-

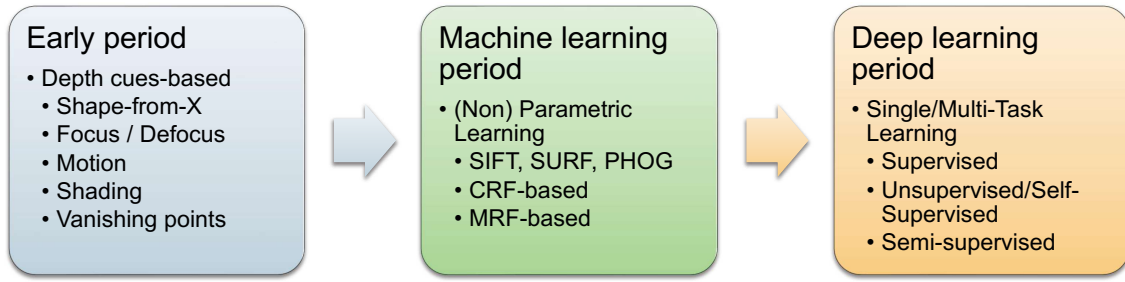


Figure 2.3: Evolution history of monocular depth estimation.

togram of oriented gradient (PHOG) [92], Conditional Random Field (CRF) [93], and Markov Random Field (MRF) [94], which were leveraged to predict monocular depth maps with parametrical and non-parametrical learning in the machine learning process [5].

In the deep learning period of depth estimation research (2014-2021) [5], monocular depth estimation has been tackled as a task of inferring a depth map from a single two-dimensional color image through a deep neural network. Eigen et al. [95] was the first work to propose this, using a coarse-to-fine framework, where the coarse network learned the global depth on the entire image, to obtain a rough depth map, and the fine network learned the local features to refine the depth [5]. Several works were published afterward, leveraging deep learning for the estimation of monocular depth [42, 96–99].

Typically, methods for monocular depth estimation based on deep learning employ an encoder-decoder network that receives as input an RGB image input and generates a depth map as output [5]. The encoder network comprises convolution and pooling layers that capture the representational features of depth, and the decoder network includes deconvolution layers that infer the pixel-level depth map through regression, with the same size as the input image [5]. To preserve the features of each scale, the corresponding layers of the encoder and decoder networks are concatenated with skip connections [5]. Finally, the resulting deep neural network is trained to minimize a depth loss function and converges when it reaches some predefined threshold that indicates that high-quality depth maps can be estimated with the current network parameters [5].

Gradient descent variants are traditionally used to train deep neural networks for monocular depth estimation, since they are guaranteed to reach a local minimum of the depth loss function [5]. However, the quality of the local minimum depends on the hyperparameters used and the initialization of the network. For example, it is generally necessary to resize the images during initialization to meet the needs of deep neural network training. Regarding hyperparameters, the initial learning rate, optimizer parameters,

batch size, and mini-batch size will usually have the greatest influence on image feature learning [5]. Stochastic Gradient Descent (SGD) has been the default choice for the learning method, and Adam has been mostly used as an optimizer [5]. The network is assumed to converge when the loss function stabilizes, which is characterized by the absence of significant changes to the gradient [5].

2.2.2 Deep Learning for Monocular Depth Estimation

Regarding the training paradigm, deep learning [100] for the estimation of monocular depth has been explored using supervised, unsupervised, and semi-supervised approaches.

Supervised monocular depth estimation approaches train deep neural networks using ground truth depth maps. Thus, these techniques require reliable ground truth that is usually generated using human annotations, which can have a high cost considering the time required to perform accurate depth annotations in numerous scenes [5]. In Figure 2.4 we illustrate an example architecture in which the depth estimation network is trained following this paradigm.

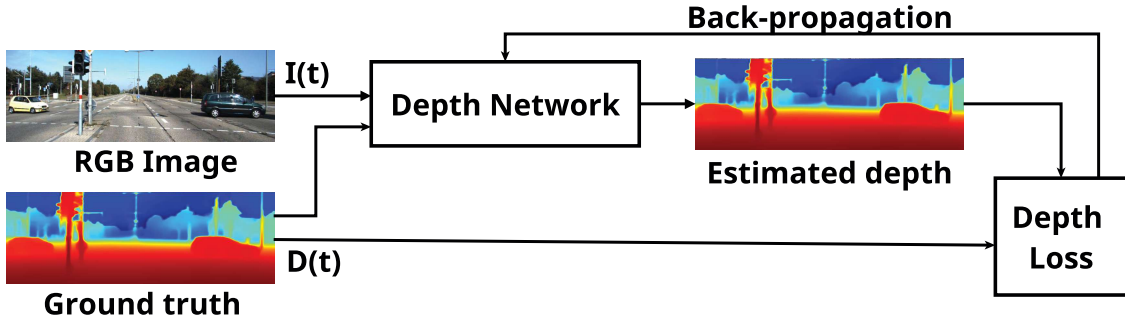


Figure 2.4: Generic model of supervised learning for monocular depth estimation, whose inputs are the RGB image at instant t and the ground truth depth map. The output is the estimated depth map. The loss is computed comparing the ground truth and output maps.

Semi-supervised monocular depth estimation approaches have been proposed to enable the utilization of large sets of unlabeled data to improve learning performance without incurring much higher costs. These approaches used synthetic data, surface normals, and LIDAR sensors to reduce dependence on ground truth depth maps [5]. In Figure 2.5 we illustrate an example architecture in which the depth estimation network is trained following this paradigm. Nevertheless, both supervised and semi-supervised approaches will typically require some amount of labeled data, which might not be sufficiently representative for training truly general models for depth estimation in the heterogeneous domains where AVs will be deployed. To address this issue, several unsupervised methods were

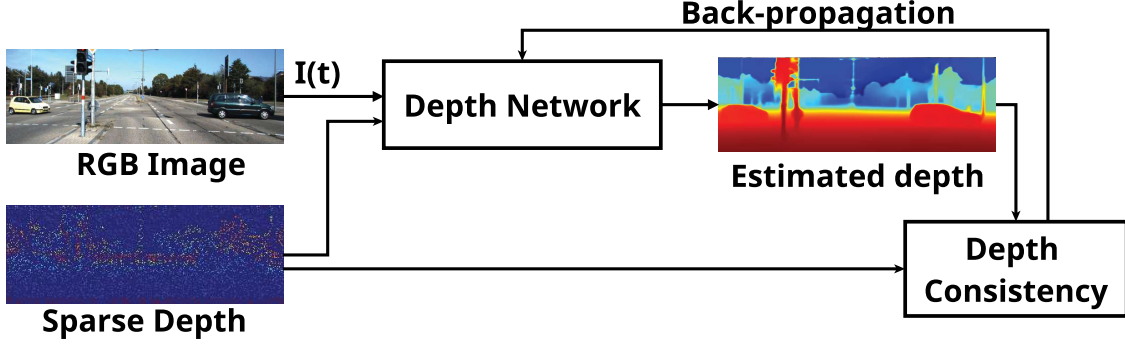


Figure 2.5: Generic model of semi-supervised learning for monocular depth estimation, whose inputs are the RGB image at instant t and the alternative ground truth signal (Sparse depth map obtained from a LIDAR sensor). The output is the estimated depth map.

proposed to learn visual features from large datasets of unlabeled images or videos without using any human annotations [13]. Due to their ability to generate pseudo-labels from raw data, these methods have also been classified as self-supervised, which is how they are most frequently referred to in the recent literature [13].

To learn visual features from unlabeled data, these methods will typically make use of one or more pretext tasks. Deep neural networks can be trained by optimizing the objective functions of the pretext tasks, which will cause the network to learn the higher-order representational features that will enable it to predict the desired visual features, such as the image depth [13]. In Figure 2.6 we illustrate an example architecture in which the depth network is trained in conjunction with a pose network, which performs the pose estimation pretext task.

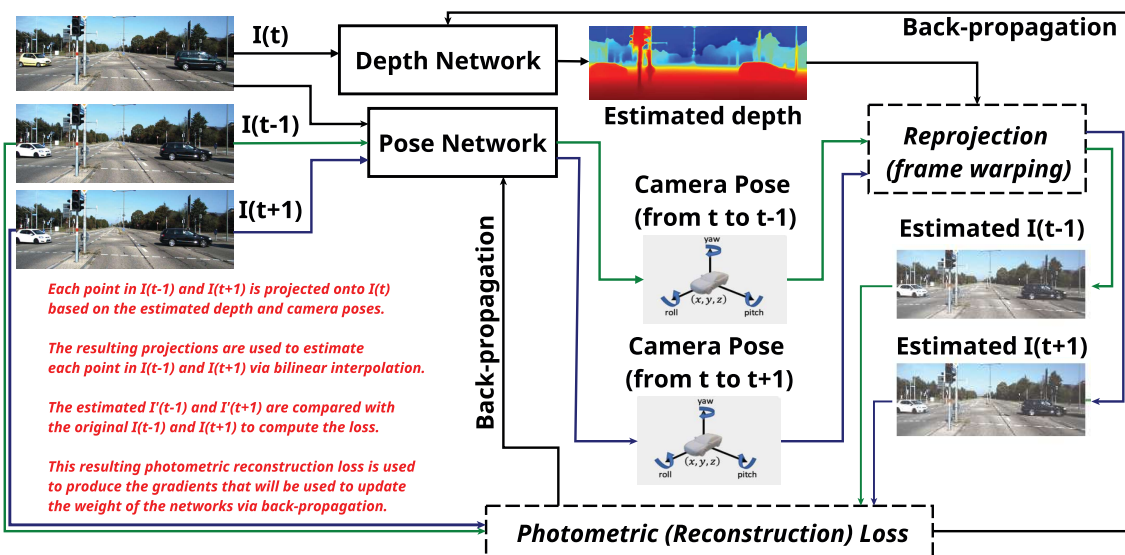


Figure 2.6: Generic model of self-supervised learning for depth estimation based on monocular sequences of RGB images, collected at instants $t - 1$, t and $t + 1$.

Various pretext tasks have been proposed for self-supervised learning, including image colorization, image inpainting, and image jigsaw puzzle [5]. The use of pretext tasks can also be referred to as multi-task learning, which has also been explored with other training paradigms. Although we can train a single network for depth estimation, which we call single-task learning, as proposed in self-supervised learning, we can combine depth estimation with other related tasks to learn shared representations that are useful for multiple related tasks, such as depth estimation, semantic segmentation, and optical flow prediction [5, 13].

2.3 Self-Supervised (or Unsupervised) Federated Learning

The term federated unsupervised representation learning was first introduced in 2020, by Berlo et al. [101]. In this work, the authors leveraged the combination of unsupervised representation learning with FL on unlabeled IoT data, being the first to demonstrate its efficacy in transferring pre-trained deep neural network features to downstream tasks with reduced labeled data.

In this section, we present an overview of self-supervised learning, FL, and the history of works that combined them to address different learning problems in diverse domains.

2.3.1 Self-Supervised Learning

Self-supervised learning has revolutionized the field of machine learning by enabling models to learn from large amounts of unlabeled data [102–104]. This approach has led to breakthroughs in natural language processing, computer vision, and other modalities, such as video, audio, and time series [105–110]. Self-supervised learning involves defining a pretext task based on unlabeled inputs to produce descriptive and intelligible representations [100, 111]. In natural language processing, a common self-supervised learning objective is to mask a word in the text and predict the surrounding words, which encourages the model to capture relationships between words in the text without the need for labels [104].

The same representations of self-supervised learning models can be used in a range of downstream tasks, such as translating text between languages, and summarizing or generating text, among others [104]. In computer vision, analogous objectives exist, and some models are learning to predict masked patches of an image or representation [112, 113]. Other self-supervised learning objectives encourage two views of the same image,

formed by adding color or cropping, to be assigned to similar representations [104].

The ability to train on vast amounts of unlabeled data is of many benefits. Unlike traditional supervised learning methods, which are trained on a specific task often known a priori based on available labeled data, self-supervised learning methods learn generic representations useful in many tasks [104]. This approach can be especially useful in domains such as medicine, where labels are expensive, or the specific task cannot be known a priori [114, 115]. Furthermore, self-supervised learning models are more robust to adversarial examples, label corruption, and input perturbations, and more fair, compared to their supervised counterparts [116, 117]. As a result, self-supervised learning has become a promising approach to advance machine learning, with potential applications that extend far beyond the traditional boundaries of supervised learning.

2.4 Federated Learning

Big Data-based machine learning systems traditionally collect, clean, and aggregate data into one or multiple central servers deployed in a cloud infrastructure for model training and update [16]. In recent years, however, privacy has become one critical aspect concerning the deployment of Big Data platforms. Data used for training typically belong to different parties that could require different policies and privacy restrictions for sharing data with the platform, such as GDPR [26], LGPD [28], and the CCPA [29]. In addition, while cloud servers provide highly scalable computational power and storage, collecting and transferring data from distributed agents to the cloud could demand high bandwidth from the network infrastructure and incur high communication delays [16].

To address these issues, in a seminal work [118] published in 2017, Google researchers proposed an FL method to allow the joint training of models by multiple parties. In this approach, the model is assumed to be a neural network, whose parameter updates can be shared with a central server without the need to transfer raw data through the network [16]. Traditionally, the central server, also referred to as aggregator agent [15], orchestrates the training process and determines how often and how many distributed agents, also called federated nodes [15], will contribute to the global model update.

A typical FL architecture, illustrated in Figure 2.7, will consist of multiple federated nodes and an aggregator agent. The aggregator agent generates the initial global model and distributes it to the federated nodes through the network. When the federated nodes receive the model, they will be responsible for refining it with their locally stored data, without sharing it with other federated nodes or the aggregator agent. After a pre-

determined number of iterations is performed by the federated nodes, they send only a summary of the weight updates generated during local training to the aggregator agent, which merges the model updates performed by each federated node following some aggregation scheme, such as Federated Averaging (FedAvg) [118], to generate a new global model, which is then distributed to all federated nodes to initiate a new FL iteration [15]. These steps are repeated until the aggregator agent achieves a satisfactory global model, according to some predefined metric. Thus, since all local data sets collected by the federated nodes are not transferred and stored on the central server, privacy risks and network consumption are significantly reduced [16].

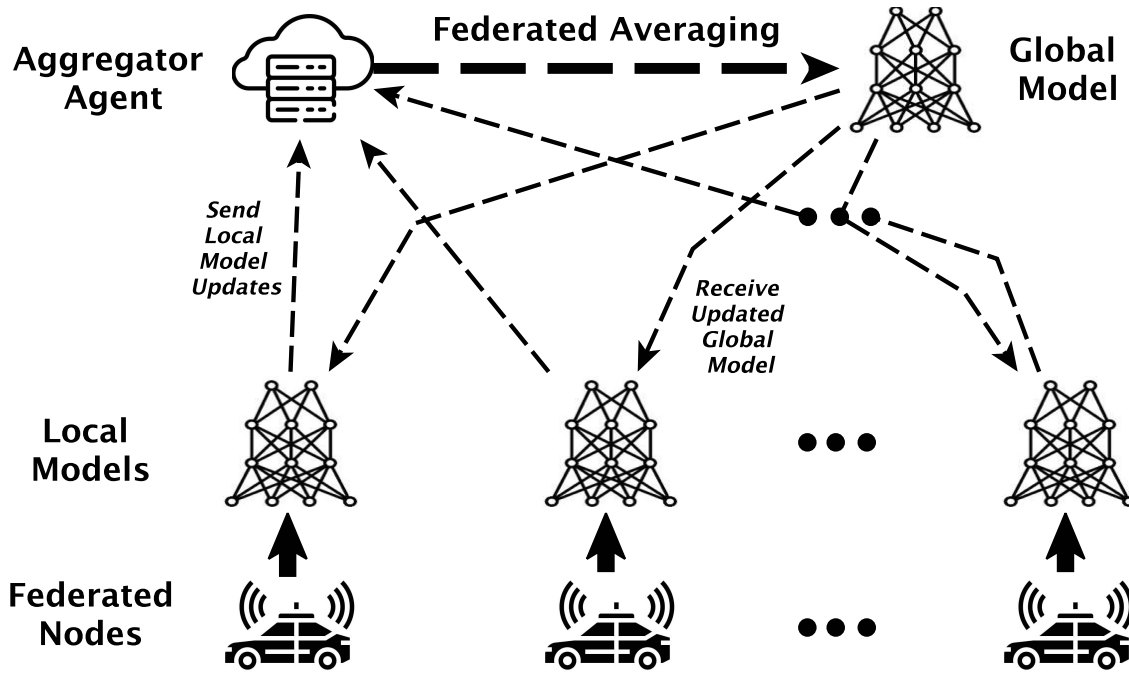


Figure 2.7: Example of FL architecture with AVs as the federated nodes.

Extensions of the initial proposal for FL include decentralized FL architectures [119] and multi-layer FL approaches [120] where the aggregation task is not performed by a single aggregator agent, but rather by multiple federated nodes that act as aggregator agent, or by multiple edge aggregator agents, that perform aggregation of model updates generated by federated nodes of specific geolocations, combined with a central aggregator agent deployed in the cloud infrastructure [120].

The use of FL has improved privacy protection in the detection of failures of industrial Internet of Things (IoT) devices, often coupled with blockchain technology [121]. Similarly, FL combined with blockchain has been investigated for Unmanned Aerial Vehicle (UAV) applications [122], introducing a blockchain-based FL-enabled UAV Mobile Edge Computing (MEC) structure. Blockchain has also been applied to develop a Credible Federated Self-Learning Framework (CSFL) [123] that addresses challenges related

to non-IID data. CSFL adapts to unknown data, maintains a secure training environment, and demonstrates resilience against attacks, outperforming existing personalized FL methods with a 4.11% increase in average accuracy.

These advancements highlight the adaptability of FL in diverse domains. Moreover, FL has been applied in recent ITS research [124], where FL-based license plate recognition systems ensure privacy, accuracy, and manageable communication costs.

In particular, recent work by Zeng et al. [125] proposes a Homophily Learning-Based Federated Intelligence (HLFI) approach that addresses the issues of varying weight distributions in deep neural networks during FL. HLFi leverages hierarchical FL and dynamic elimination learning strategies to improve equipment failure prediction by up to 9.32%, showcasing FL advantages for IoT.

2.4.1 Self-Supervised Federated Learning History

In the same year in which Berlo et al. [101] published their seminal work, a Dual Averaging algorithm [126] addressed self-supervised FL, aggregating agent gradients centrally with two weight computation methods, and simulations showed that it matched the performance of the centralized cloud processing. Also in 2020, Saeed et al. [127] tackled self-supervised FL by applying a wavelet transform to unlabeled sensor data. Their approach excelled in linear classification, low-data scenarios, and semi-supervised learning, surpassing supervised networks and autoencoder-based pre-training.

In 2021, Zhuang et al. [128] addressed non-IID data with the FedU protocol. They uploaded only deep neural network encoder layers to the aggregator agent and incorporated a dynamic predictor update module to improve performance, outperforming single-party training by more than 5% and other methods by more than 14%.

In 2022, Wang et al. [129] discussed the resilience of self-supervised decentralized learning to data heterogeneity, producing valuable representations for object classification, detection, and segmentation, even when combined with basic FedAvg. They also introduced feature alignment and clustering to enhance performance with highly non-IID data. In the same year, Wu et al. [130] proposed a collaborative contrastive learning framework. This framework used feature fusion and neighborhood matching to unify feature spaces among federated nodes, leading to better data representations. Concurrently, Qu et al. [131] conducted FL experiments, showing that replacing convolutional neural networks (CNNs) with Transformers mitigated forgetting and expedited convergence to superior global models in non-IID scenarios. Meanwhile, Li et al. [132] validated the use

of an aggregated online network for target network updates in FL.

In 2023, Mu et al. [133] introduced a local-global model architecture with prototypical contrastive loss, achieving a strong alignment between local and global training. Their experiments demonstrated improved non-IID accuracy from 1.6% to 7.9%, outperforming state-of-the-art methods while maintaining reasonable computation costs. Similarly, Park et al. [134] proposed a distributed self-supervised masked sampling distillation method that improves privacy for vision transformers, utilizing specialized encryption to ensure privacy without constant communication. Yang et al. [135] introduced Iterative Similarity Fusion, ensuring consistent aggregator agent-federated node predictions for unlabeled data and generating reliable pseudo-labels per federated node through incremental confidence. For lightweight federated nodes, an FL approach involving pre-training a single block Masked AutoEncoder (MAE) [136], cascading multiple pre-trained one-block MAEs in the aggregator agent for a downstream multi-block backbone. Lastly, Zhao et al. [137] presented FedUSC, a dynamic update mechanism for local models based on weight divergence, achieving impressive results across multiple datasets, exceeding the state-of-the-art and highlighting its applicability to complex data scenarios.

2.5 Bayesian Optimization

Design problems are prevalent in scientific and industrial domains and require complex decision-making that is difficult for individuals to comprehend [138]. For example, the IBM ILOG CPLEX [139] optimization solver has dozens of manually adjustable parameters, making it challenging to find domain experts to tune them. Moreover, software tools and libraries can have hundreds or thousands of free choices, which can be even more challenging and even unfeasible to tune. Thus, automating design choices is crucial in these and various other applications, such as optimizing nurse scheduling in healthcare, adjusting games in real-time using data analysis, and automatically choosing the best forecasting technique in data science [138]. Advancements in automated design can lead to rapid improvements and innovation in multiple domains, including advertising, healthcare informatics, banking, information mining, life sciences, control engineering, computing systems, manufacturing, e-commerce, and entertainment. Therefore, automating design choices is essential to drive innovation in various industries [138].

BO is a sequential model-based optimization technique [140], originally proposed by Pelikan et al. in 1999 [141] and further detailed in Pelikan’s book in 2002 [142], that has emerged as a solution for these varied design problems and that has recently regained

attention as an efficient approach to optimize expensive black-box functions in machine learning, such as the selection of optimal hyperparameters for a machine learning model, also known as hyperparameter tuning [138, 143]. In recent decades, it has been applied in a wide range of areas, including robotics [144], sensor networks [145], interactive user interfaces [146], reinforcement learning [147], combinatorial optimization [140], environmental monitoring [148], adaptive Monte Carlo [149], experimental design [150], information extraction [151], and automated machine learning [19, 21, 22].

One of the reasons why BO has become a mainstream approach to tackle a diverse set of problems is its high data efficiency, due to its ability to incorporate prior beliefs about the problem to help guide the sampling of new data, and to achieve a good balance between exploration and exploitation in the search [152]. Consider the maximization of a black-box function f , that is expensive to evaluate, we can define the objective function to be optimized as [152]:

$$x^* = \arg \max_{x \in X} f(x) \quad (2.1)$$

where X denotes the search or decision space, and x^* represents the global maximum. The approach builds a probabilistic model (also known as a surrogate model) that defines a distribution over the objective function, refining it with new data. First, a prior distribution over the function is specified, expressing our initial belief about the objective function. Then, using Bayes' rule, the posterior distribution is calculated based on the observed data and the prior, which quantifies our updated belief about the unknown objective function. The next sample is determined by optimizing acquisition functions, which are auxiliary functions used in BO to identify the best candidate [152].

In other words, BO involves two core tasks: (1) Modeling an objective function, calculating its mean and variance; (2) Constructing an acquisition function to estimate the next point to be evaluated.

These core tasks are detailed in the following subsections.

2.5.1 Probabilistic Surrogate Model

The most commonly used probabilistic surrogate model for approximating the true objective function in BO is the Gaussian process, characterized by a prior mean function $\mu(\cdot)$ and a covariance function $k(\cdot, \cdot)$ [152, 153]. Given a finite collection of data pairs $\mathcal{D}_n = (X, y)$ of the unknown function $y = f(X) + \epsilon$ with noise $\epsilon \sim \mathcal{N}(0, \sigma_\epsilon^2)$, where $X = [x_1, x_2, \dots, x_n]^T$ is the input, $y = [y_1, y_2, \dots, y_n]^T$ is the output resulting from the true objective evaluations, and n is the number of samples, the Gaussian process model as-

sumes that the observed data are drawn from a multivariate Gaussian distribution. Therefore, for a new data point x , the joint distribution of the observed outputs y and the predicted output $f(x)$ can be formally defined as [152]:

$$\begin{bmatrix} y \\ y \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} K(X, X) + \sigma_\epsilon^2 I & K(X, x) \\ K(X, x)^T & \kappa(x, x) \end{bmatrix}\right) \quad (2.2)$$

where T denotes matrix transposition, $K(X, X) = [k(x_i, x_j)]_{x_i, x_j \in X}$ represents a $n \times n$ correlation matrix, and $K(X, x) = [k(x_i, x)]_{x_i \in X}$ denotes a correlation vector evaluated at all pairs of training and test points. As described in [152, 153], the conditional distribution $p(y|x, X, y) \sim \mathcal{N}(\mu(x), \sigma^2(x))$ is a multivariate Gaussian distribution, where the mean and variance of the predicted output y can be estimated as:

$$\mu(x) = K(x, X)(K(X, X) + \sigma_\epsilon^2 I)^{-1}y \quad (2.3)$$

$$\sigma^2(x) = K(x, x) - K(X, x)^T (K(X, X) + \sigma_\epsilon^2 I)^{-1} K(X, x) \quad (2.4)$$

where I is the identity matrix and $\mu(x)$ is the prior mean function evaluated at x [153].

2.5.2 Acquisition Function

Assuming that the function $f(x)$ is drawn from a Gaussian process prior and that our observations are of the form $x_n, y_{n=1}^N$, where $y_n \sim \mathcal{N}(f(x_n), v)$ and v is the variance of noise introduced into the function observations, this prior and these data induce a posterior over the functions, which will be the acquisition function [22]. This function can be defined as $a : X \rightarrow \mathbb{R}^+$, and determines what point in X should be evaluated next via a proxy optimization $x_{next} = \arg \max_x a(x)$.

Several acquisition functions have been proposed in the literature and typically depend on the previous observations, as well as the Gaussian process hyperparameters. This dependence can be denoted as $a(x; x_n, y_n, \theta)$ [22]. Under the Gaussian process prior, these functions depend on the model solely through its predictive mean function $\mu(x; x_n, y_n, \theta)$ and predictive variance function $\sigma^2(x; x_n, y_n, \theta)$.

In the following subsections, we will discuss four of the most common acquisition functions found in literature and in standard implementations of BO, such as the BO package [154] for the Python programming language. In particular, this package was used

in our implementations, and we have used its accompanying notebooks ¹ to produce the charts of Figure 2.8, which illustrate a simple BO using the three most basic acquisition functions we discuss.

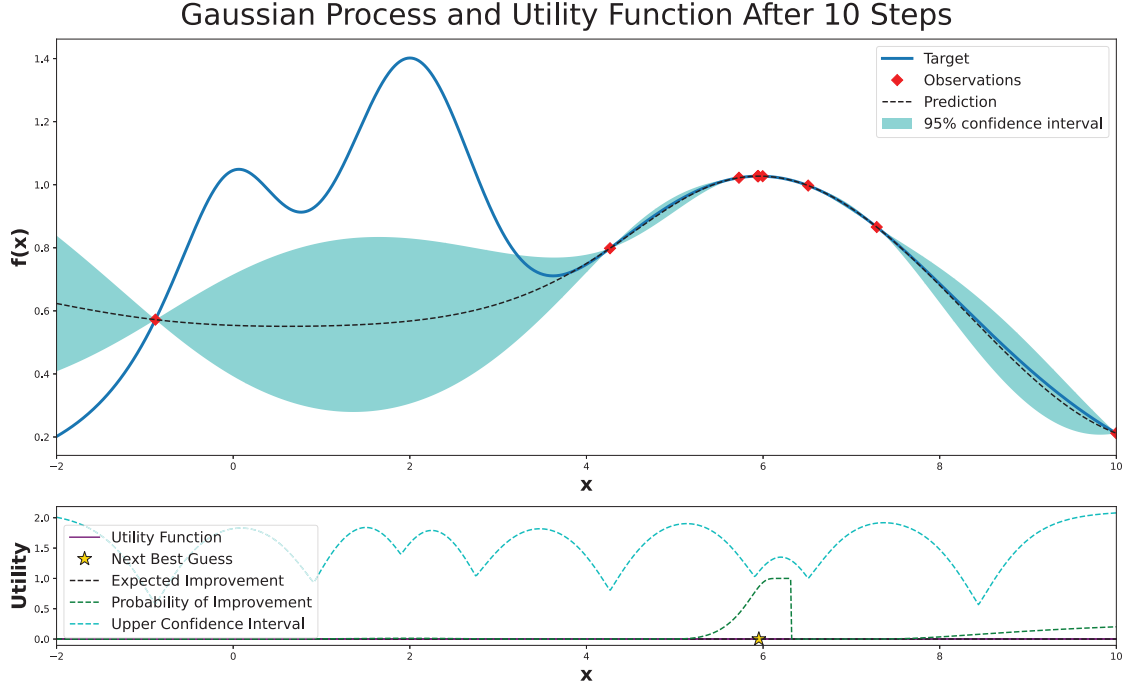


Figure 2.8: Visualization of BO with Gaussian Process for 10 iterations using Expectation of Improvement as main acquisition function. The lower chart shows alternative acquisition function results using the Probability of Improvement and Upper Confidence Interval.

2.5.2.1 Probability of Improvement

Let $\hat{f}^*$ represent the best function value found so far, and let $\Phi(\cdot)$ and $\phi(\cdot)$ denote the cumulative distribution function (CDF) and the probability density function (PDF) of the standard normal distribution, respectively. The first acquisition function is to maximize the probability of improvement (PI) [155] over the current best value $\hat{f}^*$, expressed as [152]:

$$\text{PI}(x) = P(f(x) \geq \hat{f}^*) = \Phi\left(\frac{\mu(x) - \hat{f}^*}{\sigma(x)}\right) \quad (2.5)$$

where P is the probability of finding a better objective function value in a position x than the current best value $\hat{f}^*$, which, for a Gaussian process, is given by the cumulative distribution function of the standard normal distribution [152]:

¹<https://github.com/bayesian-optimization/BayesianOptimization>

$$\Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^x \exp(-t^2/2) dt \quad (2.6)$$

2.5.2.2 Expected Improvement

Expected Improvement (EI) is the most widely used acquisition function, as it is one of the most intuitive [156]. Suppose x_n denotes the point sampled at iteration n and y_n denotes the observed value. Assume that we can only return a solution that we have evaluated as our final solution to the objective function. Suppose further that we have no evaluations left to make and must return a solution based on those we have already performed. Since we observe f without noise, the optimal choice is the point previously evaluated with the largest observed value. Let $f_n^* = \max_{m \leq n} f(x_m)$ be the value of this point, where n is the number of times we have evaluated f thus far [156].

Now suppose that we actually have one more evaluation to perform and that we can perform it anywhere. If we evaluate at x , we will observe $f(x)$. After this new evaluation, the value of the best point we have observed will be $f(x)$ (if $f(x) > f_n^*$) or f_n^* (if $f(x) \leq f_n^*$). The improvement in the value of the best observed point is then $f(x) - f_n^*$ if this quantity is positive and 0 otherwise. We can write this improvement more compactly as $[f(x) - f_n^*]^+$, where $[a]^+ = \max(a, 0)$ indicates the positive part. Our goal is to choose x so that this improvement is large. However, $f(x)$ is unknown until after the evaluation. Therefore, we can take the expected value of this improvement and choose x to maximize it. Thus, the expected improvement can be formally defined as [156]:

$$EI_n(x) := E_n [[f(x) - f_n^*]^+] \quad (2.7)$$

Here, $E_n[\cdot] = E[\cdot \mid x_{1:n}, y_{1:n}]$ denotes the expectation taken under the posterior distribution given the evaluations of f at $x_{1:n}$. This posterior distribution is given by the Gaussian process: $f(x)$ given $x_{1:n}, y_{1:n}$ is normally distributed with mean $\mu_n(x)$ and variance $\sigma_n^2(x)$.

Thus, the expected improvement can be evaluated in closed form using integration by parts [156–158]:

$$EI_n(x) = [\Delta_n(x)]^+ + \sigma_n(x) \varphi\left(\frac{\Delta_n(x)}{\sigma_n(x)}\right) - |\Delta_n(x)| \Phi\left(\frac{\Delta_n(x)}{\sigma_n(x)}\right) \quad (2.8)$$

where $\Delta_n(x) := \mu_n(x) - f_n^*$ represents the expected difference in quality between the proposed point x and the previous best. The expected improvement algorithm then evalu-

ates the point with the largest expected improvement as $x_{n+1} = \arg \max \text{EI}_n(x)$ [156, 157, 159].

2.5.2.3 Upper/Lower Confidence Bound

The confidence bound criteria, such as the upper confidence bound (UCB) for maximization problems and the lower confidence bound (LCB) for minimization problems, aim to achieve optimal regret in the multi-armed bandit community by combining uncertainty and expected reward [145, 152]. The UCB is calculated as:

$$\text{UCB}(x) = \mu(x) + \beta\sigma(x) \quad (2.9)$$

where $\beta > 0$ is a parameter that balances the exploration-exploitation trade-off. The LCB is calculated as

$$\text{LCB}(x) = \mu(x) - \beta\sigma(x) \quad (2.10)$$

Moreover, the Gaussian process upper confidence bound (GP-UCB [145]) algorithm was proposed as an acquisition function with provable cumulative regret bounds. It has also been generalized to other Bayesian models by considering upper quantiles [138].

2.5.2.4 Gaussian Process Hedge

No single acquisition strategy is universally superior for all problem instances. Empirical evidence suggests that the optimal strategy can vary at different stages of the sequential optimization process. To address this challenge, Hoffman et al. [160] proposed the use of a portfolio containing multiple acquisition strategies. At each iteration, each strategy in the portfolio generates a candidate query point, and a meta-criterion is employed to select the next query point from these candidates. The meta-criterion functions similarly to an acquisition function at a higher level, as it is optimized within the set of candidates suggested by its base strategies, rather than the entire input space [138].

The earlier approach of Hoffman et al. [160] is based on a modification of the well-known Hedge algorithm [161], which was initially designed for the full-information adversarial multi-armed bandit. This particular portfolio algorithm uses the past performance of each acquisition function to predict future performance, where performance is measured by the objective function [138].

In formal terms, Hedge is a probabilistic algorithm that, at each time step t , chooses an action i with probability $p_t(i)$ based on the cumulative rewards accumulated for that action [160]. Following the selection of an action, the algorithm receives a reward r_t^i for each action and updates its gain vector accordingly. In the context of BO, we can conceptualize N bandits, each associated with a distinct acquisition function. Choosing action i corresponds to sampling from the point designated by the function u_i , *i.e.*, $\mathbf{x}_t^i = \arg \max_{\mathbf{x}} u_i(\mathbf{x} \mid \mathcal{D}_{1:t-1})$ for $i = 1, \dots, N$ [160]. Unlike traditional BO methods, which sample the objective function only once per iteration, Hedge is a full-information strategy that requires a reward for every action at every time step. To achieve this, we define the reward at $\mathbf{x}_t^i$ as the expected value of the Gaussian process model at $\mathbf{x}_t^i$, *i.e.*, $r_t^i = \mu_t(\mathbf{x}_t^i)$. We refer to this approach as GP-Hedge. Provided that the objective function is smooth, this reward definition is reasonable and enables Hedge to effectively explore the search space [160].

2.6 Continual, Incremental, and Lifelong Learning

Intelligence is rooted in the ability of systems to adapt to changing environments [162]. In response to external stimuli, evolution has endowed humans and other organisms with the capacity to continuously acquire, refine, and exploit knowledge [163–165]. Consequently, we anticipate that AI systems should also be able to adapt similarly. This drives the study of continual learning, where a common scenario involves learning a sequence of tasks one by one and treating them as if they were observed simultaneously. Such tasks could include acquiring new skills, encountering new examples of existing skills, navigating different environments, or operating in diverse contexts while incorporating realistic challenges [165, 166]. As tasks are presented incrementally over a lifetime, continual learning is also termed incremental learning or lifelong learning in much of the literature, without a strict distinction [164, 167]. This concept has garnered significant attention in recent years, as it has the potential to enable AI systems to continuously learn and improve, much like humans do.

Unlike traditional machine learning models that assume a fixed data distribution, continual learning is distinguished by its ability to learn from dynamic and evolving data distributions [162]. A significant hurdle in this context is the phenomenon of catastrophic forgetting [23, 168], where adapting to a new distribution often comes at the cost of a significant loss of ability to capture the original. This conundrum is a manifestation of the trade-off between learning flexibility and memory retention: an overemphasis on the former can compromise the latter, and vice versa. Rather than simply striking a balance

between these two competing demands, a desirable solution for continual learning should strive to achieve strong generalizability, allowing the model to effectively adapt to distributional differences within and between tasks [162].

As a naive baseline, reusing all previous training data (if possible) not only provides a straightforward solution to the aforementioned difficulties, but also generates substantial computational and storage demands, as well as potential privacy concerns [162]. Moreover, the primary objective of continual learning is to optimize the resource utilization of model updates, ideally by processing only new training data [162].

2.6.1 Types of continual learning scenarios

Depending on how the aspect of the data that changes over time relates to the function or mapping that must be learned, there are three fundamental ways in which a learning problem can be continual [166]. In Figure 2.9 we illustrate how one could identify the type of continual learning scenario based on whether context identity is known at test time and if it is not, whether it must be inferred [166].

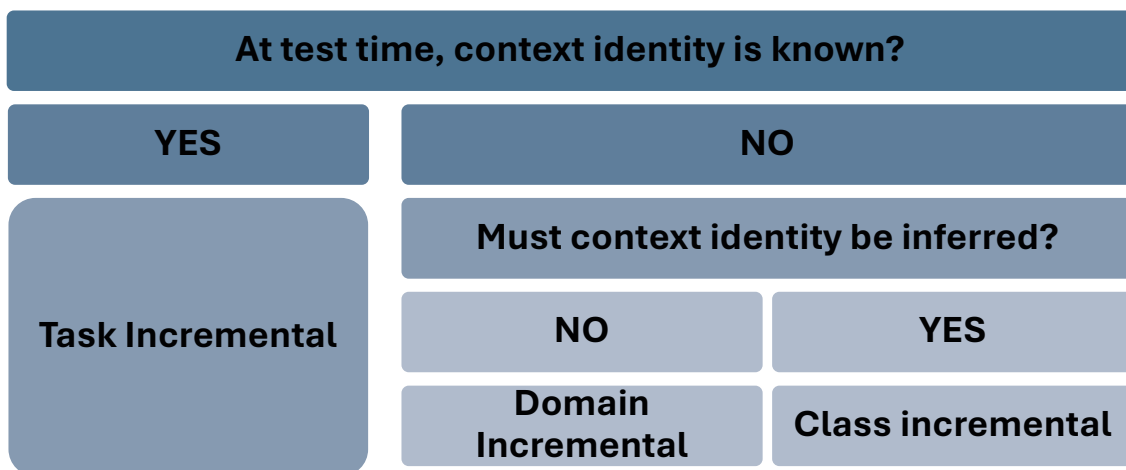


Figure 2.9: Decision tree for the three continual learning scenarios.

The initial scenario type that we consider in continual learning is referred to as “task-incremental learning” (or Task-IL) [166]. This type of scenario can be characterized as a situation in which an algorithm must incrementally acquire a set of distinct skills [169–171]. The defining characteristic of task-incremental learning is that the algorithm is always aware of the specific task it must perform, both during training and at the test time [169–171]. In practice, this could involve explicit task identification or tasks that are easily distinguishable. In this scenario type, it is possible to train models with task-specific components (for example, separate output layers for each task) or even have a separate network for each task to be learned. In the latter case, there is no forgetting at

all. The key challenge in task-incremental learning is to develop strategies that effectively share knowledge between tasks, balance performance and computational efficiency, and leverage information learned in one task to improve performance in others (*i.e.*, achieve positive forward or backward transfer between tasks) [172, 173]. These challenges remain open. Real-world examples of task-incremental learning include learning to play different sports or musical instruments, where it is usually clear which sport or instrument should be played [166].

The second type of scenario is called “domain-incremental learning” (or Domain-IL) [166]. In this type of scenario, the problem structure remains constant, but the context or input distribution changes (*e.g.*, domain-shifts [174, 175]). Similarly to task-incremental learning, this scenario type involves an algorithm incrementally learning a set of “domains” (which can be thought of as “tasks”), with the key difference being that, at least at test time, the algorithm does not know to which domain a sample belongs. However, identifying the domain is not necessary as each domain has the same possible output (*e.g.*, the same classes are used in each domain). Using task-specific components in this type of scenario is only possible if an algorithm first identifies the domain [176–179], but that is not necessarily the most efficient strategy. Preventing forgetting by design is, therefore, not possible with domain-incremental learning, and alleviating catastrophic forgetting remains an important unsolved challenge [174, 175]. Examples of scenarios of this type include incrementally learning to recognize objects under variable lighting conditions [180] (*e.g.*, indoors versus outdoors) or learning to drive in different weather conditions [175].

The third type of scenario in continual learning is called “class-incremental learning” (or Class-IL) [166]. This type of scenario can be characterized as a situation in which an algorithm must incrementally learn to distinguish between a growing number of objects or classes [181, 182]. A common setup for this scenario type involves a sequence of classification-based tasks (which can be thought of as “episodes”), where each task contains different classes, and the algorithm must learn to distinguish between all classes [177, 183]. In this case, task identification is necessary to solve the problem, as it determines which possible classes the current sample might belong to. In other words, the algorithm should be able to solve both each task (*i.e.*, distinguish between classes within an episode) and identify which task a sample belongs to (*i.e.*, distinguish between classes from different episodes). For example, an agent might initially learn to recognize cats and dogs, and later learn to recognize cows and horses; while with task-incremental learning, the agent would not be expected to distinguish between animals encountered in different episodes (*e.g.*, between cats and cows), with class-incremental learning, this is a require-

ment. A significant challenge in this type of scenario is learning to differentiate between classes that are not observed together, which has proven to be a significant challenge for deep neural networks, especially when storing examples of previously seen classes is not allowed [184, 185].

2.6.2 Classes of continual learning methods

Continual learning methods proposed in the literature can be grouped into several categories [162, 166]. In this section, we summarize and discuss the two most common categories found in the related literature: Regularization-Based and Rehearsal or Replay-Based methods. Figure 2.10 illustrates the approaches included in each of those categories of methods, along with their subdirections.

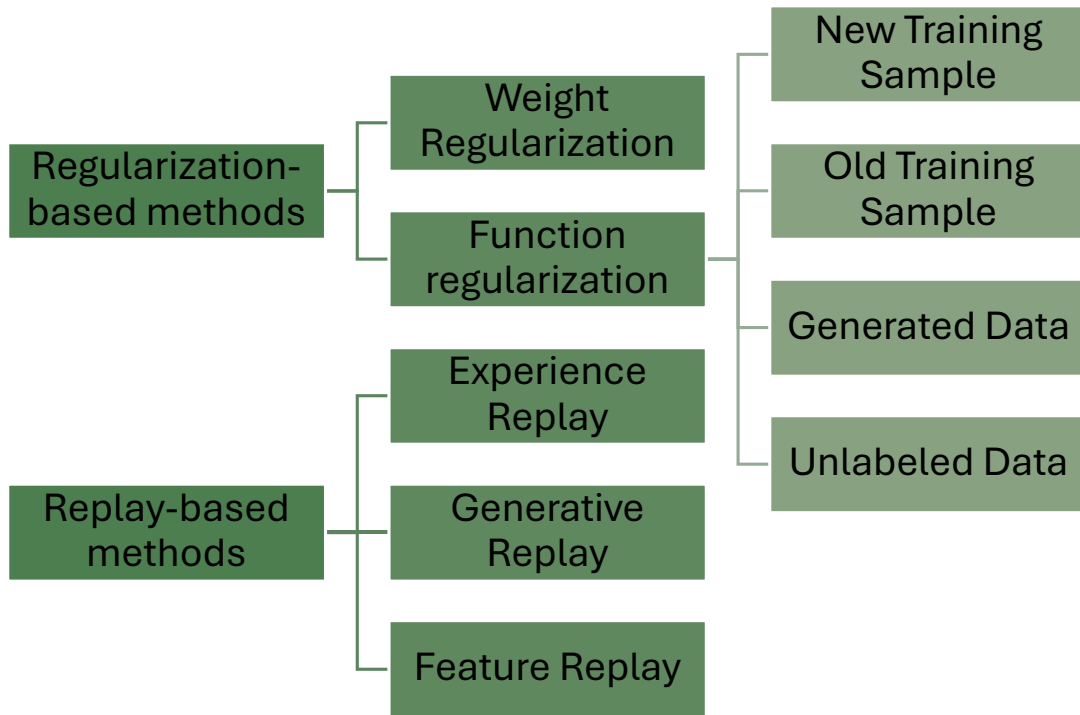


Figure 2.10: Taxonomy of most common categories continual learning methods.

2.6.2.1 Regularization-Based Methods

Regularization-based methods are characterized by incorporating explicit regularization terms to balance the old and new tasks, which typically requires storing a frozen copy of the old model for reference. These methods can be categorized into two approaches based on the target of regularization [162, 166].

The first approach is weight regularization, which selectively regularizes the variation of network parameters [162, 166]. A common implementation is to add a quadratic

penalty to the loss function that penalizes the variation of each parameter based on its contribution or “importance” to performing the old tasks, as originally derived from the online Laplace approximation of the posterior under the Bayesian framework. The importance can be calculated using the Fisher information matrix (FIM), such as Elastic Weight Consolidation (EWC) [166, 186] and some more advanced versions [187, 188]. Meanwhile, numerous efforts have been devoted to designing a better importance measurement. Synaptic Intelligence (SI) [166, 189] approximates the importance of each parameter by its contribution to the total loss variation and its update length over the entire training trajectory. MAS [190] accumulates the importance measurements based on the sensitivity of predictive results to parameter changes, which is online and unsupervised. RWalk [191] combines the regularization terms of SI [189] and EWC [186] to integrate their advantages. Interestingly, these importance measurements are all equivalent to an approximation of the FIM [192], although they come from different motivations.

The second approach is function regularization, which targets the intermediate or final output of the prediction function [162, 166]. This strategy typically uses the previously learned model as the teacher and the model currently trained as the student, while implementing knowledge distillation (KD) [193] to mitigate catastrophic forgetting. Ideally, the KD target should be all old training samples, which are not available in continual learning. However, alternatives can be new training samples [194–197], a small fraction of old training samples [198, 199], unlabeled external data [200], generated data [201, 202], etc., which suffer from different degrees of distribution shift.

2.6.2.2 Replay-Based or Rehearsal-Based Methods

Replay-Based or Rehearsal-Based Methods aim to approximate and recover past data distributions. These methods can be categorized into three approaches, each with its objectives and challenges, as discussed in [162, 166].

The first approach is experience replay, which typically involves storing a limited number of old training samples in a small memory buffer. The key challenges in this approach are to construct and exploit the memory buffer [162, 166]. To construct the buffer, old training samples must be carefully selected, compressed, augmented, and updated to adaptively recover past information. Previous work has employed fixed principles for sample selection, such as Reservoir Sampling [203, 204], Ring Buffer [172], and Mean-of-Feature [181], which select an equal number of old training samples closest to the feature mean of each class.

The second approach is generative replay or pseudo-rehearsal, which typically in-

volves training an additional generative model to replay generated data [162, 166]. This approach is closely related to the continual learning of generative models, which also requires incremental updates. DGR [183] provides an initial framework for learning each generation task while replaying generated data sampled from the old generative model to inherit previously learned knowledge. MeRGAN [201] also employs replay alignment to enforce the consistency of the generative data sampled with the same random noise between the old and new generative models, similar to the role of function regularization [162, 166].

The third approach is feature replay, which maintains feature-level rather than data-level distributions to take advantage of improved efficiency and privacy compared to the other alternatives. However, a central challenge is the representation shift caused by sequentially updating the feature extractor, which reflects catastrophic forgetting at the feature level. To address this issue, GFR [205], FA [195], and DSR [206] perform the feature distillation between the old and new models. IL2M [207] and SNCL [208] recover statistics of feature representations (e.g., mean and covariance) based on experience replay. RER [209] explicitly estimates the representation shift to update the old preserved features. FeTrIL [210] employs a fixed feature extractor learned from the initial task and replays the generated features afterward.

3. Privacy-Preserving Training of Monocular Depth Estimators for Computer Vision in Autonomous Vehicles via Self-Supervised Federated Learning

3.1 Introduction

Recent monocular depth estimation solutions leverage self-supervised learning of deep neural networks along with tasks such as ego-motion (pose estimation) and semantic segmentation [13, 14]. However, existing methods fail to address privacy preservation, low network consumption, efficient computational cost distribution, and resilience to connectivity issues. As highlighted in Chapter 1, FL [118] has been explored in ITS and AV research [15, 16] to meet these requirements. We hypothesize that combining FL with self-supervised learning can enable learning from unlabeled and private AV data with efficacy and efficiency comparable to state-of-the-art monocular depth estimation methods. Although several works have explored FL and self-supervised learning in computer vision [128–137], none have been evaluated on vehicle image datasets such as benchmark monocular depth estimation datasets [47, 48]. Thus, this work aims to develop a highly effective and efficient solution for monocular depth estimation in AVs, generating depth maps from images captured by moving AVs’ monocular cameras.

To the best of our knowledge, this is the first work to present and discuss empirical evidence of the applicability of self-supervised FL to monocular depth estimation for AVs by tackling the research questions **RQ1** and **RQ2**, detailed in Chapter 1, Section 1.5. To answer the research questions, we provide the contributions **RC1** and **RC2**, also detailed in Chapter 1, Section 1.6.

Section 3.2 discusses related work. Section 3.3 presents the proposed method, and Section 3.4 details the evaluation experiments. Section 3.5 discusses the results obtained and Section 3.6 presents conclusions and future work.

3.2 Related Work

In this section, we discuss prior work on self-supervised learning for monocular depth estimation and self-supervised FL.

3.2.1 Self-Supervised Learning for Monocular Depth Estimation

A significant series of advances in self-supervised learning for monocular depth estimation is the SC-Depth (SCD) series [211–213]. SCD1 [211] addressed scale inconsistency, enabling scale-consistent depth estimation in videos. SCD2 [212] addressed rotation issues in handheld camera videos through an auto-rectification network. SCD3 [213] targeted dynamic objects and blurred boundaries. They introduced an externally pre-trained monocular depth estimation model to generate a depth prior, pseudo-depth, which informs novel losses enhancing self-supervised learning. This approach yields models predicting sharp and accurate depth maps, even from videos of highly dynamic scenes.

In this study, we employ SCD3 [213] as our baseline method for centralized self-supervised learning of monocular depth estimation models. This method yielded impressive results on widely used monocular depth estimation benchmarks in AVs: KITTI [214] and DDAD [48]. The readily available source code¹ from its creators facilitated experiment replication and integration into our FL solution.

Besides SCD3, we compare our results with DepthFormer [215], MonoFormer [216], Ranjan et al. [217], EPC++ [218].

DepthFormer [215] and MonoFormer [216] are recent transformer-based techniques that currently excel in KITTI Eigen’s Split among self-supervised learning-based methods, with a resolution of 640×192 . DepthFormer improves self-supervised learning-driven monocular depth estimation by improving the matching of features between images during the generation of cost volume [215]. Meanwhile, MonoFormer blends CNN-Transformer hybrid networks to heighten shape bias using Transformers. It compensates for their weak locality bias through adaptive multi-level representation fusion [216].

Ranjan et al. [217] and EPC++ [218] are more directly comparable to SCD3 [213] and our method since they also employ the resolution of 832×256 . Ranjan et al. [217], propose a Competitive Collaboration framework that coordinates the training of several specialized neural networks to solve complex problems. The networks behave as competitors to explain pixels that correspond to static or dynamic regions, but also collaborate through

¹https://github.com/JiawangBian/sc_depth_pl

a moderator that assigns pixels to either belong to a static or dynamic region. Meanwhile, EPC++ (Every Pixel Counts ++) [218] is a method that jointly addresses three-dimensional geometry per pixel and motion estimation from unlabeled video frames, using three parallel networks to predict camera motion, dense depth maps, and optical flow, which are then processed by a holistic three-dimensional motion parser to disentangle and recover the three-dimensional motion of both static backgrounds and moving objects. Also, although recent works such as AFNet [219], SPIdepth [220], ScaleDepth [221], and PlaneDepth [222] can beat the results obtained by SCD3 [213], DepthFormer [215], and others, they make additional assumptions, such as the use of stereo pairs in combination with monocular videos [220, 222] and multi-view or multimodal input [219, 221], which make it difficult to compare in a fair manner with SCD3, and our method.

3.2.2 Self-Supervised Federated Learning

In this section, we present recent works that have explored the combination of self-supervised learning and FL, with promising results.

As presented in Table 3.1, while many of the related works addressed computer vision tasks, none of them were evaluated on vehicular datasets like KITTI [47] or DDAD [48]. Consequently, while these methods could potentially be adapted for AV scenarios, none provided empirical support for the successful adoption of self-supervised FL in AV contexts. This highlights the specific gap our study addresses within the literature.

Table 3.1: Characterization of self-supervised FL methods, regarding their use on AVs (AV) based on the datasets in which they were tested, and if they were evaluated on computer vision (CV) tasks with IID and non-IID (NIID) data.

ID	AV	CV	Methodology	IID	NIID
[127]	✗	✗	Wavelet-transformed self-supervised learning for enhancing representation learning in low-data federated environments.	✓	✗
[101]	✗	✗	Decentralized pre-training of models on distributed unlabeled data for efficient and privacy-preserving deep learning.	✓	✗
[126]	✗	✗	Privacy-preserving and latency-optimized FL with adaptive weight computation for non-IID data in unsupervised learning.	✗	✓

[223]	✗	✓	Federated contrastive learning with coreset selection for privacy-preserving and efficient edge data training.	✗	✓
[133]	✗	✓	Prototypical contrastive learning for non-IID data alignment in FL.	✗	✓
[134]	✗	✓	Encrypted self-supervised distillation for vision transformers in privacy-preserving medical image analysis.	✗	✓
[224]	✗	✓	Representation-aligned FL through dictionary-based consistency and cross-client alignment.	✓	✓
[128]	✗	✓	Selective contrastive learning for decentralized non-IID image data in privacy-preserving federated unsupervised frameworks.	✓	✓
[225]	✗	✓	HotFed enhances FL by leveraging self-supervised learning to initialize client models, improving accuracy and reducing communication overhead.	✓	✓
[132]	✗	✓	Asymmetrical networks and novel aggregation strategy to enhance performance and robustness in semi-supervised FL amidst non-IID data.	✓	✓
[226]	✗	✓	Local model update with divergence-aware exponential moving average.	✓	✓
[129]	✗	✓	Decentralized self-supervised learning for robust handling of unlabeled, heterogeneous data in distributed systems.	✓	✓
[130]	✗	✓	Feature fusion and neighborhood matching for unified representation learning on non-IID, unlabeled data.	✓	✓
[131]	✗	✓	Enhancing performance and convergence in FL while mitigating catastrophic forgetting using Transformer architectures.	✓	✓
[227]	✗	✓	Self-supervised FL framework for heterogeneous systems with theoretical convergence guarantees and empirical superiority.	✓	✓
[135]	✗	✓	Semi-supervised FL framework leveraging iterative similarity fusion and pseudo-labeling for accelerated convergence.	✓	✓

[136]	✗	✓	Efficient federated pre-training of image models using client-side masked autoencoders and multi-block integration.	✓	✓
[137]	✗	✓	Dynamic update-based federated unsupervised learning for improved image classification in IoT environments.	✓	✓
Ours	✓	✓	Combines FL and deep self-supervision to enable the learning of monocular depth estimators with comparable efficacy and superior efficiency.	✓	✓

In the context of computer vision tasks that employ FL, most related studies focused primarily on image classification tasks. These tasks often used variations of the CIFAR [228] and ImageNet [229] datasets. Moreover, most of these works were assessed with both IID and non-IID data, often synthetically generated based on the number of images featuring a specific object class. This is where our study diverges from the norm, as we maintain the natural sample imbalance inherent to data collection instead of synthetically creating one based on assumed distribution skew, as seen in previous work [129].

3.3 Proposed Method

In this section, we detail the FedSCDepth method, which combines a self-supervised learning-based monocular depth estimation component and an FL component, presented in the following subsections.

3.3.1 Self-Supervised Monocular Depth Estimation

In this section, we describe the monocular depth estimation model and the formalization of frame warping and self-supervision losses.

3.3.1.1 Model Architecture

As in [42, 230, 231], the core of the model architecture is composed of a DepthNet and a PoseNet, both using ResNet18 [97] as their backbone.

Our depth estimation model (DepthNet) is an encoder-decoder neural network with skip connections, depicted in Figure 3.1. More specifically, a fully convolutional U-Net architecture is used for the encoder [232] with a DispNet [233] as the decoder.

The activations are ELU non-linearities at every layer except the output, where sigmoids are used. The sigmoid output σ is converted to depth with $D = \frac{1}{(a\sigma+b)}$, where a and b are chosen to constrain D between 0.1 and 100 units [42]. Reflection padding is used in the decoder, returning the value of the closest border pixels in the source image when samples land outside the image boundaries [231].

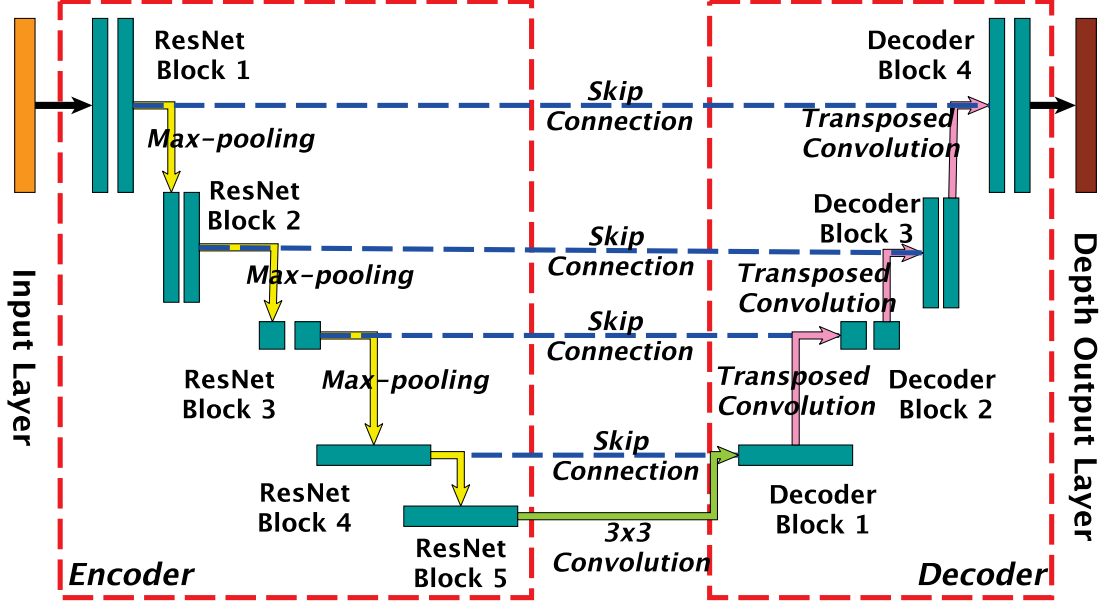


Figure 3.1: Depth estimation neural network architecture.

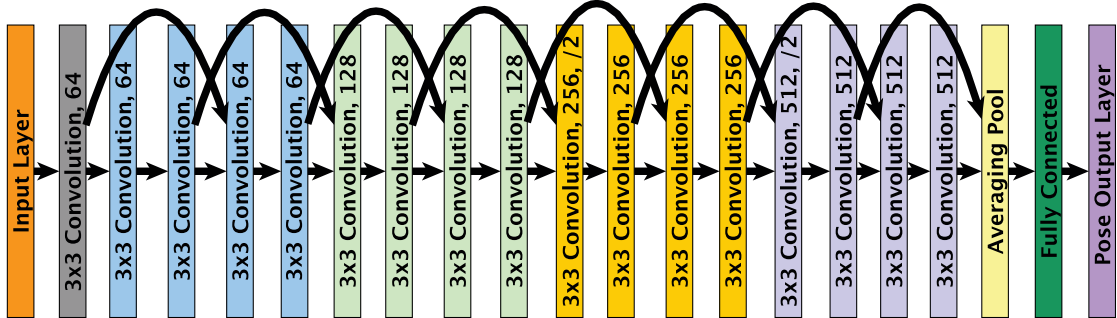


Figure 3.2: Pose estimation neural network architecture.

In addition to predicting depth, the network has to also estimate the camera pose between frames. This estimated camera pose is only needed during training to help constrain the depth estimation network [231]. Therefore, as in [42, 230, 231] we predict the rotation using an axis-angle representation and scale the rotation and translation outputs by 0.01. The pose estimation network (PoseNet), shown in Figure 3.2, is also based on the ResNet18 [234] architecture, modified to accept a pair of color images (or six channels) as input and to predict a single 6-DoF (Degrees of Freedom) relative pose [231].

List of Symbols

Main mathematical symbols used in the formal descriptions of the proposed method.

		L_g	Geometry Consistency Loss
I_a	Image frame a	L_{ern}	Edge-Aware Relative Normal Loss
I_b	Image frame b	θ	deep neural network weights
D_a	Depth Map of I_a	ε_θ	Combined depth and pose estimation deep neural networks
D_b	Depth Map of I_b	$\mathbb{C}$	Set of all AVs available for Federated Training
PD_a	Pseudo-Depth Map of I_a	$\mathbb{C}_r$	Set of AVs actively participating in an FL round r
L_{self}	Combined Self-Supervision Loss	$\mathbb{R}$	Set of all FL rounds
L_p	Photometric Loss	r	Indexer of FL rounds
L_p^M	Mask-Weighted Photometric Loss	q	Fraction of AVs active on each FL round r
L_s	Edge-Aware Smoothness Loss	φ	Projected size of $\mathbb{C}$ based on q
L_{cdr}	Confident Depth Ranking Loss		
L_n	Normal Matching Loss		

As proposed in SCD3 [213] we leverage a PseudoDepthNet to generate pseudo-depth. During the training of DepthNet and PoseNet, PseudoDepthNet generates a single-image depth prior, which is used to boost self-supervised learning.

The combination of the DepthNet and PoseNet architectures and with PseudoDepthNet is presented in Figure 3.3. In summary, given a training sample (*i.e.*, image pair I_a and I_b) the L_{self} is computed. Meanwhile, a PD_a is generated using an auxiliary pre-trained monocular depth estimation network (PseudoDepthNet), while depth maps (D_a and D_b) are produced by the monocular depth estimation network (DepthNet), and the pose estimation network (PoseNet) outputs the pose estimate (P_{ab}). PD_a and D_a are also fed to the Dynamic Region-Refinement (DRR) and Local Structure Refinement (LSR) modules.

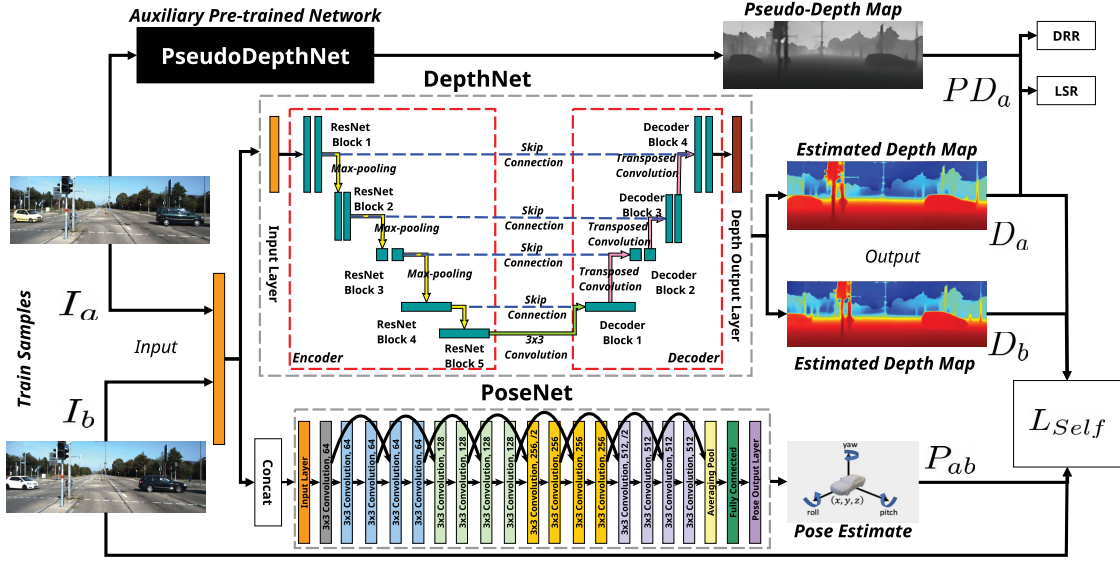


Figure 3.3: Self-supervised learning component overview.

3.3.1.2 Frame Warping

Given a sequence of image frames captured by a moving monocular camera, the reconstruction $I'_{s \rightarrow t}$ of a target frame I_t at time t can be obtained from a source frame I_s at time s by performing a bi-linear interpolation over the reprojected frame coordinates. This interpolation, also referred to as warping flow (ρ), can be formalized as [230]:

$$\rho = I'_{s \rightarrow t}(p_t) = I_s(\hat{p}_s) \quad (3.1)$$

where $\hat{p}_s$ is the reprojection of point p_t into frame I_s . To obtain the mapping from p_t to $\hat{p}_s$, p_t needs to be back-projected into a 3D point X using the camera's intrinsic matrix Y and the depth map D_t corresponding to I_t . Then X is transformed to account for camera movement $J_{t \rightarrow s}$ and projected onto the image plane [230]. This transformation is formalized as:

$$\hat{p}_s \sim Y J_{t \rightarrow s} \underbrace{D_t(p_t) Y^{-1}}_X. \quad (3.2)$$

3.3.1.3 Photometric Loss

For a consecutive pair of images (I_a, I_b) randomly sampled from a sequence of monocular images, their depths (D_a, D_b) and their 6-DoF camera pose (P_{ab}) are predicted by forwarding the DepthNet and PoseNet, respectively [213]. Provided that the warping flow (ρ_{ab}) between I_a and I_b can be generated using D_a, D_b and P_{ab} , and a synthetic reconstruc-

tion of I_a (I'_a) can be generated using ρ_{ab} and I_b via bi-linear interpolation [233]. Thus, the L_p between I_a and I'_a can be used as a self-supervision signal for both networks [211]. L_1 loss is used to reduce the impact of outliers, however, since it is not invariant to illumination changes, an additional image dissimilarity loss (SSIM [235, 236]) is used, as it normalizes the pixel illumination [211]. The SSIM is formalized as,

$$SSIM(x, y) = \frac{(2\mu_x\mu_y + C_1) + (2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} \quad (3.3)$$

where μ_x is the pixel sample mean of x , μ_y is the pixel sample mean of y , σ_x^2 is the variance of x , σ_y^2 is the variance of y , σ_{xy} is the covariance of x , $C_1 = (k_1L)^2$, $C_2 = (k_2L)^2$, L is the dynamic range of the pixel-values (typically this is $2^{\#bits/pixel} - 1$, $k_1 = 0.01$ and $k_2 = 0.03$ by default). The modified L_p is formally defined as,

$$L_p = \frac{1}{|V|} \sum_{p \in V} (\lambda_i ||I_a(p) - I'_a(p)||_1 + \lambda_s \frac{1 - SSIM_{aa'}(p)}{2}), \quad (3.4)$$

where V corresponds to the valid points that are successfully projected from I_a to the image plane of I_b , $|V|$ stands for the number of points in V [211], and $SSIM_{aa'}$ is the element-wise similarity between I_a and I'_a [236]. $\lambda_i = 0.15$, $\lambda_s = 0.85$ [42].

3.3.1.4 Mask-Weighted Photometric Loss

To mitigate the adverse impact of moving objects and occlusions, a weight mask $M = 1 - D_{diff}$ is used to assign low weights to inconsistent pixels and high weights to consistent pixels [211]. Thus, L_p^M can be formalized as,

$$L_p^M = \frac{1}{|V|} \sum_{p \in V} (M(p) \cdot L_p(p)). \quad (3.5)$$

By replacing L_p with L_p^M , the gradients of inaccurately predicted regions have a lower impact on back-propagation [211].

3.3.1.5 Combined Self-Supervision Loss Function

As in [213], L_s [217] is used to regularize the estimated depth maps, since L_p is not very informative in low-texture images or homogeneous regions. In addition, to ensure that D_a and D_b correspond to the same 3D scene structure, another loss was introduced in [211], based on a depth inconsistency map. In addition, to mitigate the impact of moving objects and occlusions, a weight mask is used to assign low weights to inconsistent pixels and high weights to consistent ones [211]. Thus, by replacing L_p with L_p^M , the

gradients of inaccurately predicted regions have less impact in back-propagation [211].

Finally, as in [213] the signals produced by PseudoDepthNet are used to compute additional losses that help regularize self-supervised learning: L_{cdr} and L_n that replaces L_s [213]. In addition, L_g encourages the predicted depths to be consistent with each other in the 3D space [213] and L_{ern} helps constrain the relative normal angles of sampled point pairs to be consistent with pseudo-depth [213]. Thus, by combining these losses, a robust self-supervision signal is obtained. Formally,

$$L_{Self} = \alpha L_p^M + \beta L_g + \gamma L_n + \delta L_{cdr} + \epsilon L_{ern}. \quad (3.6)$$

As in [213], $\alpha = 1$, $\beta = 0.5$, $\gamma = 0.1$. Auto-masking and per-pixel minimum reprojection loss are used to filter stationary and non-best points during training [231] and $\gamma = \delta = \epsilon$ [213].

3.3.2 Federated Learning

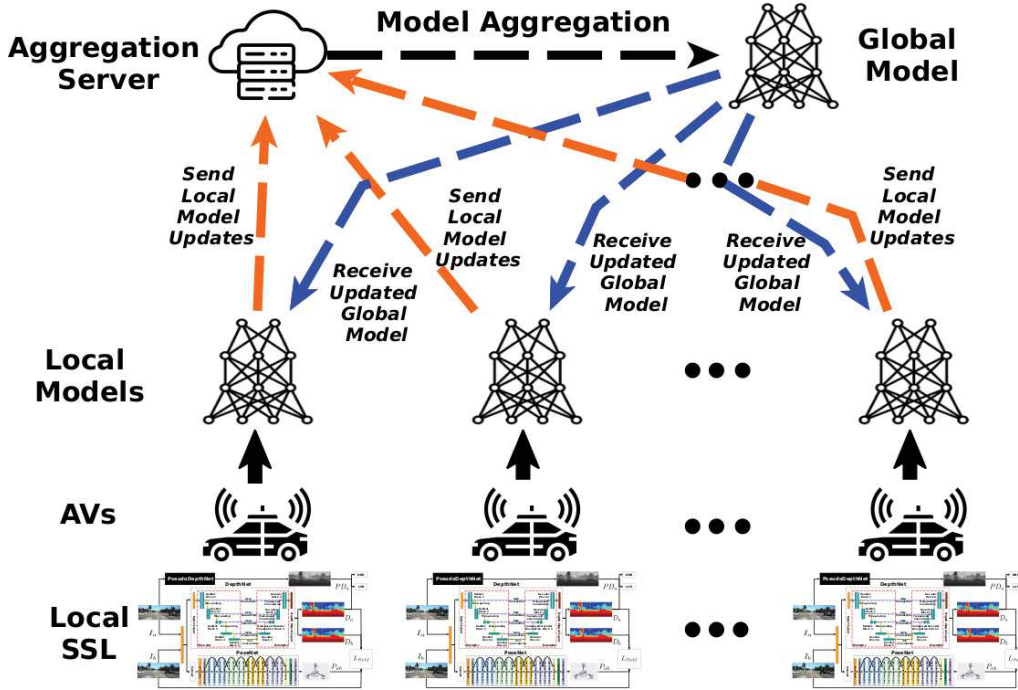


Figure 3.4: Federated self-supervised learning of a monocular depth estimator for AVs.

Figure 3.4 illustrates the FL of our self-supervised monocular depth estimation model using AVs. The aggregation server dispatches the initial global models to the participating AVs, as depicted by dashed blue lines. The AVs perform self-supervised learning on their local copies of the global models on their private data and send the model updates back to the server, illustrated with dashed orange lines. The models are aggregated to produce

a new version of the global model that is sent back to the AVs. This process is repeated until a stop condition is reached.

The main goal of FL is to learn a global model from highly distributed and heterogeneous data by aggregating locally trained models in remote devices [127], such as AVs. Considering that the combination of our monocular depth and pose estimation models (DepthNet and PoseNet) is represented as ϵ^θ , for simplicity, our FL goals can be formally defined as:

$$\min_{\theta} \epsilon^\theta, \text{ where } \epsilon^\theta := \sum_c \frac{h_c}{h} \epsilon_c^\theta \quad (3.7)$$

where $\mathbb{C}$ represents the set of participating AVs in a FL round, h_c is the total number of instances available for client $c \in \mathbb{C}$ with $h = \sum_c h_c$. Lastly, ϵ_c^θ denotes the local monocular depth estimation model parameterized with weights θ . To produce a global model, FedAvg [118] is applied to accumulate client updates after every FL round $r \in \mathbb{R}$.

Algorithm 1 FedAvg of ϵ_θ . The $|\mathbb{C}|$ participating AVs are indexed by c , q is the fraction of AVs active on each FL round $r \in \mathbb{R}$, e is the number of training passes each AV makes over its local dataset on each round (local epochs), and ϕ is the number of AVs selected for each round.

Input: Global ϵ^θ , represented as ϵ for simplicity;

- 1: Distribute a copy of ϵ (ϵ^c) to each AV $c \in \mathbb{C}$;
- 2: **for** each round $r = 1, 2, \dots \mid r \in \mathbb{R}$ **do**
- 3: $\phi = \max(q \times |\mathbb{C}|, 1)$;
- 4: $\mathbb{C}_r = \text{Random Sample of } \phi \text{ AVs from } \mathbb{C}$;
- 5: **for** each AV $c \in \mathbb{C}_r$ **in parallel do**
- 6: $\epsilon_{r+1}^c = \text{LocalUpdate}(c, \epsilon_r^c)$;
- 7: **end for**
- 8: $\epsilon^* = \epsilon_{r+1} = \sum_c \frac{h_c}{h} \epsilon_{r+1}^c$;
- 9: **end for**
- 10: **procedure** *LocalUpdate*(c, ϵ')
- 11: **for** each local epoch i from 1 to e **do**
- 12: Execute Self-Supervised Training with ϵ' on AV c ;
- 13: Evaluate updated ϵ' with Validation Dataset on AV c ;
- 14: **end for**
- 15: return updated ϵ' ;
- 16: **end procedure**

Output: Updated $\epsilon^\theta = \epsilon^*$;

In Algorithm 1, we detail how FedAvg is applied to learn global monocular depth estimation and pose estimation models. In each FL round r , the AVs that will actively

participate in that training round are selected (lines 3–4). Next, each active AV performs local updates on its copies of DepthNet and PoseNet (ϵ_c) using its private training samples and shared validation samples (“gold standard” set of samples collected before training and shared with all participants), using the self-supervised learning method described in Section 3.3.1 (lines 5–7). Local updates are performed over a predefined number of epochs (lines 11–12). After each local epoch, the validation samples are used to assess how generalizable its training was (line 13). This information is used to select the final weights of a local update, which will typically be the weights that present the best performance on the validation set (line 15). After the local updates are computed, the final weights are shared with the central server, which performs the averaging of the local models weighted by the number of training samples available for each AV (line 8).

3.4 Evaluation Experiments

To address the research questions of the present study, **RQ1** and **RQ2**, we design and execute evaluation experiments detailed in the following subsections.

3.4.1 Datasets and Scenarios

Our evaluation experiments are conducted with the publicly available KITTI [47] and DDAD [48] datasets. KITTI contains monocular images and 3D scans of scenes captured by cameras and sensors mounted on top of moving vehicles. Following the approach of [211–213], we also adopt Eigen’s split [95], with the maximum depth set to 80 meters and images resized to a resolution of 832 x 256 pixels for training. Meanwhile, DDAD contains monocular images and 3D scans captured by cameras and sensors mounted on self-driving cars moving in cross-continental urban environments [48]. As in [213], we adopt the standard train/test split, with the max depth set at 200 meters and the images resized to 640 x 384 pixels for training.

In our experiments, we assume that the drives present in the training dataset (34 for KITTI and 150 for DDAD) correspond to distinct AVs (although some were collected by the same vehicle in different drives). Based on this assumption, we characterize three base experimental scenarios: Centralized Training (CT), FT-IID (Federated Training with IID data), and FT-NIID (Federated Training with non-IID data). Federated learning base scenarios were derived into 18 scenarios by combining the use of IID or NIID data, multiple fractions of active participants in each round, and varying numbers of local training epochs per active participant. All federated learning scenarios simulated training with

12 participants for 12 rounds. This restriction in the number of participants and rounds was mostly to enable the timely evaluation of multiple scenarios (in the hardware we had available for experimentation), since each experiment execution could take from days to weeks. Also, since 12 is divisible by 2 and 3, it made it easier to evaluate the scenarios with $1/3$ and $1/2$ of the participants active per round. Finally, although there were 34 trips available in KITTI, only a subset of them had a significantly large number of samples, which made it difficult to train with the maximum number of trips.

3.4.1.1 Centralized Training

All vehicles upload their samples to a central server that will train the depth prediction model and distribute the final version to all participants.

3.4.1.2 Federated Training with IID samples

Before the FL begins, the train samples are randomly distributed among participants, preserving an equal number of samples between all participants, as shown in Figure 3.5. Next, when the FL starts, the global model is distributed to all the participants, along with the validation samples, that will be used as the gold standard for validating the models produced in local training. Each participant trains their local copy of the global model, using their random subset of train samples, and computes the validation losses against the gold standard at the end of every epoch. After each FL round, each participant (that was selected for that round) uploads their local model to the aggregation server, which computes the FedAvg and then distributes the updated global model to all participants.

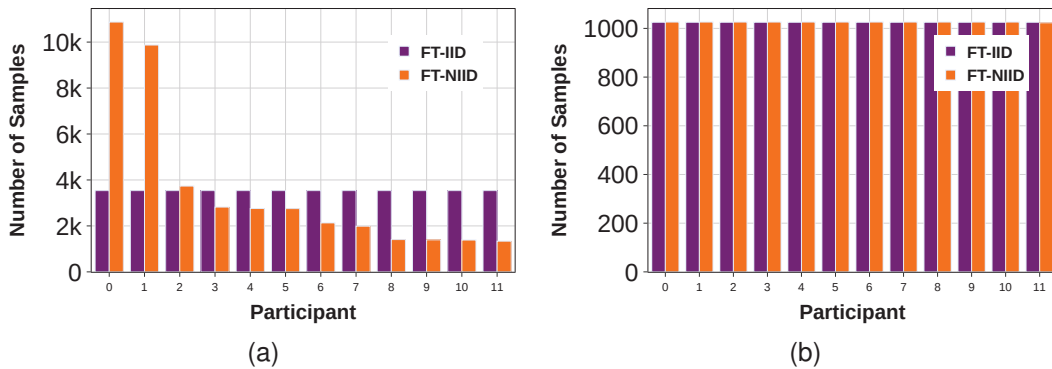


Figure 3.5: Sample count by participant in FT-IID and FT-NIID scenarios, with redistribution of remaining samples, in KITTI (a) and DDAD (b) datasets.

3.4.1.3 Federated Training with non-IID samples

This scenario is similar to the previous one, except that the train samples are distributed according to the drives in which they were collected, reflecting the natural imbal-

ance of data collection. Furthermore, since the number of samples per drive was highly skewed when selecting a subset of the 34 drives, we first picked those with the most train samples. However, participants selected for each FL round were randomly selected, without replacement, to ensure that the training would go over every participant at least once, given a sufficient number of FL rounds. Furthermore, to avoid creating too much advantage for the IID scenario, we randomly redistributed the remaining samples (from participants with the least number of train samples) between the selected participants. Thus, both the IID and the non-IID scenarios had access to all the samples, changing only their distribution among the participants. As shown in Figure 3.5, this redistribution did not remove the great imbalance present in the original distribution by drive in KITTI, although in DDAD this imbalance cannot be observed since its original amount of samples per trip is much more standardized, compared to KITTI.

3.4.2 Metrics

To evaluate the efficacy of the final models, we adopt standard depth evaluation metrics [211–213] that include absolute relative error (*AbsRel*), squared relative error (*SqRel*), linear root mean squared error (*RMS*), log *RMS* (*RMSlog*), and accuracy under threshold (Δ_j), formally defined as:

$$AbsRel = \frac{1}{|T|} \sum_{y_i \in T} \frac{|y_i - y_i^*|}{y_i^*}, \quad SqRel = \frac{1}{|T|} \sum_{y_i \in T} \frac{\|y_i - y_i^*\|^2}{y_i^{*2}} \quad (3.8)$$

$$RMS = \sqrt{\frac{1}{|T|} \sum_{y_i \in T} \|y_i - y_i^*\|^2}, \quad RMSlog = \sqrt{\frac{1}{|T|} \sum_{y_i \in T} \|\log y_i - \log y_i^*\|^2} \quad (3.9)$$

$$\Delta^i = \max \left(\frac{y_i^*}{y_i}, \frac{y_i}{y_i^*} \right), \quad \Delta_j = \frac{1}{T} \sum_{y_i \in T} 1(\Delta^i < 1.25^j) \quad (3.10)$$

where y_i and y_i^* are the estimated depth and ground-truth depth at pixel i , respectively [95, 237]. *AbsRel* and *SqRel* reduce the effect of large errors in the distance, where the image resolution is lower, and a higher magnitude of the error is more likely to occur [237]. Meanwhile, *RMS* and *RMSlog* are the root value of the average squared error and its log [237]. Moreover, Δ_j is the proportion of $i \in T$ where y_i is within a certain relative error (Δ^i) threshold of y_i^* . Thus, $1(\Delta^i < 1.25^j)$ defines an indicator function that equals 1 when the condition ($\Delta^i < 1.25^j$) is satisfied and 0 otherwise, where $1.25^j, j = 1, 2, 3$ is

the threshold for Δ^i [237, 238]. In addition, as in [213], the predicted depth maps are multiplied by a scalar that matches the median with the ground truth for evaluation. Therefore, in summary, lower values of $AbsRel$, $SqRel$, RMS , and $RMSlog$ are preferred, while higher values of Δ_j indicate better accuracy, with $j = 1$ indicating the most strict accuracy threshold (higher quality depth estimates per pixel) and $j = 3$ the least strict threshold.

For assessing the efficiency of the training methods, we consider the $AbsRel$ computed over the validation set (referred to as Λ from now on) during the training as our reference for effective learning, and, adapting a communication cost estimation approach proposed in [239] for FL, we formally compute its upper bound ($\mathfrak{U}_{max}$) as

$$\mathfrak{U}_{max} = 2 \times |\hat{\mathbb{R}}| \times (|\mathbb{C}| \times |\omega_{\beta}|) \quad (3.11)$$

where $|\mathbb{C}|$ is the total number of participants, $|\hat{\mathbb{R}}|$ is the total number of communication rounds (or FL rounds) elapsed, and $|\omega|$ is the number of model parameters. In our formulation, we replace $|\omega|$ with $|\omega_{\beta}|$ to make it explicit that what we consider is the size of the model parameters in Bytes (β). The main motivation for this is to make the comparisons with the estimated cost for CT more direct, since these will be estimated based on the dataset size, which is also measured in Bytes. Additionally, we estimate its lower bound ($\mathfrak{U}_{min}$) considering only the participants selected for training the model as,

$$\mathfrak{U}_{min} = 2 \times |\hat{\mathbb{R}}| \times (q \times |\mathbb{C}| \times |\omega_{\beta}|) = \mathfrak{U}_{max} \times q \quad (3.12)$$

where q is the fraction of participants that were selected for training the model locally on each FL round. In this estimate, we assume that instead of updating the global model for every participant in every round, only those performing local training will have access to the latest global model version.

Finally, we also analyze the number of training steps as a proxy for a computational cost estimate since these represent the number of batches the model has seen during the learning process (including images repeated across epochs). Formally, the number of training steps in a given epoch of the CT is defined as $\hat{\tau} = \hat{e} \times \rho$ and the final number of training steps for CT is defined as $\tau = e \times \rho$, where $\hat{e}$ is the number of epochs in which the model has already been trained on, while e is the total number of epochs defined for the training, and ρ is the number of batches per epoch, assuming every epoch is trained over the same number of batches. We fix ρ at $1k$, to enable comparison with SCD3's original results [213].

For FL scenarios, the computation of the training steps is adjusted to account for the

number of participants. Thus, we define the steps in a given FT round ($\hat{\tau}_{FL}$) as,

$$\hat{\tau}_{FL} = \sum_r^{\hat{\mathbb{R}}} \left(\sum_c^{\mathbb{C}_r} e_c \times \rho_c \right) \quad (3.13)$$

where $|\hat{\mathbb{R}}|$ is the number of FL rounds elapsed, $\mathbb{C}_r$ are the participants that performed training on the round r (which is a subset of $\mathbb{C}$), and e_c is the number of epochs through which the participant $c \in \mathbb{C}_r$ iterated over ρ_c batches.

Although there may be fewer batches available for a given c , in the non-IID scenarios, we have configured the training NOT to resample the available batches randomly until $\rho_c = \rho$, as this resampling could limit the potential efficiency gains of the FT-NIID.

3.4.3 Implementation Details

Our FedSCDepth implementation was based on SCD3 [213] and Dec-SSL [129] source codes, which were made publicly available on GitHub by their authors. Our implementation was also publicly shared on GitHub².

As in SCD3 [213], the DNN implementation used PyTorch Lightning³, with Adam optimizer and learning rate set to 10^{-4} . The DNN encoder was initialized using ImageNet [229] pre-trained weights. As mentioned in Section 3.4.2, the maximum number of batches per epoch (ρ) was set as $1k$ and the batch size as 4 to allow comparison of CT and FT results.

FT experiments were conducted with $|\mathbb{C}| = 12$ participants for $|\mathbb{R}| = 12$ rounds with a total of 18 different setups resulting from the combination of the following parameter values: $Q = 1, \frac{1}{2}, \frac{1}{3}$; $E = 1, 2, 3$; $IID \in \{true, false\}$.

As in Dec-SSL [129], the local updates were simulated in the same process in which the FedAvg aggregation was computed. Although this simulation strategy might not be sufficiently realistic for estimating all possible metrics, it does not affect the metrics we adopt to estimate the efficacy of the trained models and the efficiency of the training.

The experiments were run on a bare metal server with a CPU i3-12100F 4.3 GHz, 32GB DDR4 RAM, RTX 2060 GPU with 12 GB GDDR6 RAM, and SSD M.2 2280 1TB. The server was configured with Ubuntu 22.04.2 LTS, Python 3.8.15, Conda 4.12.0, Pip 22.3.1, and CUDA version 12.1.

²<https://github.com/eltonfss/federated-sc-depth/tree/kitti-v2>

³<https://www.pytorchlightning.ai/>

3.4.4 Ablation Studies

In this section, we present the results of ablation studies [240, 241], in which we analyze the sensitivity of the federated training to variations of some of its parameters. For instance, we observe the impact of the number of participants per round ($\phi = q \times |\mathcal{C}|$), the number of local training epochs (e), the number of communication/federation rounds elapsed ($|\hat{\mathcal{R}}|$), and the data heterogeneity (*IID*) on the lowest global validation losses (Λ) and their corresponding communication ($\bar{\mathcal{O}}_{max}$ and $\bar{\mathcal{O}}_{min}$) and computational costs ($\hat{\tau}_{FL}$), shown in Figures 3.6, 3.7, 3.8, and 3.9.

3.4.4.1 Impact of number of participants per round

In Figure 3.6 (a), we find that the best Λ in KITTI is lower when $\phi = 6$. Meanwhile, in Figure 3.6 (c), the $\bar{\mathcal{O}}_{max}$ corresponding to the best Λ is lower when $\phi = 4$, while in Figure 3.6 (e), $\bar{\mathcal{O}}_{min}$ is lower when $\phi = 4$. Finally, in Figure 3.6 (g), the $\hat{\tau}_{FL}$ for the best Λ is lower when $\phi = 4$. One point to note is that in Figure 3.6 (c), for FT-NIID the $\bar{\mathcal{O}}_{max}$ increases greatly from $\phi = 4$ to $\phi = 6$, but has a slight decrease from $\phi = 6$ to $\phi = 12$. Meanwhile, for FT-IID, there is a meaningful decrease in $\bar{\mathcal{O}}_{max}$ from $\phi = 4$ to $\phi = 6$, and an equivalent increase from $\phi = 6$ to $\phi = 12$. This likely occurs because the random sampling performed in local training has a greater impact on the validation loss improvement than the value of ϕ when $\phi \geq 6$ in the NIID scenario. Meanwhile, for the IID scenario, the value of ϕ is likely to have low impact on the validation loss reduction.

In Figure 3.7 (a), we find that the best Λ in DDAD is also lower when $\phi = 6$. Meanwhile, in Figure 3.7 (c), the $\bar{\mathcal{O}}_{max}$ corresponding to the best Λ is the same for all values of ϕ , while in Figure 3.7 (e), $\bar{\mathcal{O}}_{min}$ is also lower when $\phi = 4$. Finally, in Figure 3.7 (g), the $\hat{\tau}_{FL}$ for the best Λ is lower when $\phi = 4$.

3.4.4.2 Impact of number of local epochs

In Figure 3.6 (b), we find that the best Λ in KITTI is lower when $e = 2$. Meanwhile, in Figure 3.6 (d), the $\bar{\mathcal{O}}_{max}$ corresponding to the best Λ is lower when $e = 3$, while in Figure 3.6 (f), the $\bar{\mathcal{O}}_{min}$ is also lower when $e = 3$. Finally, in Figure 3.6 (h), the $\hat{\tau}_{FL}$ corresponding to the best Λ is also lower when $e = 3$.

In Figure 3.7 (b), we find that the best Λ in DDAD is lower when $e = 3$. Meanwhile, in Figure 3.7 (d), the $\bar{\mathcal{O}}_{max}$ corresponding to the best Λ is lower when $e = 2$, while in Figure 3.7 (f), the $\bar{\mathcal{O}}_{min}$ is also lower when $e = 3$. Finally, in Figure 3.7 (h), the $\hat{\tau}_{FL}$ corresponding to the best Λ is also lower when $e = 2$ for NIID data and $e = 1$ for IID data.

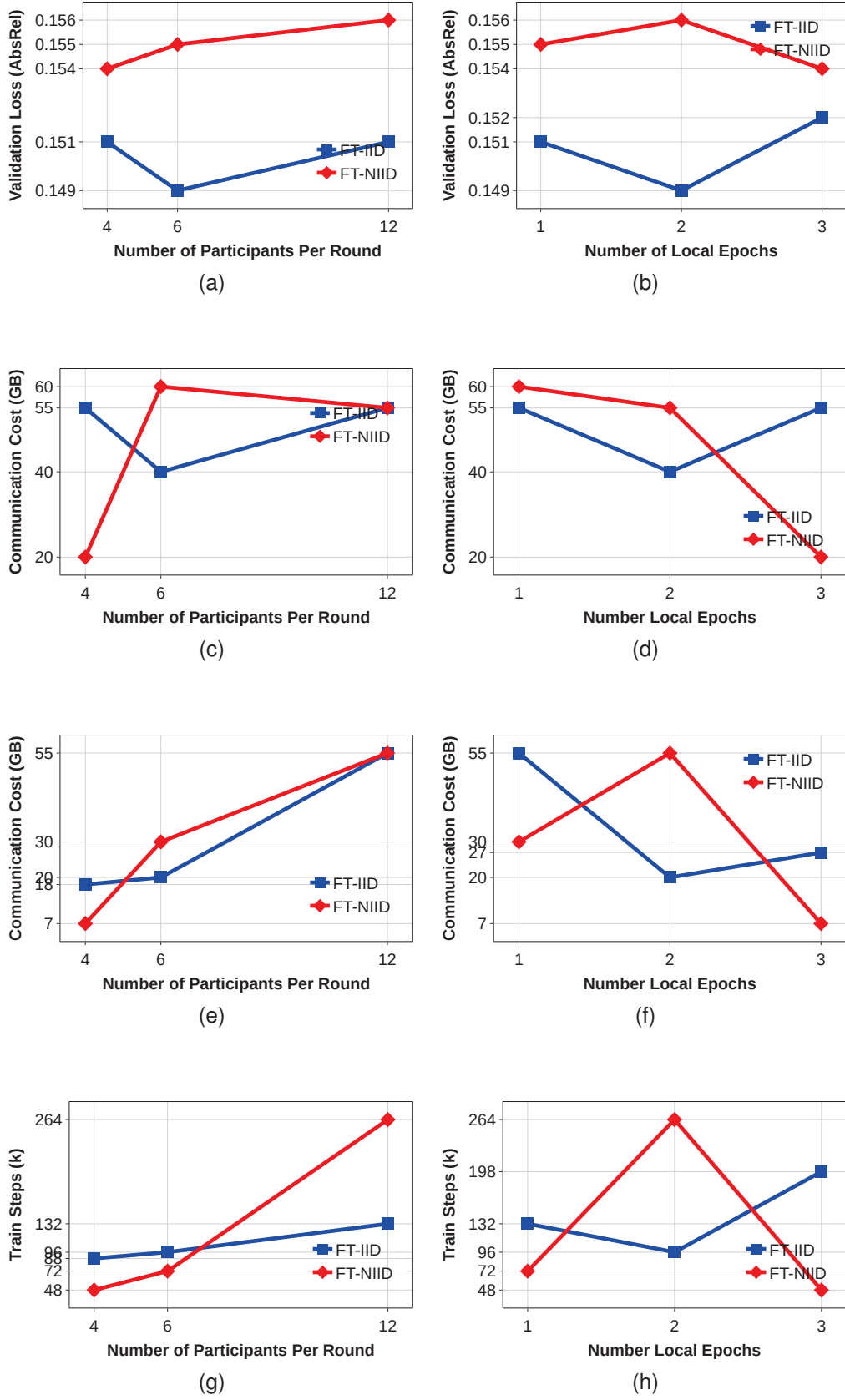


Figure 3.6: Lowest Λ (a), its $\bar{\mathcal{U}}_{max}$ (c), $\bar{\mathcal{U}}_{min}$ (e), and $\hat{\tau}_{FL}$ to obtain it (g), by number of AVs (ϕ). Lowest Λ (b), its $\bar{\mathcal{U}}_{max}$ (d), $\bar{\mathcal{U}}_{min}$ (f), and $\hat{\tau}_{FL}$ to obtain it (h), by number of epochs (e), in KITTI.

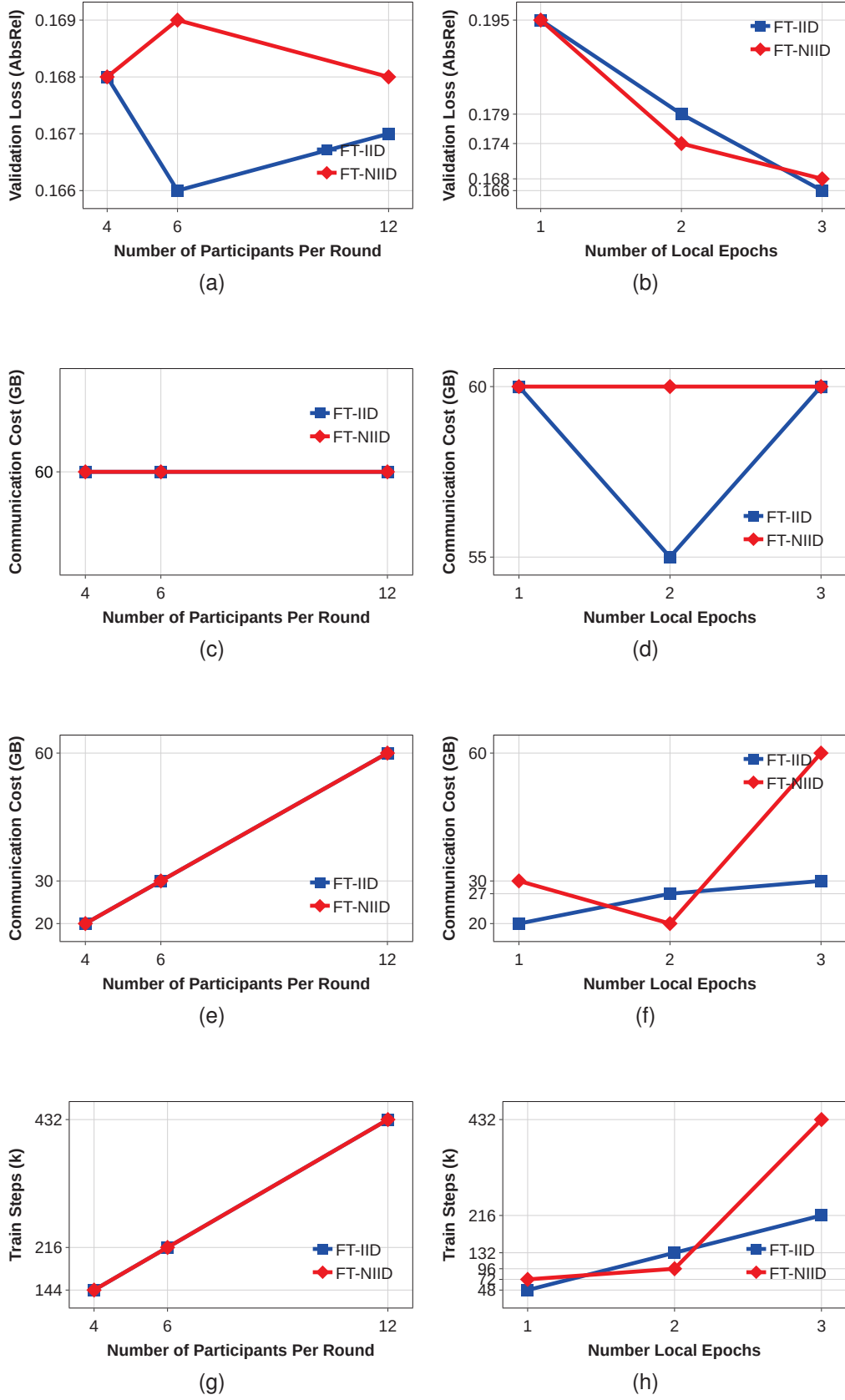


Figure 3.7: Lowest Λ (a), its $\bar{\mathcal{U}}_{max}$ (c), $\bar{\mathcal{U}}_{min}$ (e), and $\hat{\tau}_{FL}$ to obtain it (g), by number of AVs (ϕ). Lowest Λ (b), its $\bar{\mathcal{U}}_{max}$ (d), $\bar{\mathcal{U}}_{min}$ (f), and $\hat{\tau}_{FL}$ to obtain it (h), by number of epochs (e), in DDAD.

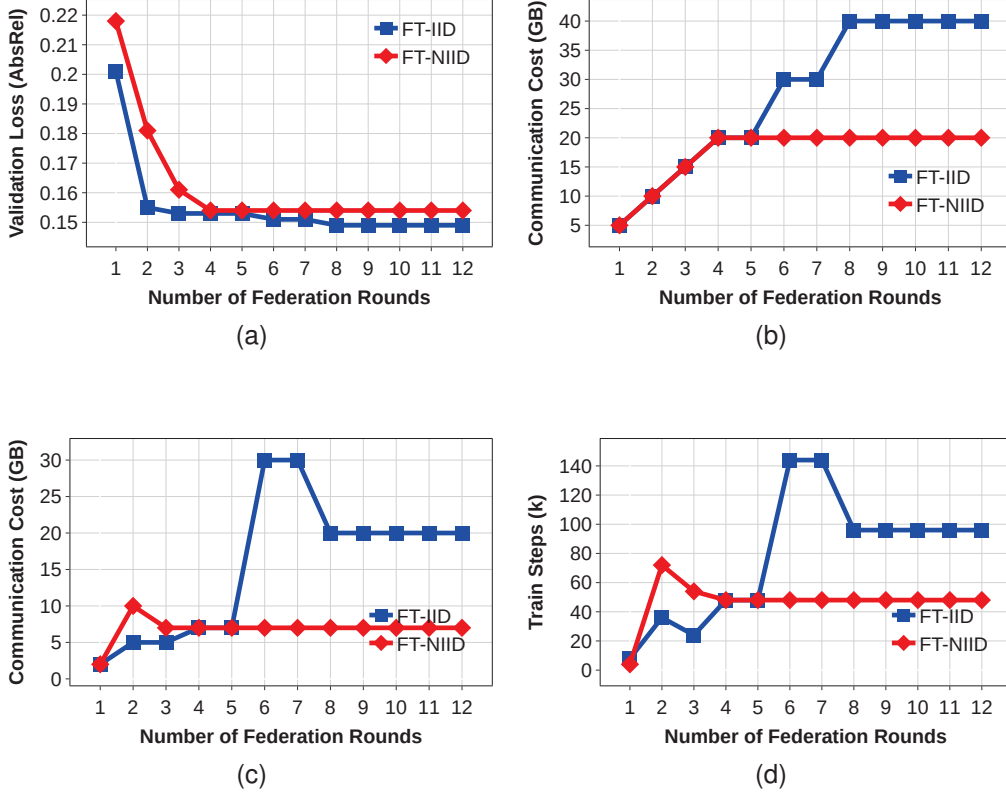


Figure 3.8: Lowest Λ (a), its $\bar{\mathcal{U}}_{max}$ (b), $\bar{\mathcal{U}}_{min}$ (c), and $\hat{\tau}_{FL}$ (d), by FL rounds elapsed ($|\hat{\mathcal{R}}|$), in KITTI.

One point to highlight is that both in Figure 3.6 (d), (f), and (h), as well as Figure 3.7 (d) and (f), we observe large decreases in the y-axis from $e = 1$ to $e = 2$, and increases from $e = 2$ to $e = 3$, for FT-IID (except Figure 3.7 (f), in which the variation is small). Meanwhile, we observe an inverse trend, for FT-NIID (except for Figure 3.6 (d) and Figure 3.7 (d) in which $\bar{\mathcal{U}}_{min}$ always decreases or remains constant). This also likely occurs because the random sampling during local training can produce variations in performance that can significantly impact the reduction of the validation loss than the value of e . Such variations are likely more impactful in the NIID case, which is reflected by the larger variation we observe for this scenario.

3.4.4.3 Impact of number of federation rounds

In Figure 3.8 (a), we find that the best Λ in KITTI decreases rapidly until $|\hat{\mathcal{R}}| = 4$, with a modest decrease afterward. However, the lowest values are observed with $|\hat{\mathcal{R}}| = 8$. Meanwhile, in Figure 3.8 (b) and (c), we observe that the $\bar{\mathcal{U}}_{max}$ and $\bar{\mathcal{U}}_{min}$ for the best Λ scale to about 40 GB and 30 GB, respectively, while in Figure 3.8 (d), the $\hat{\tau}_{FL}$ to the best Λ grows to approximately 144k steps, at $|\hat{\mathcal{R}}| = 7$.

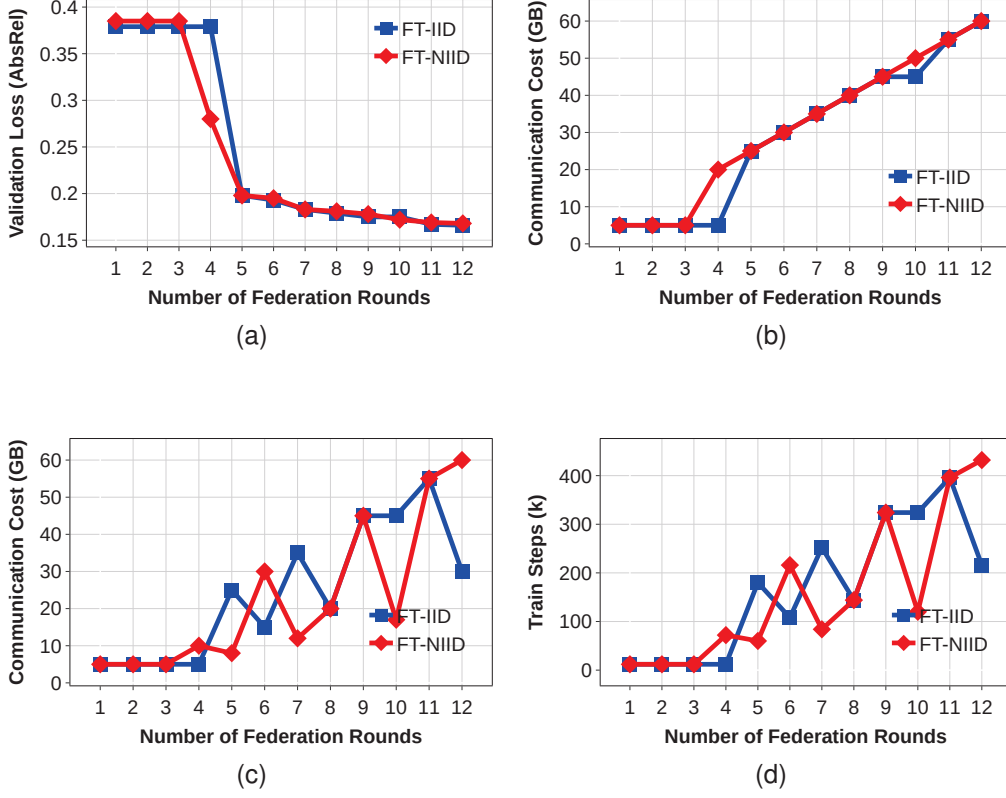


Figure 3.9: Lowest Λ (a), its $\bar{\mathcal{U}}_{max}$ (b), $\bar{\mathcal{U}}_{min}$ (c), and $\hat{\tau}_{FL}$ (d), by FL rounds elapsed ($|\hat{\mathcal{R}}|$), in DDAD.

In Figure 3.9 (a), we find that the best Λ in DDAD decreases rapidly to $|\hat{\mathcal{R}}| = 5$, with a modest decrease afterward. However, the lowest values are observed with $|\hat{\mathcal{R}}| = 12$. Meanwhile, in Figure 3.9 (b) and (c), we observe that the $\bar{\mathcal{U}}_{max}$ and $\bar{\mathcal{U}}_{min}$ for to the best Λ scale up to about 60 GB, while in Figure 3.9 (d), the $\hat{\tau}_{FL}$ to the best Λ grows up to about 216k steps for IID data and to about 432k steps for NIID data, at $|\hat{\mathcal{R}}| = 12$.

It is important to note that the values displayed Figure 3.8 (b), (c), and (d), as well as in Figure 3.9 (b), (c), and (d), reflect the $\bar{\mathcal{U}}_{max}$, $\bar{\mathcal{U}}_{min}$, and the $\hat{\tau}_{FL}$ required to reach the best Λ up to each round, considering all the variations of ϕ and e . Moreover, since Λ can increase or decrease from one round to another, the best Λ in the fourth round of one experiment can have been achieved in the second round. In that case, the values of $\bar{\mathcal{U}}_{max}$, $\bar{\mathcal{U}}_{min}$, and the $\hat{\tau}_{FL}$ considered for the fourth round would be those of the second round. Thus, the variations we observe more evidently in Figure 3.9 (c) and (d), reflect the fact the data points in each round don't necessary come from the same experiment (same values of ϕ and e).

3.4.4.4 Impact of data heterogeneity

Analyzing Figures. 3.6, 3.7, 3.8 and 3.9, we find that the proposed method showed robustness to the data heterogeneity inherent to data collection, obtaining very close lowest Λ with $IID = true$ and $IID = false$ (NIID), with lowest NIID's Λ about 3% and 1% higher than the lowest Λ with IID data in KITTI and DDAD, respectively. Meanwhile, the impact on communication and computational costs was low, as the maximum data transfer cost and the maximum number of training steps were lower with NIID data in KITTI, while in DDAD the $\mathcal{U}_{max}$ was the same for IID and NIID, and the maximum value of $\hat{\tau}_{FL}$ for NIID was the double of IID.

3.4.4.5 Final Recommendation

After analyzing the different FT configurations in the ablation studies, we concluded that the one that showed the best overall results was the one with $\phi = 6$, $e = 3$, and $|\hat{\mathcal{R}}| = 12$. This was mostly because, with $\phi = 6$, FT achieved its best *AbsRel* in both KITTI and DDAD, while with $e = 3$ it achieved its best value in DDAD. Also with $e = 3$, it also achieved its best Ω_{min} in both KITTI and DDAD. Moreover, since the best *AbsRel* values were obtained in the last rounds ($|\hat{\mathcal{R}}| = 12$) of the IID scenarios, we considered it made sense to compare this setting with the centralized training. This would allow for a fairer comparison since in the centralized training the data would be sampled similarly.

3.4.5 Comparison with Centralized Training

In this section, we compare the best FT configuration ($\phi = 6$, $e = 3$, and $|\hat{\mathcal{R}}| = 12$) with the centralized training, in the IID data setting. In Figures. 3.10 (a) and (b), we observe that the selected FT configuration reached a Λ about 4.3% and 11.4% worse than the best Λ obtained with CT (in KITTI and DDAD, respectively) at an additional computational cost overhead of approximately 53.7% and 44.4%, respectively.

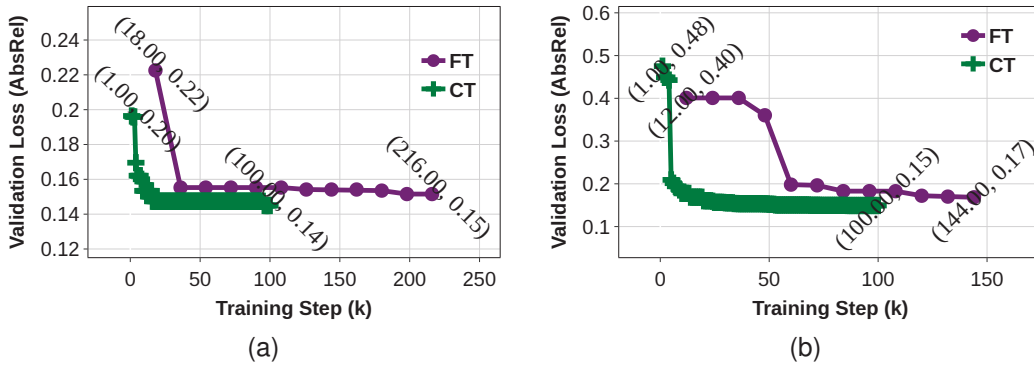


Figure 3.10: Lowest Λ by training steps (τ and $\hat{\tau}_{FL}$), in KITTI (a) and DDAD (b).

One thing to note here is that, since the Λ are calculated at the end of each epoch for CT, its first data point was obtained in the $1k$ training steps. Meanwhile, the first Λ is computed in the FT after the first FL round is complete. Thus, the number of training steps of its first data point will depend on the values of ϕ , e , and $\hat{\tau}_{FL}$ after the first round (as defined in Section 3.4.2). This is also true for the number of training steps for the last data point, which will also be affected by $|\hat{\mathcal{R}}|$. Therefore, it is expected that both the first and last data points for FT and CT in Figures. 3.10, 3.11, and 3.12 will be positioned at different horizontal axis values.

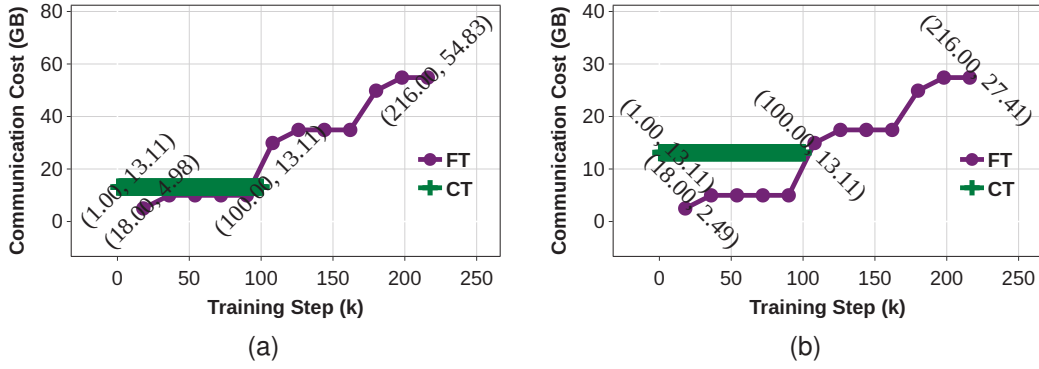


Figure 3.11: $\bar{U}_{max}$ (a) and $\bar{U}_{min}$ (b) required to reach lowest Λ , by number of training steps (τ and $\hat{\tau}_{FL}$), in KITTI.

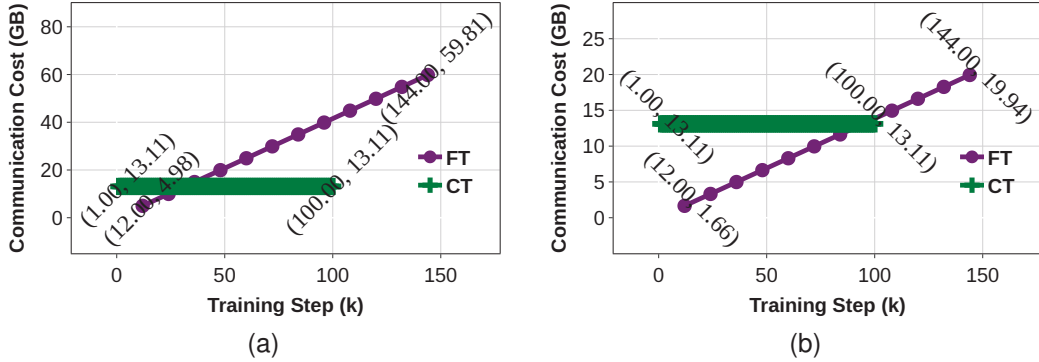


Figure 3.12: $\bar{U}_{max}$ (a) and $\bar{U}_{min}$ (b) required to reach lowest Λ , by number of training steps (τ and $\hat{\tau}_{FL}$), in DDAD.

In Figures. 3.11 (a) and (b), we find that the final $\bar{U}_{min}$ and $\bar{U}_{max}$ were about 51.7% and 76.1% higher with FT than CT, respectively, in KITTI. Furthermore, in Figures. 3.12 (a) and (b), we observe that the final $\bar{U}_{min}$ and $\bar{U}_{max}$ were approximately 34.3% and 78.1% higher with FT than with CT, respectively, in DDAD. However, the total CT communication cost needs to be paid right in the first round, while in FT, this cost is split

between $|\hat{\mathbb{R}}| = 12$ rounds. Considering that there would be $|\mathbb{C}| = 12$ AVs involved in the data collection and training, we would have to transmit, on average, about 1.293 GB (in KITTI) or 1.093 GB (in DDAD) per AV with CT (in the first round only) and something between 0.188 GB and 0.381 GB (in KITTI) or 0.138 GB and 0.415 (in DDAD) per AV with FT (on each round), which indicates that the communication cost paid by each AV on the first round would be on average about 65.1% to 87.4% lower with FT. Also, if we consider that in CT, the computational cost of 100k training steps has to be paid by the central server at the first round, while in FT, the computational cost of 180k (in KITTI) or 144k (in DDAD) training steps is shared by the 12 AVs, with an average of about 1k to 1.25k training steps performed by each AV on each round, we conclude that FT promotes a more efficient cost distribution overall.

Finally, in Tables 3.2 and 3.3, we observe that the efficacy metrics obtained in KITTI test set with the best model obtained with FT were very close to the ones obtained with CT, even obtaining a slight advantage on the *AbsRel*, *SqRel*, and *RMSlog* metrics calculated over static regions (image regions NOT classified as vehicles or pedestrians [213]). However, in Tables 3.4 and 3.5, we observe that the efficacy metrics obtained in the DDAD test set, with the best FT model, were not as close to those obtained with the CT model. In this dataset, the FT results were closer to those of CT in the dynamic regions. The results presented correspond to the 95% confidence interval of each metric’s mean computed with 1k bootstrap [242] samples.

Table 3.2: Efficacy comparison with CT on KITTI Dynamic Regions (Eigen Split).

Scenario	<i>AbsRel</i>	<i>SqRel</i>	<i>RMS</i>	<i>RMSlog</i>
FT	0.220 ± 0.013	2.816 ± 0.398	7.709 ± 0.386	0.295 ± 0.010
CT	0.191 ± 0.009	2.072 ± 0.245	7.111 ± 0.323	0.282 ± 0.009
Absolute % Difference	(from 3.181% to 26.701%)	(from 3.586% to 66.940%)	(from 1.439% to 18.379%)	(from 2.034% to 11.347%)

Table 3.3: Efficacy comparison with CT on KITTI Static Regions (Eigen Split).

Scenario	<i>AbsRel</i>	<i>SqRel</i>	<i>RMS</i>	<i>RMSlog</i>
FT	0.105 ± 0.002	0.623 ± 0.030	4.276 ± 0.101	0.157 ± 0.004
CT	0.106 ± 0.003	0.638 ± 0.034	4.275 ± 0.104	0.159 ± 0.004
Absolute % Difference	(from 3.774% to 5.714%)	(from 7.680% to 12.680%)	(from 4.771% to 4.819%)	(from 3.774% to 6.369%)

Table 3.4: Efficacy comparison with CT on DDAD Dynamic Regions.

Scenario	<i>AbsRel</i>	<i>SqRel</i>	<i>RMS</i>	<i>RMSlog</i>
FT	0.231 ± 0.004	5.041 ± 0.209	17.963 ± 0.401	0.314 ± 0.004
CT	0.221 ± 0.005	4.707 ± 0.213	16.454 ± 0.300	0.297 ± 0.004
Absolute %	(from 0.433%	(from 1.746%	(from 4.498%	(from 2.866%
Difference	to 8.597%)	to 16.061%)	to 13.431%)	to 8.417%)

Table 3.5: Efficacy comparison with CT on DDAD Static Regions.

Scenario	<i>AbsRel</i>	<i>SqRel</i>	<i>RMS</i>	<i>RMSlog</i>
FT	0.162 ± 0.002	3.588 ± 0.089	17.950 ± 0.225	0.273 ± 0.003
CT	0.148 ± 0.002	3.149 ± 0.120	15.656 ± 0.178	0.242 ± 0.002
Absolute %	(from 6.173%	(from 6.410%	(from 10.535%	(from 9.524%
Difference	to 12.162%)	to 20.578%)	to 17.227%)	to 14.876%)

3.4.6 Comparison with State-of-the-Art on SSL-based monocular depth estimation

In Table 3.6, we present the monocular depth estimation efficacy metrics obtained in the test dataset with the best-performing FT configuration of the FedSCDepth in KITTI. We also compare its results with the results reported by three state-of-the-art SSL-based (non-distributed) monocular depth estimation methods: SCD3 [213], which was the baseline of our SSL monocular depth estimation component; DepthFormer (DF) [215], the best performing SSL-based monocular depth estimation methods in the KITTI Eigen Split [95]; Ranjan et al. [217], EPC++ [218], and MonoFormer (MF) [216].

Table 3.6: Efficacy Comparison with state-of-the-art on KITTI (Eigen Split).

Method	Resolution	<i>AbsRel</i>	<i>SqRel</i>	<i>RMS</i>	<i>RMSlog</i>	Δ_1	Δ_2	Δ_3
DF [215]	640×192	0.090	0.661	4.149	0.175	0.905	0.967	0.984
MF [216]	640×192	0.104	0.846	4.580	0.183	0.891	0.962	0.982
Ranjan et al. [217]	832×256	0.148	1.149	5.464	5.464	0.815	0.935	0.973
EPC++ [218]	832×256	0.141	1.029	5.350	0.216	0.816	0.941	0.976
<i>SCD3 [213]</i>	832×256	0.118	0.756	4.709	0.188	0.864	0.960	0.984
<i>Ours</i>	832×256	0.115	0.753	4.593	0.183	0.869	0.963	0.985

Analyzing Table 3.6, we can observe that the results obtained by FedSCDepth were better than SCD3 on all metrics. Meanwhile, its *AbsRel* was about 21.7% worse than DF, which was the best overall. However, FedSCDepth reached the highest Δ_3 overall, and its *SqRel*, *RMS*, *RMSlog*, Δ_1 , and Δ_2 were about 12.2%, 9.7%, 4.4%, 4%, and 0.4% worse than with DF, respectively. Based on these results, we conclude that most of the difference between FedSCDepth and DF is due to the transformer-based technique (and the lower resolution) employed by the latter, which produced a highly superior efficacy than SCD3, since a similar gap is observed between SCD3 and MF, which is another transformer-based technique (that also applied a lower resolution). This difference is also visible when we compare them with MF, which presents a much closer performance to DF than SCD3. Thus, our solution outperformed the efficacy of SCD3, which represents the pre-transformer SSL-based monocular depth estimation state-of-the-art and our main baseline.

Table 3.7: Efficacy Comparison with state-of-the-art on DDAD.

Method	Resolution	<i>AbsRel</i>	<i>SqRel</i>	<i>RMS</i>	<i>RMSlog</i>	Δ_1	Δ_2	Δ_3
DF [215]	Undisclosed	0.135	2.953	12.477	-	0.836	-	-
SCD3 [213]	640×192	0.142	3.031	15.868	0.248	0.813	0.922	0.963
<i>Ours</i>	640×192	0.169	3.531	17.659	0.278	0.758	0.906	0.952

In Table 3.7, we present the monocular depth estimation efficacy metrics obtained in the test dataset with the best-performing FT configuration of the FedSCDepth in DDAD. We also compare its results with the results reported only by SCD3 [213] and DF [215], since Ranjan et al. [217], EPC++ [218], and MF [216] did not report results on DDAD.

Analyzing Table 3.7, we can observe that the results obtained by SCD3 were better than FedSCDepth. Meanwhile, FedSCDepth’s *AbsRel* was about 20.1% worse than DF, which was the best overall, although it was not possible to determine which resolution was used in its training. Moreover, DF reached the highest on most metrics, and SCD3 had the second-best results. Based on these results, we conclude that in DDAD the efficacy gap was larger between FedSCDepth and SCD3, than between SCD3 and DF.

3.5 Discussion

After analyzing efficiency metrics for FT and CT, we conclude the following about **RQ1**: FT is more efficient than CT, since CT incurs communication costs upfront, and FT distributes these costs across multiple rounds, requiring about 65.1% to 87.4% less data

transfer per round, as presented in Section 3.4.5. In addition, in CT, the central server covers the entire computational cost (100k training steps), while in FT, the participating AVs share this cost (about 1k to 1.25k training steps per AV per round on average, totaling 12k to 15k steps per AV at the 12th round). FT’s efficiency was higher with IID data compared to non-IID data in most cases. However, the overall efficiency did not differ significantly between these two FT setups, suggesting that the proposed solution would perform well in real-world AV deployments, which typically exhibit non-IID data. Meanwhile, upon comparing the best FT model’s efficacy with the state-of-the-art, we conclude the following for **RQ2**: monocular depth estimation models from FT outperform those from CT in the static regions of KITTI, but are outperformed the dynamic regions and in DDAD. Moreover, the efficacy of the FT model remains consistent with both IID and non-IID data in most scenarios, highlighting its suitability for practical AV deployments, where data collection is often unbalanced. Finally, a potential deployment challenge for the proposed method is the absence of an extra privacy preservation technique for secure parameter transmission from AVs to the server, such as differential privacy, secure multi-party computation, or homomorphic encryption [243]. However, integrating these techniques into the method is feasible, although assessing their impact on performance might be necessary. Additionally, the recent achievements of blockchain in FL [121, 122] offer a chance to harness these technologies in practical AV deployments.

3.6 Conclusion

In this chapter, we address monocular depth estimation for autonomous vehicles using a collaborative approach. Our method employs federated and self-supervised learning with unlabeled vehicle data to train a depth estimator effectively, efficiently, and with privacy preservation. Prototype testing on the KITTI and DDAD datasets demonstrates near state-of-the-art performance, achieving this with lower per-vehicle computation and communication costs per round compared to centralized training. Notably, the method remains robust with non-IID data, even with the use of conventional federated learning aggregation with FedAvg. Future research involves exploring alternative aggregation functions and optimization strategies to further reduce computational and communication costs, as well as assessing generalizability across different datasets and domain adaptation scenarios.

4. Enhancing Federated Averaging of Self-Supervised Monocular Depth Estimators for Autonomous Vehicles with Bayesian Optimization

4.1 Introduction

As discussed in Chapter 3, FL offers significant benefits in addressing essential requirements in real AV deployments, including data privacy, network consumption, computational cost distribution, and connectivity resilience, providing a promising solution to address the challenges of monocular depth estimation in AVs [15, 16, 118].

Before the present study, only FedSCDepth [45, 46] (detailed in Chapter 3) explored self-supervised FL as an alternative for monocular depth estimation in AVs. However, it relies on FedAvg [118], known for issues with data that are not IID [18]. Thus, to address FedSCDepth’s main limitation, we draw inspiration from recent work in traffic flow prediction [44], demonstrating that BO [141] improves the averaging weights based on the global loss of the averaged model on each round.

As highlighted in Chapter 2, BO is a versatile approach extensively applied in recent works [19] for automated machine learning and has been used to improve local training in FL schemes [20]. Unlike traditional optimization strategies, such as grid or random search, BO analyzes past evaluations of an objective function and an acquisition function to select new inputs to the objective function with the highest expected enhancement in outputs [21]. Thus, BO usually finds better inputs with fewer objective function evaluations [22]. Provided that, we hypothesize that combining self-supervised FL with BO can enable learning monocular depth estimation models from unlabeled and private data captured by AVs with comparable efficacy and superior efficiency than existing methods in self-supervised FL for monocular depth estimation on AVs. Thus, the primary objective of this work is to develop a solution for monocular depth estimation in AVs, generating depth maps from images captured by monocular cameras in moving AVs.

This study is the first to present empirical evidence of the usage of BO to enhance self-supervised FL of monocular depth estimation models for AVs, by addressing **RQ3** and **RQ4**, detailed in Chapter 1, Section 1.5. To answer these research questions, we make contributions **RC3** to **RC6**, also detailed in Chapter 1, Section 1.6.

Section 4.2 examines related work, Section 4.3 presents the proposed method, and Section 4.4 details the evaluation experiments. Section 4.5 discusses the results, Section 4.6 explains its limitations, and Section 4.7 outlines the conclusions and future work.

4.2 Related Work

In this section, we discuss prior works on self-supervised FL for AVs and the use of BO to enhance FL.

4.2.1 Self-Supervised Federated Learning

In Table 4.1, we characterize recent self-supervised FL works with key aspects, including their use of BO and applicability to AV use cases, based on the datasets in which they were evaluated.

Table 4.1: Characterization of self-supervised FL works regarding their use of BO, applicability to AVs (AV), evaluation on computer vision (CV) tasks, experimentation with IID and NIID, and Datasets used.

Ref.	BO	AV	CV	IID	NIID	Datasets
[132]	✗	✗	✓	✓	✓	CIFAR
[129]	✗	✗	✓	✓	✓	ImageNet, CIFAR, MS-COCO, Amazon
[227]	✗	✗	✓	✓	✓	CIFAR, Tiny-ImageNet, LEAF
[223]	✗	✗	✓	✗	✓	CIFAR, SVHN
[130]	✗	✗	✓	✓	✓	CIFAR, Fashion-MNIST
[131]	✗	✗	✓	✓	✓	Retina, CIFAR, CelebA
[133]	✗	✗	✓	✗	✓	CIFAR, Tiny-ImageNet
[134]	✗	✗	✓	✗	✓	TCIA PET-CT, MICCAI 2015, CTI ICH D&S
[135]	✗	✗	✓	✓	✓	MNIST, CIFAR
[136]	✗	✗	✓	✓	✓	(Mini-)ImageNet, CIFAR, Mini-INAT2021
[137]	✗	✗	✓	✓	✓	CIFAR, SVHN, STL-10, COVID-19, Mini-ImageNet
[45,46]	✗	✓	✓	✓	✓	KITTI
Ours	✓	✓	✓	✓	✓	KITTI, DDAD

Although many self-supervised FL studies focus primarily on computer vision tasks, it should be noted that only one of these works has been evaluated with vehicular datasets, such as KITTI [47] or DDAD [215]. Moreover, none of them explore the use of BO to enhance self-supervised FL in computer vision for AVs.

Although these methodologies show potential for AV scenarios, few provide empirical evidence of self-supervised FL success in this context, and none investigate the synergistic combination of BO and self-supervised FL for it. This highlights the specific gap in the literature that our study addresses.

4.2.2 Self-Supervised Federated Learning for Monocular Depth Estimation in Autonomous Vehicles

To our knowledge, the only related work that proposed the use of self-supervised FL for monocular depth estimation is FedSCDepth [45, 46], detailed in Chapter 3. This method integrates a self-supervised monocular depth estimator learning component, based on SCD [213], and an FL component, based on FedAvg [118].

The architecture of the self-supervised monocular depth estimator learning component comprises a monocular depth estimation network (DepthNet) and a pose estimation network (PoseNet). Additionally, a pre-trained monocular depth estimation network (PseudoDepthNet) plays a crucial role in generating pseudo-depth. During DepthNet and PoseNet training, PseudoDepthNet generates a single image depth prior, improving self-supervised learning [45, 46]. Concurrently, the FL component governs the collaborative training of PoseNet and DepthNet through the standard FedAvg algorithm [118]. In each federation round r , the AVs participating in that round of training are selected. Subsequently, each participating AV conducts local updates on the DepthNet and PoseNet weights using its private training samples. Local updates occur over a predefined number of epochs.

Following each local epoch, the shared validation samples (“gold standard” set of samples collected before the training and shared with all participants) are used to assess the generalizability of the training efforts. This assessment informs the selection of the final weights for a local update, typically favoring those that exhibit superior performance on the validation set. Once local updates are calculated, the resulting weights are sent to the central server, which calculates their averaging based on the number of training samples available for each AV [45, 46].

4.2.3 Bayesian Optimization for Federated Learning

In FL, the aggregation function (typically FedAvg [118]) can be classified as a black box function if we consider that the weights of local models in the global model can be tuned based on their contribution to its performance [44]. Thus, instead of relying on assumptions or statistics about the training of local models [244], we can dynamically adjust the input vector that represents the weight of each local model in the global model by running an optimization process of the aggregation function, whose only requirement is that we can evaluate the performance of the global model for a sufficient number of weight combinations.

Recent works have explored the use of BO (detailed in Chapter 2) in FL [20]. In [143], a Federated Thompson Sampling algorithm (FTS) is proposed, using BO to optimize local training. It uses Thompson sampling as the acquisition function and the Gaussian process as a surrogate model for the objective function. Random Fourier features are shared in each FL round and used to approximate the Gaussian process. The authors demonstrate that FTS converges with both IID and non-IID data.

In [44], the authors argue that dynamic approaches, such as BO, are more likely to show better results in each round of aggregation. BO is used to support FL for traffic flow prediction and, similar to FTS [143], the Gaussian process is used as an objective function surrogate. However, they used expected improvement instead of Thompson sampling as the acquisition function.

The use of BO in our solution is strongly inspired by [44], as they also use global model loss as an objective function for BO. We follow a configuration that is similar to the one they used, with Gaussian process regression for the objective function. However, instead of using expected improvement as the acquisition function, we adopt GP-Hedge [160, 245], since it can probabilistically choose the best of the following acquisition functions during each iteration: Expected Improvement, Probability of Improvement, and Lower/Upper Confidence Bound [246] (All of these functions are detailed in Chapter 2). Moreover, it is worth noting that, differently from prior work, we define explicitly how the number of initial random points for BO is set, which significantly influences its ability to explore other regions of the search space. Additionally, we also specify the total number of iterations that BO is allowed to run, which is directly related to the computational cost on the central server and the extra delay for computing and sharing the updated global weights.

4.3 Proposed Method

In this section, we detail our proposed method, denoted BOFedSCDepth [58, 61], which is composed of an aggregation function, formalized as BOFedAvg, and the self-supervised FL solution for monocular depth estimation proposed in FedSCDepth [45, 46]. Figures 4.1 and 4.2 show an overview of BOFedSCDepth and BOFedAvg with AVs.

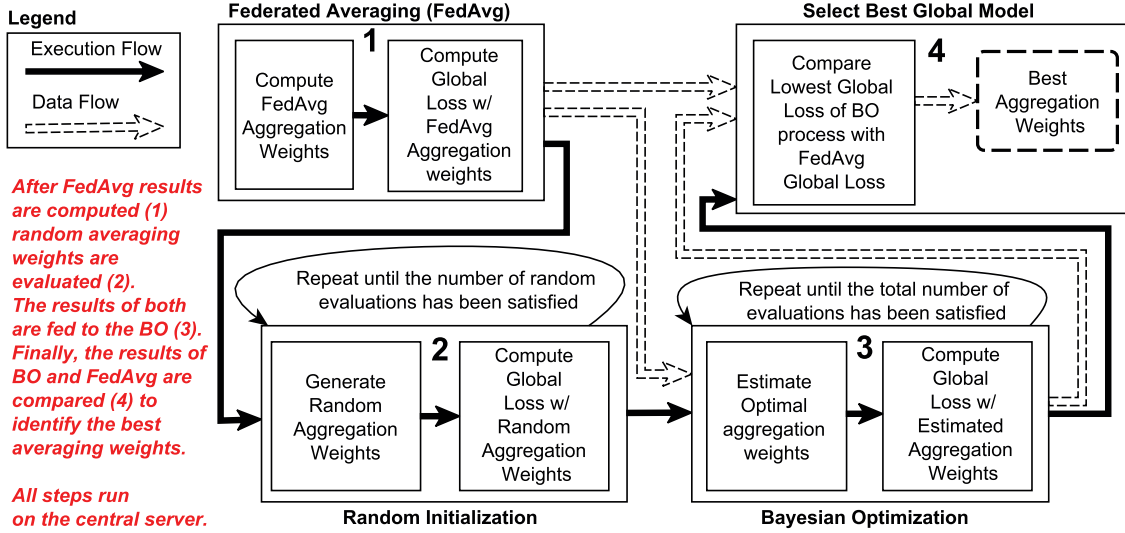


Figure 4.1: BOFedAvg overview.

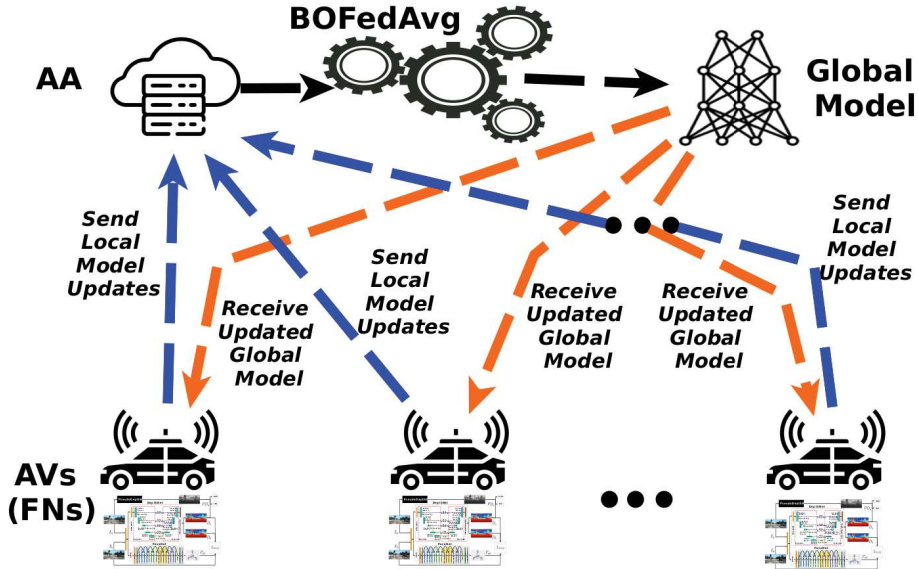


Figure 4.2: Bayes optimized self-supervised FL of monocular depth estimators for AVs.

In Figure 4.2, the central server (AA) dispatches the initial global models to the FNs, as depicted by dashed orange lines. The FNs perform the self-supervised learning on their private data and send the local model updates back to the AA, depicted with dashed blue lines. The AA runs BOFedAvg to produce a new global model that is sent to the AVs.

4.3.1 Bayes Optimized FedAvg (BOFedAvg)

Considering that the deep neural network to be trained is represented as $\epsilon_{DP}^\theta = \epsilon^\theta = \epsilon$ for simplicity, the result of FedAvg on each round r (ξ^μ) can be formalized as:

$$\xi^\mu = \sum_u \frac{m_u}{m} \epsilon_u = \sum_u w_u \epsilon_u \quad (4.1)$$

where $\mathbb{U}$ represents a subset of the total AVs set $\mathbb{C}$ that participate on a round $r \in \mathbb{R}$, $\mathbb{R}$ being the set of all rounds. m_u is the total number of samples available for AV $u \in \mathbb{U}$, and $m = \sum_c m_c$. Lastly, $\epsilon_u = \epsilon_u^\theta$ denotes the local monocular depth estimation model parameterized with weights θ , and $w_u = \frac{m_u}{m}$. Thus, we define $W_{\mathbb{U}} = [w_1, w_2, \dots, w_{|\mathbb{U}|}]$ as the vector of all w_u that are applied to each ϵ_u in the round r .

The goal of BOFedAvg is to find an alternative to $W_{\mathbb{U}}$ produced by Equation 4.1, which we denote as $W_{\mathbb{U}}^*$, that produces an optimally averaged model (ξ^*), that achieves a lower mean absolute relative error [213] (*AbsRel*) than that produced by FedAvg (ξ^μ).

To achieve this, BOFedAvg leverages the shared validation set to compute *AbsRel*, which we will now refer to as Λ , obtained by applying $W_{\mathbb{U}}$ to the local models learned in r (ξ^μ). This value will be called Λ^μ and will be the input to BO and the reference to decide whether $W_{\mathbb{U}}^*$ is better than $W_{\mathbb{U}}$, since, in each BO iteration $z \in \mathbb{Z}$, where $\mathbb{Z}$ is the set of all iterations of the round r , we will also calculate the value Λ^z for $W_{\mathbb{U}}^z$ and the final $W_{\mathbb{U}}^*$, which corresponds to $W_{\mathbb{U}}^z$ with the lowest Λ^z . Provided that, the core BO process receives as input $W_{\mathbb{U}}$, Λ^μ , the function that computes the Λ over a validation set for a given $W_{\mathbb{U}}'$ ($\Lambda = \text{AbsRel}(\text{Normalize}(W_{\mathbb{U}}'))$), where *Normalize* is a standard normalization function that guarantees that the evaluated weights will always sum to one¹), the local monocular depth estimation models learned by each AV in r (deep neural networks), and the search range hyperparameter parameter Ψ , which is used to compute: (1) the number of random $W_{\mathbb{U}}^z$ (defined as $|W_{\mathbb{U}}^\pi| = (|\mathbb{U}| \times \Psi) - 1$) that will be evaluated² before trying to predict a $W_{\mathbb{U}}^z$ that presents the highest expected enhancement in Λ ; (2) the total number of BO iterations $z \in \mathbb{Z}$ (defined as $|\mathbb{Z}| = \max(|W_{\mathbb{U}}^\pi| \times 2, 5)$), that includes the combined set of $W_{\mathbb{U}}^\pi$ and $W_{\mathbb{U}}$ evaluations (referred as $W_{\mathbb{U}}^\Omega$).

Note that Ψ can be modified to increase or decrease both the level of exploration, since a higher $|W_{\mathbb{U}}^\pi|$ implies exploration of more random regions in the search space, and exploitation, with a higher $|\mathbb{Z}|$ allowing deeper exploitation of favorable regions in the

¹This is especially important for the weights generated in the initial random configurations, that might produce non-convex sums of weights.

²($W_{\mathbb{U}}, \Lambda^\mu$) replaces one of the random initial evaluations.

search space levels. This could suggest that a greater Ψ would always be desirable. However, z has a non-negligible cost, especially during the evaluation of Λ^z . Thus, after preliminary experiments, we selected a default value of $\Psi = 6$. Moreover, in Section 4.4.5.5 we evaluated alternative values of Ψ and confirmed that the best trade-off between Λ improvement and the computational overhead introduced was also obtained with $\Psi = 6$.

After evaluating $W_{\mathbb{U}}^{\Omega}$, we obtain the corresponding values of Λ . Thus, we assume that they contribute to a vector $Y^r = \Lambda(W_{\mathbb{U}}^Z)$ obeying $Y^r \sim (\mu_z(W_{\mathbb{U}}^Z), \vartheta_z(W_{\mathbb{U}}^Z, W_{\mathbb{U}}^Z))$, a z -dimensional normal distribution, where $\vartheta_z(W_{\mathbb{U}}^Z, W_{\mathbb{U}}^Z)$ is the covariance matrix, and $Z = 1 : z = [1, \dots, z]$. Then, we can estimate the covariance matrix by using the kernel function matrix $\mathbf{K}$, with $W_{\mathbb{U}}^i = W_i \mid 1 \leq i \leq z$, for simplicity:

$$\mathbf{K} = \vartheta_z(W_Z, W_Z) = \begin{bmatrix} k(W_1, W_1) & \dots & k(W_1, W_z) \\ \dots & \dots & \dots \\ k(W_z, W_1) & \dots & k(W_z, W_z) \end{bmatrix} \quad (4.2)$$

where

$$k(W_1, W_2) = \sigma_0 \exp \left(-\frac{1}{2\sigma^2} \|W_1 - W_2\| \right) \quad (4.3)$$

and $k(W_i, W_j), (i, j = 1, 2, \dots, z)$ is the Gaussian kernel function, σ_0 and σ are the parameters of the kernel function [44]. In the following iterations, the GP-Hedge [245] is used as an acquisition function [44] to predict W_{z+1} with the desired maximum enhancement in Λ (denoted $W_{\star}$). Thus, the BO prediction is defined as

$$W_{\star} = \arg \max_{W'_{\mathbb{U}} \in \mathcal{X}} \Lambda(W'_{\mathbb{U}}) \quad (4.4)$$

where $\mathcal{X}$ is the search space, $\Lambda^{\ominus} = \min_{i=1,2,\dots,z} \Lambda(W_{\mathbb{U}}^z)$ denotes the minimum value observed in the first z iterations, the strategy to compute $W_{\star}$ is

$$\begin{aligned} W_{\star} &= \arg \max_{x \in \mathcal{X}} \mathbf{E}_{\Lambda(x) \sim \mathcal{N}(\mu_z(x), \sigma_z^2(x))} [\max(\Lambda^{\ominus} - \Lambda(x), 0)] \\ W_{\star} &= \arg \max_{x \in \mathcal{X}} (\Lambda_z^{\star} - \mu_z(x)) \Phi(\Upsilon_z) + \sigma_z(x) \phi(\Upsilon_z) \end{aligned} \quad (4.5)$$

where $W'_{\mathbb{U}} = x$ for conciseness, $[\max(\Lambda^{\ominus} - \Lambda(x), 0)]$ defines how much lower the value Λ is than the minimum observed so far, $\Phi(\cdot)$ is the cumulative density function, $\phi(\cdot)$ is

the probability density function and Υ_z is formally defined as

$$\Upsilon_z = \frac{\Lambda^\ominus - \mu_z(x)}{\sigma_z(x)} = \frac{\Lambda^\ominus - \mu_z(W'_\mathbb{U})}{\sigma_z(W'_\mathbb{U})} \quad (4.6)$$

When a new prediction W_{z+1} is available, a Gaussian vector following the distribution $\mathbf{N}(\mu(W_{z+1}), k(W_{z+1}, W_{z+1}))$ can be derived by $\vartheta_z(W_Z, W_Z)$ and $W_\mathbb{U}^\Omega$. The predicted mean μ and covariance σ^2 can be obtained by adjusting the joint Gaussian prior distribution over the observations, following the formula for the conditional distribution of the multidimensional normal distribution, defined as

$$\mu_z = k^{\hat{\mathbb{Z}}} \mathbf{K}^{-1} (\Lambda(W_Z) - \mu(W_Z) + \mu(W_{z+1})) \quad (4.7)$$

where $\hat{\mathbb{Z}}$ is the subset of $\mathbb{Z}$ that ends at the current z and

$$\vartheta_z^2 = k(W_{z+1}, W_{z+1}) - k^{\hat{\mathbb{Z}}} \mathbf{K}^{-1} k \quad (4.8)$$

Algorithm 2 BOFedAvg aggregation procedure.

```

1: procedure BOFedAvg( $\mathbb{U}, W_\mathbb{U}, \Lambda^\mu, \Lambda, \mathbb{E}_\mathbb{U}, \Psi$ )
2:    $h = |W_\mathbb{U}^\pi| = (|\mathbb{U}| \times |\Psi| - 1; |\mathbb{Z}| = \max(|W_\mathbb{U}^\pi| \times 2, 5);$ 
3:    $W_\mathbb{U}^\Omega = \text{Set of } h \text{ random } W_\mathbb{U}^z \text{ where } 0 \geq w_u \geq 1, w_u \in W_\mathbb{U}^z;$ 
4:   for each  $W_\mathbb{U}^z \in W_\mathbb{U}^\Omega$  do
5:      $\Lambda^z = \Lambda(W_\mathbb{U}^z);$ 
6:   end for
7:   for each BO iteration  $z = h + 1, h + 2, \dots, |\mathbb{Z}|$  do
8:     Estimate the covariance matrix  $\mathbf{K}(W, W)$  using Equations 4.2, 4.3;
9:     Compute  $W_\mathbb{U}^z = W_\star$  using Equations 4.4, 4.5, 4.6;
10:     $\Lambda^z = \Lambda(W_\mathbb{U}^z);$ 
11:    Adjust prior distribution with  $(W_\mathbb{U}^z, \Lambda^z)$  using Equations 4.7, 4.8;
12:  end for
13:   $\Lambda^\ominus = \min \Lambda^z; W_\mathbb{U}^* = W_\mathbb{U}^z \text{ where } \Lambda^z = \Lambda^\ominus; \Lambda^* = \Lambda(W_\mathbb{U}^*);$ 
14:  if  $\Lambda^\mu < \Lambda^*$  then
15:     $W_\mathbb{U}^* = W_\mathbb{U};$ 
16:  end if
17:  Apply  $W_\mathbb{U}^*$  to  $\mathbb{E}_\mathbb{U}$  to produce  $\xi^*$  and return  $\xi^*$ ;
18: end procedure
```

As illustrated in Algorithm 2, the core BO process will repeat this procedure multiple times until the maximum number of iterations $|\mathbb{Z}|$ is reached. Afterward, we take the best averaging weights found ($W_\mathbb{U}^*$) and compare its validation loss Λ^* with Λ^μ . If $\Lambda^\mu < \Lambda^*$, $W_\mathbb{U}^*$ is replaced by $W_\mathbb{U}$ and ξ^* will be equivalent to the model produced by standard FedAvg

for round r (ξ^μ). In this algorithm, $\mathbb{E}_\mathbb{U}$ is the set of trained local models in a given round.

4.3.2 Bayes Optimized FedSCDepth (BOFedSCDepth)

In BOFedSCDepth, we extend FedSCDepth [45, 46] by replacing its FL component with one that optimizes the aggregation weights with BOFedAvg and verifies if the global model produced in a given round (ξ^μ) is better than the one produced in the previous round (ξ^{r-1}), based on the same evaluation function (Λ) that is used in BOFedAvg. Algorithms 3 and 4 detail our method, reusing the procedures of Algorithm 2. In this algorithm, AVs $\mathbb{C}$ are indexed by c (while participating AVs $\mathbb{U}$ are indexed by u), v is the fraction of active AVs in each FL round, e is the number of training passes each AV makes in its local dataset in each round (local epochs), $|\mathbb{U}|$ is the number of AVs selected for each round, and $\mathbb{R}$ is the set of rounds indexed by r .

Algorithm 3 BOFedSCDepth FL procedure.

Input: $\varepsilon, \mathbb{C}, v, e, \mathbb{R}, \Psi$;

```

1:  $\varepsilon^0 = \varepsilon$ ;  $\Lambda_0 \cong +\infty$ ;
2: for each round  $r \in \mathbb{R}$  do
3:    $t = r - 1$ ;  $|\mathbb{U}| = \max(v \times |\mathbb{C}|, 1)$ ;
4:    $\mathbb{U} = \text{Random Set of } |\mathbb{U}| \text{ AVs that will be active in round } r$ ;
5:    $\hat{\mathbb{U}} = \text{Random Set of } |\hat{\mathbb{U}}| \text{ AVs that were active in round } t$ ;
6:    $\mathbb{E}_\mathbb{U} = \text{empty vector with } |\mathbb{U}| \text{ positions}$ ;
7:   for each AV  $u \in \mathbb{U}$  in parallel do
8:     Compute  $\varepsilon_u^r = \text{LocalUpdate}(u, \varepsilon^t, e)$  and Save  $\varepsilon_u^r$  into  $\mathbb{E}_\mathbb{U}[u]$ ;
9:   end for
10:  Compute  $\xi^\mu$  using Equation 4.1 and extract vector  $W_\mathbb{U}$  from  $\xi^\mu$ ;
11:   $\Lambda = \text{AbsRel}(\text{Normalize}(W'_\mathbb{U}))$ ;  $\Lambda^\mu = \Lambda(W_\mathbb{U})$ ;
12:   $\xi^* = \text{BOFedAvg}(B, W_\mathbb{U}, \Lambda^\mu, \Lambda, \mathbb{E}_\mathbb{U}, \Psi)$ ;
13:  Extract aggregation weights vector  $W_\mathbb{U}^*$  from  $\xi^*$ ;
14:   $\Lambda^\mu = \Lambda(W_\mathbb{U}^*)$ ;
15:  if  $\Lambda_t < \Lambda^\mu$  then
16:     $\Lambda^\mu = \Lambda_t$ ;  $W_\mathbb{U} = W_{\hat{\mathbb{U}}}$ ;  $\xi^\mu = \varepsilon^t$ ;
17:  end if
18: end for
Output:  $\varepsilon$ ;
```

Algorithm 4 BOFedSCDepth auxiliary procedures.

```

1: procedure  $\text{LocalUpdate}(u, \varepsilon', e)$ 
2:   for each local epoch  $i$  from 1 to  $e$  do
3:     Execute self-supervised learning with  $\varepsilon'$  on AV  $u$  using loss defined in Eq. 3.6;
4:     Compute  $\Lambda$  of updated  $\varepsilon'$  with Validation Dataset;
5:   end for
6:   return updated  $\varepsilon'$ ;
7: end procedure
```

4.4 Evaluation Experiments

To address the research questions of the present study, **RQ3** and **RQ4**, we design and execute evaluation experiments detailed in the following subsections.

4.4.1 Datasets

As in Chapter 3, our evaluation experiments are conducted with the publicly available KITTI [47] and DDAD [48] datasets.

KITTI contains monocular images and 3D scans from scenes captured by cameras and sensors mounted on moving cars [47]. As in Chapter 3, we use Eigen’s split [95], with the max depth set to 80 meters and train images resized to a resolution of 832×256 pixels.

DDAD contains monocular images and 3D scans captured by cameras and sensors mounted on self-driving cars moving in urban environments on all continents [48]. As in Chapter 3, we adopt the standard train/test split, with the max depth set at 200 meters and the images resized to 640×384 pixels for training.

4.4.2 Scenarios

As in Chapter 3, we assume that the drives present in the training datasets (34 for KITTI and 150 for DDAD) correspond to different AVs, although some were collected by the same vehicle in different periods. Provided that, we adopt two of the experimental scenarios proposed in Chapter 3: FT-IID and FT-NIID.

As in Chapter 3, FL base scenarios were derived into 18 scenarios by combining the use of IID or NIID data ($s \in S$, where $s = true$ corresponds to IID data), multiple fractions of active participants in each round ($v \in V$) and the numbers of local epochs of training per active participant ($e \in E$): $S = \{true, false\}$, $V = \{1, \frac{1}{2}, \frac{1}{3}\}$, and $E = \{1, 2, 3\}$.

All FL scenarios simulated training with 12 participants ($|\mathbb{C}|$) for 12 rounds ($|\mathbb{R}|$) and adopted the same sample redistribution strategy adopted in Chapter 3.

4.4.3 Metrics

To evaluate the efficacy of the final model, related to **RQ4**, we use standard depth metrics [45,46], detailed in Chapter 3: mean absolute relative error (*AbsRel*), squared relative error (*SqRel*), root mean squared error (*RMS*), root mean squared log error (*RMSlog*), and

accuracy under threshold ($\Delta_j < 1.25^j, j = 1, 2, 3$) detailed in Eigen’s work [95]. The predicted depth maps, as in [213], are scaled by the median for evaluation.

For evaluating training efficiency, related to **RQ3**, we employ FedSCDepth’s initial evaluation metrics [45, 46]: (1) *AbsRel* in the validation set during training (Λ); (2) upper ($\mathcal{O}_{max}$) and lower ($\mathcal{O}_{min}$) communication cost bounds [45, 46]; (3) computational cost estimates, including the number of training steps (τ_{train}) for FT (τ_{train}^{FT}) [45, 46].

The metrics are computed following [45, 46], except for τ_{train}^{FT} , where random batch re-sampling is disabled to avoid limiting efficiency gains in FT-NIID scenarios. In addition, we assess two additional computational cost metrics: the number of inference batches (Validation Steps) and averaging operations (η) in each federation round. Due to BOFedAvg’s multiple weight combination evaluations, we examine their potential impact on FL’s overall computational cost.

Formally, we compute the number of Validation Steps (τ_{val}) required by FedAvg (τ_{val}^μ) as $\tau_{val}^\mu = \sum_r^{\hat{\mathbb{R}}} \tau_{val}^\mu(r)$ with

$$\tau_{val}^\mu(r) = \rho_{val} + \sum_u^{\mathbb{U}} (\rho_{val} \times e_u) \quad (4.9)$$

where $\rho_{val} = \frac{\phi_{val}}{v}$, with $v = 4$ as the batch size, $\hat{\mathbb{R}}$ is the set of FL rounds elapsed, $\mathbb{U}$ are the AVs that performed training on the round r , and e_u is the number of epochs through which AV p performed validation over ρ_{val} batches [45, 46]. ϕ_{val} is the number of labeled validation samples on which inference is computed to assess Λ of the local models and the averaged models, where the validation set is shared by all AVs and the central server. Thus, we estimate the τ_{val} of BOFedAvg (τ_{val}^*) as

$$\tau_{val}^*(r) = \tau_{val}^\mu(r) + \rho_{val} \times |\mathbb{Z}| \quad (4.10)$$

$$\tau_{val}^* = \sum_r^{\hat{\mathbb{R}}} \tau_{val}^*(r) \quad (4.11)$$

where $\mathbb{Z}$ and r are as defined in section 4.3.1. η on FedAvg (η^μ) is always equal to $|\hat{\mathbb{R}}|$ (one averaging per round r), while for BOFedAvg we have $\eta^* = |\hat{\mathbb{R}}| \times (1 + |\mathbb{Z}|)$.

4.4.4 Implementation Details

Our prototype³ was based on FedSCDepth’s source code. As in Chapter 3, our deep neural network used Adam optimizer, and the learning rate was set to 10^{-4} . The deep

³<https://github.com/eltonfss/federated-sc-depth/tree/bofscd>

neural network encoder was initialized using ImageNet [229] pre-trained weights. The maximum number of batches per epoch was set as $1k$ and the batch size as 4. The 18 FL scenarios ran with $|\mathbb{R}| = 12$, $|\mathbb{C}| = 12$ and all combinations of $(v \in V, e \in E, s \in S)$, reusing the hardware of FedSCDepth’s experiments, detailed in Chapter 3.

4.4.5 Ablation Studies

In this section, we analyze the impact of the number of autonomous active vehicles per round ($v \times |\mathbb{C}| = |\mathbb{U}|$), the number of local epochs (e), the number of rounds (r) and the heterogeneity of the data (s) on the lowest global validation losses (Λ_{min}) and their corresponding communication and computational costs, based on the results presented at Tables 4.2, 4.3, 4.4, 4.5, and Figure 4.3.

Table 4.2: Lowest global validation loss (Λ_{min}), its max and min communication costs ($\bar{O}_{max}/\bar{O}_{min}$), number of training steps (τ_{train}), validation steps (τ_{val}) and averaging operations (η), by number of active AVs per round ($|\mathbb{U}|$), grouped by dataset (DS) and FT scenarios (Scenario). *Best results per group in bold.

DS	Scenario	$ \mathbb{U} $	Λ_{min}	$\bar{O}_{max}$	$\bar{O}_{min}$	τ_{train}	τ_{val}	η
KITTI	FT-IID	4	0.1418	39.9 GB	13.3 GB	96k	271.9k	392
		6	0.1452	44.8 GB	22.4 GB	108k	428.3k	657
		12	0.1445	49.8 GB	49.8 GB	240k	951.7k	1450
KITTI	FT-NIID	4	0.1522	49.8 GB	16.6 GB	120k	339.9k	490
		6	0.1507	54.8 GB	27.4 GB	66k	486.1k	803
		12	0.1490	49.8 GB	49.8 GB	240k	951.7k	1450

Table 4.3: Lowest global validation loss (Λ_{min}), its $\bar{O}_{max}/\bar{O}_{min}$, number of training steps (τ_{train}), validation steps (τ_{val}) and averaging operations (η), by number of active AVs per round ($|\mathbb{U}|$), grouped by dataset (DS) and FT Scenarios. *Best results per group in bold.

DS	Scenario	$ \mathbb{U} $	Λ_{min}	$\bar{O}_{max}$	$\bar{O}_{min}$	τ_{train}	τ_{val}	η
DDAD	FT-IID	4	0.1781	59.8 GB	19.9 GB	144k	711k	588
		6	0.1681	59.8 GB	29.9 GB	216k	1066.5k	876
		12	0.1678	59.8 GB	59.8 GB	288k	1990.8k	1740
DDAD	FT-NIID	4	0.1626	59.8 GB	19.9 GB	144k	711k	588
		6	0.1768	59.8 GB	29.9 GB	216k	1066.5k	876
		12	0.1672	59.8 GB	59.8 GB	432k	2133k	1740

4.4.5.1 Impact of number of participants per round

Tables 4.2 and 4.3 show that the lowest validation losses in KITTI and DDAD were obtained with $|\mathcal{U}| = 4$ or $|\mathcal{U}| = 12$. This was also the case for $\bar{\mathcal{O}}_{max}$ and $\bar{\mathcal{O}}_{min}$, although in DDAD $\bar{\mathcal{O}}_{max}$ was the same for all values of $|\mathcal{U}|$. The computational costs followed a similar trend on both datasets, except for the τ_{train} of $|\mathcal{U}| = 6$ in FT-NIID, which was the lowest overall.

4.4.5.2 Impact of number of local epochs

Table 4.4: Lowest global validation loss (Λ_{min}), its max and min communication costs ($\bar{\mathcal{O}}_{max}/\bar{\mathcal{O}}_{min}$), number of training steps (τ_{train}), validation steps (τ_{val}) and averaging operations (η), by number of epochs (e), grouped by dataset (DS) and FT scenarios (Scenario).

DS	Scenario	e	Λ_{min}	$\bar{\mathcal{O}}_{max}$	$\bar{\mathcal{O}}_{min}$	τ_{train}	τ_{val}	η
KITTI	FT-IID	1	0.1461	59.8 GB	59.8 GB	144k	1060.5k	1740
		2	0.1445	49.8 GB	49.8 GB	240k	951.7k	1450
		3	0.1418	39.9 GB	13.3 GB	96k	271.9k	392
KITTI	FT-NIID	1	0.1499	34.9 GB	34.9 GB	84k	618.6k	1015
		2	0.1490	49.8 GB	49.8 GB	240k	951.7k	1450
		3	0.1521	54.8 GB	54.8 GB	396k	1121.7k	1595

Table 4.5: Lowest global validation loss (Λ_{min}), its max and min communication costs ($\bar{\mathcal{O}}_{max}/\bar{\mathcal{O}}_{min}$), number of training steps (τ_{train}), validation steps (τ_{val}) and averaging operations (η), by number of epochs (e), grouped by dataset (DS) and FT scenarios (Scenario).

DS	Scenario	e	Λ_{min}	$\bar{\mathcal{O}}_{max}$	$\bar{\mathcal{O}}_{min}$	τ_{train}	τ_{val}	η
DDAD	FT-IID	1	0.1902	59.8 GB	59.8 GB	144k	1848.6k	1740
		2	0.1678	59.8 GB	59.8 GB	288k	1990.8k	1740
		3	0.1681	59.8 GB	29.9 GB	216k	1066.5k	876
DDAD	FT-NIID	1	0.1902	59.8 GB	59.8 GB	144k	1848.6k	1740
		2	0.1736	59.8 GB	59.8 GB	288k	1990.8k	1740
		3	0.1626	59.8 GB	19.9 GB	144k	711k	588

In Tables 4.4, and 4.5, the lowest validation losses were observed with $e = 2$ or $e = 3$, both in KITTI and DDAD. This was also true in most cases for $\bar{\mathcal{O}}_{max}$ and $\bar{\mathcal{O}}_{min}$, although, in the FT-NIID (KITTI) scenario, they were lower with $e = 1$ and, in DDAD, $\bar{\mathcal{O}}_{max}$ was

the same for all values of e . The computational costs followed a similar trend on both datasets, except for FT-NIID (KITTI), in which they were all lower with $e = 1$, and on FT-IID (DDAD), where τ_{train} was also lower with $e = 1$.

4.4.5.3 Impact of number of federation rounds

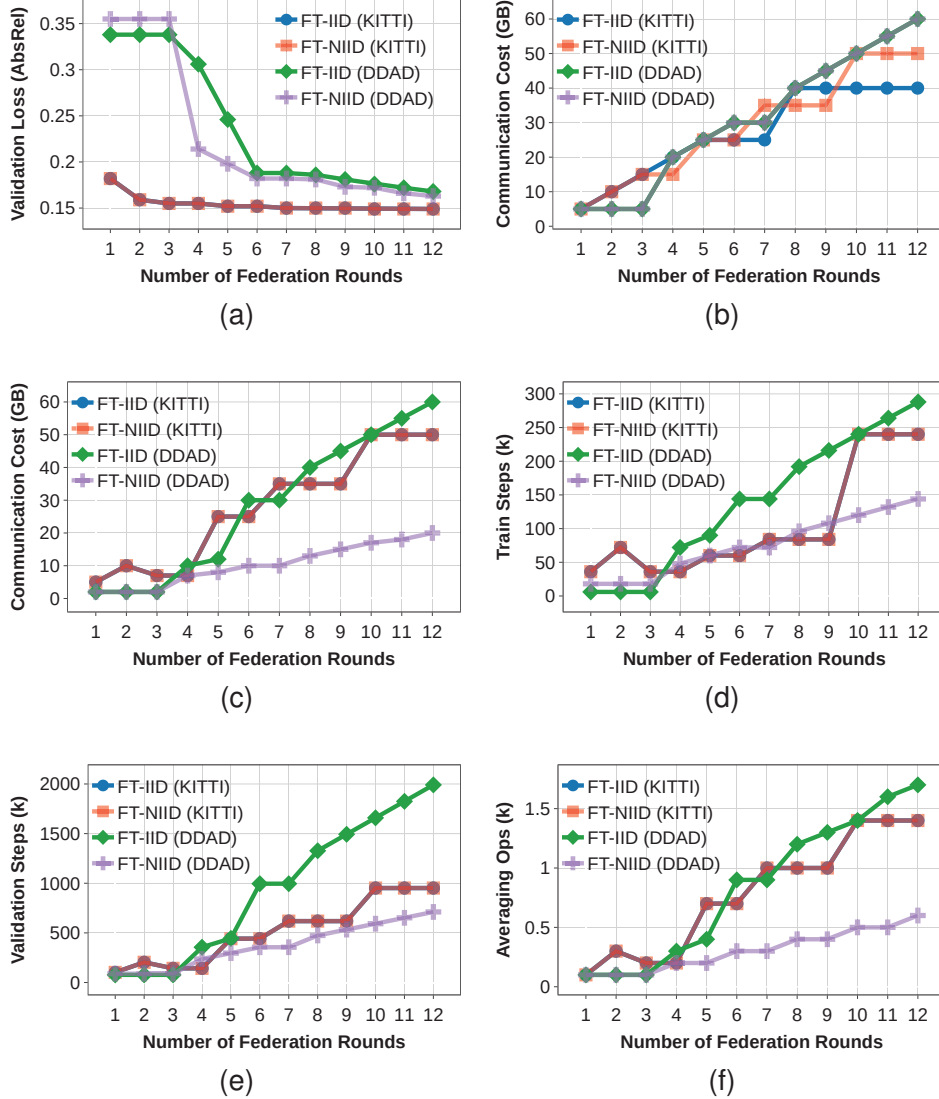


Figure 4.3: Lowest global validation loss (a), its communication cost upper bound (b), and lower bound (c), and number of train steps (d), validation steps (e), and averaging operations (f) to obtain it, by the number of rounds.

In Figure 4.3 (a), validation loss on KITTI improves rapidly until $r = 5$, with a modest decrease afterward. In DDAD, validation loss starts to improve significantly at $r = 4$, dropping rapidly until $r = 6$. Regarding the communication cost metrics in Figure 4.3 (b, c) on KITTI, both exhibit an approximately linear increase with r , reaching up to about 50 GB in $r = 12$. However, for FT-NIID, the lower bound requires 2 to 5 times higher computational cost than FT-IID, which reaches a maximum of about 20 GB at $r = 4$. In

DDAD, a linear trend is observed, with a maximum of about 60 GB at $r = 12$. Unlike KITTI, on DDAD, the FT-IID lower bound is up to 3x higher than FT-NIID's, which reaches a maximum of about 20 GB at $r = 12$. Concerning the computational cost metrics in Figure 4.3 (d, e, f), the same trends observed in the lower bound of communication costs in KITTI reach a maximum of approximately 250k τ_{train} , 951.7k τ_{val} and 1.4k η with FT-NIID. In DDAD, similar trends are observed in the lower bound of communication costs, reaching almost 300k τ_{train} , 1990.8k τ_{val} , and more than 1.5k η with FT-IID.

4.4.5.4 Impact of data heterogeneity

Analyzing Figure 4.3 and Tables 4.2, 4.3, 4.4, and 4.5, we can observe that the proposed method exhibits robustness to the inherent data heterogeneity in data collection, as evidenced by the closely comparable lowest validation loss with NIID and IID data in both KITTI and DDAD experiments. However, a significant impact on communication and computational costs is observed in KITTI, with FT-NIID requiring more data transfer and training steps to achieve its best validation loss compared to FT-IID.

4.4.5.5 Impact of Search Range Hyperparameter

Considering the configurations $|\mathcal{U}| = 4$, $e = 3$, and $s = true$, which yielded the best results on KITTI, and $|\mathcal{U}| = 4$, $e = 3$, and $s = false$, which yielded the best results on DDAD, we have performed additional ablation experiments to determine the optimal value for Ψ , considering the values 4, 6 (default), 8, 10, and 12, as options. The results obtained are depicted in Figures 4.4 and 4.5.

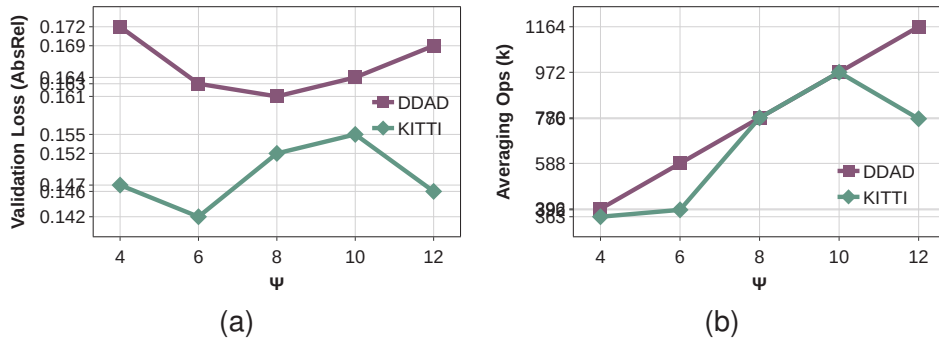


Figure 4.4: Lowest validation loss (a) and number of averaging operations required for obtaining it (b), by value of Ψ for KITTI and DDAD datasets.

In Figure 4.4 (a), we show that the lowest validation loss was obtained with $\Psi = 6$ (for KITTI) and $\Psi = 8$ (for DDAD), suggesting that increasing the value Ψ above these values might not present an interesting trade-off, considering the computation overhead that might be introduced by additional iterations of BO, as illustrated in Figure 4.4 (b).

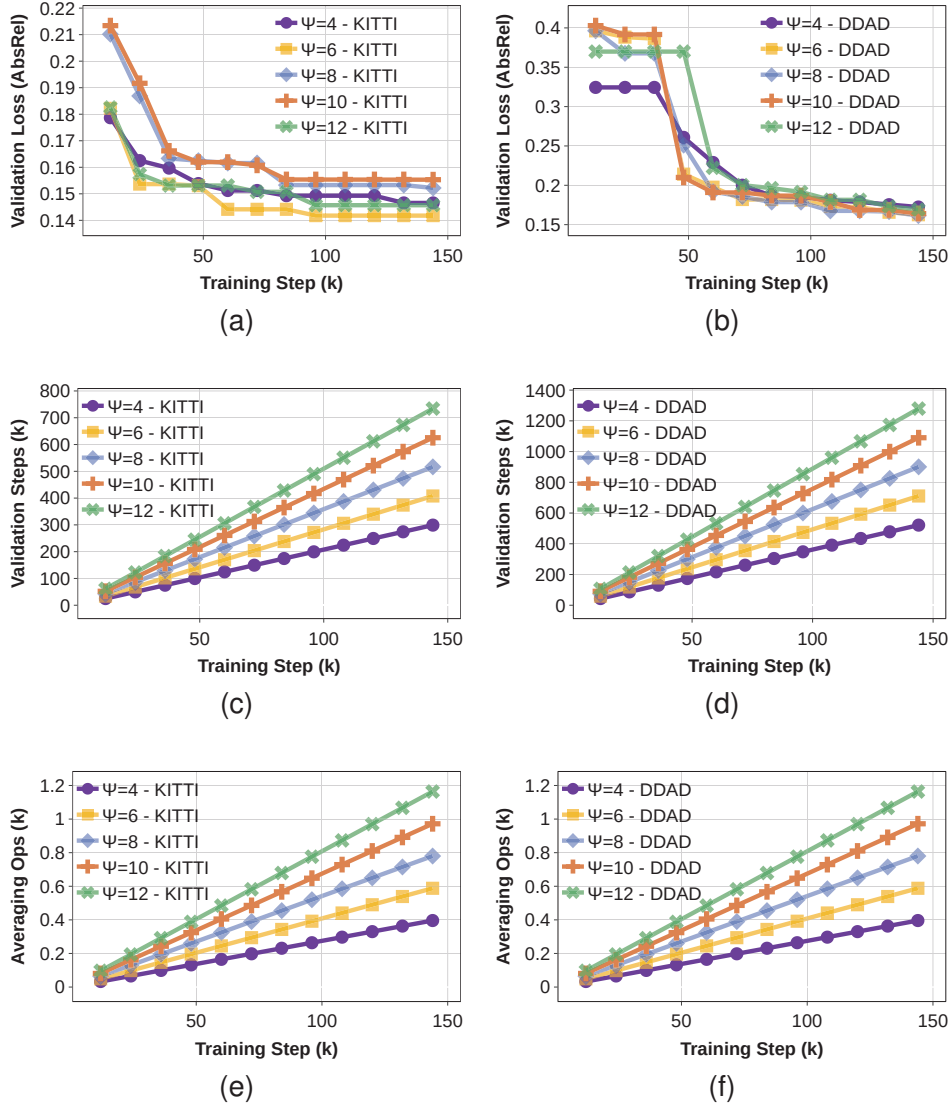


Figure 4.5: Lowest validation loss obtained with varying values of Ψ (a, b), and its required number of validation steps (c, d), and averaging operations (e, f), by the number of training steps, for KITT and DDAD datasets.

Moreover, in Figures 4.5 (a) and (b), we observe that there is no clear trend regarding higher values of Ψ leading to lower initial validation losses (at the end of the first round). Furthermore, in the following rounds of training, any advantage observed tends to dissipate, and at the end of the training we can see that training with $\Psi = 6$ and $\Psi = 8$ obtained the lowest validation losses in KITT and DDAD, respectively. However, the difference in losses obtained with other values of Ψ was not that large, especially for DDAD.

Regarding computational cost metrics, in Figure 4.5 (c), (d), (e) and (f), we observe that the number of validation steps and averaging operations grows almost linearly with the number of training steps for all values of Ψ , with the final number of validation steps of $\Psi = 12$ being about 1.8 times larger than with $\Psi = 6$, for example. Moreover, the final

number of averaging operations performed with $\Psi = 12$ was approximately two times larger than with $\Psi = 6$, as expected.

4.4.5.6 Impact of Acquisition Function Choice

Taking as a reference the same configurations used in Section 4.4.5.5, and the selected value of $\Psi = 6$, we compare the performance of the results obtained with the GP-Hedge (GPH) acquisition function against the results obtained with the other options presented in Chapter 2: Expected Improvement (EI), Probability of Improvement (PI), and Lower/Upper Confidence Bound (CB).

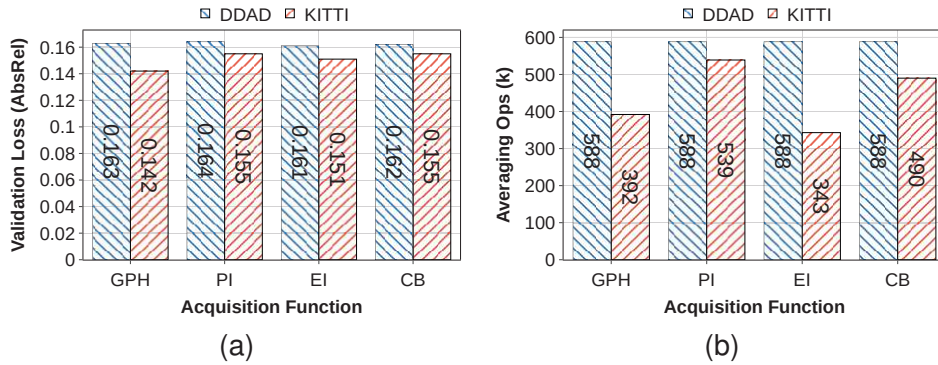


Figure 4.6: Lowest validation loss (a) and number of averaging operations for obtaining it (b), per acquisition function for KITTI and DDAD datasets.

In Figure 4.6 (a), we show that the lowest validation losses were obtained with GP-Hedge and Expected Improvement, for KITTI and DDAD, respectively. Meanwhile, in Figure 4.6 (b), we show that the number of averaging operations required to reach the lowest validation loss was the same with all acquisition functions for KITTI, while for DDAD it was lower with Expected Improvement as the acquisition function.

Provided that the difference in the validation losses obtained with different acquisition functions for DDAD was very small (maximum of 0.003) and the improvement observed in this metric when using the GP-Hedge acquisition function compared to the best alternative (Expected Improvement) was three times greater (0.009), we maintain GP-Hedge as our choice of acquisition function for the proposed method.

4.4.6 Comparison with State-of-the-Art on Self-Supervised FL of Monocular Depth Estimators

From the ablation studies, we concluded that the combination of $|\mathbb{U}| = 4$, $e = 3$, $\Psi = 6$, and $s = \text{true}$ yielded the best results in all metrics considered in KITTI, while the

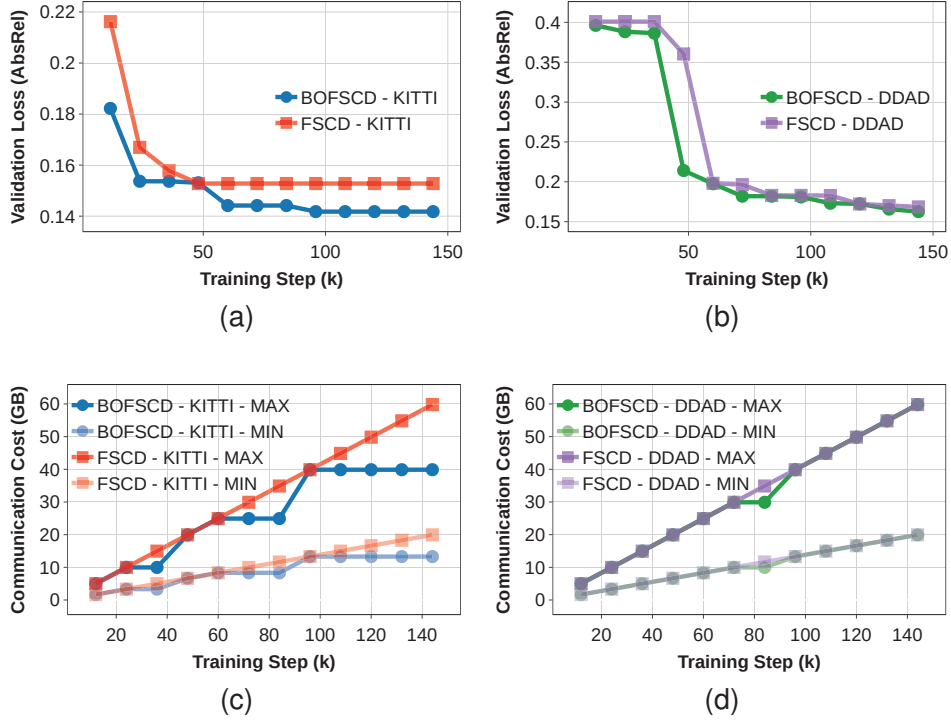


Figure 4.7: Lowest validation loss obtained with BOFSCD and FSCD (a, b), and its estimated communication cost upper and lower bounds (c, d), by the number of training steps, for KITTI and DDAD datasets.

same was true for DDAD, but with $s = false$. Although in DDAD the lowest validation loss was obtained with $\Psi = 8$, the improvement compared to $\Psi = 6$ was so low that we decided to use $\Psi = 6$ since it introduces less computational overhead and allows a more standardized comparison with KITTI results.

To provide a fair comparison of our method’s results with FedSCDepth [45, 46], we have reproduced its training using the configurations chosen from our ablation studies, since in its published experiments its authors only trained it with 9 to 10 AVs, while our training we have used 12 in total. Thus, we have calculated the efficiency metrics of the BOFedSCDepth (BOFSCD) and FedSCDepth (FSCD) methods during training, which are presented in Figures 4.7 and 4.8 based on the number of training steps required for each federation round.

In Figures 4.7 (a, b), we can observe that the largest differences between the BOFSCD and FSCD validation losses are observed in the initial rounds, with the BOFSCD validation loss always being better or equal to the FSCD validation loss. Meanwhile, in Figures 4.7 (c, d), we can observe that the communication cost of FSCD grows linearly with the number of training steps, while for BOFSCD it only grows when the updated global model improves or maintains the validation loss of the previous round. At the end

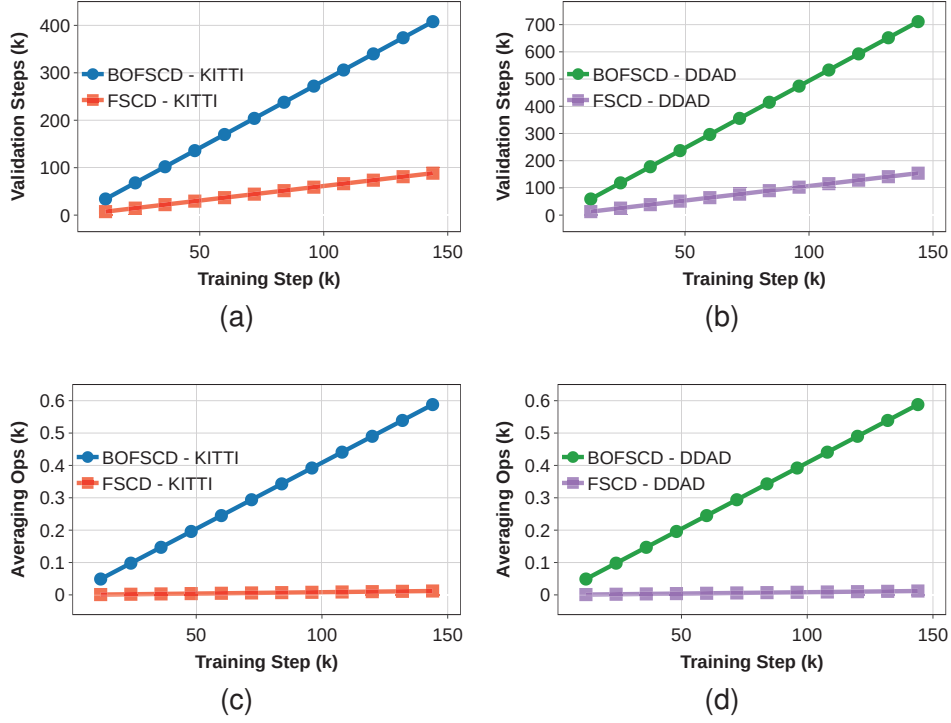


Figure 4.8: The number of validation steps (a, b) and the number of averaging operations (c, d), by the number of training steps, for KITTI and DDAD datasets.

of the training $\bar{\mathcal{U}}_{max}$ and $\bar{\mathcal{U}}_{min}$ were approximately 33.3% lower with BOFSCD in KITTI and the same as with FSCD in DDAD. In addition, in Figures 4.8 (a, b), we observe that τ_{val} grows linearly at each round with both methods, but its growth rate is approximately $4.6\times$ higher with BOFSCD due to its multiple evaluations of the intermediary global models generated by varying the averaging weights of each local model. The growth rate is also larger with DDAD than with KITTI because it has a larger validation set.

Finally, in Figures 4.8 (c, d), we note that the η of BOFSCD is always $49\times$ the η of FSCD, which is always equal to the number of rounds, both for KITTI and DDAD.

Table 4.6: Efficacy of BOFSCD (Ours) and FSCD (Baseline) methods on KITTI (Eigen Split) dataset (DS), at round (r), grouped by DS and r . *Best results per group in bold.

r	DS	Method	$AbsRel$	$SqRel$	RMS	$RMSlog$	Δ_1	Δ_2	Δ_3
4	KITTI	FSCD	0.120 ± 0.002	0.762 ± 0.026	4.667 ± 0.079	0.186 ± 0.003	0.859 ± 0.004	0.961 ± 0.002	0.985 ± 0.001
		BOFSCD	0.125 ± 0.002	0.777 ± 0.024	4.757 ± 0.075	0.190 ± 0.003	0.848 ± 0.005	0.959 ± 0.002	0.985 ± 0.001
6	KITTI	FSCD	0.121 ± 0.002	0.775 ± 0.026	4.630 ± 0.078	0.186 ± 0.003	0.859 ± 0.004	0.962 ± 0.002	0.985 ± 0.001
		BOFSCD	0.119 ± 0.002	0.753 ± 0.024	4.754 ± 0.079	0.189 ± 0.003	0.859 ± 0.004	0.960 ± 0.002	0.984 ± 0.001
12	KITTI	FSCD	0.119 ± 0.002	0.781 ± 0.027	4.631 ± 0.073	0.185 ± 0.003	0.863 ± 0.004	0.963 ± 0.002	0.985 ± 0.001
		BOFSCD	0.119 ± 0.002	0.762 ± 0.025	4.581 ± 0.078	0.184 ± 0.003	0.865 ± 0.004	0.962 ± 0.002	0.985 ± 0.001

In Tables 4.6 and 4.7, we analyze the efficacy metrics obtained in the KITTI and

DDAD test sets. The results presented correspond to the 95% confidence interval of each metric's mean computed with 1k bootstrap [242] samples. In addition to the results obtained by the global models generated at the end of the training with each method ($r = 12$), we also compute the efficacy metrics of the global models generated in a third and half of the training rounds ($r = 4$ and $r = 6$). Observing the efficiency metrics at different stages of federated training, we can complement the insights obtained by the analysis of Figures 4.7 (a, b), regarding the advantage of BOFSCD in the initial rounds of training.

Table 4.7: Efficacy of BOFSCD (Ours) and FSCD (Baseline) methods on DDAD dataset (DS), at round (r), grouped by DS and r . *Best results per group in bold.

r	DS	Method	$AbsRel$	$SqRel$	RMS	$RMSlog$	Δ_1	Δ_2	Δ_3
4	DDAD	FSCD	0.364 ± 0.003	8.693 ± 0.205	26.120 ± 0.171	0.742 ± 0.002	0.436 ± 0.002	0.617 ± 0.002	0.699 ± 0.002
		BOFSCD	0.215 ± 0.001	5.412 ± 0.055	22.755 ± 0.155	0.381 ± 0.002	0.615 ± 0.002	0.827 ± 0.002	0.906 ± 0.001
6	DDAD	FSCD	0.196 ± 0.001	4.340 ± 0.054	20.144 ± 0.160	0.322 ± 0.002	0.692 ± 0.002	0.871 ± 0.001	0.934 ± 0.001
		BOFSCD	0.182 ± 0.001	4.251 ± 0.055	19.996 ± 0.154	0.314 ± 0.002	0.715 ± 0.002	0.881 ± 0.001	0.936 ± 0.001
12	DDAD	FSCD	0.169 ± 0.001	3.531 ± 0.064	17.659 ± 0.146	0.278 ± 0.002	0.758 ± 0.002	0.906 ± 0.001	0.952 ± 0.001
		BOFSCD	0.162 ± 0.001	3.433 ± 0.060	17.321 ± 0.148	0.273 ± 0.002	0.770 ± 0.002	0.908 ± 0.001	0.954 ± 0.001

Starting from the final global models ($r = 12$), we note that the models trained with BOFSCD were able to achieve better results (than those trained with FSCD) on most of the efficacy metrics in the KITTI and DDAD datasets. However, the difference in the values of these metrics was very small, with $AbsRel$ the same for both KITTI and about 4.1% better with BOFSCD on DDAD. Meanwhile, if we look at the global models obtained after the first half of the federation rounds ($r = 6$), we can observe that the advantage of BOFSCD in DDAD is a bit higher, with $AbsRel$ about 7.1% better. Also, this is the only round (out of the three analyzed) in which we could observe an improvement of $AbsRel$ on KITTI when using BOFSCD, about 1.6% better than with FSCD.

Finally, the results obtained by the global models generated in the first third of the federation rounds ($r = 4$) show a much more significant advantage for BOFSCD in DDAD, with $AbsRel$ about 40.1% better than with FSCD, which highlights what we believe is the greatest potential benefit of using BOFSCD. However, all efficacy metrics on KITTI, in the same round, were better with FSCD than with BOFSCD, $AbsRel$ being 4% worse with our proposed method. Thus, one could conclude that the usage of BOFSCD was not as useful in KITTI, since in most rounds of training, the results were very close and would not justify the additional computational cost introduced by the use of BO during the aggregation phase. However, this additional cost would probably be worth the improvement in efficacy observed in DDAD, at least in the initial rounds of federated training, where the advantage of using BOFSCD is more evident.

4.5 Discussion

Regarding **RQ3**, the combination of BO with FL outperforms the use of standard FL for self-supervised learning of monocular depth estimators for AVs in terms of $\bar{U}_{max}$, $\bar{U}_{min}$ and τ_{train} , achieving better Λ with fewer training steps and communication cost. Despite an increased central server computational load, due to its higher η and τ_{val} , the use of BO is compensated for by the reduction in the communication and computational burdens in AVs, highlighting the importance of efficient use of the network bandwidth and the computing power of AVs above all. In addition, this novel approach shows good efficiency with IID and non-IID data.

Concerning **RQ4**, we find that the final models trained with the use of BO during aggregation of the local models outperform those trained with standard FedAvg aggregation in both KITTI and DDAD. However, the efficacy advantage of the BOFedSCDepth model on KITTI was very small, whereas on DDAD it was more significant, especially in the initial rounds. Furthermore, the model’s efficacy remains consistent with both IID and non-IID data across most scenarios, highlighting its suitability for practical AV deployments where data collection and its characteristics are often imbalanced.

4.6 Limitations and Threats to Validity

Concerning the limitations of applicability of the proposed method, we highlight the assumption that the central aggregation server has sufficiently advanced hardware to run multiple iterations of inference with the global model. In standard FedAvg, this assumption is not required, as the server is only responsible for computing the averaging of the models. Another assumption required for our method is that there is a public validation dataset available to be used during optimization (to compute the global model loss). Although in most datasets such a “gold standard” set of samples is available, in real AV deployments, these could be more difficult to obtain. For example, to enable a large-scale FL campaign in a new area/domain, at least one AV would need to collect these validation samples in this area/domain. These samples would be used as the “gold standard” with which optimization would be performed.

Threats to validity include the fact that our experiments were conducted in a simulated setting, in which the local training of each AV was performed sequentially (without parallelization) due to the constraints of the hardware available for experimentation. Moreover, although the datasets used for experimentation are two of the state-of-the-art datasets for

monocular depth estimation, they may not sufficiently represent all the domains in which AVs might be deployed. Therefore, the results and conclusions obtained by this work are directly influenced and limited by the quality and representativeness of these datasets.

4.7 Conclusion

This chapter addresses monocular depth estimation for AVs by leveraging a fusion of FL and BO. Our approach combines these two techniques with self-supervised learning using unlabeled vehicle data to train a depth estimator efficiently, ensuring both high efficacy and privacy preservation. Prototype testing on KITTI and DDAD datasets shows that models trained with our method achieve state-of-the-art performance. Importantly, our method converges faster and incurs zero per-vehicle computational and communication costs overhead compared to our baseline. Future research will explore the generalizability of learned models in domain adaptation.

5. Continual Federated Learning of Monocular Depth for Autonomous Vehicles

5.1 Introduction

As discussed in Chapter 3, FL offers a promising solution by improving data privacy, limiting network usage, dispersing computational burdens, and strengthening connection resilience [15, 118]. Before this work, only FedSCDepth (Chapter 3) and BOFedSCDepth (Chapter 4) delved into self-supervised FL for monocular depth estimation in AVs. However, they did not tackle the challenges of continual learning. To overcome this limitation, we draw inspiration from recent research [24] demonstrating that Experience Replay (ER) [166] can help prevent “catastrophic forgetting” [23] of previously encountered data during training with novel datasets. Moreover, regularization-based methods [162] have also been employed to mitigate such forgetting. Thus, we hypothesize that the combination of experience replay and regularization with self-supervised FL facilitates the continual learning of monocular depth estimation models from unlabeled confidential AV data with enhanced efficiency, in contrast to existing methodologies. Hence, the objective of this study is to develop a solution for monocular depth estimation in AVs, generating depth maps from images captured by monocular cameras installed in moving AVs.

This research marks the first empirical demonstration that the use of experience replay and regularization can enhance continual self-supervised FL of monocular depth estimation models for AVs, addressing the research questions **RQ5**, **RQ6** and **RQ7**, detailed in Chapter 1, Section 1.5. To answer these questions, we provide the contributions **RC7** to **RC11**, also detailed in Chapter 1, Section 1.6.

Section 5.2 examines related work, Section 5.3 presents the proposed method, and Section 5.4 details the evaluation experiments. Section 5.5 discusses the results obtained. Section 5.6 outlines the conclusions and future work.

5.2 Related Work

In standard learning protocols, new datasets must be shared with a central server to retrain models using both previously used datasets and the new data. This incurs a high computational cost and exposure to raw data collected by AVs [24]. Meanwhile, in the continual learning protocol (detailed in Chapter 2), models can be re-trained incrementally by each AV using their data, without sharing them or downloading the entire data set used in initial training [24]. However, a key challenge in harnessing naive continuous learning to estimate monocular depth in practical AV deployments is the catastrophic forgetting problem [23]. This forgetting can be defined as the loss of previously acquired capabilities in a trained model when retraining it with new datasets with a different data distribution (*e.g.*, images of new locations or traffic conditions) or to perform a new task (*e.g.*, image classification or segmentation) [166].

For instance, assuming that an initial training campaign for a new AV model was conducted in a suburban area of California (USA), and we want to make sure that this AV will also perform well in a urban center of Rio de Janeiro (Brazil). We could just conduct a second training campaign in an urban center of Rio but, if we do naive continual learning, this could cause the forgetting of some of the features learned in the suburban area of California, causing the AV model to have poorer performance on that scenario.

Several alternatives to naive continual learning have been proposed in the literature, utilizing regularization, parameter isolation, rehearsal, and spatio-temporal relations to address forgetting in computer vision tasks [24]. In particular, rehearsal-based methods, such as experience replay [24, 203], systematically store data used in previous train iterations in a buffer. This buffer is periodically sampled to enhance new train data, forcing the model to “remember” the characteristics of old samples while learning on the new samples. Given a buffer of sufficient size, experience replay has been shown to outperform other rehearsal-based methods [24] while being among the best-performing methods for continual learning in computer vision tasks [166]. Thus, in the present work, we take inspiration from previous rehearsal-based and regularization-based techniques for continual learning in monocular depth estimation to devise novel experience replay and regularization methods. Our methods assume more realistic constraints associated with the FL setting, such as the limited access to samples of previous train datasets (to preserve privacy), as well as computational resource limitations of AVs (which might not allow the storage and sampling of the experience replay buffer within each AV during local training).

5.3 Proposed Methods

In this section, we describe our proposed methods, namely ERFedSCDepth [59], RFedSCDepth, and ERRFedSCDepth.

5.3.1 Experience Replay FedSCDepth

ERFedSCDepth consists of an experience replay augmented aggregation function called ERFedAvg, integrated into FedSCDepth’s [45] self-supervised FL framework for monocular depth estimation. Figure 5.1 provides an overview of ERFedAvg and its combination with FedSCDepth, highlighting each step of the learning process and the interactions between the *Central Server* and the *AVs*.

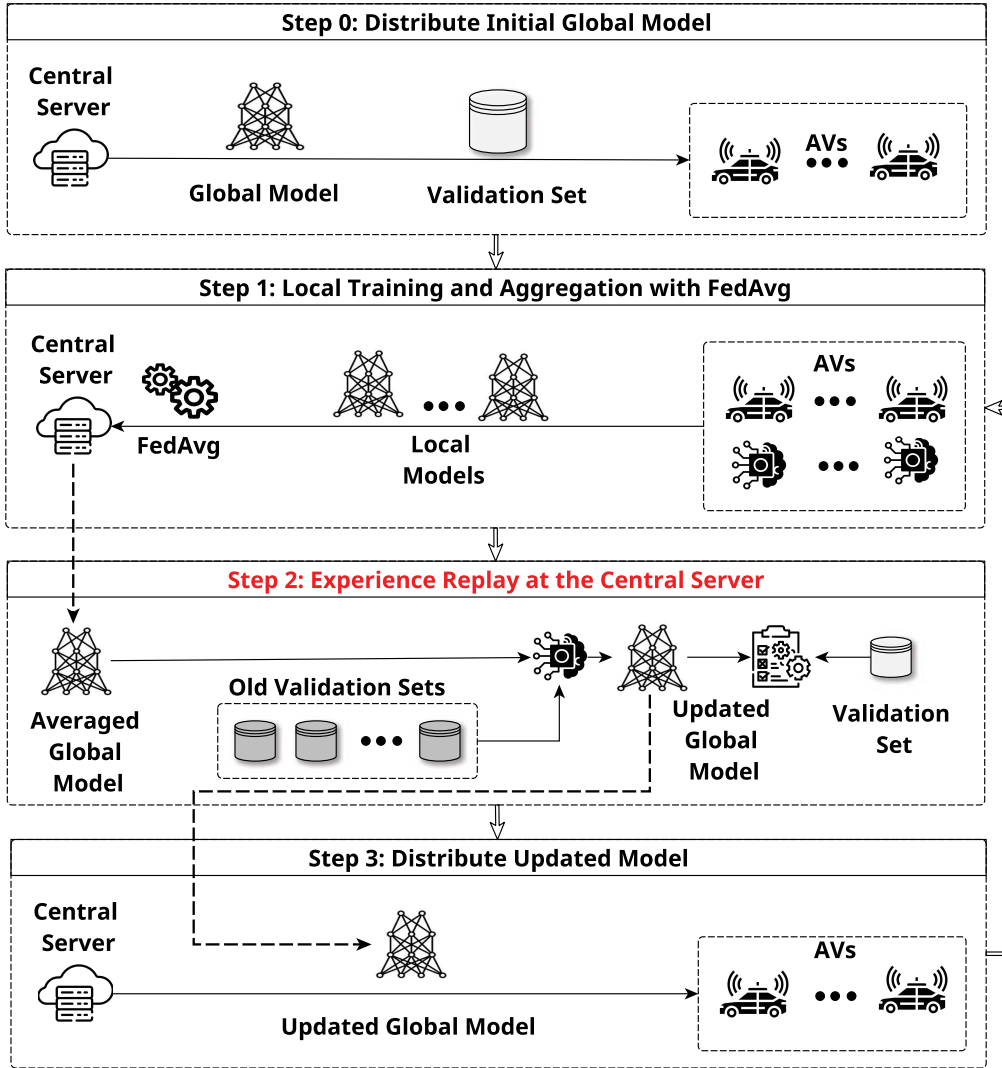


Figure 5.1: ERFedSCDepth overview.

In *Step 0*, the *Central Server* distributes the *Global Model* (trained with previous

datasets) and the *Validation Set*¹ of the new dataset to the AVs. After receiving those and collecting a representative set of new training samples, *Step 1* begins with each AV performing local self-supervised learning using the *Global Model* weights for initialization and the shared *Validation Set* for local evaluation. Afterward, the *Central Server* receives the *Local Models* trained locally by AVs and aggregates them using the standard FedAvg [118] technique, generating the *Averaged Global Model*.

Next, in *Step 2*, the *Central Server* performs self-supervised learning using *Averaged Global Model* weights for initialization and the *Old Validation Sets* (shared in previous datasets) as input data. This experience replay generates the *Updated Global Model* that is evaluated using the *Validation Set* of the current dataset. Finally, on *Step 3*, the *Central Server* distributes the *Updated Global Model* to the AVs, and the process goes back to *Step 1*. This loop is repeated until some stop condition is reached, such as the maximum number of federation rounds or a predefined performance threshold.

Formally, the deep neural network produced by combining DepthNet and PoseNet can be represented as $\epsilon_{DP}^\theta = \epsilon^\theta = \epsilon$. Thus, the result of standard FedAvg on each round r (ξ^μ) can be defined as:

$$\xi^\mu = \sum_u^{\mathbb{U}} \frac{m_u}{m} \epsilon_u \quad (5.1)$$

$\mathbb{U}$ represents a subset of the total set of AVs $\mathbb{C}$ participating in a round $r \in \mathbb{R}$, where $\mathbb{R}$ is the set of all rounds, m_u is the total number of samples available for AV $u \in \mathbb{U}$, and $m = \sum_c^{\mathbb{C}} m_c$. Moreover, v is the fraction of AVs active in each round r .

Thus, in Algorithm 5 we define ERFedAvg as a function of ξ^μ , the shared *Validation Set* of the current dataset (X_{val}), and the experience replay buffer composed by the *Old Validation Sets* ($\mathcal{ER} = [X_{val}^1, X_{val}^2, \dots, X_{val}^\Psi]$ where $\Psi = |\mathcal{ER}|$).

Algorithm 5 ERFedAvg as a function of ξ^μ , X_{val} , and $\mathcal{ER}$.

```

1: procedure ERFedAvg( $\xi^\mu, X_{val}, \mathcal{ER}$ )
2:    $\xi^* = \xi^\mu$ ;
3:   for each  $X_{val}^i \in \mathcal{ER}$  do
4:      $\xi^* = \text{Self-Supervised Learning of } \xi^* \text{ with } X_{val}^i$ ;
5:     Evaluate  $\xi^*$  with  $X_{val}$ ;
6:   end for
7:   return  $\xi^*$ ;
8: end procedure
```

Based on Equation 5.1 and Algorithms 5, 6, and 7, Algorithm 8 defines how ERFedAvg is used within ERFedSCDepth. Equation 5.1 is used to implement the function *FedAvg*, and Algorithm 5 is used in the final step of each round r . For simplicity, we

¹“gold standard” set of samples collected before the training and shared with all the participants.

have omitted some parameters and implementation details of the function *LocalUpdate*, such as the number of epochs used in local training, as these were further detailed and investigated in FedSCDepth ablation studies [45].

Algorithm 6 Local model update procedure.

```

1: procedure LocalUpdate( $u, \epsilon', X_{val}$ )
2:   Execute self-supervised learning with  $\epsilon'$  on AV  $u$ ;
3:   Evaluate updated  $\epsilon'$  with  $X_{val}$ ;
4:   return updated  $\epsilon'$ ;
5: end procedure

```

Algorithm 7 FedAvg aggregation function procedure.

```

1: procedure FedAvg( $\mathbb{U}, \mathbb{E}_{\mathbb{U}}$ )
2:   Compute and return  $\xi^\mu$  using Equation 5.1;
3: end procedure

```

Algorithm 8 ERFedSCDepth FL procedure.

Input: $\epsilon, \mathbb{C}, v, \mathbb{R}, \mathcal{ER}$;

```

1:  $\epsilon^0 = \epsilon$ ;
2: for each round  $r \in \mathbb{R}$  do
3:    $t = r - 1$ ;  $|\mathbb{U}| = \max(v \times |\mathbb{C}|, 1)$ ;
4:    $\mathbb{U}$  = Random Set of  $|\mathbb{U}|$  AVs that will be active in round  $r$ ;
5:    $\mathbb{E}_{\mathbb{U}}$  = empty vector with  $|\mathbb{U}|$  positions;
6:   for each AV  $u \in \mathbb{U}$  in parallel do
7:      $\epsilon_u^r = \text{LocalUpdate}(u, \epsilon^t, X_{val})$ ;
8:     Save  $\epsilon_u^r$  into  $\mathbb{E}_{\mathbb{U}}[u]$ ;
9:   end for
10:   $\xi^\mu = \text{FedAvg}(\mathbb{U}, \mathbb{E}_{\mathbb{U}})$ ;
11:   $\epsilon = \epsilon^r = \text{ERFedAvg}(\xi^\mu, X_{val}, \mathcal{ER})$ ;
12: end for

```

Output: ϵ ;

5.3.2 Regularized FedSCDepth

RFedSCDepth consists of the combination of the FedAvg aggregation function with a regularization step using the global models obtained with previous datasets, integrated into FedSCDepth's [45] self-supervised FL framework for monocular depth estimation. The combined aggregation function is denoted RFedAvg. Figure 5.2 provides an overview of RFedAvg and its combination with FedSCDepth, highlighting each step of the learning process and the interactions between the *Central Server* and the AVs.

Step 0, *Step 1*, and *Step 3* are the same as in ERFedAvg. The only exception is *Step 2*, in which the *Central Server* regularizes the *Averaged Global Model* weights by averaging them with the *Old Global Models* weights (obtained at the end of the training with each of the previous datasets). This regularization step generates the *Updated Global Model* that is used in *Step 3*.

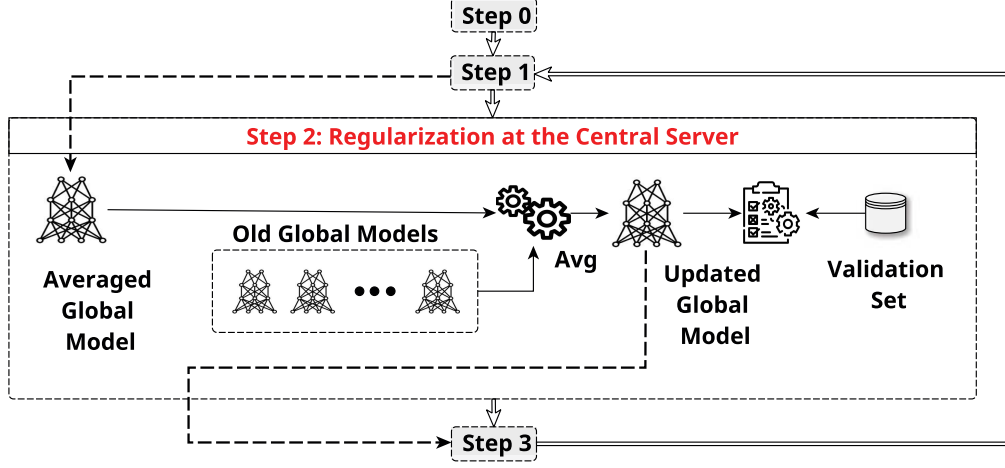


Figure 5.2: RFedSCDepth overview.

Algorithm 9 RFedAvg as a function of ξ^μ and $\mathcal{GM}$.

```

1: procedure RFedAvg( $\xi^\mu, \mathcal{GM}$ )
2:    $\xi^\Sigma = \xi^\mu$ ;
3:   for each model weight  $\xi_{final}^i \in \mathcal{GM}$  do
4:      $\xi^\Sigma = \xi^\Sigma + \xi_{final}^i$ ;
5:   end for
6:    $\xi^* = \frac{\xi^\Sigma}{|\mathcal{GM}|+1}$ ;
7:   return  $\xi^*$ 
8: end procedure

```

In Algorithm 9, we define RFedAvg as a function of ξ^μ (defined in Equation 5.1) and the set of *Old Global Models* denoted as $\mathcal{GM} = [\xi_{final}^1, \xi_{final}^2, \dots, \xi_{final}^\Psi]$ where $\Psi = |\mathcal{GM}|$ and ξ_{final}^1 denotes the final Global Model trained with the first dataset, ξ_{final}^2 the Global Model trained with the second dataset, and ξ_{final}^Ψ is the Global Model Trained with the last dataset (before the current one).

Based on Equation 5.1 and Algorithms 5, 6, and 7, Algorithm 10 defines how RFedAvg is used within RFedSCDepth.

5.3.3 Experience Replay Regularized FedSCDepth

ERRFedSCDepth consists of the combination of the ERFedAvg and RFedAvg aggregation functions, integrated into FedSCDepth’s [45] self-supervised FL framework for monocular depth estimation. The combined aggregation function is denoted ERRFedAvg. Figure 5.3 provides an overview of ERRFedAvg and its combination with FedSCDepth, highlighting each step of the learning process and the interactions between the *Central Server* and the AVs.

Algorithm 10 RFedSCDepth FL procedure.

Input: $\varepsilon, \mathbb{C}, v, \mathbb{R}, \mathcal{GM}$;

- 1: $\varepsilon^0 = \varepsilon$;
- 2: **for** each round $r \in \mathbb{R}$ **do**
- 3: $t = r - 1$; $|\mathbb{U}| = \max(v \times |\mathbb{C}|, 1)$;
- 4: $\mathbb{U} = \text{Random Set of } |\mathbb{U}| \text{ AVs that will be active in round } r$;
- 5: $\mathbb{E}_{\mathbb{U}} = \text{empty vector with } |\mathbb{U}| \text{ positions}$;
- 6: **for** each AV $u \in \mathbb{U}$ **in parallel do**
- 7: $\varepsilon_u^r = \text{LocalUpdate}(u, \varepsilon^t, X_{val})$;
- 8: Save ε_u^r into $\mathbb{E}_{\mathbb{U}}[u]$;
- 9: **end for**
- 10: $\xi^\mu = \text{FedAvg}(\mathbb{U}, \mathbb{E}_{\mathbb{U}})$;
- 11: $\varepsilon = \varepsilon^r = \text{RFedAvg}(\xi^\mu, \mathcal{GM})$;
- 12: **end for**

Output: ε ;

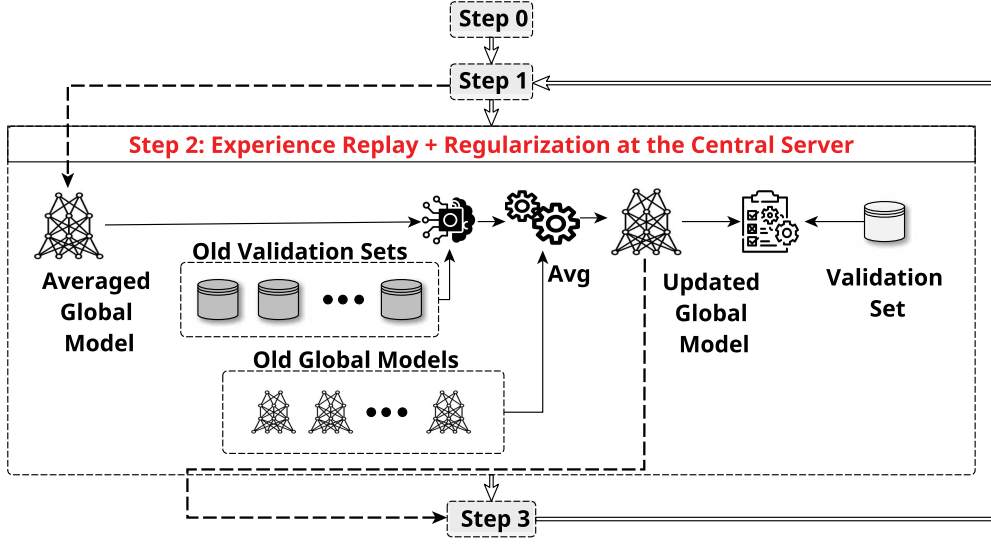


Figure 5.3: ERRFedSCDepth overview.

Step 0, *Step 1*, and *Step 3* are the same as in ERFedAvg. The only exception is *Step 2*, in which the *Central Server* performs self-supervised learning using *Averaged Global Model* weights for initialization and the *Old Validation Sets* (shared in previous datasets) as input data and then regularizes the resulting model weights by averaging them with the *Old Global Models* weights (obtained at the end of the training with each of the previous datasets). This step generates the *Updated Global Model* that is used in *Step 3*.

In Algorithm 11, we define ERFedAvg as a function of ξ^μ , the shared *Validation Set* of the current dataset (X_{val}), the experience replay buffer composed by the *Old Validation Sets* ($\mathcal{ER} = [X_{val}^1, X_{val}^2, \dots, X_{val}^\Psi]$ where $\Psi = |\mathcal{ER}|$), and the set of *Old Global Models* denoted as $\mathcal{GM} = [\xi_{final}^1, \xi_{final}^2, \dots, \xi_{final}^\Psi]$ where $\Psi = |\mathcal{GM}|$.

Based on Equation 5.1 and Algorithms 9, 11, 6, and 7, Algorithm 12 defines how ERFedAvg is used within ERRFedSCDepth.

Algorithm 11 ERRFedAvg as a function of ξ^μ , X_{val} , $\mathcal{ER}$, and $\mathcal{GM}$.

```

1: procedure ERRFedAvg( $\xi^\mu, X_{val}, \mathcal{ER}, \mathcal{GM}$ )
2:    $\xi^{ER} = \text{ERFedAvg}(\xi^\mu, X_{val}, \mathcal{ER})$ ;
3:    $\xi^{ERR} = \text{RFedAvg}(\xi^*, \mathcal{GM})$ ;
4:   return  $\xi^{ERR}$ ;
5: end procedure

```

Algorithm 12 ERRFedSCDepth FL procedure.

Input: $\varepsilon, \mathbb{C}, v, \mathbb{R}, \mathcal{ER}, \mathcal{GM}$;

```

1:  $\varepsilon^0 = \varepsilon$ ;
2: for each round  $r \in \mathbb{R}$  do
3:    $t = r - 1$ ;  $|\mathbb{U}| = \max(v \times |\mathbb{C}|, 1)$ ;
4:    $\mathbb{U} = \text{Random Set of } |\mathbb{U}| \text{ AVs that will be active in round } r$ ;
5:    $\mathbb{E}_{\mathbb{U}} = \text{empty vector with } |\mathbb{U}| \text{ positions}$ ;
6:   for each AV  $u \in \mathbb{U}$  in parallel do
7:      $\varepsilon_u^r = \text{LocalUpdate}(u, \varepsilon^t, X_{val})$ ;
8:     Save  $\varepsilon_u^r$  into  $\mathbb{E}_{\mathbb{U}}[u]$ ;
9:   end for
10:   $\xi^\mu = \text{FedAvg}(\mathbb{U}, \mathbb{E}_{\mathbb{U}})$ ;
11:   $\varepsilon = \varepsilon^r = \text{ERRFedAvg}(\xi^\mu, X_{val}, \mathcal{ER}, \mathcal{GM})$ ;
12: end for
Output:  $\varepsilon$ ;

```

5.4 Evaluation Experiments

To address the research questions of the present study (**RQ5**, **RQ6**, and **RQ7**) we design and execute evaluation experiments detailed in the following subsections.

5.4.1 Datasets and Scenarios

Our evaluations are conducted with the publicly available KITTI [47] and DDAD [48] datasets, with the same settings used in Chapters 3 and 4. We designed experiments comparing naive² continual learning (FedSCDepth) and our proposed methods (ERFedSCDepth, RFedSCDepth, ERRFedSCDepth) with two continual learning scenarios:

- DDAD to KITTI - Retraining the best model learned with DDAD on KITTI, using the same setting that was used in its federated training: 12 AVs in total, 4 AVs active per round, three local training epochs, IID data;
- KITTI to DDAD - Retraining the best model learned with KITTI on DDAD, reusing its federated training setting, like the previous one, but with Non-IID data.

²Naive here indicates that the retraining is performed using the best model trained with the source dataset (e.g., DDAD) as the initialization for the training with the target dataset (e.g., KITTI), without any adjustment to avoid the catastrophic forgetting.

5.4.2 Metrics

To evaluate efficacy, we use standard depth metrics [45]: mean absolute relative error (*AbsRel*), squared relative error (*SqRel*), root mean squared error ($RMS = RMSE$), root mean squared log error ($RMSlog = RMSElog$), and accuracy below threshold ($A_j = \Delta_j, j = 1, 2, 3$), detailed in Chapter 3. The predicted depth maps are scaled by the median, as in [213]. Moreover, to compare different continual learning methods, we analyze the proportional difference of the values obtained in each efficacy metric on the test set of each dataset before and after retraining. Given a metric v and a dataset τ , the proportional difference (PD_v^τ) is computed as:

$$PD_v^\tau = \frac{v_{before}^\tau - v_{after}^\tau}{v_{before}^\tau} \quad (5.2)$$

where v_{before}^τ is the value of v obtained with the test set of τ before retraining and v_{after}^τ is the value after retraining. The proportional difference allows us to compare the efficacy metrics variations on both datasets during continual learning more fairly, as their absolute values might vary between significantly different upper and lower bounds. Additionally, we compute the Stability-Plasticity Trade-Off (SPTO) [24] of each strategy in each scenario. SPTO has been recently proposed in the literature to measure how well the continual learning strategy handles the dilemma of retaining performance on previously seen tasks/datasets while preserving the ability to learn new tasks/datasets [162]. It is formally defined as [24]:

$$SPTO = \frac{2 \times \mathcal{AS} \times \mathcal{AP}}{\mathcal{AS} + \mathcal{AP}} \quad (5.3)$$

where stability ($\mathcal{AS} = \frac{1}{n_t} \sum_{j=1}^{n_t} \mathcal{A}_{m,j}$) is the average performance on all previously seen tasks after training the model on the final task and plasticity ($\mathcal{AP} = \frac{1}{n_t} \sum_{i=1}^{n_t} \mathcal{A}_{i,i}$) is the average performance of the tasks after the model is trained on them for the first time [162]. Also, $\mathcal{A}$ is any of the efficacy metrics (*AbsRel*, *SqRel*, *RMSE*, *RMSELog*, A_1 , A_2 , A_3).

5.4.3 Implementation Details

Our prototype was based on FedSCDepth’s [45] source code, which its authors made publicly available on GitHub. We have shared our implementation³ in the same way. Our deep neural network, as in [45], used the Adam optimizer, with the learning rate set to 10^{-4} . The deep neural network encoder was initialized using ImageNet [229] pre-trained weights. We set the maximum number of batches per epoch as $1k$ and the batch size as

³<https://github.com/eltonfss/federated-sc-depth/tree/clfedscdepth>

4, ensuring a comprehensive approach to our research. The continual learning ran on the same hardware used in the federated training of the source models, detailed in Chapter 3.

5.4.4 Results Analysis

In Figures 5.4 and 5.5, we present the difference proportion obtained with all methods on the error metrics (*AbsRel*, *SqRel*, *RMSE*, *RMSElog*) and the accuracy metrics (*A1*, *A2*, *A3*). Moreover, in Table 5.2 and Figure 5.6, we present the SPTO results obtained with all the methods in the same metrics. Finally, in Tables 5.4 and 5.3, we present the final error and accuracy metrics of the models learned with all the methods. For simplicity, we abbreviate the method names in the legends using the acronyms of Table 5.1.

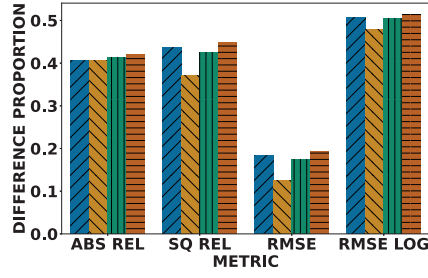
Table 5.1: Abbreviations of method names.

Method Name	Acronym
FedSCDepth	FSCD
ERFedSCDepth	ERFSCD
RFedSCDepth	RFSCD
ERRFedSCDepth	ERRFSCD

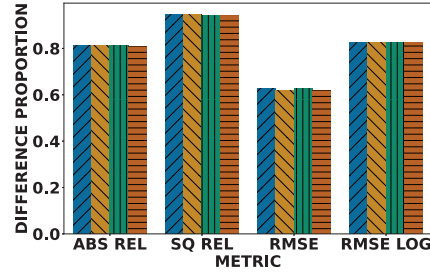
5.4.4.1 Difference Proportions Analysis

Analyzing Figures 5.4 (a) and (b) we observe that when retraining a model initially trained with DDAD in KITTI samples, all evaluated methods present similar results in all metrics, except for metrics *SqRel*, *RMSE*, and *RMSElog* (SQ REL, RMSE, and RMSE LOG on the chart) for which the difference proportions were substantially lower with ERRFSCD while slightly higher with RFSCD, compared to the others tested with DDAD.

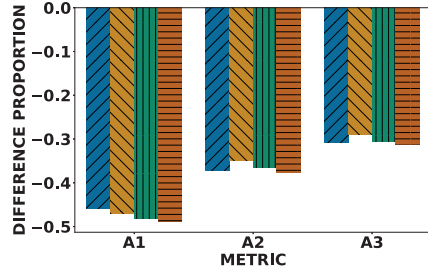
Looking only at this particular result, one could conclude that the regularization component introduced in RFSCD was beneficial, while the experience replay introduced in the ERFSCD component was not. Nonetheless, when we look at the results of the ERFSCD method we observe that its results were also substantially better than those of ERRFSCD, and very close to those of RFSCD. Moreover, another reason for which this result is not yet very conclusive is that our initial expectation was that the difference proportions would be negative on DDAD, since the model initially trained on DDAD would likely forget some particular characteristics of DDAD samples after being retrained on KITTI, which turned out to be false for all methods, including the naive continual learning with FSCD. In contrast, in Figures 5.5 (a) and (b), we can see that when adapting a model from KITTI to DDAD, the experience replay mechanism introduced in ERFSCD allowed it to



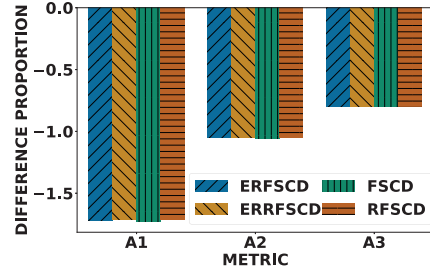
(a) Difference proportions of error metrics obtained on test dataset of DDAD



(b) Difference proportions of error metrics obtained on test dataset of KITTI



(c) Difference proportions of accuracy metrics obtained on DDAD test dataset



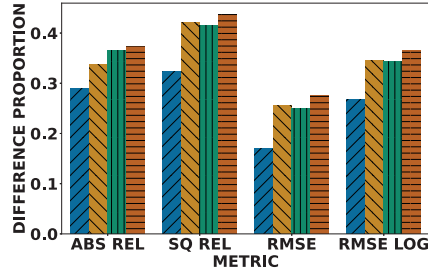
(d) Difference proportions of accuracy metrics obtained on KITTI test dataset

Figure 5.4: Difference proportion results obtained when retraining a model from DDAD on KITTI. According to Equation 5.2, positive values in (a) and (b) indicate a reduction in error metrics, while negative values in (c) and (d) indicate an increase in accuracy metrics.

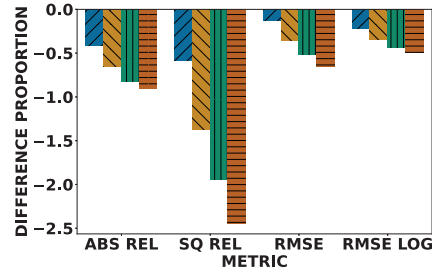
preserve a substantially lower value in most error metrics in KITTI (its difference proportion was closer to zero), while sacrificing a smaller proportion of the improvements obtained in DDAD, when compared to the other methods.

In this particular setting, one can observe the expected effect of the “catastrophic forgetting” problem, since after retraining the model in DDAD, the error metrics presented higher (worse) values in the KITTI test dataset. In that sense, we note that the forgetting was substantially higher with RFSCD, followed by FSCD (which should be the worst). Meanwhile, when looking at the error metrics obtained in DDAD, RFSCD allowed greater improvements, while ERFSCD was the method that limited the learning with the new dataset the most. Therefore, looking only at these results, one could conclude that ERRFSCD presented the greatest balance of reducing the forgetting of KITTI samples while allowing the learning of new samples from DDAD.

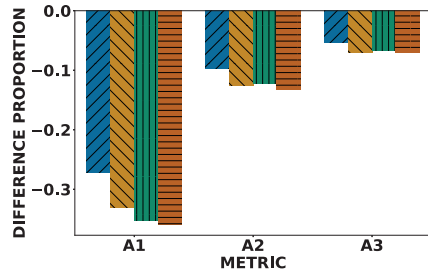
Regarding the difference proportions in the accuracy metrics, shown in Figures 5.4 (c) and (d), we only observe a marginal advantage for RFSCD in the DDAD test and slightly worse overall results for the methods that used Experience Replay (ERFSCD and ERRFSCD), which is in line with what was observed in Figures 5.4 (a) and (b). Furthermore,



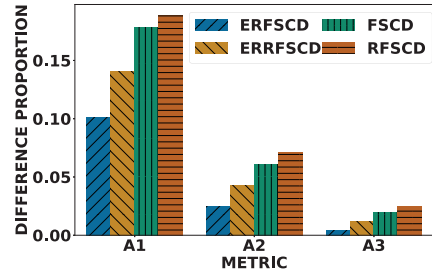
(a) Difference proportions of error metrics obtained on test dataset of DDAD



(b) Difference proportions of error metrics obtained on test dataset of KITTI



(c) Difference proportions of accuracy metrics obtained on DDAD test dataset



(d) Difference proportions of accuracy metrics obtained on KITTI test dataset

Figure 5.5: Difference proportion results obtained when retraining a model from KITTI on DDAD. According to Equation 5.2, positive values in (a) and (b) indicate a reduction in error metrics, while negative values in (c) and (d) indicate an increase in accuracy metrics.

based on Figure 5.4 (d), one could conclude that the accuracy in KITTI was very similar for all methods after retraining in this dataset. However, it is important to note that, for accuracy metrics, lower difference proportion values indicate the better (higher) final accuracy values⁴. Furthermore, in Figures 5.5 (c) and (d), we note that when retraining from KITTI to DDAD, the trends are inversely similar to those observed in Figures 5.5 (a) and (b), although the proportion of the accuracy metrics sacrificed in DDAD is much closer to what was preserved in KITTI than for the error metrics, indicating a worse trade-off.

5.4.4.2 Stability-Plasticity Trade-Off Analysis

When analyzing the radar chart of Figure 5.6 (a) we can observe that when retraining a model trained on DDAD with KITTI, the SPTO of the error metrics is best with RFSCD, while with ERFSCD it is only substantially better than FSCD on *RMSE* and *SQRel* metrics. Similarly to what was observed in the difference proportion results of this same scenario, ERRFSCD has the worst results for the error metrics.

⁴An increase in an accuracy metric will result in a negative difference proportion, while a reduction in an error metric will result in a positive difference proportion, according to Equation 5.2.

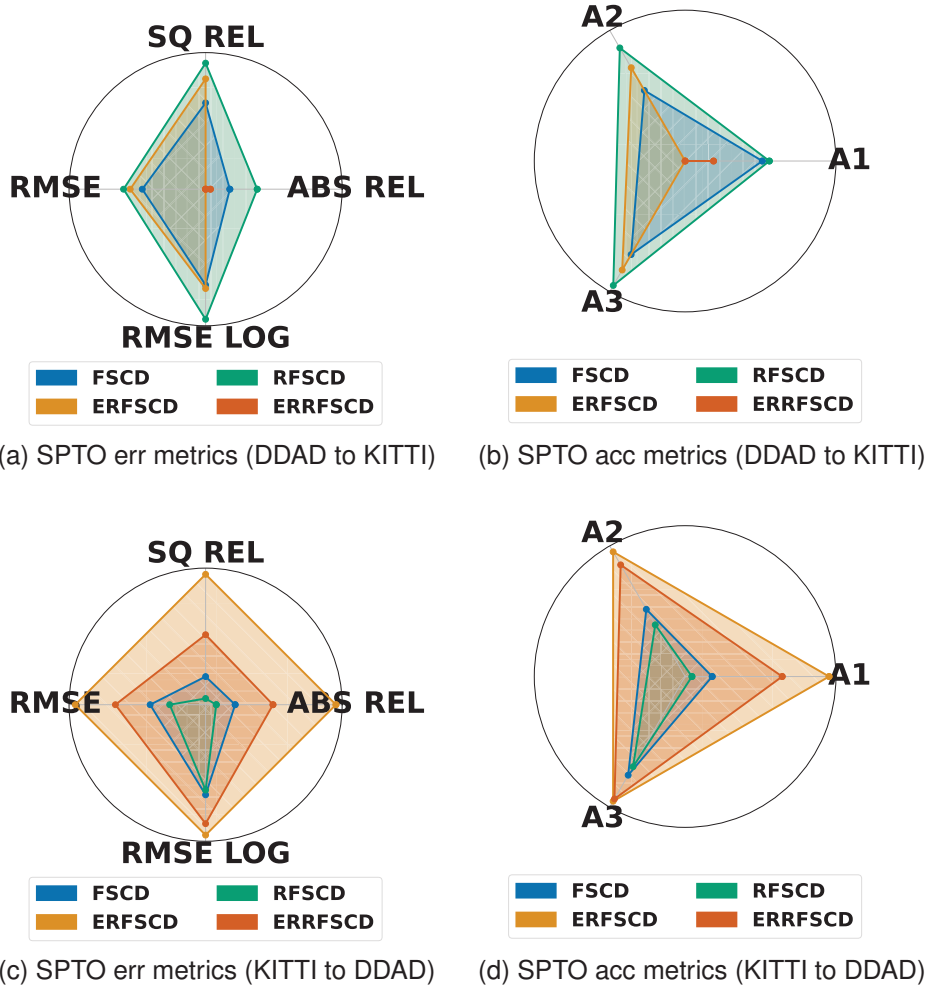


Figure 5.6: Normalized SPTO error (err) metrics (a, c) and accuracy (acc) metrics (b, d) obtained with test dataset, when retraining a model from DDAD to KITTI, and vice versa.

Table 5.2: SPTO of (non-normalized) error and accuracy metrics obtained with test datasets, when retraining a model learned with Dataset 1 (DS1) on Dataset 2 (DS2).

DS1	DS2	Method	<i>AbsRel</i>	<i>SqRel</i>	<i>RMS</i>	<i>RMSlog</i>	A1	A2	A3
DDAD	KITTI	FSCD	0.227	4.866	17.950	0.410	0.648	0.815	0.876
		ERFSCD	0.228	4.812	17.862	0.409	0.643	0.817	0.877
		RFSCD	0.226	4.776	17.814	0.406	0.649	0.818	0.878
		ERRFSCD	0.228	5.064	18.413	0.422	0.645	0.810	0.871
KITTI	DDAD	FSCD	0.169	2.166	8.661	0.247	0.759	0.918	0.967
		ERFSCD	0.158	1.596	7.326	0.235	0.785	0.930	0.972
		RFSCD	0.171	2.288	9.010	0.248	0.755	0.915	0.966
		ERRFSCD	0.165	1.933	8.042	0.238	0.774	0.928	0.971

In Figure 5.6 (b), we observe a similar ranking of the results concerning the accuracy metrics. Meanwhile, in Figures 5.6 (c) and (d) we observe an overall dominance of ERFSCD when retraining a model in KITTI with DDAD, while the worst results are those obtained with RFSCD and the second-best results are those obtained with ERRFSCD. These results are also in line with the analysis of difference proportions, and the insights obtained from Figure 5.6 are reinforced by analyzing the raw (non-normalized) SPTO metrics obtained for each scenario, with each method, presented in Table 5.2.

5.4.4.3 Final Efficacy Metrics Analysis

In Table 5.3, we highlight that in the DDAD to KITTI retraining scenario, ERFSCD obtained the best result in all metrics except *SqRel* for KITTI, that was better with ER-RFSCD. In addition, all methods were able to achieve the highest value of *A3* on KITTI. Also, when testing on the DDAD dataset, RFSCD obtained the best results across all metrics followed by FSCD on *AbsRel* and *A1*, ERFSCD on *SqRel*, *RMS*, *RMSlog*, *A2*, and *A3*. ERRFSCD obtained the worst results overall in this scenario.

Table 5.3: Efficacy of models trained with DDAD and retrained on KITTI with all methods, evaluating on Test Dataset (TD). *Best of results of each scenario highlighted in bold.

TD	Method	<i>AbsRel</i>	<i>SqRel</i>	<i>RMS</i>	<i>RMSlog</i>	<i>A1</i>	<i>A2</i>	<i>A3</i>
KITTI	FSCD	0.119 ± 0.002	0.794 ± 0.029	4.607 ± 0.081	0.186 ± 0.003	0.865 ± 0.004	0.963 ± 0.002	0.985 ± 0.001
	ERFSCD	0.117 ± 0.002	0.761 ± 0.026	4.600 ± 0.077	0.185 ± 0.003	0.869 ± 0.004	0.963 ± 0.002	0.985 ± 0.001
	RFSCD	0.121 ± 0.002	0.808 ± 0.027	4.698 ± 0.083	0.187 ± 0.003	0.861 ± 0.004	0.962 ± 0.002	0.985 ± 0.001
	ERRFSCD	0.118 ± 0.002	0.738 ± 0.022	4.728 ± 0.075	0.187 ± 0.003	0.861 ± 0.004	0.962 ± 0.002	0.985 ± 0.001
DDAD	FSCD	0.214 ± 0.001	4.996 ± 0.052	21.585 ± 0.155	0.367 ± 0.002	0.646 ± 0.003	0.842 ± 0.001	0.913 ± 0.001
	ERFSCD	0.217 ± 0.001	4.999 ± 0.055	21.482 ± 0.155	0.370 ± 0.002	0.638 ± 0.003	0.842 ± 0.001	0.912 ± 0.001
	RFSCD	0.211 ± 0.001	4.801 ± 0.048	21.108 ± 0.147	0.360 ± 0.002	0.649 ± 0.003	0.849 ± 0.001	0.917 ± 0.001
	ERRFSCD	0.216 ± 0.001	5.467 ± 0.053	22.839 ± 0.149	0.386 ± 0.002	0.641 ± 0.002	0.832 ± 0.001	0.901 ± 0.001

Table 5.4: Efficacy of models trained with KITTI and retrained on DDAD with all methods, evaluating on Test Dataset (TD). *Best of results of each scenario highlighted in bold.

TD	Method	<i>AbsRel</i>	<i>SqRel</i>	<i>RMS</i>	<i>RMSlog</i>	<i>A1</i>	<i>A2</i>	<i>A3</i>
KITTI	FSCD	0.219 ± 0.004	2.239 ± 0.070	7.099 ± 0.105	0.268 ± 0.004	0.706 ± 0.006	0.903 ± 0.003	0.966 ± 0.002
	ERFSCD	0.170 ± 0.003	1.204 ± 0.039	5.266 ± 0.073	0.226 ± 0.003	0.772 ± 0.006	0.937 ± 0.003	0.981 ± 0.001
	RFSCD	0.229 ± 0.004	2.623 ± 0.089	7.727 ± 0.112	0.278 ± 0.003	0.697 ± 0.006	0.893 ± 0.003	0.961 ± 0.002
	ERRFSCD	0.198 ± 0.003	1.807 ± 0.059	6.329 ± 0.091	0.249 ± 0.003	0.738 ± 0.006	0.920 ± 0.003	0.973 ± 0.001
DDAD	FSCD	0.156 ± 0.001	3.433 ± 0.057	17.543 ± 0.148	0.272 ± 0.002	0.783 ± 0.002	0.907 ± 0.001	0.952 ± 0.001
	ERFSCD	0.175 ± 0.001	3.970 ± 0.053	19.397 ± 0.157	0.303 ± 0.002	0.736 ± 0.002	0.887 ± 0.001	0.941 ± 0.001
	RFSCD	0.154 ± 0.001	3.296 ± 0.065	16.938 ± 0.143	0.263 ± 0.002	0.787 ± 0.003	0.915 ± 0.001	0.956 ± 0.001
	ERRFSCD	0.163 ± 0.001	3.392 ± 0.054	17.386 ± 0.143	0.271 ± 0.002	0.770 ± 0.002	0.910 ± 0.001	0.955 ± 0.001

In the KITTI-DDAD retraining scenario, depicted in Table 5.4, ERFSCD was also able

to achieve the best results across all error and accuracy metrics on KITTI, while RFSCD also reached the best results on KITTI. Moreover, the results presented in Tables 5.3 and 5.4 correspond to the 95% confidence interval of each metric’s mean computed with 1k bootstrap [242] samples.

5.4.5 Combination with Bayesian Optimization

To provide additional insights regarding the interaction of the aggregation functions in this chapter and the BOFedAvg aggregation function proposed in Chapter 4, we conducted additional experiments combining the BOFedSCDepth method with the ERFedSCDepth, RFedSCDepth and ERRFedSCDepth methods. Thus, in this section, we present the results obtained with the BOFedSCDepth, BOERFedSCDepth, BORFedSCDepth, and BOERRFedSCDepth that resulted from these combinations, on the continual learning scenarios analyzed in the previous section.

In Figures 5.7 and 5.8, we present the difference proportions obtained with all methods on the error metrics (*AbsRel*, *SqRel*, *RMSE*, *RMSElog*) and the accuracy metrics (*A1*, *A2*, *A3*). Moreover, in Table 5.6 and Figure 5.9, we present the SPTO results obtained with all the methods in the same metrics. Finally, in Tables 5.8 and 5.7, we present the final error and accuracy metrics of the models learned with all the methods. For simplicity, we also abbreviate the method names in the legends using the acronyms of Table 5.5.

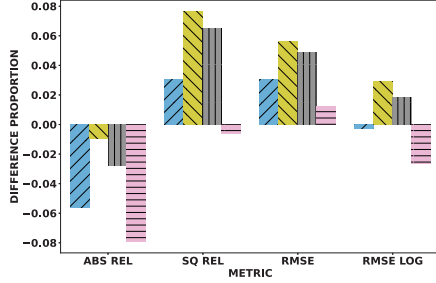
Table 5.5: Abbreviations of method names (with BO).

Method Name	Acronym
BOFedSCDepth	BOFSCD
BOERFedSCDepth	BOERFSCD
BORFedSCDepth	BORFSCD
BOERRFedSCDepth	BOERRFSCD

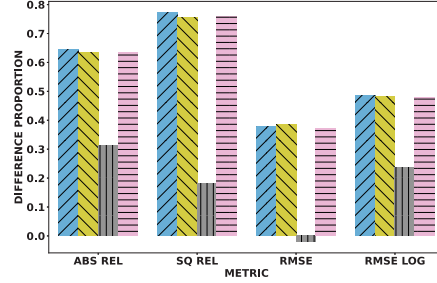
5.4.5.1 Difference Proportions Analysis

Analyzing Figures 5.7 (a) and (b) we observe that when retraining a model initially trained with DDAD on samples from KITTI, the methods that leveraged Experience Replay (ERBOFSCD and ERRBOFSCD) presented the most stable overall results, with modest improvements of error metrics when testing with KITTI and very slight improvement/worsening of those metrics when testing with DDAD. Meanwhile, the naive continual learning method (BOFSCD) and the method based solely on regularization (BORFSCD) presented the highest performance loss with DDAD, although its scale was much

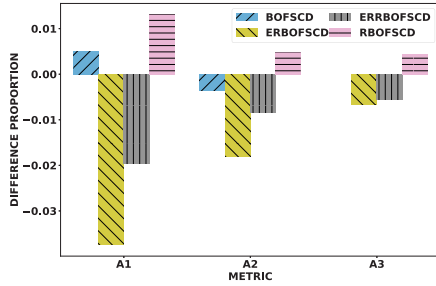
lower than the performance gain in KITTI, which was higher overall with the naive approach, as expected. Moreover, we can draw similar conclusions, if we consider the accuracy metrics presented in Figures 5.7 (c) and (d).



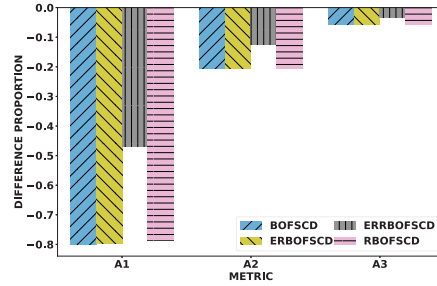
(a) Error metrics testing on DDAD



(b) Error metrics testing on KITTI



(c) Accuracy metrics testing on DDAD

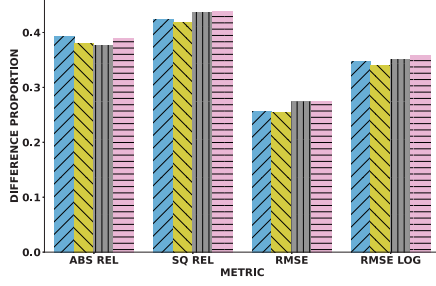


(d) Accuracy metrics testing on KITTI

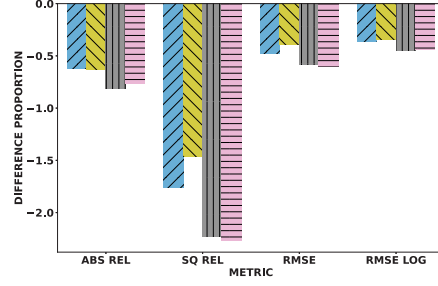
Figure 5.7: Difference proportion results when retraining a model from DDAD on KITTI.

In contrast, analyzing Figures 5.8 (a) and (b), we observe that, when retraining a model initially trained with KITTI on DDAD samples, the methods that leveraged regularization were the most unstable, obtaining the greatest improvements in the error metrics on DDAD, but also the greatest worsening of those same metrics on KITTI. Meanwhile, naive continual learning and the use of experience replay alone showed more stable results, although the difference proportions obtained were not very different from the ones obtained with the other methods overall. Also, analyzing Figures 5.8 (c) and (d), we can draw similar conclusions about accuracy metrics.

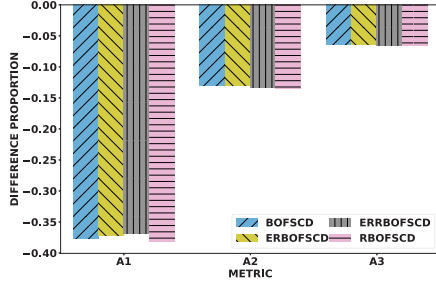
Finally, contrasting those results with those observed in Section 5.4.4.1 we can conclude that, regarding the difference proportions of the error and accuracy metrics, the methods based on regularization performed worse when adding the BO component to the aggregation function. Meanwhile, the use of Experience Replay alone provided better overall results when combined with this same component. Conversely, naive continual learning achieved better absolute values without BO, but was better in comparison with the other methods when using it.



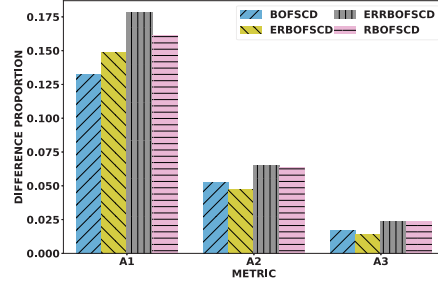
(a) Error metrics testing on DDAD



(b) Error metrics testing on KITTI



(c) Accuracy metrics testing on DDAD



(d) Accuracy metrics testing on KITTI

Figure 5.8: Difference proportion results when retraining a model from KITTI on DDAD.

5.4.5.2 Stability-Plasticity Trade-Off Analysis

Table 5.6: SPTO of (non-normalized) error and accuracy metrics obtained with test datasets, when retraining a model learned with Dataset 1 (DS1) on Dataset 2 (DS2).

DS1	DS2	Method	<i>AbsRel</i>	<i>SqRel</i>	<i>RMS</i>	<i>RMSlog</i>	A1	A2	A3
DDAD	KITTI	BOFSCD	0.191	3.865	16.900	0.325	0.671	0.861	0.925
		ERBOFSCD	0.188	3.817	16.714	0.321	0.686	0.868	0.928
		RBOFSCD	0.194	3.936	17.044	0.329	0.667	0.857	0.923
		ERRBOFSCD	0.219	4.452	17.851	0.349	0.644	0.848	0.922
KITTI	DDAD	BOFSCD	0.164	2.083	8.532	0.242	0.775	0.922	0.969
		ERBOFSCD	0.165	1.976	8.231	0.241	0.767	0.925	0.971
		RBOFSCD	0.170	2.229	8.900	0.246	0.762	0.918	0.967
		ERRBOFSCD	0.173	2.220	8.843	0.248	0.752	0.917	0.967

When analyzing the radar charts of Figures 5.9 (a) and (b), we can observe that, when retraining a model trained in DDAD with KITTI, the best SPTO of all error and accuracy metrics is reached with ERBOFSCD, with BOFSCD being the second best in all metrics. The worst results were obtained with ERRBOFSCD. Meanwhile, analyzing Figures 5.9

(c) and (d), we can observe that, when retraining a model trained on KITTI with DDAD, ERBOFSCD achieved the best SPTO in most of the error and accuracy metrics, except for *AbsRel* (ABS REL) and A1 (A1).

These results are in line with what was observed in the difference proportion results and, contrasting with the SPTO results observed in Section 5.4.4.2, we can also conclude that ERBOFSCD was the method that achieved the greatest synergy with the BO component. This is also confirmed by analyzing Table 5.6, in which we observe that ERBOFSCD achieved the best SPTO results on most of the error and accuracy metrics, on all the scenarios evaluated.

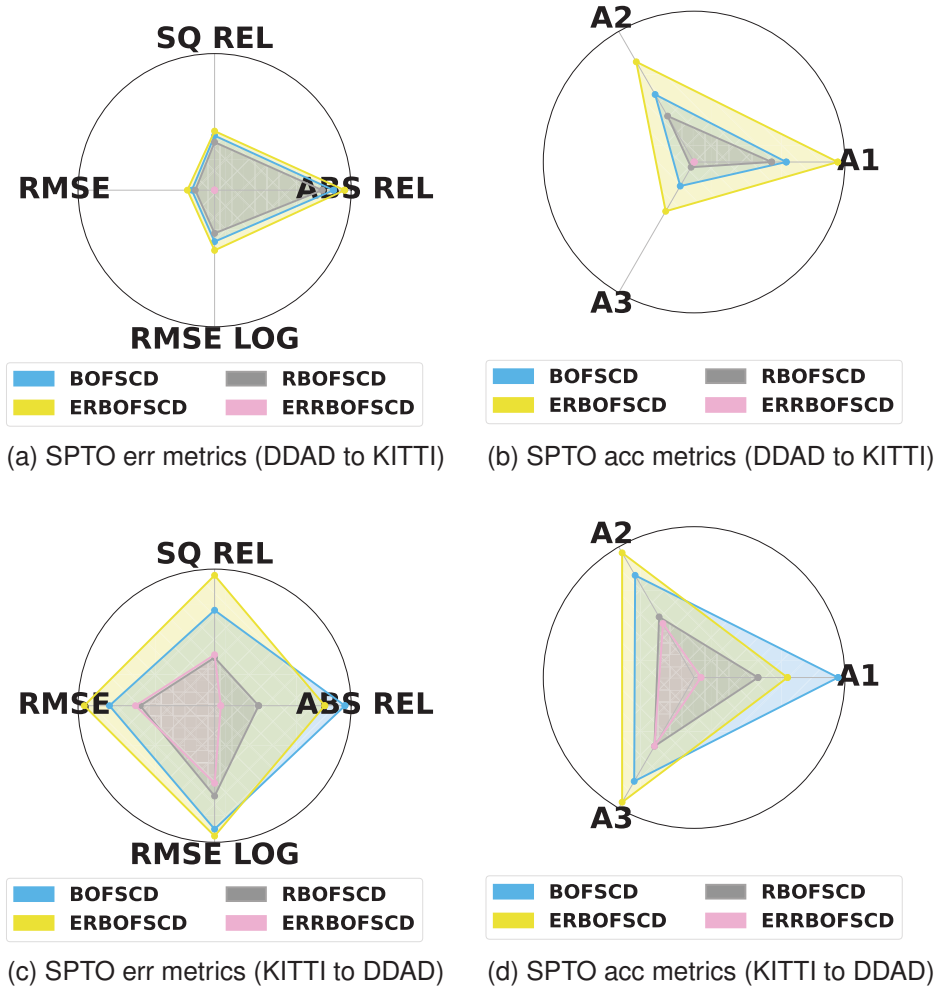


Figure 5.9: Normalized SPTO error (err) metrics (a, c) and accuracy (acc) metrics (b, d) obtained with test dataset, when retraining a model from DDAD to KITTI, and vice-versa.

5.4.5.3 Final Efficacy Metrics Analysis

In Table 5.7, we highlight that in the DDAD to KITTI scenario, BOFSCD obtained the best final error and accuracy metrics in KITTI, while ERBOFSCD achieved the best

results on DDAD. Meanwhile, on the KITTI to DDAD scenario (Table 5.8) BOFSCD obtained the best final *AbsRel* on both datasets, but while on KITTI most of the other metrics were better with ERBOFSCD, on DDAD most of the other metrics were better with RBOFSCD.

Finally, contrasting those results with those observed in Section 5.4.4.2, we can also conclude that RBOFSCD was able to achieve the best results in several metrics, in one of the test datasets, this regularization-based method was the least synergistic with the BO component, with its results being worse with it than without it, in most scenarios. Moreover, the results presented in Tables 5.7 and 5.8 correspond to the 95% confidence interval of each metric’s mean computed with 1k bootstrap [242] samples.

Table 5.7: Efficacy of models trained with DDAD and retrained on KITTI with all methods, evaluating on Test Dataset (TD). *Best of results of each scenario highlighted in bold.

TD	Method	<i>AbsRel</i>	<i>SqRel</i>	<i>RMS</i>	<i>RMSlog</i>	A1	A2	A3
KITTI	BOFSCD	0.114 ± 0.002	0.708 ± 0.022	4.636 ± 0.080	0.184 ± 0.003	0.871 ± 0.004	0.963 ± 0.002	0.985 ± 0.001
	ERBOFSCD	0.119 ± 0.002	0.767 ± 0.026	4.632 ± 0.082	0.187 ± 0.003	0.863 ± 0.004	0.962 ± 0.002	0.985 ± 0.001
	RBOFSCD	0.117 ± 0.002	0.751 ± 0.024	4.697 ± 0.078	0.186 ± 0.003	0.865 ± 0.004	0.961 ± 0.002	0.984 ± 0.001
	ERRBOFSCD	0.119 ± 0.002	0.761 ± 0.026	4.626 ± 0.079	0.186 ± 0.003	0.866 ± 0.004	0.961 ± 0.002	0.984 ± 0.001
DDAD	BOFSCD	0.227 ± 0.001	5.246 ± 0.050	22.061 ± 0.150	0.382 ± 0.002	0.612 ± 0.003	0.830 ± 0.001	0.906 ± 0.001
	ERBOFSCD	0.216 ± 0.001	4.896 ± 0.048	21.308 ± 0.139	0.366 ± 0.002	0.636 ± 0.002	0.846 ± 0.001	0.915 ± 0.001
	RBOFSCD	0.232 ± 0.001	5.446 ± 0.053	22.477 ± 0.160	0.391 ± 0.002	0.607 ± 0.002	0.823 ± 0.001	0.902 ± 0.001
	ERRBOFSCD	0.221 ± 0.001	5.061 ± 0.050	21.645 ± 0.150	0.374 ± 0.002	0.627 ± 0.003	0.834 ± 0.001	0.911 ± 0.001

Table 5.8: Efficacy of models trained with KITTI and retrained on DDAD with all methods, evaluating on Test Dataset (TD). *Best of results of each scenario highlighted in bold.

TD	Method	<i>AbsRel</i>	<i>SqRel</i>	<i>RMS</i>	<i>RMSlog</i>	A1	A2	A3
KITTI	BOFSCD	0.203 ± 0.003	2.141 ± 0.074	7.010 ± 0.105	0.259 ± 0.003	0.736 ± 0.005	0.909 ± 0.003	0.968 ± 0.002
	ERBOFSCD	0.204 ± 0.003	1.912 ± 0.058	6.602 ± 0.093	0.255 ± 0.004	0.722 ± 0.006	0.914 ± 0.003	0.971 ± 0.002
	RBOFSCD	0.221 ± 0.004	2.537 ± 0.083	7.622 ± 0.112	0.273 ± 0.004	0.711 ± 0.006	0.898 ± 0.003	0.962 ± 0.002
	ERRBOFSCD	0.226 ± 0.004	2.509 ± 0.085	7.541 ± 0.119	0.275 ± 0.004	0.697 ± 0.006	0.897 ± 0.003	0.962 ± 0.002
DDAD	BOFSCD	0.150 ± 0.001	3.279 ± 0.069	17.040 ± 0.146	0.264 ± 0.002	0.790 ± 0.002	0.913 ± 0.001	0.956 ± 0.001
	ERBOFSCD	0.153 ± 0.001	3.310 ± 0.070	17.095 ± 0.139	0.267 ± 0.002	0.787 ± 0.002	0.913 ± 0.001	0.956 ± 0.001
	RBOFSCD	0.151 ± 0.001	3.197 ± 0.070	16.627 ± 0.143	0.259 ± 0.002	0.793 ± 0.002	0.917 ± 0.001	0.958 ± 0.001
	ERRBOFSCD	0.154 ± 0.001	3.204 ± 0.067	16.621 ± 0.132	0.262 ± 0.002	0.786 ± 0.002	0.916 ± 0.001	0.958 ± 0.001

5.5 Discussion

Regarding **RQ5**, the combination of experience replay with FL has been shown to provide a better trade-off of stability and plasticity of the learning, compared to a naive continual (federated) learning strategy, especially when retraining a model initially trained

in KITTI with DDAD. However, if we only consider the final efficacy metrics obtained, the addition of experience replay alone did not improve the final test results substantially.

Concerning **RQ6**, the combination of the proposed regularization strategy with FL provided not only better difference proportions and overall SPTO results, when adapting a model from DDAD to KITTI, but also the best final efficacy metrics in most scenarios, indicating that this combination is very likely to improve the efficacy of continual self-supervised FL for monocular depth estimation, within the AV domain.

Finally, concerning **RQ7**, the combination of experience replay, regularization, and FL achieved competitive difference proportions and SPTO in most scenarios, with ER-RFSCD achieving better final error metrics than RFSCD in one of the testing scenarios. However, since the simple combination of regularization and FL performed better in most metrics and scenarios, while the addition of experience replay caused it to perform worse than naive continual learning (FSCD) in several settings, we conclude that the combination of these three components does not substantially improve the efficacy of the solution.

Furthermore, the experiments that combine the BO component proposed in Chapter 4 indicated that our conclusion about **RQ5** and **RQ7** is maintained if we consider BOF-SCD as our baseline, instead of FSCD. However, the conclusion of **RQ6** is changed, as the regularization-based method was unable to achieve a better overall performance, compared to the baseline.

5.6 Conclusion

This chapter addresses continual learning of monocular depth for AVs using FL, regularization, and experience replay. Our approach combines these three techniques with self-supervised learning using unlabeled vehicle data to train a depth estimator efficiently, with high efficacy and preservation of privacy. Prototype testing on KITTI and DDAD datasets shows that models trained with our method achieve state-of-the-art performance. Through evaluation experiments, we show that variations of the proposed method help reduce catastrophic forgetting while incurring zero per-vehicle computational and communication costs overhead compared to naive continual FL, considering both IID and non-IID data. Future research will explore refining our experience replay and regularization mechanisms, extending our assessment with additional datasets, and combining alternative strategies to enhance continual FL.

6. Conclusions and Future Work

In this chapter, the main conclusions of this work are summarized, as well as the next steps to be taken for future research. The central focus of this study has been on self-supervised monocular depth estimation for AVs, addressing the challenge of achieving state-of-the-art efficacy with increased learning efficiency, with regard to communication and computational costs, while improving privacy preservation, resilience to connectivity losses, and robustness to continual learning scenarios. Thus, this work introduced three incremental methodologies to tackle these challenges, each addressing specific facets of the problem:

- Privacy-Preserving Training of Monocular Depth Estimators for Computer Vision in Autonomous Vehicles via Self-Supervised Federated Learning - Chapter 3;
- Enhancing Federated Averaging of Self-Supervised Monocular Depth Estimators for Autonomous Vehicles with Bayesian Optimization - Chapter 4;
- Continual Federated Learning of Monocular Depth for Autonomous Vehicles - Chapter 5.

6.1 Key Contributions and Results

Chapter 3 pioneers a privacy-preserving method for self-supervised monocular depth estimation using FL. The method allows for collaborative training of depth estimators with unlabeled data from multiple vehicles, while preserving privacy and minimizing computational overhead. The results of the KITTI dataset reveal the following:

- Federated training is more efficient than centralized, since centralization incurs communication costs upfront, and federation distributes these costs across multiple rounds, requiring about 65.1% to 87.4% less data transfer per round (**RQ1**);

- Monocular depth estimation models learned with federated training can outperform those learned with centralized training (**RQ2**).

Building on the first method, Chapter 4 integrates BO to enhance the effectiveness of federated averaging for monocular depth estimation. The results with KITTI and DDAD datasets suggest that:

- The combination of BO with FL is more efficient than the use of standard FL for self-supervised learning of monocular depth estimators for AVs, achieving better efficacy with fewer training steps and communication cost (**RQ3**);
- The final models trained with the use of BO during the aggregation of the local models outperform those trained with standard FedAvg aggregation (**RQ4**).

Chapter 5 introduces the final methodological advance, using regularization and experience replay to address catastrophic forgetting in continual FL. By combining the methods proposed in Chapters 3 and 4 with those new components, the results with KITTI and DDAD datasets indicate that:

- The combination of experience replay with FL provides a better tradeoff of stability and plasticity of the learning when compared to naive continual FL strategy (**RQ5**), synergizing well with BO;
- The combination of regularization with FL provided better tradeoff of stability and plasticity results, and the best final efficacy metrics on most of the scenarios (**RQ6**), but did not achieve good synergy with BO;
- The combination of experience replay, regularization, and FL achieved a competitive tradeoff of stability and plasticity in most of the scenarios. However, the combination of these three components does not substantially improve the efficacy of the solution, compared to the use of only one of them (**RQ7**).

In summary, the results presented in Chapters 3, 4, and 5 suggest that the ideal approach for privacy-preserving, efficient, and effective, self-supervised monocular depth estimation in AVs involves:

- FL for decentralized training while minimizing communication costs;
- Bayesian Optimization (BO) with FL to enhance learning efficiency and model performance;

- Continual FL with Experience Replay or Regularization to balance stability and plasticity, mitigating catastrophic forgetting.

6.2 Future Research Directions

This research addresses critical challenges in self-supervised monocular depth estimation, including privacy preservation, non-IID data handling, and efficient model updates. However, several avenues remain for future exploration.

6.2.1 Privacy-Preservation Gaps in Federated Learning

Despite FL’s natural advantage in privacy preservation compared to (traditional) centralized machine learning approaches [25], recent work has shown multiple ways through which the privacy of the data can be breached in a federated training [30].

In addition to the fact that multiple works have tried to combine traditional privacy preservation techniques in the FL framework [31], none has evaluated the suitability of these solutions to the problem of self-supervised monocular depth estimation in AVs.

6.2.2 Efficiency & Efficacy Gaps in Federated Learning

Regardless of the advances and contributions made in the present work, there is still room for improvements in efficiency and efficacy. In particular, adaptations of the proposed methods with transformer-based monocular depth estimation methods [134, 215, 247] present a promising avenue for investigation for future research. Moreover, federated decentralized learning strategies [119, 120] can provide even greater improvements in the efficiency and robustness of training. Novel metrics and simulation strategies to allow the evaluation of large-scale FL scenarios, with lower computational cost than what is currently required, will also be of high value to the research community of this area.

6.2.3 Extended Continual Federated Learning Studies

Continual learning evaluations are typically more expensive and more complex than single dataset evaluations, especially in a federated training setting. In addition, although there is a wide variety of computer vision datasets publicly available, only a subset provides monocular images with corresponding depth maps (at least for the validation set of samples) in a format that is easy to integrate into FL experiments. Moreover, several

datasets that have already been used for monocular depth estimation experiments, such as CityScapes [248], only provide depth annotations in the form of disparity maps, which are more commonly used for stereo (binocular) vision [249] tasks. Thus, new studies are required to implement their preprocessing pipelines to convert disparity maps to depth maps, since existing studies have not publicly shared their preprocessing code in a way that is easily reusable and extensible, making it hard to standardize the evaluations of different studies and compare their results under fair conditions. Therefore, future work related to the present thesis can include the extension of our continual learning evaluations with additional datasets, such as CityScapes [248] and DrivingStereo [250], which will require additional preprocessing efforts but could present novel insights on the performance and generalizability of the proposed methods.

6.2.4 Foundation Models for Monocular Depth Estimation

Recent works have started exploring the use of foundation models for zero-shot monocular depth estimation [251, 252] with promising results. Future explorations and evaluations of these models in AVs and ITS could also produce significant contributions to the research community and the industry. In particular, assessments of the reliability, feasibility, and efficiency of the deployment, and eventual fine-tuning, of those models in AVs, possibly leveraging FL and other privacy-preservation strategies, might present novel insights not only for the AV research but also for the AI research overall.

6.2.5 Applications of Federated Learning in Intelligent Transportation Systems

Despite FL’s recent exploration of several problems of ITS and AVs [15, 16], there are many open problems in which FL, and other Privacy Enhancing Technologies [32–38] can be leveraged, to improve privacy preservation and efficiency during model training. For example, a recent survey [253] outlines several applications in which FL could potentially be explored, such as recognition of traffic signs, pedestrian detection and tracking, and vehicle detection and classification.

Bibliography

- [1] A Khamis. Smart mobility: Exploring foundational technologies and wider impacts. apress, 2021.
- [2] Li Zhu, Fei Richard Yu, Yige Wang, Bin Ning, and Tao Tang. Big data analytics in intelligent transportation systems: A survey. *IEEE Transactions on Intelligent Transportation Systems*, 2018.
- [3] Hexuan Hu, Bo Tang, Xuejiao Gong, Wei Wei, and Huihui Wang. Intelligent fault diagnosis of the high-speed train with big data based on deep neural networks. *IEEE Transactions on Industrial Informatics*, 2017.
- [4] Abdulaziz Aldakkelallah and Milan Simic. Autonomous vehicles in intelligent transportation systems. In *HCIS*, pages 185–198. Springer, 2021.
- [5] Yue Ming, Xuyang Meng, Chunxiao Fan, and Hui Yu. Deep learning for monocular depth estimation: A review. *Neurocomputing*, 438:14–33, 2021.
- [6] Junning Zhang, Qunxing Su, Cheng Wang, and Hongqiang Gu. Monocular 3d vehicle detection with multi-instance depth and geometry reasoning for autonomous driving. *Neurocomputing*, 403:182–192, 2020.
- [7] Alexander N Gorban, Evgeny M Mirkes, and Ivan Y Tyukin. How deep should be the depth of convolutional neural networks: a backyard dog case study. *Cognitive Computation*, 12(2):388–397, 2020.
- [8] Fei Liu, Shubo Zhou, Yunlong Wang, Guangqi Hou, Zhenan Sun, and Tieniu Tan. Binocular light-field: Imaging theory and occlusion-robust depth perception application. *IEEE Transactions on Image Processing*, 29:1628–1640, 2019.
- [9] Hamid Laga. A survey on deep learning architectures for image-based depth reconstruction. *arXiv preprint arXiv:1906.06113*, 2019.

- [10] Ang Li, Anning Hu, Wei Xi, Wenxian Yu, and Danping Zou. Stereo-lidar depth estimation with deformable propagation and learned disparity-depth conversion. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2729–2736. IEEE, 2024.
- [11] Hao Ai and Lin Wang. Elite360d: Towards efficient 360 depth estimation via semantic-and distance-aware bi-projection fusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9926–9935, 2024.
- [12] Ning-Hsu Albert Wang and Yu-Lun Liu. Depth anywhere: Enhancing 360 monocular depth estimation via perspective distillation and unlabeled data augmentation. *Advances in Neural Information Processing Systems*, 37:127739–127764, 2024.
- [13] Longlong Jing and Yingli Tian. Self-supervised visual feature learning with deep neural networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.
- [14] Jierui Liu, Zhiqiang Cao, Xilong Liu, Shuo Wang, and Junzhi Yu. Self-supervised monocular depth estimation with geometric prior and pixel-level sensitivity. *IEEE Transactions on Intelligent Vehicles*, 8(3):2244–2256, 2023.
- [15] Dimitrios Michael Manias and Abdallah Shami. Making a case for federated learning in the internet of vehicles and intelligent transportation systems. *IEEE Network*, 35(3):88–94, 2021.
- [16] Zhaoyang Du, Celimuge Wu, Tsutomu Yoshinaga, Kok-Lim Alvin Yau, Yusheng Ji, and Jie Li. Federated learning for vehicular internet of things: Recent advances and open issues. *IEEE Open Journal of the Computer Society*, 1, 2020.
- [17] Stefano Savazzi, Monica Nicoli, Mehdi Bennis, Sanaz Kianoush, and Luca Barbi-eri. Opportunities of federated learning in connected, cooperative, and automated industrial systems. *IEEE Communications Magazine*, 59, 2021.
- [18] Xiaodong Ma, Jia Zhu, Zhihao Lin, Shanxuan Chen, and Yangjie Qin. A state-of-the-art survey on solving non-iid data in federated learning. *Future Generation Computer Systems*, 135:244–258, 2022.
- [19] Elton F. de S. Soares, Carlos Alberto V. Campos, and Sidney C. de Lucena. Online travel mode detection method using automated machine learning and feature engineering. *Future Generation Computer Systems*, 101:1201–1212, 2019.
- [20] Longbing Cao, Hui Chen, Xuhui Fan, Joao Gama, Yew-Soon Ong, and Vipin Kumar. Bayesian federated learning: A survey, 2023.

- [21] Chris Thornton, Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. Auto-weka: Combined selection and hyperparameter optimization of classification algorithms. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '13, pages 847–855, New York, NY, USA, 2013. ACM.
- [22] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 2951–2959. Curran Associates, Inc., 2012.
- [23] Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier, 1989.
- [24] Hemang Chawla, Arnav Varma, Elahe Arani, and Bahram Zonooz. Continual learning of unsupervised monocular depth from videos. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 8419–8429, 2024.
- [25] Nguyen Truong, Kai Sun, Siyao Wang, Florian Guitton, and YiKe Guo. Privacy preservation in federated learning: An insightful survey from the gdpr perspective. *Computers & Security*, 110:102402, 2021.
- [26] General Data Protection Regulation. Regulation eu 2016/679 of the european parliament and of the council of 27 april 2016. *Official Journal of the European Union*, 2016.
- [27] Houda Rafi, Yannick Benezeth, Fan Yang Song, Philippe Reynaud, Emmanuel Arnoux, and Cedric Demonceaux. Tree-based personalized clustered federated learning: A driver stress monitoring through physiological data case study. *IEEE Internet of Things Journal*, 12(2):1688–1698, 2025.
- [28] Lara Rocha Garcia, Edson Aguilera-Fernandes, Rafael Augusto Moreno Gonçalves, and Marcos Ribeiro Pereira-Barretto. *Lei Geral de Proteção de Dados (LGPD): Guia de implantação*. Editora Blucher, 2020.
- [29] Stuart L Pardo. The california consumer privacy act: towards a european-style privacy regime in the united states. *J. Tech. L. & Pol'y*, 23:68, 2018.
- [30] Xuefei Yin, Yanming Zhu, and Jiankun Hu. A comprehensive survey of privacy-preserving federated learning: A taxonomy, review, and future directions. *ACM Computing Surveys (CSUR)*, 54(6):1–36, 2021.

- [31] Jingxue Chen, Hang Yan, Zhiyuan Liu, Min Zhang, Hu Xiong, and Shui Yu. When federated learning meets privacy-preserving computation. *ACM Computing Surveys*, 56(12):1–36, 2024.
- [32] Victoria L Lemieux and John Werner. Protecting privacy in digital records: The potential of privacy-enhancing technologies. *ACM Journal on Computing and Cultural Heritage*, 16(4):1–18, 2024.
- [33] Organisation for Economic Co-operation and Development (OECD). Report on the implementation of the OECD privacy guidelines. Technical Report 361, OECD Publishing, Paris, 2023.
- [34] David W Archer, Borja de Balle Pigem, Dan Bogdanov, Mark Craddock, Adria Gascon, Ronald Jansen, Matjaž Jug, Kim Laine, Robert McLellan, Olga Ohrimenko, et al. Un handbook on privacy-preserving computation techniques. *arXiv preprint arXiv:2301.06167*, 2023.
- [35] Royal Society. From privacy to partnership. *The role of privacy enhancing technologies in data governance and collaborative analysis*, 2023.
- [36] Elif Ustundag Soykan, Leyli Karacay, Ferhat Karakoc, and Emrah Tomur. A survey and guideline on privacy enhancing technologies for collaborative machine learning. *IEEE Access*, 10:97495–97519, 2022.
- [37] Shi-Cho Cha, Tzu-Yang Hsu, Yang Xiang, and Kuo-Hui Yeh. Privacy enhancing technologies in the internet of things: Perspectives and challenges. *IEEE Internet of Things Journal*, 6(2):2159–2187, 2018.
- [38] Nesrine Kaaniche, Maryline Laurent, and Sana Belguith. Privacy enhancing technologies for solving the privacy-personalization paradox: Taxonomy and survey. *Journal of Network and Computer Applications*, 171:102807, 2020.
- [39] Vasileios Mavroeidis, Kameran Vishni, Mateusz D Zych, and Audun Jøsang. The impact of quantum computing on present cryptography. *arXiv preprint arXiv:1804.00200*, 2018.
- [40] Andy Majot and Roman Yampolskiy. Global catastrophic risk and security implications of quantum computers. *Futures*, 72:17–26, 2015.
- [41] Amlaan Bhoi. Monocular depth estimation: A survey, 2019.
- [42] Clément Godard, Oisín Mac Aodha, and Gabriel J Brostow. Unsupervised monocular depth estimation with left-right consistency. In *IEEE/CVF CVPR*, 2017.

- [43] Kenneth C Laudon and Jane P Laudon. *Essentials of management information systems*. Pearson, 2017.
- [44] Lu Zang, Yang Qin, and Ruonan Li. Traffic flow prediction based on federated learning with joint pca compression and bayesian optimization. In *IEEE SMC*, pages 3330–3335, 2022.
- [45] Elton F. de S. Soares and Carlos Alberto V. Campos. Privacy-preserving training of monocular depth estimators via self-supervised federated learning. In *2024 IEEE 100th Vehicular Technology Conference (VTC2024-Fall)*, pages 1–6, 2024.
- [46] Elton F. de S. Soares and Carlos Alberto V. Campos. Federated self-supervised learning of monocular depth estimators for autonomous vehicles. *arXiv preprint arXiv:2310.04837*, 2023.
- [47] Jonas Uhrig, Nick Schneider, Lukas Schneider, Uwe Franke, Thomas Brox, and Andreas Geiger. Sparsity invariant cnns. In *2017 International Conference on 3D Vision (3DV)*, pages 11–20, 2017.
- [48] Vitor Guizilini, Rareş Ambruş, Sudeep Pillai, Allan Raventos, and Adrien Gaidon. 3d packing for self-supervised monocular depth estimation. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2482–2491, 2020.
- [49] L von Bertalanffy. General system theory—a critical review. *General systems*, 7(1):1–20, 1962.
- [50] George A Miller. The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological review*, 63(2):81, 1956.
- [51] Michael J Kearns, Umesh Virkumar Vazirani, and Umesh Vazirani. *An introduction to computational learning theory*. MIT press, 1994.
- [52] Jan Recker. *Scientific research in information systems: a beginner’s guide*. Springer Nature, 2021.
- [53] Auguste Comte and John Henry Bridges. *A general view of positivism*. Routledge, 2015.
- [54] Yoon Soo Park, Lars Konge, and Anthony R Artino Jr. The positivism paradigm of research. *Academic medicine*, 95(5):690–694, 2020.
- [55] Markus Gabriel. *Fields of sense: a new realist ontology*. Edinburgh University Press, 2015.

- [56] Dvora Yanow. Neither rigorous nor objective?: Interrogating criteria for knowledge claims in interpretive science. In *Interpretation and Method*, pages 97–119. Routledge, 2015.
- [57] Stephen Kwadwo Antwi and Kasim Hamza. Qualitative and quantitative research paradigms in business research: A philosophical reflection. *European journal of business and management*, 7(3):217–225, 2015.
- [58] Elton F. de S. Soares, Emilio Vital Brazil, and Carlos Alberto V. Campos. Leveraging bayesian optimization to enhance self-supervised federated learning of monocular depth estimators. *2024 27th IEEE International Conference on Intelligent Transportation Systems (ITSC 2024)*, Edmonton, Canada, Sept 2024.
- [59] Elton F de S Soares and Carlos Alberto V Campos. Towards continual federated learning of monocular depth for autonomous vehicles. In *2024 IEEE 100th Vehicular Technology Conference (VTC2024-Fall)*, pages 1–5. IEEE, 2024.
- [60] Elton F. de S. Soares and Carlos Alberto V. Campos. Self-supervised learning of monocular depth estimators in autonomous vehicles with federated learning. *Engineering Applications of Artificial Intelligence*, 151:110572, 2025.
- [61] Elton F de S Soares, Emilio Vital Brazil, and Carlos Alberto V Campos. Enhancing federated averaging of self-supervised monocular depth estimators for autonomous vehicles with bayesian optimization. *Future Generation Computer Systems*, page 107752, 2025.
- [62] Rudolf Giffinger and Nataša Pichler-Milanović. *Smart cities: Ranking of European medium-sized cities*. Centre of Regional Science, Vienna University of Technology, 2007.
- [63] Clara Benevolo, Renata Paola Dameri, and Beatrice D’Auria. Smart mobility in smart city. In *Empowering Organizations*, pages 13–28. Springer, 2016.
- [64] Agachai Sumalee and Hung Wai Ho. Smarter and more connected: Future intelligent transportation system. *IATSS Research*, 2018.
- [65] Hojat Behrooz and Yeganeh M Hayeri. Machine learning applications in surface transportation systems: A literature review. *Applied Sciences*, 12(18):9156, 2022.
- [66] Jasmin Praful Bharadiya. Artificial intelligence in transportation systems a critical review. *American Journal of Computing and Engineering*, 6(1):35–45, 2023.

- [67] S Abirami, M Pethuraj, M Uthayakumar, and P Chitra. A systematic survey on big data and artificial intelligence algorithms for intelligent transportation system. *Case Studies On Transport Policy*, page 101247, 2024.
- [68] Liping Fu and LR Rilett. Estimation of time-dependent, stochastic route travel times using artificial neural networks. *Transportation planning and technology*, 24(1):25–48, 2000.
- [69] Qing Ye, Wai Yuen Szeto, and Sze Chun Wong. Short-term traffic speed forecasting based on data recorded at irregular intervals. *IEEE Transactions on Intelligent Transportation Systems*, 13(4):1727–1737, 2012.
- [70] Dipti Srinivasan, Saptak Sanyal, and Vishal Sharma. Freeway incident detection using hybrid fuzzy neural network. *IET Intelligent Transport Systems*, 1(4):249–259, 2007.
- [71] Mohammad S Ghanim and Ghassan Abu-Lebdeh. Real-time dynamic transit signal priority optimization for coordinated traffic networks using genetic algorithms and artificial neural networks. *Journal of Intelligent Transportation Systems*, 19(4):327–338, 2015.
- [72] Linsen Chong, Montasir M Abbas, Alejandra Medina Flintsch, and Bryan Higgs. A rule-based neural network approach to model driver naturalistic behavior in traffic. *Transportation Research Part C: Emerging Technologies*, 32:207–223, 2013.
- [73] Yang Zhang and Yuncai Liu. Traffic forecasting using least squares support vector machines. *Transportmetrica*, 5(3):193–213, 2009.
- [74] Yunlong Zhang and Yuanchang Xie. Travel mode choice modeling with support vector machines. *Transportation Research Record*, 2076:141 – 150, 2008.
- [75] Fang Yuan and Ruey Long Cheu. Incident detection using support vector machines. *Transportation Research Part C: Emerging Technologies*, 11(3-4):309–328, 2003.
- [76] Jianli Xiao and Yuncai Liu. Traffic incident detection using multiple-kernel support vector machine. *Transportation research record*, 2324(1):44–52, 2012.
- [77] Jie Sun and Jian Sun. Real-time crash prediction on urban expressways: identification of key variables and a hybrid support vector machine model. *IET intelligent transport systems*, 10(5):331–337, 2016.

- [78] Ni Dong, Helai Huang, and Liang Zheng. Support vector machine in crash prediction at the level of traffic analysis zones: assessing the spatial proximity effects. *Accident Analysis & Prevention*, 82:192–198, 2015.
- [79] Xiaoyu Zhu, Yifei Yuan, Xianbiao Hu, Yi-Chang Chiu, and Yu-Luen Ma. A bayesian network model for contextual versus non-contextual driving behavior assessment. *Transportation research part C: emerging technologies*, 81:172–187, 2017.
- [80] Dawei Li, Tomio Miwa, and Takayuki Morikawa. Modeling time-of-day car use behavior: A bayesian network approach. *Transportation research part D: transport and environment*, 47:54–66, 2016.
- [81] Enrique Castillo, José María Menéndez, and Santos Sánchez-Cambronero. Predicting traffic flow using bayesian networks. *Transportation Research Part B: Methodological*, 42(5):482–509, 2008.
- [82] Zheng Zhu, Bo Peng, Chenfeng Xiong, and Lei Zhang. Short-term traffic flow prediction with linear conditional gaussian bayesian network. *Journal of Advanced Transportation*, 50(6):1111–1123, 2016.
- [83] Moinul Hossain and Yasunori Muromachi. A bayesian network based framework for real-time crash prediction on the basic freeway segments of urban expressways. *Accident Analysis & Prevention*, 45:373–381, 2012.
- [84] Andreas Gregoriades and Kyriacos C Mouskos. Black spots identification through a bayesian networks quantification of accident risk index. *Transportation Research Part C: Emerging Technologies*, 28:28–43, 2013.
- [85] Olga Mindlina. Depth estimation: problem, use cases, solutions and practical experiments using eden — multimodal synthetic dataset of enclosed garden scenes. <https://medium.com/@olga.mindlina>, 2024. Accessed: 22/11/2024.
- [86] Nikolai Smolyanskiy, Alexey Kamenev, and Stan Birchfield. On the importance of stereo for accurate depth estimation: An efficient semi-supervised deep neural network approach. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2018.
- [87] Yi-Min Tsai, Yu-Lin Chang, and Liang-Gee Chen. Block-based vanishing line and vanishing point detection for 3d scene reconstruction. In *2006 international symposium on intelligent signal processing and communications*, pages 586–589. IEEE, 2006.

- [88] Chang Tang, Chunping Hou, and Zhanjie Song. Depth recovery and refinement from a single image using defocus cues. *Journal of Modern Optics*, 62(6):441–448, 2015.
- [89] Ruo Zhang, Ping-Sing Tsai, James Edwin Cryer, and Mubarak Shah. Shape-from-shading: a survey. *IEEE transactions on pattern analysis and machine intelligence*, 21(8):690–706, 1999.
- [90] David G Lowe. Object recognition from local scale-invariant features. In *Proceedings of the seventh IEEE international conference on computer vision*, volume 2, pages 1150–1157. Ieee, 1999.
- [91] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *European conference on computer vision*, pages 404–417. Springer, 2006.
- [92] Anna Bosch, Andrew Zisserman, and Xavier Munoz. Image classification using random forests and ferns. In *2007 IEEE 11th international conference on computer vision*, pages 1–8. Ieee, 2007.
- [93] John Lafferty, Andrew McCallum, and Fernando CN Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. *International Conference on Machine Learning 2001 (ICML 2001)*, 2001.
- [94] George R Cross and Anil K Jain. Markov random field texture models. *IEEE Transactions on pattern analysis and machine intelligence*, pages 25–39, 1983.
- [95] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. *Advances in neural information processing systems*, 27, 2014.
- [96] Fayao Liu, Chunhua Shen, and Guosheng Lin. Deep convolutional neural fields for depth estimation from a single image. In *IEEE/CVF CVPR*, pages 5162–5170, 2015.
- [97] Iro Laina, Christian Rupprecht, Vasileios Belagiannis, Federico Tombari, and Nassir Navab. Deeper depth prediction with fully convolutional residual networks. In *IEEE 3DV*, pages 239–248, 2016.
- [98] Huangying Zhan, Ravi Garg, Chamara Saroj Weerasekera, Kejie Li, Harsh Agarwal, and Ian Reid. Unsupervised learning of monocular depth estimation and visual odometry with deep feature reconstruction. In *IEEE/CVF CVPR*, pages 340–349, 2018.

- [99] Shanshan Zhao, Huan Fu, Mingming Gong, and Dacheng Tao. Geometry-aware symmetric domain adaptation for monocular depth estimation. In *IEEE/CVF CVPR*, June 2019.
- [100] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.
- [101] Bram van Berlo, Aaqib Saeed, and Tanir Ozcelebi. Towards federated unsupervised representation learning. In *EdgeSys*, pages 31–36, 2020.
- [102] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 1597–1607. PMLR, 13–18 Jul 2020.
- [103] Ishan Misra and Laurens van der Maaten. Self-supervised learning of pretext-invariant representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.
- [104] Randall Balestriero, Mark Ibrahim, Vlad Sobal, Ari Morcos, Shashank Shekhar, Tom Goldstein, Florian Bordes, Adrien Bardes, Gregoire Mialon, Yuandong Tian, Avi Schwarzschild, Andrew Gordon Wilson, Jonas Geiping, Quentin Garrido, Pierre Fernandez, Amir Bar, Hamed Pirsiavash, Yann LeCun, and Micah Goldblum. A cookbook of self-supervised learning, 2023.
- [105] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020.
- [106] M Popel, M Tomkova, J Tomek, Ł Kaiser, J Uszkoreit, O Bojar, and Z Žabokrtský. Transforming machine translation: a deep learning system reaches news translation quality comparable to human professionals. *Nature Communications*, 11(1), 2020.
- [107] Priya Goyal, Mathilde Caron, Benjamin Lefaudeux, Min Xu, Pengchao Wang, Vivek Pai, Mannat Singh, Vitaliy Liptchinsky, Ishan Misra, Armand Joulin, and Piotr Bojanowski. Self-supervised pretraining of visual features in the wild, 2021.

- [108] Kristoffer Wickstrøm, Michael Kampffmeyer, Karl Øyvind Mikalsen, and Robert Jenssen. Mixing up contrastive learning: Self-supervised representation learning for time series. *Pattern Recognition Letters*, 155:54–61, March 2022.
- [109] Shuo Liu, Adria Mallol-Ragolta, Emilia Parada-Cabeleiro, Kun Qian, Xin Jing, Alexander Kathan, Bin Hu, and Bjoern W. Schuller. Audio self-supervised learning: A survey, 2022.
- [110] Madeline C. Schiappa, Yogesh S. Rawat, and Mubarak Shah. Self-supervised learning for videos: A survey. *ACM Computing Surveys*, 55(13s):1–37, July 2023.
- [111] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *Overview of Supervised Learning*, pages 9–41. Springer New York, New York, NY, 2009.
- [112] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, Bilal Piot, koray kavukcuoglu, Remi Munos, and Michal Valko. Bootstrap your own latent - a new approach to self-supervised learning. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 21271–21284. Curran Associates, Inc., 2020.
- [113] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16000–16009, June 2022.
- [114] Rayan Krishnan, Pranav Rajpurkar, and Eric J Topol. Self-supervised learning in medicine and healthcare. *Nature biomedical engineering*, 6(12):1346—1352, December 2022.
- [115] Ozan Ciga, Tony Xu, and Anne Louise Martel. Self supervised contrastive learning for digital histopathology. *Machine Learning with Applications*, 7:100198, 2022.
- [116] Dan Hendrycks, Mantas Mazeika, Saurav Kadavath, and Dawn Song. Using self-supervised learning can improve model robustness and uncertainty. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

- [117] Priya Goyal, Quentin Duval, Isaac Seessel, Mathilde Caron, Ishan Misra, Levent Sagun, Armand Joulin, and Piotr Bojanowski. Vision models are more robust and fair when pretrained on uncurated images without supervision, 2022.
- [118] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*. PMLR, 2017.
- [119] Luca Barbieri, Stefano Savazzi, Mattia Brambilla, and Monica Nicoli. Decentralized federated learning for extended sensing in 6g connected vehicles. *Vehicular Communications*, page 100396, 2021.
- [120] Xiaokang Zhou, Wei Liang, Jinhua She, Zheng Yan, I Kevin, and Kai Wang. Two-layer federated learning with heterogeneous model aggregation for 6g supported internet of vehicles. *IEEE Transactions on Vehicular Technology*, 70(6):5308–5317, 2021.
- [121] Weishan Zhang, Qinghua Lu, Qiuyu Yu, Zhaotong Li, Yue Liu, Sin Kit Lo, Shiping Chen, Xiwei Xu, and Liming Zhu. Blockchain-based federated learning for device failure detection in industrial iot. *IEEE Internet of Things Journal*, 2020.
- [122] Chaoyang Zhu, Xiao Zhu, Junyu Ren, and Tuanfa Qin. Blockchain-enabled federated learning for uav edge computing network: Issues and solutions. *Ieee Access*, 10:56591–56610, 2022.
- [123] Weishan Zhang, Zhicheng Bao, Yuru Liu, Liang Xu, Qinghua Lu, Huansheng Ning, Xiao Wang, Su Yang, Fei-Yue Wang, and Zengxiang Li. Cfsl: A credible federated self-learning framework. *IEEE Internet of Things Journal*, 2023.
- [124] Xiangjie Kong, Kailai Wang, Mingliang Hou, Xinyu Hao, Guojian Shen, Xin Chen, and Feng Xia. A federated learning-based license plate recognition scheme for 5g-enabled internet of vehicles. *IEEE Transactions on Industrial Informatics*, 17(12):8523–8530, 2021.
- [125] Xingjie Zeng, Zepei Yu, Weishan Zhang, Xiao Wang, Qinghua Lu, Tao Wang, Mu Gu, Yonglin Tian, and Fei-Yue Wang. Homophily learning-based federated intelligence: A case study on industrial iot equipment failure prediction. *IEEE Internet of Things Journal*, 2022.
- [126] Mykola Servetnyk, Carrson C Fung, and Zhu Han. Unsupervised federated learning for unbalanced data. In *IEEE GLOBECOM*, pages 1–6, 2020.

- [127] Aaqib Saeed, Flora D Salim, Tanir Ozcelebi, and Johan Lukkien. Federated self-supervised learning of multisensor representations for embedded intelligence. *IEEE Internet of Things Journal*, 8(2), 2020.
- [128] Weiming Zhuang, Xin Gan, Yonggang Wen, Shuai Zhang, and Shuai Yi. Collaborative unsupervised visual representation learning from decentralized data. In *IEEE/CVF international conference on computer vision*, 2021.
- [129] Lirui Wang, Kaiqing Zhang, Yunzhu Li, Yonglong Tian, and Russ Tedrake. Does learning from decentralized non-iid unlabeled data benefit from self supervision? In *The Eleventh International Conference on Learning Representations*, 2022.
- [130] Yawen Wu, Zhepeng Wang, Dewen Zeng, Meng Li, Yiyu Shi, and Jingtong Hu. Decentralized unsupervised learning of visual representations, 2022.
- [131] Liangqiong Qu, Yuyin Zhou, Paul Pu Liang, Yingda Xia, Feifei Wang, Ehsan Adeli, Li Fei-Fei, and Daniel Rubin. Rethinking architecture design for tackling data heterogeneity in federated learning. In *IEEE/CVF CVPR*, pages 10061–10071, June 2022.
- [132] Simou Li, Yuxing Mao, Jian Li, Yihang Xu, Jinsen Li, Xueshuo Chen, Siyang Liu, and Xianping Zhao. Fedutn: federated self-supervised learning with updating target network. *Applied Intelligence*, pages 1–14, 2022.
- [133] Xutong Mu, Yulong Shen, Ke Cheng, Xueli Geng, Jiaxuan Fu, Tao Zhang, and Zhiwei Zhang. Fedproc: Prototypical contrastive federated learning on non-iid data. *Future Generation Computer Systems*, 143:93–104, 2023.
- [134] Sangjoon Park, Ik-Jae Lee, Jun Won Kim, and Jong Chul Ye. Single-round self-supervised distributed learning using vision transformer, 2023.
- [135] Nan Yang, Dong Yuan, Charles Z Liu, Yongkun Deng, and Wei Bao. Fedil: Federated incremental learning from decentralized unlabeled data with convergence analysis, 2023.
- [136] Nan Yang, Xuanyu Chen, Charles Z. Liu, Dong Yuan, Wei Bao, and Lizhen Cui. Fedmae: Federated self-supervised learning with one-block masked auto-encoder, 2023.
- [137] Chen Zhao, Zhipeng Gao, Yang Yang, Qian Wang, Zijia Mo, and Xinlei Yu. Fedusc: Collaborative unsupervised representation learning from decentralized data for internet of things. *IEEE Internet of Things Journal*, 10(15):13601–13611, 2023.

- [138] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P Adams, and Nando De Freitas. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2015.
- [139] Philippe Laborie, Jérôme Rogerie, Paul Shaw, and Petr Vilím. Ibm ilog cp optimizer for scheduling. *Constraints*, 23(2):210–250, April 2018.
- [140] Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In Carlos A. Coello Coello, editor, *Learning and Intelligent Optimization*, pages 507–523, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.
- [141] Martin Pelikan, David E. Goldberg, and Erick Cantú-Paz. Boa: the bayesian optimization algorithm. In *GECCO*, page 525–532, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.
- [142] Martin Pelikan. *Bayesian optimization algorithm: From single level to hierarchy*. University of Illinois at Urbana-Champaign, 2002.
- [143] Zhongxiang Dai, Bryan Kian Hsiang Low, and Patrick Jaillet. Federated bayesian optimization via thompson sampling. *Advances in Neural Information Processing Systems*, 33:9687–9699, 2020.
- [144] Daniel J Lizotte, Tao Wang, Michael H Bowling, Dale Schuurmans, et al. Automatic gait optimization with gaussian process regression. In *IJCAI*, volume 7, pages 944–949, 2007.
- [145] Niranjan Srinivas, Andreas Krause, Sham M Kakade, and Matthias Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. *arXiv preprint arXiv:0912.3995*, 2009.
- [146] Eric Brochu, Tyson Brochu, and Nando De Freitas. A bayesian interactive optimization approach to procedural animation design. In *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 103–112, 2010.
- [147] Eric Brochu, Vlad M Cora, and Nando De Freitas. A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *arXiv preprint arXiv:1012.2599*, 2010.
- [148] Roman Marchant and Fabio Ramos. Bayesian optimisation for intelligent environmental monitoring. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pages 2242–2249. IEEE, 2012.

- [149] Nimalan Mahendran, Ziyu Wang, Firas Hamze, and Nando De Freitas. Adaptive mcmc with bayesian optimization. In *Artificial Intelligence and Statistics*, pages 751–760. PMLR, 2012.
- [150] Javad Azimi, Ali Jalali, and Xiaoli Z Fern. Hybrid batch bayesian optimization. In *Proceedings of the 29th International Conference on International Conference on Machine Learning*, pages 315–322, 2012.
- [151] Ziyu Wang, Babak Shakibi, Lin Jin, and Nando Freitas. Bayesian multi-scale optimistic optimization. In *Artificial Intelligence and Statistics*, pages 1005–1014. PMLR, 2014.
- [152] Xilu Wang, Yaochu Jin, Sebastian Schmitt, and Markus Olhofer. Recent advances in bayesian optimization. *ACM Computing Surveys*, 55(13s):1–36, 2023.
- [153] Carl Edward Rasmussen. Gaussian processes in machine learning. In *Summer school on machine learning*, pages 63–71. Springer, 2003.
- [154] Fernando Nogueira. Bayesian Optimization: Open source constrained global optimization tool for Python, 2014–.
- [155] HJ Kushner. A new method of locating the maximum point of an arbitrary multi-peak curve in the presence of noise. *Journal of Basic Engineering*, 86(1):97–106, 1964.
- [156] Peter I Frazier. A tutorial on bayesian optimization. *arXiv preprint arXiv:1807.02811*, 2018.
- [157] Donald R Jones, Matthias Schonlau, and William J Welch. Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, 13:455–492, 1998.
- [158] Charles E Clark. The greatest of a finite set of random variables. *Operations Research*, 9(2):145–162, 1961.
- [159] Jonas Mockus. On bayesian methods for seeking the extremum. In *Proceedings of the IFIP Technical Conference*, pages 400–404, 1974.
- [160] Matthew Hoffman, Eric Brochu, Nando De Freitas, et al. Portfolio allocation for bayesian optimization. In *UAI*, pages 327–336, 2011.
- [161] Peter Auer, Nicolo Cesa-Bianchi, Yoav Freund, and Robert E Schapire. Gambling in a rigged casino: The adversarial multi-armed bandit problem. In *Proceedings of IEEE 36th annual foundations of computer science*, pages 322–331. IEEE, 1995.

- [162] Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. A comprehensive survey of continual learning: theory, method and application. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.
- [163] Raia Hadsell, Dushyant Rao, Andrei A Rusu, and Razvan Pascanu. Embracing change: Continual learning in deep neural networks. *Trends in cognitive sciences*, 24(12):1028–1040, 2020.
- [164] Dhireesha Kudithipudi, Mario Aguilar-Simon, Jonathan Babb, Maxim Bazhenov, Douglas Blackiston, Josh Bongard, Andrew P Brna, Suraj Chakravarthi Raja, Nick Cheney, Jeff Clune, et al. Biological underpinnings for lifelong learning machines. *Nature Machine Intelligence*, 4(3):196–210, 2022.
- [165] German I Parisi, Ronald Kemker, Jose L Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural networks*, 113:54–71, 2019.
- [166] Guido M Van de Ven, Tinne Tuytelaars, and Andreas S Tolias. Three types of incremental learning. *Nature Machine Intelligence*, 4(12):1185–1197, 2022.
- [167] Zhiyuan Chen and Bing Liu. Lifelong machine learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 12(3):1–207, 2018.
- [168] James L McClelland, Bruce L McNaughton, and Randall C O’Reilly. Why there are complementary learning systems in the hippocampus and neocortex: insights from the successes and failures of connectionist models of learning and memory. *Psychological review*, 102(3):419, 1995.
- [169] P Ruvolo and Eaton E Ella. An efficient lifelong learning algorithm. In *Proceedings of the 30th International Conference on Machine Learning*, pages 507–515, 2013.
- [170] Nicolas Y Masse, Gregory D Grant, and David J Freedman. Alleviating catastrophic forgetting using context-dependent gating and synaptic stabilization. *Proceedings of the National Academy of Sciences*, 115(44):E10467–E10475, 2018.
- [171] Rahul Ramesh and Pratik Chaudhari. Model zoo: A growing" brain" that learns continually. *arXiv preprint arXiv:2106.03027*, 2021.
- [172] David Lopez-Paz and Marc' Aurelio Ranzato. Gradient episodic memory for continual learning. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

- [173] Joshua T Vogelstein, Jayanta Dey, Hayden S Helm, Will LeVine, Ronak D Mehta, Tyler M Tomita, Haoyin Xu, Ali Geisa, Qingyang Wang, Gido M van de Ven, et al. Representation ensembling for synergistic lifelong learning with quasilinear complexity. *arXiv preprint arXiv:2004.12908*, 2020.
- [174] Zixuan Ke, Bing Liu, Hu Xu, and Lei Shu. Classic: Continual and contrastive learning of aspect sentiment classification tasks. *arXiv preprint arXiv:2112.02714*, 2021.
- [175] M Jehanzeb Mirza, Marc Masana, Horst Possegger, and Horst Bischof. An efficient domain-incremental learning approach to drive in all weather conditions. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3001–3011, 2022.
- [176] Rahaf Aljundi, Punarjay Chakravarty, and Tinne Tuytelaars. Expert gate: Life-long learning with a network of experts. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3366–3375, 2017.
- [177] Johannes Von Oswald, Christian Henning, Benjamin F Grewe, and João Sacramento. Continual learning with hypernetworks. *arXiv preprint arXiv:1906.00695*, 2019.
- [178] Mitchell Wortsman, Vivek Ramanujan, Rosanne Liu, Aniruddha Kembhavi, Mohammad Rastegari, Jason Yosinski, and Ali Farhadi. Supermasks in superposition. *Advances in Neural Information Processing Systems*, 33:15173–15184, 2020.
- [179] James B Heald, Máté Lengyel, and Daniel M Wolpert. Contextual inference underlies the learning of sensorimotor repertoires. *Nature*, 600(7889):489–493, 2021.
- [180] Vincenzo Lomonaco and Davide Maltoni. Core50: a new dataset and benchmark for continuous object recognition. In *Conference on robot learning*, pages 17–26. PMLR, 2017.
- [181] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H. Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.
- [182] Xiaoyu Tao, Xiaopeng Hong, Xinyuan Chang, Songlin Dong, Xing Wei, and Yihong Gong. Few-shot class-incremental learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12183–12192, 2020.

- [183] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. *Advances in neural information processing systems*, 30, 2017.
- [184] Eden Belouadah, Adrian Popescu, and Ioannis Kanellos. A comprehensive study of class incremental learning algorithms for visual tasks. *Neural Networks*, 135:38–54, 2021.
- [185] Marc Masana, Xialei Liu, Bartłomiej Twardowski, Mikel Menta, Andrew D. Bagdanov, and Joost van de Weijer. Class-incremental learning: Survey and performance evaluation on image classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(5):5513–5533, 2023.
- [186] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017.
- [187] Hippolyt Ritter, Aleksandar Botev, and David Barber. Online structured laplace approximations for overcoming catastrophic forgetting. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.
- [188] Jonathan Schwarz, Wojciech Czarnecki, Jelena Luketina, Agnieszka Grabska-Barwinska, Yee Whye Teh, Razvan Pascanu, and Raia Hadsell. Progress & compress: A scalable framework for continual learning. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4528–4537. PMLR, 10–15 Jul 2018.
- [189] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 3987–3995. PMLR, 06–11 Aug 2017.
- [190] Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory aware synapses: Learning what (not) to forget. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

- [191] Arslan Chaudhry, Puneet K. Dokania, Thalaiyasingam Ajanthan, and Philip H. S. Torr. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.
- [192] Frederik Benzing. Unifying importance based regularisation methods for continual learning. In Gustau Camps-Valls, Francisco J. R. Ruiz, and Isabel Valera, editors, *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*, volume 151 of *Proceedings of Machine Learning Research*, pages 2372–2396. PMLR, 28–30 Mar 2022.
- [193] Jianping Gou, Baosheng Yu, Stephen J. Maybank, and Dacheng Tao. Knowledge distillation: A survey. *Int. J. Comput. Vision*, 129(6):1789–1819, June 2021.
- [194] Prithviraj Dhar, Rajat Vikram Singh, Kuan-Chuan Peng, Ziyang Wu, and Rama Chellappa. Learning without memorizing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.
- [195] Ahmet Iscen, Jeffrey Zhang, Svetlana Lazebnik, and Cordelia Schmid. Memory-efficient incremental learning through feature adaptation. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision – ECCV 2020*, pages 699–715, Cham, 2020. Springer International Publishing.
- [196] Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(12):2935–2947, 2018.
- [197] Amal Rannen, Rahaf Aljundi, Matthew B. Blaschko, and Tinne Tuytelaars. Encoder based lifelong learning. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.
- [198] Francisco M. Castro, Manuel J. Marin-Jimenez, Nicolas Guil, Cordelia Schmid, and Karteek Alahari. End-to-end incremental learning. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.
- [199] Arthur Douillard, Matthieu Cord, Charles Ollion, Thomas Robert, and Eduardo Valle. Podnet: Pooled outputs distillation for small-tasks incremental learning. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision – ECCV 2020*, pages 86–102, Cham, 2020. Springer International Publishing.

- [200] Kibok Lee, Kimin Lee, Jinwoo Shin, and Honglak Lee. Overcoming catastrophic forgetting with unlabeled data in the wild. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.
- [201] Chenshen Wu, Luis Herranz, Xialei Liu, yaxing wang, Joost van de Weijer, and Bogdan Raducanu. Memory replay gans: Learning to generate new categories without forgetting. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.
- [202] Mengyao Zhai, Lei Chen, Frederick Tung, Jiawei He, Megha Nawhal, and Greg Mori. Lifelong gan: Continual learning for conditional image generation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.
- [203] Matthew Riemer, Ignacio Cases, Robert Ajemian, Miao Liu, Irina Rish, Yuhai Tu, and Gerald Tesauro. Learning to learn without forgetting by maximizing transfer and minimizing interference. *arXiv preprint arXiv:1810.11910*, 2018.
- [204] Jeffrey S. Vitter. Random sampling with a reservoir. *ACM Trans. Math. Softw.*, 11(1):37–57, March 1985.
- [205] Xialei Liu, Chenshen Wu, Mikel Menta, Luis Herranz, Bogdan Raducanu, Andrew D. Bagdanov, Shangling Jui, and Joost van de Weijer. Generative feature replay for class-incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2020.
- [206] Kai Zhu, Wei Zhai, Yang Cao, Jiebo Luo, and Zheng-Jun Zha. Self-sustaining representation expansion for non-exemplar class-incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9296–9305, June 2022.
- [207] Eden Belouadah and Adrian Popescu. Il2m: Class incremental learning with dual memory. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.
- [208] Zheng Gong, Kun Zhou, Xin Zhao, Jing Sha, Shijin Wang, and Ji-Rong Wen. Continual pre-training of language models for math problem understanding with syntax-aware memory network. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio, editors, *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5923–5933, Dublin, Ireland, May 2022. Association for Computational Linguistics.

- [209] Marco Toldo and Mete Ozay. Bring evanescent representations to life in lifelong class incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16732–16741, June 2022.
- [210] Grégoire Petit, Adrian Popescu, Hugo Schindler, David Picard, and Bertrand Delezoide. Fetril: Feature translation for exemplar-free class-incremental learning. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 3911–3920, January 2023.
- [211] Jiawang Bian, Zhichao Li, Naiyan Wang, Huangying Zhan, Chunhua Shen, Ming-Ming Cheng, and Ian Reid. Unsupervised scale-consistent depth and ego-motion learning from monocular video. *Advances in neural information processing systems*, 32, 2019.
- [212] Jia-Wang Bian, Huangying Zhan, Naiyan Wang, Tat-Jun Chin, Chunhua Shen, and Ian Reid. Auto-rectify network for unsupervised indoor depth estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(12):9802–9813, 2022.
- [213] Libo Sun, Jia-Wang Bian, Huangying Zhan, Wei Yin, Ian Reid, and Chunhua Shen. Sc-depthv3: Robust self-supervised monocular depth estimation for dynamic scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(1):497–508, 2024.
- [214] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.
- [215] Vitor Guizilini, Rareş Ambruş, Dian Chen, Sergey Zakharov, and Adrien Gaidon. Multi-frame self-supervised depth with transformers. In *IEEE/CVF CVPR*, pages 160–170, June 2022.
- [216] Jinwoo Bae, Sungho Moon, and Sunghoon Im. Deep digging into the generalization of self-supervised monocular depth estimation. In *AAAI*, 2023.
- [217] Anurag Ranjan, Varun Jampani, Lukas Balles, Kihwan Kim, Deqing Sun, Jonas Wulff, and Michael J. Black. Competitive collaboration: Joint unsupervised learning of depth, camera motion, optical flow and motion segmentation. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12232–12241, 2019.
- [218] Chenxu Luo, Zhenheng Yang, Peng Wang, Yang Wang, Wei Xu, Ram Nevatia, and Alan Yuille. Every pixel counts++: Joint learning of geometry and motion with 3d

- holistic understanding. *IEEE transactions on pattern analysis and machine intelligence*, 42(10):2624–2641, 2019.
- [219] Junda Cheng, Wei Yin, Kaixuan Wang, Xiaozhi Chen, Shijie Wang, and Xin Yang. Adaptive fusion of single-view and multi-view depth for autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10138–10147, June 2024.
- [220] Mykola Lavreniuk. Spidepth: Strengthened pose information for self-supervised monocular depth estimation, 2024.
- [221] Ruijie Zhu, Chuxin Wang, Ziyang Song, Li Liu, Tianzhu Zhang, and Yongdong Zhang. Scaleddepth: Decomposing metric depth estimation into scale prediction and relative depth estimation, 2024.
- [222] Ruoyu Wang, Zehao Yu, and Shenghua Gao. Planedepth: Self-supervised depth estimation via orthogonal planes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 21425–21434, June 2023.
- [223] Jiahe Shi, Yawen Wu, Dewen Zeng, Jun Tao, Jingtong Hu, and Yiyu Shi. Self-supervised on-device federated learning from unlabeled streams. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 42(12):4871–4882, December 2023.
- [224] Fengda Zhang, Kun Kuang, Zhaoyang You, Tao Shen, Jun Xiao, Yin Zhang, Chao Wu, Yueting Zhuang, and Xiaolin Li. Federated unsupervised representation learning. *arXiv preprint arXiv:2010.08982*, 2020.
- [225] Juan Zhao, Ruixuan Li, Haozhao Wang, and Zijun Xu. Hotfed: Hot start through self-supervised learning in federated learning. In *2021 IEEE HPCC/DSS/SmartCity/DependSys*, pages 149–156. IEEE, 2021.
- [226] Weiming Zhuang, Yonggang Wen, and Shuai Zhang. Divergence-aware federated self-supervised learning. *arXiv preprint arXiv:2204.04385*, 2022.
- [227] Disha Makhija, Nhat Ho, and Joydeep Ghosh. Federated self-supervised learning for heterogeneous clients, 2022.
- [228] Alex Krizhevsky. Learning multiple layers of features from tiny images. *University of Toronto*, 05 2012.
- [229] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *IEEE CVPR*, pages 248–255, 2009.

- [230] Yevhen Kuznetsov, Marc Proesmans, and Luc Van Gool. Comoda: Continuous monocular depth adaptation using past experiences. In *IEEE/CVF WACV*, pages 2907–2917, 2021.
- [231] Clément Godard, Oisin Mac Aodha, Michael Firman, and Gabriel J Brostow. Digging into self-supervised monocular depth estimation. In *CVPR*, 2019.
- [232] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*. Springer, 2015.
- [233] Tinghui Zhou, Matthew Brown, Noah Snavely, and David G. Lowe. Unsupervised learning of depth and ego-motion from video. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6612–6619, 2017.
- [234] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE/CVF CVPR*, pages 770–778, 2016.
- [235] Z. Wang, E.P. Simoncelli, and A.C. Bovik. Multiscale structural similarity for image quality assessment. In *The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003*, volume 2, pages 1398–1402 Vol.2, 2003.
- [236] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.
- [237] Uchitha Rajapaksha, Ferdous Sohel, Hamid Laga, Dean Diepeveen, and Mohammed Bennamoun. Deep learning-based depth estimation methods from monocular image and videos: A comprehensive survey. *ACM Computing Surveys*, 2024.
- [238] Akhil Gurram and Antonio M Lopez. On the metrics for evaluating monocular depth estimation. *arXiv preprint arXiv:2302.10007*, 2023.
- [239] Su Liu, Jiong Yu, Xiaoheng Deng, and Shaohua Wan. Fedcpf: An efficient-communication federated learning approach for vehicular edge computing in 6g communication networks. *IEEE Transactions on Intelligent Transportation Systems*, 23(2):1616–1629, 2021.
- [240] Andrew Hard, Kurt Partridge, Cameron Nguyen, Niranjana Subrahmanya, Aishanee Shah, Pai Zhu, Ignacio Lopez Moreno, and Rajiv Mathews. Training keyword spotting models on non-iid data with federated learning. *arXiv preprint arXiv:2005.10406*, 2020.

- [241] Jian Xu, Meiling Yang, Wenbo Ding, and Shao-Lun Huang. Stabilizing and improving federated learning with non-iid data and client dropout. *arXiv preprint arXiv:2303.06314*, 2023.
- [242] Bradley Efron. Estimating the error rate of a prediction rule: improvement on cross-validation. *Journal of the American statistical association*, 78(382):316–331, 1983.
- [243] Bimal Ghimire and Danda B Rawat. Recent advances on federated learning for cybersecurity and cybersecurity for federated learning for internet of things. *IEEE Internet of Things Journal*, 2022.
- [244] Hung T Nguyen, Vikash Sehwal, Seyyedali Hosseinalipour, Christopher G Brinton, Mung Chiang, and H Vincent Poor. Fast-convergent federated learning. *IEEE Journal on Selected Areas in Communications*, 39(1):201–218, 2020.
- [245] Thiago de P. Vasconcelos, Daniel Augusto R.M.A. de Souza, Gustavo C. de M. Virgolino, César L.C. Mattos, and João P.P. Gomes. Self-tuning portfolio-based bayesian optimization. *Expert Systems with Applications*, 188:115847, 2022.
- [246] Tim Head, Manoj Kumar, Holger Nahrstaedt, Gilles Louppe, and Iaroslav Shcherbatyi. scikit-optimize/scikit-optimize, September 2020.
- [247] Chaoqiang Zhao, Youmin Zhang, Matteo Poggi, Fabio Tosi, Xianda Guo, Zheng Zhu, Guan Huang, Yang Tang, and Stefano Mattoccia. Monovit: Self-supervised monocular depth estimation with a vision transformer. In *2022 International Conference on 3D Vision (3DV)*, pages 668–678. IEEE, 2022.
- [248] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3213–3223, 2016.
- [249] Rostam Affendi Hamzah and Haidi Ibrahim. Literature survey on stereo vision disparity map algorithms. *Journal of Sensors*, 2016(1):8742920, 2016.
- [250] Guorun Yang, Xiao Song, Chaoqin Huang, Zhidong Deng, Jianping Shi, and Bolei Zhou. Drivingstereo: A large-scale dataset for stereo matching in autonomous driving scenarios. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 899–908, 2019.

- [251] Denis Zavadski, Damjan Kalšan, and Carsten Rother. Primedepth: Efficient monocular depth estimation with a stable diffusion preimage. In *Proceedings of the Asian Conference on Computer Vision*, pages 922–940, 2024.
- [252] Shangshu Yu, Meiqing Wu, and Siew-Kei Lam. Vfm-depth: Leveraging vision foundation model for self-supervised monocular depth estimation. *IEEE Transactions on Circuits and Systems for Video Technology*, 2024.
- [253] Ruhul Amin Khalil, Ziad Safelnasr, Naod Yemane, Mebruk Kedir, Atawulrahman Shafiqurrahman, and Nasir Saeed. Advanced learning technologies for intelligent transportation systems: Prospects and challenges. *IEEE Open Journal of Vehicular Technology*, 2024.