DEVELOPMENT AND ASSESSMENT OF PREDICTIVE MODELS FOR IMPROVED SWINE FARMING

By

Junjie Han

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

Animal Science—Doctor of Philosophy
Computational Mathematics, Science and Engineering—Dual Major

2022

ABSTRACT

DEVELOPMENT AND ASSESSMENT OF PREDICTIVE MODELS FOR IMPROVED SWINE FARMING

By

Junjie Han

Prediction of outcomes is critical in both swine breeding and management. This necessitates the development of predictive models that address challenges in swine farming. For predictive modeling, there have been significant advances in deep learning. Nevertheless, there are needs to adapt deep learning-based models for specific swine farming problems including genomic prediction and behavior analysis. Furthermore, there is not yet a clear guideline on how to validate a model in this field. The overarching goal of this dissertation was to validate a collection of predictive models for improved swine farming with applications to precision management, phenotyping, and breeding. The first study addressed the pig genomic prediction problem. Differential evolution was utilized to optimize deep learning (DL) hyperparameters that affected the predictive performance of DL models. Performance of optimized DL was compared with “best practice” DL architectures selected from literature and baseline DL models with randomly specified hyperparameters. Optimized models showed clear improvement. Further, differential evolution saved considerable time compared to traditional optimization approaches e.g., grid search. Despite the success of genomic prediction, phenotyping has become a bottleneck in breeding programs as it is still time-consuming and labor-intensive. Computer vision (CV) can be used to automate the phenotyping process. Nonetheless, there are limited amount of public data for CV development in livestock farming. Most published CV applications to livestock farming were developed using rather small datasets, and their broader validity remained unknown. Therefore, the second study aimed at reviewing publicly available image

datasets that were used for CV algorithms in livestock farming and the validation methods in the related work. Through the review, we could not find public datasets that addressed pigs' agonistic behaviors (negative social behaviors), which is an important topic in swine farming. Given this, the third study aimed at collecting a video dataset to study pig's agonistic behavior and adapting a state-of-the-art DL pipeline to classify pigs' agonistic behaviors through video analysis. The pipeline was validated through various training-validation data partitions, where the training data were used for model development and the validation data were used for model evaluation. Results showed that splitting the training and validation sets at random led to over-optimistic estimates of model performance. The last study focused on developing and validating a statistical model for the analysis of pigs' social interactions. Generalized linear mixed models were fitted, and a Bayesian framework was used for parameter estimation and posterior predictive model checking. The predictive performance of the models varied depending on the validation strategy, where three strategies were defined: random cross-validation, block-by-social-group cross-validation, and block-by-focal-animals validation. In conclusion, this dissertation provides information about how state-of-the-art models can be adapted for and validated in swine farming applications. Future directions of this research could aim at creating reference imagery datasets in swine farming that provides a platform for CV applications and developing integrated computer vision systems, which eventually assists in prediction tasks for improved pig management and breeding.

To my wife, my mother, and my father.

ACKNOWLEDGEMENTS

I sincerely appreciate my family, my dissertation committee members, MSU faculty and staff, and friends. Without their support, it would not be possible for me to accomplish my doctoral dissertation. Furthermore, my interactions with colleagues, professors, and students, and my personal positive experience on being mentored gave me the opportunity to understand and appreciate the beauty of both science and humankind.

To my major advisor, Dr. Juan P. Steibel. Juan has patiently and selflessly guided me to be a qualified PhD student in many aspects- critical thinking, scientific writing, presentation skills, and attitude, etc. He has always supported my decisions and given his wholehearted advice on every single question I asked. He was always approachable and has spent so much time mentoring me. In personal life, I am grateful that he treated me as a friend and gave me advice beyond academic scope. Great thanks to Juan!

To my dissertation committee members: Dr. Janice Siegford, Dr. Cedric Gondro, Dr. Robert Tempelman, Dr. Tami Brown-Brandl, and Dr. Dirk Colbry. The high standard they have set and expected for me motivated myself to push the limit. Their encouragement and constructive suggestions guided me to become a better student. They are academic role models for me, and they have influenced me to set up my career goals as a scientist. It was my honor to be mentored by these great people.

I would not be able to finish my PhD program without the support from my family. My father, Hongming Han, and my mother, Zhenai Piao, have always solidly got my back no matter what happened. Special thanks to my wife, Jinhua Qian. She has been incredibly supportive and inspired me to overcome challenges. She has always been there to share my

happiness and low morale. She was understanding and considering throughout my PhD journey, and she always cheered me up.

I thank my collaborators and researchers from whom I learned different aspects to think of problems. Dr. Kenneth Reid, Dr. Joao Dorea, Dr. Tomas Norton, Andrea Parmiggiani, Dr. Daniel Morris, Raymond Lesiyon, Anna Bosgraaf, Chen Chen, and Dr. Gustavo de los Campos have provided valuable suggestions and help in my research projects. It was my pleasure to work with my collaborators. Also, I thank Kevin Turner and Chris Rozeboom for their assistance in my experiments at MSU Swine Teaching & Research Center.

Finally, thanks to my fellow students from departments of animal science & computational mathematics, science and engineering at MSU. Special thanks to my great friends Lingkun Li, Xing Lu, Daoyang Chen, Mingzhe Li, Fei Zhang, Kangxu Wang, and Zinan Wang for their company during my PhD journey. They shared valuable thoughts with me and comforted me when I was in low morale.

"I'm a million miles ahead of where I'm from, but I still have another million miles to go."

Tim Bergling

PREFACE

Chapter 2 was formatted for publication in *G3: Genes|Genomes|Genetics* and corresponds to the peer-reviewed version of Han, J., C. Gondro, K. Reid, and J. P. Steibel. 2021. Heuristic hyperparameter optimization of deep learning models for genomic prediction. G3 Genes|Genomes|Genetics. 11. doi:10.1093/g3journal/jkab032.

Chapter 3 was formatted for the preprint version of Han, J., J. R. Dorea, T. Norton, A. Parmiggiani, D. Morris, J. Siegford, and J. P. Steibel. Publicly Available Datasets for Computer Vision in Precision Livestock Farming: A Review.

Chapter 4 was formatted for publication in *Computers and Electronics in Agriculture* and corresponds to the pending review version of Han, J., J. Siegford, D. Colbry, R. Lesiyon, A. Bosgraaf, C. Chen, T. Norton, and J. P. Steibel. 2022. Under review. Evaluation of Computer Vision for Detecting Agonistic Behavior of Pigs in a Single-Space Feeding Stall Through Blocked Cross-Validation Strategies.

Chapter 5 was formatted for publication in *Applied Animal Behaviour Science*.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

KEY TO ABBREVIATIONS

adam = Adaptive moment estimation

adj = Adjusted

AUC = Area under receiver operating characteristics curve

CNN = Convolutional neural network

CPU = Central processing unit

CV = Computer vision

DE = Differential evolution

DL = Deep learning

DOI = Digital object identifier

EB = Ear-to-body

elu = Exponential linear unit

Eq = Equation

GBLUP = Genomic best linear unbiased prediction

GLMM = Generalized linear mixed models

GPU = Graphics processing unit

h^2 = Heritability

HB = Head-to-body

hr = Hour

ID = Identification

L = Levering

LSTM = Long short-term memory

M = Mounting

mAP = mean average precision

MCMC = Markov chain Monte Carlo

MLP = Multilayer perceptron

MOTA = Multiple objects tracking accuracy

MSE/mse = Mean squared error

N = Number of samples

nadam = Nesterov-accelerated adaptive moment estimation

NC = No-contact

obs = Observed

Opt = Optimized

OSF = Open science framework

P = P-value

PLF = Precision livestock farming

QTL = Quantitative trait loci

R2 = R squared

relu = Rectified linear unit

RGB = Red-green-blue

RGB-D = Red-green-blue and depth

RMSE = Root mean square error

rmsprop = Root mean square propagation

ROC = Receiver operating characteristics

ROI = Region of interest

SD = Standard deviation

selu = Scaled exponential linear unit

SG = Social group

sgd = Stochastic gradient descent

SNP = Single nucleotide polymorphism

softmax = Normalized exponential function

US = United States

US = United States of America

USDA = United States Department of Agriculture

CHAPTER 1: GENERAL INTRODUCTION

1. INTRODUCTION

Currently, the United States (US) is the world's third-largest producer and consumer of pork products, exporting over 20% of commercial pork produced (USDA, 2019). Such success is partially due to structural and organizational changes to the US swine industry (Pairis-Garcia et al., 2016). In the past decades, the US swine industry has transitioned from small and private farms into larger and integrated systems (Pairis-Garcia et al., 2016).

Swine farmers are interested in stable and predictable conditions for pig production and marketing, which helps achieve acceptable margins between costs and profits (Commandeur, 2006). To improve profitability in swine farming, a common strategy is to scale up the production (Öhlund et al., 2017). Nevertheless, current internal migration trends from rural to urban areas are leading to decreasing number of farmers (Berckmans, 2017), resulting in increased workload per farmer. Furthermore, as both the population and income increase globally, there are ever growing demand for pork products (Alexandratos and Bruinsma, 2012). Production systems need to continue to adapt to meet these challenges.

Breeding is a powerful tool to improve swine farming efficiency (Oliviero et al., 2019). In livestock farming, breeding refers to the selection of individuals as parents to produce better performing offspring (Hill, 2001). Livestock breeding has enabled great increase in efficiency and economic benefit of swine farming (Hayes et al., 2013). An important prediction problem in livestock breeding is how to estimate animals' breeding values. Traditional breeding programs simply relied on the animal pedigree and phenotypic values that are quantifiable traits e.g., weight in pounds and feed intake in grams. Modern breeding programs use genomic prediction,

which refers to the prediction of individual phenotypic/breeding value using genomic information. To implement genomic prediction, advanced prediction models are needed that relate animal genomics to phenotypic values.

Management is another powerful tool for improved swine farming efficiency (Peltoniemi et al., 2019). Daily management contributes to the optimization of costs and benefits, leading to increased profitability in swine farming (Agostini et al., 2014). An important fact is that, to manage pigs, farmers need to predict the status of pigs (e.g., predicting whether the pigs are fed well and predicting pigs' health condition). However, especially in large-scale pig farms, it is challenging to predict and manage pigs at individual levels. Therefore, prediction models for individual animals will positively assist in pig management.

Prediction of outcomes is critical in both swine breeding and management. As result, researchers, swine breeders and swine farmers are increasingly interested in predictive modeling that assists on-farm decision making (Tzanidakis et al., 2021). Predictive modeling refers to the process of developing a statistical or computational tool that forecasts future outcomes with the aid of existing data (Kuhn and Johnson, 2013). Although the goal of developing predictive modeling is to predict future outcomes, relatively few models are developed and validated in depth to match the goal. Validating a prediction model essentially means comparing predicted values to the actual observed outcomes in a population (Ramspek et al., 2021). For validation purposes, data are split into two parts: a training set that is used for model development and a validation set that is used for model assessment. The common practice is to split data at random, i.e., random validation. The underlying assumption of random validation is that the training set and validation set are independent. However, dependence structures may exist in the data e.g., temporal and spatial structures which violates the assumption of independence between the

training set and validation set and leads to overoptimistic results (Roberts et al., 2017). Unfortunately, data structures are often overlooked when splitting the data for validation purposes. Therefore, proper designs of validation strategy i.e., the training-validation data split, is necessary to determine a prediction model's generalizability and reproducibility to new farming environments and pigs (Ramspek et al., 2021).

Despite the substantial advancement in predictive modeling methods (Putka et al., 2018), few attempts have been made to introduce the advanced predictive modeling to practical swine farming. There are needs to adapt state-of-the-art predictive models for swine farming, as the predictive ability of advanced methods e.g., deep learning is less well studied on swine breeding and behavior recognition problems. Further, there is not yet a clear guideline on how to properly split the data when developing and validating a novel predictive model in the field. Therefore, collectively, this study aims at filling this gap by investigating and assessing several state-of-the-art predictive models that fit into the swine farming context.

1.1 Deep learning for genomic prediction

Deep learning, a part of statistical learning toolboxes, has dramatically improved state-of-the-art applications in computer vision, speech recognition and genomics (Lecun et al., 2015). Deep learning (DL) is a set of representation learning methods, where a machine can be fed with raw data and automatically discover the representations needed for prediction or classification, with multiple levels of simple but non-linear modules transforming the representation at one level into a representation at a higher, slightly more abstract level (Lecun et al., 2015). Its flexibility allows researchers to apply DL in several contexts of prediction problems. For instance, DL has already been applied to the genomic and phenotypic prediction in plants (Crossa et al., 2019; Montesinos-López et al., 2018), human traits (Bellot et al., 2018), and

breeding values of bulls (Abdollahi-Arpanahi et al., 2020). Nevertheless, the potential of DL-based genomic prediction remains unknown in swine farming. Thus, a better understanding of DL applications in swine genomic prediction will bring valuable information to the swine industry.

1.2 Deep learning for phenotyping

Genomic prediction tools have made great contributions to swine breeding programs. Thanks to the technological developments in genomic studies, genotyping costs have decreased significantly in the past years. However, phenotyping, the process of determining an individual's observable trait(s), has become a bottleneck in breeding programs, as it is still time consuming and labor intensive, which may be even more costly than genotyping (Watanabe et al., 2017). Thus, there are needs for phenotyping tools to accelerate swine breeding programs.

Deep learning is also suitable for high-throughput phenotyping through automating the analysis video recordings. As DL is predominant in computer vision (CV) that refers to the use of artificial intelligence to automatically extract useful information form computer graphics, DL can be used to predict traits of animals through digital images. Compared to the traditional approach that requires manual observation, CV has advantages of being low-cost, objective, and non-interventional, and to generate information in a continuous scale (Chen et al., 2021; Li et al., 2021). Researchers have already discovered this promising tool and started to develop DL models for pig posture/behavior analysis in their experiments (Chen et al., 2020b; Liu et al., 2020; Nasirahmadi et al., 2019; Zhang et al., 2020).

A limitation of DL centers around the data that are used to develop DL models. Deep learning is known as a data-hungry approach, and it typically requires a large number of samples in order to train models with acceptable performance (Lu et al., 2017; Marcus, 2018). For DL-

based CV in general, there exist reference datasets e.g., COCO (Lin et al., 2014) and ImageNet (Jia Deng et al., 2009), which consist thousands of images with millions of instances annotated by experts that allow developers to test algorithms. To the best of our knowledge, we do not have such datasets available in livestock farming. Therefore, significant effort is needed to create public image datasets in livestock farming that is vital to develop DL-based high-throughput phenotyping. To achieve this, a survey of public resources for the existing animal CV datasets is a necessary first step to create reference data in the filed.

1.3 Deep learning and statistical learning for animal-animal interaction

Agonistic behaviors are part of the standard repertoire of behaviors in pigs and include aggressive, submissive, and defensive behaviors in competitive situations (Gasser et al., 2009; Machado et al., 2017). In swine farming, pigs present agonistic behaviors in forms of fighting over the control of resources or space and performing shows of force to establish social hierarchy in the group (Machado et al., 2017). Consequently, pigs receive injures and suffer stress that affect their welfare and productivity (Ekkel et al., 1995). Therefore, it is necessary to target traits that are related to pigs' agonistic behaviors. Understanding this type of traits will bring valuable information to swine welfare, management, and breeding.

There are two key steps in the analysis of pigs' agonistic behaviors: 1) measuring the traits of interest and 2) analyzing the measures. It is noteworthy that animal-animal interaction is one of the intrinsic components of agonistic behavior. Measuring or recognizing interactive behaviors requires observing motions of at least two animals simultaneously, which is more complicated than measuring behaviors that pertain to an individual. To observe and measure traits about pigs' agonistic behaviors, the standard approach is through manual annotation of videos that is laborious and time-consuming. A DL-based CV model seems to be helpful to

measure the traits as it automates the behavior recognition. To date, most DL-based CV applications concentrate on individual pig behaviors, and publicly available imagery datasets in this field were designed and annotated for studying behaviors of individual pigs. However, none of the publicly available datasets so far are specifically about pig agonistic interactions. This gap necessitates creating image datasets that are publicly available for the study of pigs' agonistic behaviors and developing a CV algorithm that assists in measuring traits from images/videos.

Once the trait is measured, subsequently, another challenging task is to analyze the traits that are related to pigs' interactive behaviors. Researchers are interested in quantifying the interaction intensity as well as factors affecting the interactions between animals. Interactive behavior of pigs usually involves two pigs, namely a dyad. Given data on pairwise social interactions recorded from an experimental or observational study, it is necessary to quantify the effects of various individual- and group-level factors on social interactions. However, there is limited research on the models to analyze pairwise social interactions in pigs.

A proper way to model dyadic data is to fit generalized linear mixed models (GLMM) that include fixed and random effects to account for means and covariances depending on the actual design of the experiment (Kenny et al., 2020). The potential for using this approach remains unknown in the analysis of pig's social behaviors. A conceptualization and implementation of the GLMM framework will contribute to the better understanding of pigs' social interactions.

2. OBJECTIVES

The overarching goal of this study is to validate state-of-the-art prediction models and techniques for improved swine farming. The specific objectives for this research are:

Objective 1: to develop deep learning models for genomic prediction and assess model performance through different training-validation data splits.

Objective 2: to investigate public datasets for computer vision applications in livestock farming that are useful for phenotyping and behavior studies, and to review the validation strategies in the related work that utilized the public datasets.

Objective 3: to assess performance of deep learning-based computer vision models through different data splitting strategies by articulating temporal and spatial independence between training and validation sets.

Objective 4: to conceptualize, implement, and validate a novel statistical model for the analysis of social interaction data.

REFERENCES

REFERENCES

Abdollahi-Arpanahi, R., Gianola, D., Peñagaricano, F., 2020. Deep learning versus parametric and ensemble methods for genomic prediction of complex phenotypes. Genet. Sel. Evol. 52, 1–15. https://doi.org/10.1186/s12711-020-00531-z

Agostini, P.S., Fahey, A.G., Manzanilla, E.G., O'Doherty, J. V., De Blas, C., Gasa, J., 2014. Management factors affecting mortality, feed intake and feed conversion ratio of grow-finishing pigs. Animal 8, 1312–1318. https://doi.org/10.1017/S1751731113001912

Alexandratos, N., Bruinsma, J., 2012. World agriculture towards 2030/2050: the 2012 revision.

Bellot, P., de los Campos, G., Pérez-Enciso, M., 2018. Can deep learning improve genomic prediction of complex human traits? Genetics 210, 809–819. https://doi.org/10.1534/genetics.118.301298

Berckmans, D., 2017. General introduction to precision livestock farming. Anim. Front. 7, 6–11. https://doi.org/10.2527/af.2017.0102

Chen, C., Zhu, W., Norton, T., 2021. Behaviour recognition of pigs and cattle: Journey from computer vision to deep learning. Comput. Electron. Agric. 187, 106255. https://doi.org/10.1016/j.compag.2021.106255

Chen, C., Zhu, W., Steibel, J., Siegford, J., Wurtz, K., Han, J., Norton, T., 2020. Recognition of aggressive episodes of pigs based on convolutional neural network and long short-term memory. Comput. Electron. Agric. 169, 105166. https://doi.org/10.1016/j.compag.2019.105166

Commandeur, M.A.M., 2006. Diversity of pig farming styles: Understanding how it is structured. NJAS - Wageningen J. Life Sci. 54, 111–127. https://doi.org/10.1016/S1573-5214(06)80007-2

Crossa, J., Martini, J.W.R., Gianola, D., Pérez-Rodríguez, P., Jarquin, D., Juliana, P., Montesinos-López, O., Cuevas, J., 2019. Deep Kernel and Deep Learning for Genome-Based Prediction of Single Traits in Multienvironment Breeding Trials. Front. Genet. 10, 1–13. https://doi.org/10.3389/fgene.2019.01168

Ekkel, E.D., Van Doorn, C.E.A., Hessing, M.J.C., Tielen, M.J.M., 1995. The specific-stress-free housing system has positive effects on productivity, health, and welfare of pigs. J. Anim. Sci. 73, 1544–1551.

Gasser, P.J., Lowry, C.A., Orchinik, M., 2009. 41 - Rapid Corticosteroid Actions on Behavior: Mechanisms and Implications, in: Pfaff, D.W., Arnold, A.P., Etgen, A.M., Fahrbach, S.E., Rubin Brain and Behavior (Second Edition), R.T.B.T.-H. (Eds.), . Academic Press, San Diego, pp. 1365–1397. https://doi.org/https://doi.org/10.1016/B978-008088783-8.00041-3

Hayes, B.J., Lewin, H.A., Goddard, M.E., 2013. The future of livestock breeding: Genomic selection for efficiency, reduced emissions intensity, and adaptation. Trends Genet. 29, 206–214. https://doi.org/10.1016/j.tig.2012.11.009

Hill, W.G., 2001. Selective Breeding, in: Brenner, S., Miller, J.H.B.T.-E. of G. (Eds.), . Academic Press, New York, pp. 1796–1799. https://doi.org/https://doi.org/10.1006/rwgn.2001.1167

James, G., Witten, D., Hastie, T., Tibshirani, R., 2013. An introduction to statistical learning. Springer.

Jia Deng, Wei Dong, Socher, R., Li-Jia Li, Kai Li, Li Fei-Fei, 2009. ImageNet: A large-scale hierarchical image database 248–255. https://doi.org/10.1109/cvprw.2009.5206848

Kenny, D.A., Kashy, D.A., Cook, W.L., 2020. Dyadic data analysis. Guilford Publications.

Kuhn, M., Johnson, K., 2013. Applied predictive modeling. Springer.

Lecun, Y., Bengio, Y., Hinton, G., 2015. Deep learning. Nature. https://doi.org/10.1038/nature14539

Li, G., Huang, Y., Chen, Z., Chesser, G.D., Purswell, J.L., Linhoss, J., Zhao, Y., 2021. Practices and applications of convolutional neural network-based computer vision systems in animal farming: A review. Sensors 21, 1–42. https://doi.org/10.3390/s21041492

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L., 2014. Microsoft coco: Common objects in context, in: European Conference on Computer Vision. Springer, pp. 740–755.

Liu, D., Oczak, M., Maschat, K., Baumgartner, J., Pletzer, B., He, D., Norton, T., 2020. A computer vision-based method for spatial-temporal action recognition of tail-biting behaviour in group-housed pigs. Biosyst. Eng. 195, 27–41. https://doi.org/10.1016/j.biosystemseng.2020.04.007

Lu, Z., Jiang, X., Kot, A., 2017. Enhance deep learning performance in face recognition. 2017 2nd Int. Conf. Image, Vis. Comput. ICIVC 2017 244–248. https://doi.org/10.1109/ICIVC.2017.7984554

Machado, S.P., Caldara, F.R., Foppa, L., De Moura, R., Gonçalves, L.M.P., Garcia, R.G., De Alencar Nääs, I., Dos Santos Nieto, V.M.O., De Oliveira, G.F., 2017. Behavior of pigs reared in enriched environment: Alternatives to extend pigs attention. PLoS One 12, 1–18. https://doi.org/10.1371/journal.pone.0168427

Marcus, G., 2018. Deep Learning: A Critical Appraisal 1–27.

Montesinos-López, A., Montesinos-López, O.A., Gianola, D., Crossa, J., Hernández-Suárez, C.M., 2018. Multi-environment genomic prediction of plant traits using deep learners with dense architecture. G3 Genes, Genomes, Genet. 8, 3813–3828.

https://doi.org/10.1534/g3.118.200740

Nasirahmadi, A., Sturm, B., Edwards, S., Jeppsson, K.H., Olsson, A.C., Müller, S., Hensel, O., 2019. Deep learning and machine vision approaches for posture detection of individual pigs. Sensors (Switzerland) 19, 1–16. https://doi.org/10.3390/s19173738

Öhlund, E., Hammer, M., Björklund, J., 2017. Managing conflicting goals in pig farming: farmers' strategies and perspectives on sustainable pig farming in Sweden. Int. J. Agric. Sustain. 15, 693–707. https://doi.org/10.1080/14735903.2017.1399514

Oliviero, C., Junnikkala, S., Peltoniemi, O., 2019. The challenge of large litters on the immune system of the sow and the piglets. Reprod. Domest. Anim. 54, 12–21. https://doi.org/10.1111/rda.13463

Pairis-Garcia, M.D., Johnson, A.K., Azarpajouh, S., Colpoys, J.D., Rademacher, C.J., Millman, S.T., Webb, S.R., 2016. The U.S. swine industry: Historical milestones and the future of on-farm swine welfare assessments. CAB Rev. Perspect. Agric. Vet. Sci. Nutr. Nat. Resour. https://doi.org/10.1079/PAVSNNR201611025

Peltoniemi, O., Björkman, S., Oropeza-Moe, M., Oliviero, C., 2019. Developments of reproductive management and biotechnology in the pig. Anim. Reprod. 16, 524–538. https://doi.org/10.21451/1984-3143-AR2019-0055

Putka, D.J., Beatty, A.S., Reeder, M.C., 2018. Modern Prediction Methods: New Perspectives on a Common Problem. Organ. Res. Methods 21, 689–732. https://doi.org/10.1177/1094428117697041

Ramspek, C.L., Jager, K.J., Dekker, F.W., Zoccali, C., Van DIepen, M., 2021. External validation of prognostic models: What, why, how, when and where? Clin. Kidney J. 14, 49–58. https://doi.org/10.1093/ckj/sfaa188

Roberts, D.R., Bahn, V., Ciuti, S., Boyce, M.S., Elith, J., Guillera-Arroita, G., Hauenstein, S., Lahoz-Monfort, J.J., Schröder, B., Thuiller, W., Warton, D.I., Wintle, B.A., Hartig, F., Dormann, C.F., 2017. Cross-validation strategies for data with temporal, spatial, hierarchical, or phylogenetic structure. Ecography (Cop.). 40, 913–929. https://doi.org/10.1111/ecog.02881

Tzanidakis, C., Simitzis, P., Arvanitis, K., Panagakis, P., 2021. An overview of the current trends in precision pig farming technologies. Livest. Sci. 249, 104530. https://doi.org/10.1016/j.livsci.2021.104530

USDA, 2019. United States Department of Agriculture. Hogs & Pork.

Watanabe, K., Guo, W., Arai, K., Takanashi, H., Kajiya-Kanegae, H., Kobayashi, M., Yano, K., Tokunaga, T., Fujiwara, T., Tsutsumi, N., Iwat, H., 2017. High-throughput phenotyping of sorghum plant height using an unmanned aerial vehicle and its application to genomic prediction modeling. Front. Plant Sci. 8, 1–11. https://doi.org/10.3389/fpls.2017.00421

Zhang, K., Li, D., Huang, J., Chen, Y., 2020. Automated video behavior recognition of pigs using two-stream convolutional networks. Sensors (Switzerland) 20. https://doi.org/10.3390/s20041085

CHAPTER 2: HEURISTIC HYPERPARAMETER OPTIMIZATION OF DEEP LEARNING MODELS FOR GENOMIC PREDICTION

Junjie Han, Cedric Gondro, Kenneth Reid, and Juan P. Steibel

1. ABSTRACT

There is a growing interest among quantitative geneticists and animal breeders in the use of deep learning (DL) for genomic prediction. However, the performance of DL is affected by hyperparameters that are typically manually set by users. These hyperparameters do not simply specify the architecture of the model, they are also critical for the efficacy of the optimization and model fitting process. To date, most DL approaches used for genomic prediction have concentrated on identifying suitable hyperparameters by exploring discrete options from a subset of the hyperparameter space. Enlarging the hyperparameter optimization search space with continuous hyperparameters is a daunting combinatorial problem. To deal with this problem, we propose using differential evolution (DE) to perform an efficient search of arbitrarily complex hyperparameter spaces in DL models and we apply this to the specific case of genomic prediction of livestock phenotypes. This approach was evaluated on two pig and cattle datasets with real genotypes and simulated phenotypes (N=7,539 animals and M=48,541 markers) and one real dataset (N=910 individuals and M=28,916 markers). Hyperparameters were evaluated using cross validation. We compared the predictive performance of DL models using hyperparameters optimized by DE against DL models with "best practice" hyperparameters selected from published studies and baseline DL models with randomly specified hyperparameters. Optimized models using DE showed clear improvement in predictive performance across all three datasets. DE optimized hyperparameters also resulted in DL models

with less overfitting and less variation in predictive performance over repeated retraining compared to non-optimized DL models.

2. INTRODUCTION

Over the past decades, there have been enormous gains in the productivity of livestock, much of which was due to the rapid genetic improvement of quantitative traits e.g. growth rates, reproductive traits, and feed conversion rates (Hill, 2016). In recent years, with the rise of DNA sequencing and high throughput genotyping technology as well as with the inception of genomic prediction models (Meuwissen et al., 2001), single nucleotide polymorphisms (SNP) became widely used for genomic prediction and genomic selection.

Genomic prediction refers to the use of statistical models to estimate the genetic component of a phenotype by using data from SNP markers (Meuwissen et al., 2001; VanRaden, 2008). The same models can also be used for phenotypic prediction by associating an individual's genotype to its phenotypes which is commonly used to predict complex traits in humans (Yang et al., 2010). For animal production, both genomic prediction and phenotypic prediction have resulted in more accurate selection while genomic prediction has been useful for management decisions (e.g. market allocation). The technology has also provided a platform for the adoption of novel breeding approaches and has led to new biological insights into the underpinnings of complex quantitative traits (Hickey et al., 2017). For simplicity we will use only the term genomic prediction throughout the text.

Several models have been proposed for genomic prediction (Corvin et al., 2010; Gianola, 2013; Habier et al., 2011; VanRaden, 2008), and GBLUP is one of the most commonly used models (Fragomeni et al., 2017). A common assumption across these models is that genomic

effects are strictly additive, i.e. most models do not explicitly consider interactions between alleles within markers (dominance), nor between markers (epistasis) (Crossa et al., 2019). More recently, deep learning (Lecun et al., 2015) has been proposed as an alternative to genomic prediction models that does not depend on the typical assumptions of traditional genomic prediction methods.

Deep learning (DL) has dramatically improved state-of-the-art applications in computer vision, speech recognition and genomics (Eraslan et al., 2019; Koumakis, 2020; Lecun et al., 2015). DL methods are flexible and can potentially learn very cryptic data structures – even interactions between predictors (Crossa et al., 2019). DL has already been applied to genomic prediction in plants (Crossa et al., 2019; A. Montesinos-López et al., 2018; Montesinos-López et al., 2019), human traits (Bellot et al., 2018), and estimation of breeding values in cattle (Abdollahi-Arpanahi et al., 2020).

DL models in genomic prediction are promising tools (Bellot et al., 2018). However, one of the critical challenges of implementing DL is selection of appropriate hyperparameters since they significantly affect the performance of the prediction algorithm. Hyperparameter features are values or options typically set by users before the model is fitted that impact the algorithm's predictive performance by avoiding overfitting and underfitting (Luo, 2016). Each feature that is part of the hyperparameter set can take a range of values or options and they can interact with each other to determine the properties of the final fitted model; a properly specified hyperparameter set is fundamental for a DL model to achieve a high prediction accuracy. But, unfortunately, there is no one-size-fits-all best way to optimize these hyperparameters.

Several procedures have been used to select DL hyperparameters for genomic prediction applications; e.g. grid search (Crossa et al., 2019; Pérez-Enciso and Zingaretti, 2019) and genetic

algorithms (Bellot et al., 2018). Grid search is only feasible for a limited number of parameters and levels, which is not the case for most DL applications. On the other side, genetic algorithms are better suited for optimizing large and complex parametric spaces, but currently available implementations of genetic algorithms to tune DL hyperparameters for genomic prediction require that the options of each hyperparameter are either already discrete or discretized before the optimization process (Bellot et al., 2018).

An alternative to genetic algorithms is differential evolution (DE) which is a population based evolutionary heuristic well suited for optimization of discrete and continuous search spaces (Das et al., 2016; Storn and Price, 1997). Differential Evolution lies on the intersection between real-valued genetic algorithms and evolution strategies. DE uses the conventional population structure of genetic algorithms and the self-adapting mutation of evolution strategies; in a sense DE can be loosely viewed as a population based simulated annealing algorithm in which the mutation rate decreases as the population converges on a solution.

In this study, we propose to adapt DE to optimize the DL hyperparameter set for genomic prediction and evaluate its effectiveness to improve prediction accuracies in simulated and real datasets for two classes of DL models: multilayer perceptron (MLP) and convolutional neural network (CNN). We emphasize that the focus of this paper is on optimization of DL hyperparameters to identify a set suitable for a given specific genomic prediction problem, rather than a comparison of DL with GBLUP or other genomic prediction methods. As the predictive performance depends on the architecture of the trait and the population structure, we demonstrate the importance and the impact of proper hyperparameter specification on genomic prediction with DL.

3. MATERIAL AND METHODS

3.1 Datasets

3.1.1 Simulated datasets

Real genotypes from two livestock populations – pigs and cattle – were used to create simulated datasets for testing purposes. Genotypes from both species were edited to be of the same dimensions, comprising a total of 48,541 SNP genotypes for 7,359 individuals, from which 6,031 (80%) and 1,508 (20%) were randomly assigned to the discovery and validation populations, respectively. Phenotypes were simulated for both species by randomly assigning 1,000 SNP as quantitative trait loci (QTL) with additive effects for a heritability of 0.4 using the R simulation package *GenEval* (Cuyabano, 2020).

3.1.2 Real dataset

The real data came from an experimental F2 cross of Duroc and Pietrain pigs already previously described (Edwards et al., 2008). Briefly, four Duroc sires were mated to 15 Pietrain dams to produce 56 F1 individuals (50 females and 6 males). F1 animals were mated to produce a total of 954 F2 pigs that were phenotyped for 38 meat quality and carcass quality traits. For this study pH meat records measured 24 hours post-mortem from 910 F2 pigs were used. We purposely selected this trait as it is moderately heritable (h^2=0.19±0.05) and for which we have mapped putative QTL (Casiró et al., 2017). Two different SNP chips were used to genotype the F2 pigs, but all SNP were imputed to a common set of approximately 62,000 SNP (Gualdrón Duarte et al., 2013) with high accuracy (R2>0.97). SNP were pruned by filtering out SNP with: 1) low genotyping rates (less than 90%), 2) lack of segregation, 3) inconsistent Mendelian inheritance with the pedigree information, 4) low imputation accuracy (R2<0.64; Casiró *et al.*

2017), 5) and high correlation between markers (larger than 0.99). A final set of 28,916 SNP was used for this study.

Phenotypic records were pre-adjusted for fixed effects:

$$y_{adj} = y_{obs} - X\beta,$$

where y_{adj} is adjusted response, y_{obs} is pH measured 24 hours post-mortem, X is the incidence matrix with the fixed effects of sex, slaughter group and carcass weight, and β represents the coefficients of fixed effects.

3.2 Deep learning and genomic prediction

Deep learning (DL) methods are a set of representation learning methods, where a machine can be fed with raw data and automatically discover the representations needed for prediction or classification, with multiple levels of simple but non-linear modules that transform the representation at one level into a representation at a higher, slightly more abstract level (Lecun et al., 2015). In the context of genomic prediction, we used DL to build a system that predicts an animal's phenotypic value given its genotype. DL computes and minimizes a loss function that measures the error of prediction. In this study, we used mean squared error (mse) as the loss function:

$$mse = \frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{N},$$

where N represents the number of individuals in the training dataset, y_i represents the observed response of individual i and $\hat{y}_i$ is the predicted response of individual i. Two types of DL models were used in this study: multilayer perceptron and convolutional neural network.

3.2.1 Multilayer perceptron

This model is also known as feed-forward artificial neural network. In this paper, MLP (Figure 2.1) has: an input layer with as many nodes as SNP markers, a variable number of hidden

layer(s) with a certain number of nodes, and an output layer representing the response. Since nodes between layers are fully connected, MLP can potentially model complex and higher order interactions between predictor variables (Abdollahi-Arpanahi et al., 2020). A detailed explanation of how MLP models work is presented in the File S2.1 and also available at GitHub alongside source code (https://github.com/jun-jieh/DE_DL).

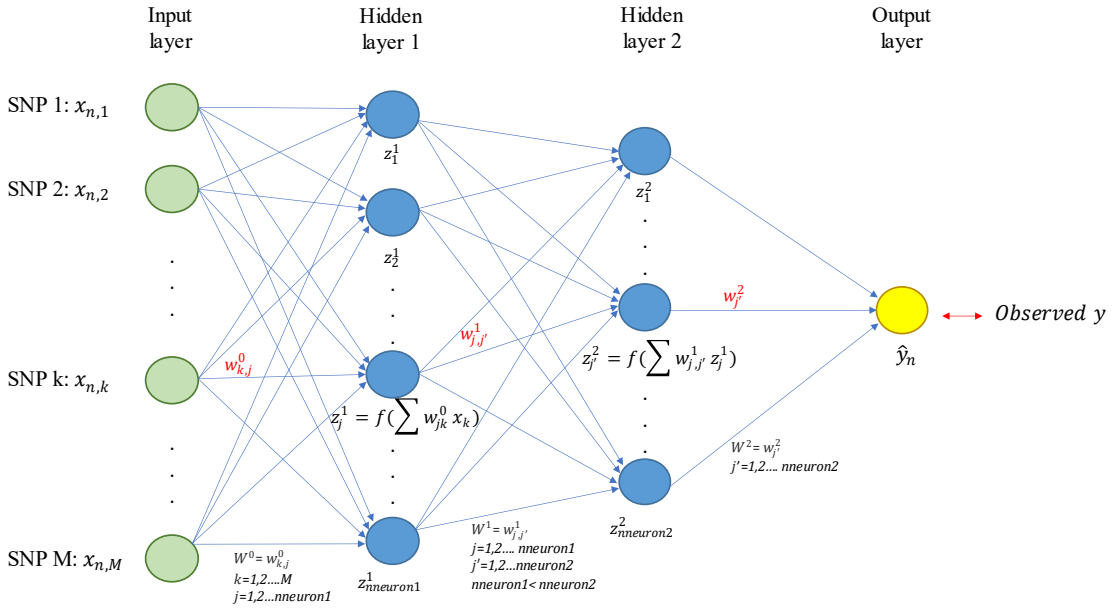

Figure 2.1 Multilayer Perceptron (MLP) for genomic prediction of a single trait with M SNP markers. The network has an input layer, two fully connected hidden layers and an output layer. Each node's input in the hidden layers is a transformation of the weighted sum of the output from the previous layer. The number of nodes in hidden layers decrease as the depth of the MLP increases, to facilitate representation learning.

As deep learning consists of transforming representations at a previous layer into its next (more abstract) layer (Lecun et al., 2015), we opted to adaptively set the number of nodes for each hidden layer based on the depth of the network so that the next hidden layer always has fewer nodes than the previous one. For instance, in an MLP with two hidden layers (Figure 2.1), the first layer can only have neurons ranging from 259 to 512 while the second layer can have any number between 4 and 258. Table S2.1 summarizes the number of nodes search space for

MLPs with one, two, and up to five layers. Other researchers may choose different adaptive rules to impose restrictions on the possible number of neurons per layer, or may even simply choose to use the same number of nodes for all layers.

3.2.2 Convolutional neural network

CNN is designed to process data that comes as multiple-array format (Lecun et al., 2015) e.g. 1d for an animal's genotype, 2d for images and 3d for videos. Typical CNN models consist of an input layer, convolutional layer(s), pooling layer(s), a flattened layer, and an output layer (Figure 2.2). In the context of genomic prediction (Figure 2.2), the input layer for a single observation in a CNN is a one-dimension array that contains an animal's genotype and the number of units in the layer will be equal to the number of markers. The output layer $\hat{y}_n$ represents the predicted response value for the phenotype or breeding value of the n^{th} individual.

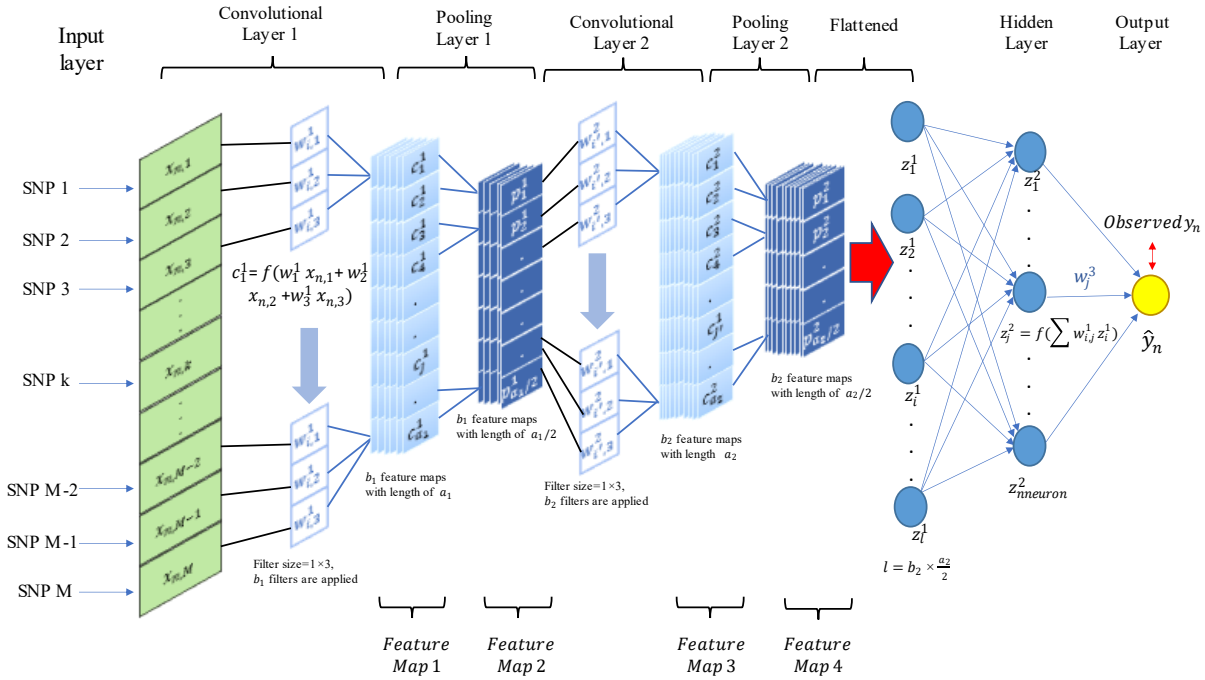

Figure 2.2 1-d Convolutional neural network (CNN) for genomic prediction of a single trait with M SNP markers. The network has an input layer, two convolutional layers with their corresponding pooling layers, a fully connected hidden layer and an output layer. Each convolutional layer applies a number of filters to the output of the previous layer and its output is subsequently summarized by a pooling layer.

Between the input and output layers, a CNN contains a variable number of convolutional layer(s) followed by pooling layer(s). Full details on CNN architecture are given in the File S2.1 and at GitHub along with source code (https://github.com/jun-jieh/DE_DL). In this study each convolutional layer applied filters of size f (a hyperparameter to be optimized) with the stride equal to the filter size (non-overlapping convolutions of the input). In CNN, several restrictions are typically assumed regarding the model architecture. When learning from a global level to a local level, more details are required to obtain the pattern at the local level (Lecun et al., 2015). Therefore, the number of filters increases as the depth of the CNN increases, to detect local motifs. To reflect this expectation we adaptively set the number of filters applied in each convolutional layer as a function on the depth of the network. Specifically, we limited the number of filters in any convolutional layer to be between 4 and 128, but this range is partitioned for each convolutional layer to make sure that the next convolutional layer will always have a number of filters larger than the previous layer. For example, in a CNN with two convolutional layers (Figure 2.2), the first convolutional layer can only have between 4 and 65 filters while the second convolutional layer can have between 66 to 128 filters. Examples of the adaptive number of filters as a function of the depth of the CNN is presented in Table S2.2. The hyperparameter space for filter size was set as an integer between 2 and 20. Although the filter size is specified by the user, the output feature (feature map in Figure 2.2) has to conform to the minimum length (the length of feature map needs to be equal to or larger than the filter size) of the feature maps in each convolutional layer and pooling layer, which is illustrated in Table S2.3. If the condition is not satisfied, instead of fixing the kernel size through all convolutional layers, we set adaptive kernel size in order to successfully execute the model fitting (see details in Figure S2.1). The adaptive kernel size is to ensure that CNN generates a valid output.

3.2.3 DL model training

TensorFlow (Abadi et al., 2015) was used to train DL models. At each iteration (epoch; a detailed description of epoch is presented in File S2.1) of the training process *TensorFlow* randomly partitioned the training data into an actual training set (80% of data), that was used for updating the model weights and a testing set (20% of data), that was used to evaluate the updated model. The data partition was performed by *TensorFlow* and we did not have control over the random partitions. At the end of each epoch, an internal validation was performed by evaluating the correlation between predicted and observed response in the testing set. A DL training procedure typically requires multiple epochs, which was one of the hyperparameters optimized in our DE procedure (see below). We also introduced an early stopping when the correlation did not change over 0.1 for ten consecutive epochs, as it was assumed that fitness (correlation) could not be improved and further exploration was unnecessary.

3.2.4 Hyperparameter optimization

Table 2.1 presents the hyperparameters optimized in this study. A plausible range of values for each hyperparameter was defined based on ranges suggested by the literature for DL applied to genomic prediction (additional details of each hyperparameter can be found in the File S2.1). These ranges were then used as constraints for the differential evolution algorithm. It is important to indicate that users can accordingly extend/reduce/modify the hyperparameter space described in Table 2.1.

Table 2.1 Parameter space for optimized hyperparameters. Hyperparameter space and range (see details in File S2.1). N represents sample size. α_1=0.001 for the simulated datasets and α_1=0.01 for the real pig dataset. α_2=0.01 for the simulated datasets and α_2=0.1 for the real pig dataset.

Hyperparameters	**Parameter space (MLP)**	**Parameter space (CNN)**	**Value Type**
Number of layers	[1,2,3,4,5]	[1,2,3,4,5]	Integer
Number of neurons	[8-512]	[8-512]	Integer
Activation	['relu', 'elu', 'sigmoid', 'selu', 'softplus', 'linear', 'tanh']	['relu', 'elu', 'sigmoid', 'selu', 'softplus', 'linear', 'tanh']	Categorical
Optimizer	['sgd', 'adam', 'adagrad', 'rmsprop', 'adadelta', 'adamax', 'nadam']	['sgd', 'adam', 'adagrad', 'rmsprop', 'adadelta', 'adamax', 'nadam']	Categorical
Dropout rate	[0-1]	[0-1]	Continuous
L2 penalty	[0-1]	[0-1]	Continuous
Batch size	[N×α_1-N×α_2]	32	Integer
Epoch	[21-50]	[21-50]	Integer
Number of filters	NA	[2-128]	Integer
Filter size	NA	[2-20]	Integer
Pooling	NA	['max', 'average']	Categorical

3.3 Differential evolution algorithm for deep learning

Differential evolution (DE) is an evolutionary algorithm that includes four steps: 1) initialization, 2) mutation, 3) crossover and 4) selection (Storn and Price, 1997). A generic version of this algorithm is described in pseudocode format (Figure 2.3). DE was used to evolve a population of numeric vectors that can be recoded to represent hyperparameter combinations through random keys. A toy example of the DE approach is provided at GitHub (https://github.com/jun-jieh/DE_DL).

```
d ← population size
α ← crossover chance
μ ← mutation chance
δ ← number of generations
pop ← generate d solutions
for i = 1 to d do
    for p in pop do
        p_f ← calculate fitness of p
    end for
    pop' ← randomly select four p ∈ pop
    pop'_f[1] ←calculate fitness of first selected solution
    pop'_m                                   ▷ solutions to be mutated
    pop'_c                                   ▷ solutions to be crossovered
    for j = 2 to 4 do
        r = random real between [0,1]
        if r ≤ μ then
            pop'_m ⌢ pop'[j]                 ▷ ⌢ appends an element to population
        end if
        if r ≤ α then
            pop'_c ⌢ pop'[j]
        end if
    end for
    pop' ← operators applied to pop'[2, 3, 4], pop'[1] is unchanged
    p'_f ← calculate fitness of p'
    if p'_f > p_f then p = p'
    end if
end for
return pop
```

Figure 2.3 Pseudocode for differential evolution algorithm

3.3.1 Random key

Random key is an encoding mechanism originally used in genetic algorithms by Bean (1994). The core of this algorithm is a set of *d H*-dimensional numeric vectors $\boldsymbol{pop_1}$, …, $\boldsymbol{pop_d}$ as a population. Each numeric vector represents a solution that is linked or mapped to a set of model hyperparameters through a mapping function (random key). Suppose that there are *K* hyperparameters to optimize, where *K*=8 for the MLP and *K*=10 for the CNN (Table 2.1). Within each hyperparameter *k*=1…*K* there are H_k loci, and if the parameter takes continuous values, then H_k =1. So, the size $H = \sum_{k=1}^{K} H_k$. Each vector $\boldsymbol{pop_i}$ is partitioned into *K* sub-blocks that

contain H_k loci, where each single locus in H represents a hyperparameter option or value. For categorical hyperparameters, there is a mapping performed from the H_k dimensional block of the numeric vector to an H_k dimensional vector MAP_k containing the names of the categories for the k^{th} hyperparameter as follows: the H_k elements are ranked according to their values and the rank of the first element is used as an index for the MAP_k vector to select the corresponding categorical value. In this way, the evolutionary operators (mutation, crossover, and selection) can be applied directly on the numerical vector pop_i but the results can always still be translated into a set of categorical (and continuous) hyperparameter values. An example of this with the hyperparameter number of layers (H_k =5) is presented in Figure 2.4. So, in a nutshell, a random key is a vector of real numbers that, once sorted, its ranking can be used to map against a set of statically ordered features. The idea is that better features will evolve to higher values in the key while worse features will evolve to lower values; the ranking of the sorted key allows sorting the features from best to worst and provides a smoother fitness surface for the DE to explore.

The main steps for the differential evolution algorithm are:

3.3.2 Initialization

We initialized d=50 H-dimensional parameter vectors $pop_1 \ldots pop_{50}$ as a population pop (line 5 Figure 2.3) from a uniform [0,1] distribution and we mapped the numeric vector to a set of hyper-parameter values as described before (Table 2.1) to obtain 50 hyperparameter sets. Then we fitted 50 models using each set of hyperparameters and recorded their correlations between predicted and observed response values. An individual of the population refers to one of the H-dimensional vectors in pop and its encoded hyperparameter set. From now on, we use the term individual to refer to a hyperparameter solution in DE.

3.3.3 Mutation

To generate a mutation, indices of two random individuals are selected from the population: $r_1, r_2, \in \{1,2, \dots, d\}$ and the target H-dimensional vectors pop_{r_1} and pop_{r_2} are extracted. Then vectors p and pop_{r_2} are mutated using

$$mu = \mu \cdot (pop_{r_1} - pop_{r_2})$$

where mu is the mutant vector and μ is the mutation parameter ($\mu \in [0,1]$). Storn and Price (1997) recommended that 0.5 is usually a good initial choice for the mutation. In this study, we set $\mu = 0.5$.

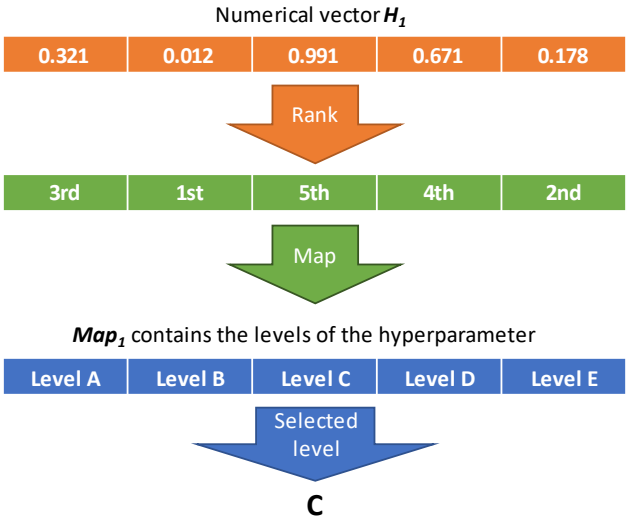

Figure 2.4 Summary of the random key (mapping function) used to transform numeric vectors into discrete levels of hyperparameters. The numeric vector can be subject to mutation and recombination. The mapping is used to transform the result into a meaningful set of hyperparameters that can be used to fit a model and obtain a fitness to select numeric vectors.

3.3.4 Crossover

To increase the diversity in hyperparameter combinations represented in the population parameters, crossover function is used to combine the mutant vector mu with other individual vectors. First, an H-dimensional vector *RN* with uniformly random numbers $\in [0,1]$ is generated. The crossover rate is defined by parameter α ($\alpha \in [0,1]$). Gämperle et al. (2002) suggested that a good choice for the crossover constant is a value between 0.3 and 0.9. In this study, we set $\alpha = 0.5$. Another H-dimensional vector (CR**)** with logical variables (True/False or 1/0) is then generated according to

$$CR_i = \begin{cases} 1 & if\ RN_i < \alpha \\ 0 & if\ RN_i \geq \alpha \end{cases},\ i = 1,2,\dots,H.$$

Then, two more individual vectors pop_{r_3} and pop_{r_4} are selected and crossover generates a new individual according to

$$Challenger_i = \begin{cases} pop_{r_3,i} & if\ CR_i = 0 \\ pop_{r_4,i} - mu_i & if\ CR_i = 1 \end{cases}, i = 1,2,\dots,H,$$

where $Challenger$ is the newly generated individual and i is the i^{th} element of pop_{r_3}, pop_{r_4}, mu, and $Challenger$. For convenience, we name pop_{r_3} $Titleholder$.

3.3.5 Selection

To decide whether or not the $Challenger$ should replace the $Titleholder$ in population pop, both vectors of numeric values $Challenger$ and $Titleholder$ were mapped into hyperparameter sets using the random key. The models were fitted based on mapped hyperparameters and Pearson correlation coefficient γ between the predicted and the observed values was computed. We averaged the correlations over epochs, as the testing sets varied in epochs. The averaged correlation was defined as the fitness of the DL model (given a hyperparameter set). Additionally, we applied a penalized fitness if any of the following three

scenarios happened during model fitting: exhausted memory when fitting a specified model, a constant generated for all predicted responses, and exploding/vanishing gradient which led to an unstable model-fitting procedure (convergence issue). A penalized individual had its fitness set to -1. If $\gamma_{Titleholder} < \gamma_{Challenger}$, we replaced $Titleholder$ by $Challenger$ in pop; otherwise, we retained $Titleholder$ in pop. Finally, steps 2)- 4) are repeated for δ iterations. For the simulated pig dataset and the simulated cattle dataset, δ of both datasets was 2,000, while $\delta = 10{,}000$ was used for the real pig dataset (after δ iterations there was no further significant improvement). It is worth noting that the initial population does not need to be random, it can be based on prior information or can even be the results from a previous run –in effect, the DE can continue evolving a population that has already optimized for some iterations if e.g. the run did not converge.

As shown in the results section, if a DL model is run multiple times with the same dataset and hyperparameters, the predictive performance differs slightly from run to run. This means that a model trained once can get a slightly higher/lower prediction accuracy compared to the average prediction accuracy that would be obtained over multiple re-trainings. This effect is more pronounced in more complex models, which are more prone overfitting. To mitigate this problem, we introduced a variation to the traditional DE algorithm by refitting the $Titleholder$ each time and updating its fitness value. Specifically, in each iteration, the $Titleholder$ was refitted, and if the $Titleholder$ won the contest, the updated fitness was retained.

3.3.6 Top model selection

At the end of the DE run, each individual solution in the population was refitted 30 times to select the best model based on two criteria to evaluate model stability through repeated training of each DL model. The best model was selected based on two measures obtained from

this repeated training: mean fitness and standard deviation (SD) of the fitness obtained by refitting each model 30 times. This is necessary because as explained above, the refitting of the selected models resulted in slightly different predictive performance. The details on how this bi-variate criteria selection was performed can be found in GitHub (https://github.com/jun-jieh/DE_DL).

3.4 Optimized model assessment through external validation

Each dataset was partitioned into five training sets and five validation sets (80% and 20% for training and validation, respectively). The DE was applied to each of the training sets to optimize hyperparameter sets for both, MLPs and CNNs. Note that the training data used for optimization was not part of the validation set. The final MLP and CNN models (2x5) from the DE runs were then refitted 30 times (with the training sets only), and each refitted model was evaluated by predicting the corresponding validation set and computing the correlation between the predicted and the observed response. The average correlation, also known as external validation, of the 30 refits as well as the SD of correlations was then calculated. This external validation is distinct from the internal validation (described in the DL model training) utilized by the DE to optimize the fitness and should be differentiated. In short, the external validation was performed using validation sets while the internal validation was performed utilizing testing sets (described in DL model training). GBLUP was used to estimate the response variable and its prediction accuracy as a comparison reference to the optimized MLPs and CNNs (GBLUP details can be found in File S2.1).

3.5 Hardware and software

The computer processor used in this study was Intel(R) Core i7-8750H CPU @ 2.20 GHz with 16GB of RAM memory and Microsoft(R) Windows 10 operating system. The GPU

(graphic card) was NVIDIA(R) GeForce GTX 1070 with 8 GB GDDR5 memory. All the analyses were implemented in R (R Core Team, 2020). For GBLUP we used the *gwaR* R package (Steibel, 2015) and for DL the R *Keras* package (Chollet et al., 2017), which is a high-level neural networks API on top of *TensorFlow* (Abadi et al., 2015) with GPU computing enabled.

3.6 Data availability

The authors state that all data necessary for confirming the conclusions presented in the article are represented fully within the article. Animal protocols were approved by the Michigan State University All University Committee on Animal Use and Care (AUF# 09/03-114-00). Custom R code used to fit MLP/CNN, implement DE, and evaluate models are available at GitHub (https://github.com/jun-jieh/DE_DL). Genotypes and phenotypes for the animals in the real pig dataset are available at GitHub (https://github.com/jun-jieh/RealPigData). Supplemental material contains the constraints for DL architecture, a summary of predictive performance for GBLUP/MLP/CNN models, the distributions of selected hyperparameters, and the architectures of DL models derived from other studies.

4. RESULTS AND DISCUSSION

As reported in many published applications of deep learning in genomic predictions (Abdollahi-Arpanahi et al., 2020; Bellot et al., 2018; Zingaretti et al., 2020), we observed that retraining of a certain DL model with the same hyperparameter configuration and the same dataset produced slightly different predictions. This forced us to consider the variation in the predictive performance under the retraining in DE and post-DE model selection (see methods section). It also had an impact in the results presented below.

4.1 Optimization runtime profiles

The DE's optimization runtime profiles (mean fitness and SD of fitness) for the three datasets (simulated cattle, simulated pig, and real pig) and the two DL models (MLP and CNN) are shown in Figures 2.5-2.7 and Table 2.2. The mean fitness increased during the DE run, but it is important to note that it can – and did – also decrease at some points due to the stochastic sampling of individual subsets that we used for model testing to avoid overfitting (panels A and C of Figures 2.5-2.7). A similar short-term decrease in fitness was observed when using DE to optimize model hyperparameters in the context of emotion recognition (Nakisa et al., 2018). In our case, the occasional drop in the mean fitness was due to the retraining of the models as the refitting of the same model could yield a lower fitness. Thus, sometimes, even if a current $Titleholder$ won the challenge, its new fitness could be lower than before due to the re-fitting. Alternatively, when a new *Challenger* won the contest, its fitness could have been higher that the refitted fitness of the *Titleholder* but still lower than the previously estimated fitness for that *Titleholder,* which resulted in a new candidate solution in the population but also in a lower fitness.

Table 2.2 Runtime profile for the DE approach. MLP, multilayer perceptron; CNN, convolutional neural network; Avg. runtime, average runtime for one DE iteration (each iteration fits two models); Num. iterations, the total number of iterations used in DE; min, minutes; hr, hours.

Model type	Dataset	Avg. runtime	Num. iterations	Total runtime
MLP	Simulated Pig	3.95 min	2,000	131.78 hr
	Simulated Cattle	4.01 min	2,000	133.67 hr
	Real Pig	0.30 min	10,000	49.81 hr
CNN	Simulated Pig	2.36 min	2,000	77.67 hr
	Simulated Cattle	2.73 min	2,000	87.73 hr
	Real Pig	0.24 min	10,000	40.15 hr

In general, DE for CNNs converged faster (reached the maximum possible average fitness) compared to DE for MLPs (panels A/C of Figures 2.5-2.7). For CNNs, DE converged after approximately 600, 700, and 1,500 iterations for the simulated pig, simulated cattle, and real pig datasets, respectively. For MLPs, DE converged in approximately 1,000, 1,000, and 2,500 iterations for the three datasets, respectively. One possible explanation is that, MLP disregards spatial information and use each neuron as an independent predictor, while CNN tends to learn from a global pattern at the beginning and then summarize the features into a local level (Lecun et al., 2015). In genomic data, linkage disequilibrium is a nonrandom relationship of alleles at different physical locations, which is a sensitive indicator that structures a genome (Slatkin, 2008). Also, Tang and Sun (2019) argued that CNN could be utilized to extract motifs from homologous sequences, where motifs are essential features for distinguishing different sequence families. Given a dataset with spatial structure, CNN potentially has advantage over MLP that CNN can deal with local connectivity.

For each dataset, evolved MLPs and CNNs converged to similar mean fitness but varied across data partitions (panels A/C of Figures 2.5-2.7). The mean fitness in the simulated pig dataset ranged from 0.27 to 0.31 and the range was (0.31, 0.33) for the simulated cattle dataset, while the real pig dataset had a range of (0.19, 0.29). Mitchell et al. (2015) trained networks with permuted datasets and also reported varying predictive performance given different data partitions.

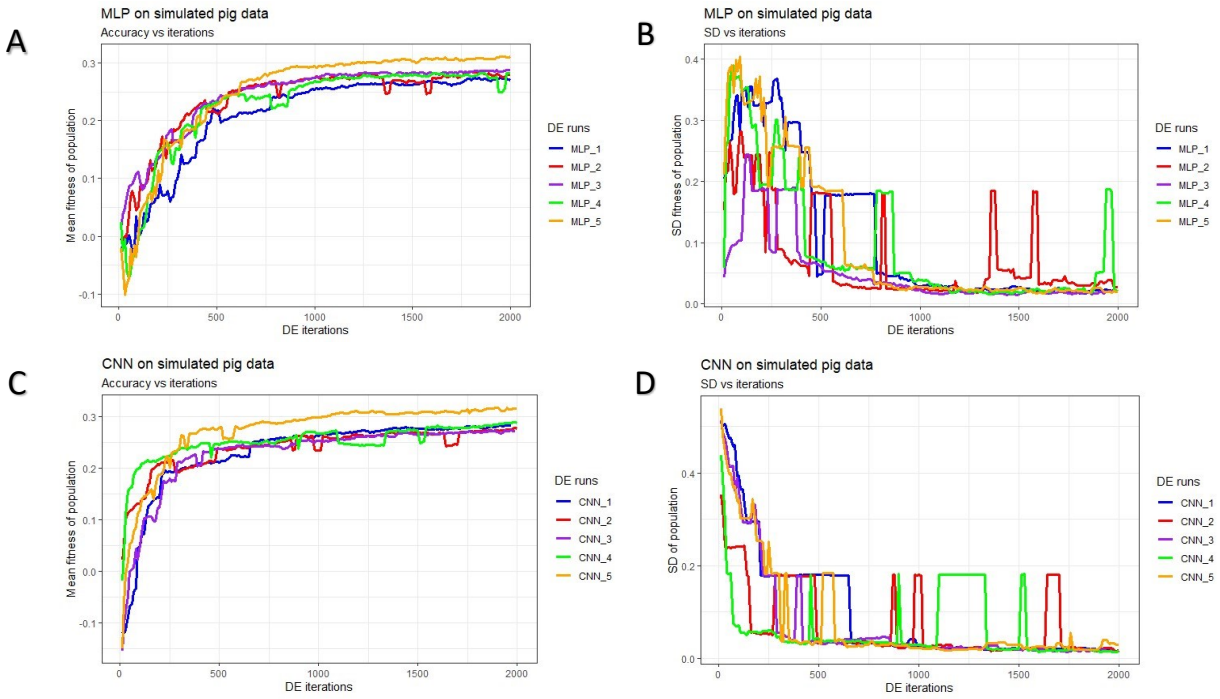

Figure 2.5 History of differential evolution by algorithm and data partition in the simulated pig dataset over 2,000 iterations. Mean and standard deviation of the fitness (correlation between the predicted and true phenotype) were computed given each population. (A) Mean fitness of five populations by fitting multilayer perceptron (MLP) models. (B) Standard deviation of fitness within each population (MLPs). (C) Mean fitness of five populations by fitting convolutional neural network (CNN) models. (D) Standard deviation of fitness within each population (CNNs).

Most evolved populations had a fitness SD smaller than 0.05. However, one exception was population 5 with CNNs in the simulated cattle dataset (panel D of Figure 2.6). Only this population had a large SD of 0.19 and the population contained a CNN hyperparameter set with penalized fitness, indicating its failure to remove a penalized individual in 2,000 iterations. DE performance is sensitive to the number of iterations set by the user and generally solutions can evolve further when the iteration number is increased (Gämperle et al., 2002; Kok and Rajendran, 2016). Thus, the solution with penalized fitness should be removed by introducing more DE iterations. On the other hand, the post-DE refitting (described in top model selection) would further exclude this solution. Overall, the within population SDs for both MLP and CNN

models were reduced over DE iterations (panels B/D of Figures 2.5-2.7), suggesting evolved models in each population had similar performance. Kim and Lee (2019) reported that deep learning models with different hyperparameters could have the same predictive performance, which indicated that the best solution may not be unique. Zhang et al. (2020) also indicated that superior solutions would prefer the closest candidates in evolutionary optimization algorithms. Therefore, we argue that DE evolves a population to where candidate solutions are increasingly similar to each other. Furthermore, distributions of evolved models showed similarities in hyperparameter options e.g. activation function, number of layers, filter size, optimizer, dropout, and pooling, while the hyperparameters were less similar in number of nodes (filters), fully connected layer in CNN, batch size, and L2 regularization (Tables S2.4-S2.15). Yu and Zhu (2020) have mentioned that in the process of optimization, hyperparameters with greater importance received preferential treatment, whereas it was difficult to quantitatively determine the significance of the hyperparameters. We argue that the heterogeneity/homogeneity in the hyperparameters resort to the less/more important hyperparameters.

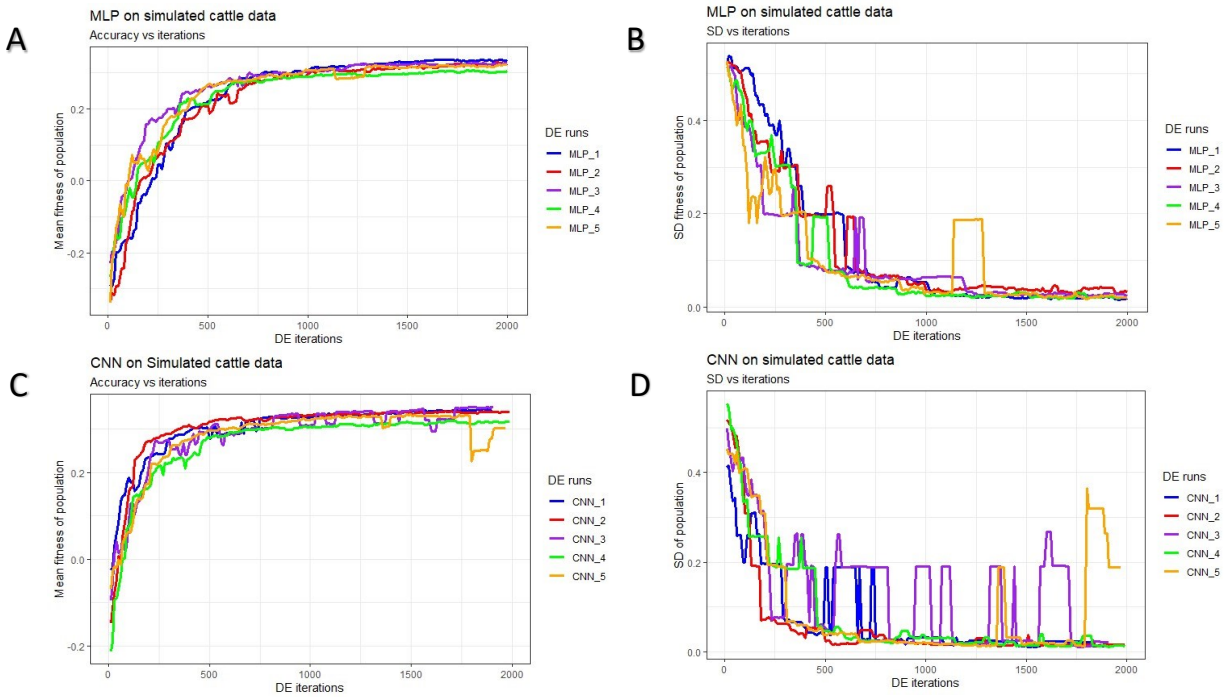

Figure 2.6 History of differential evolution by algorithm and data partition in the simulated cattle dataset over 2,000 iterations. Mean and standard deviation were computed given each population. (A) Mean fitness of five populations by fitting multilayer perceptron (MLP) models. (B) Standard deviation of fitness within each population (MLPs). (C) Mean fitness of five populations by fitting convolutional neural network (CNN). (D) Standard deviation of fitness within each population (CNNs).

Table 2.2 presents the runtime of DE approach in the three datasets. We observed that the models fitted with the real pig dataset were approximately 10 times faster compared to the two simulated datasets as there were fewer SNP loci in the real pig dataset. We also observed that CNNs fitted faster than MLPs. Depending on the dataset, the average runtime for one iteration (two DL models) ranged from 0.30-4.01 minutes for MLPs and 0.24-2.73 minutes for CNNs. As a reference, Montesinos-López *et al.* (2018) reported that training a DL model with their dataset required 3.60 hours. It is important to point out that we utilized the GPU computing that parallelized the computation in DL with thousands of graphic computing units, and this will be the major impact on computational speedup compared to the default setup (CPU computing). For comparison, we fitted GBLUP models with *gwaR* (Steibel, 2015) package, and it required 1.77

hours, 1.90 hours, and 19.08 seconds to fit the simulated pig, simulated cattle, and real pig datasets, respectively. It is important to point out that the runtime increase quadratically with samples and linearly with markers using *gwaR* package. Furthermore, we estimated the time budget for grid searches with GPU computing enabled. An exhaustive search is estimated to cost 9,352,875-404,278,022 hours according to the defined hyperparameter space (Table 2.1) and dataset, which results in up to 4,594,067 times more computing resource compared to DE approach used in this study.

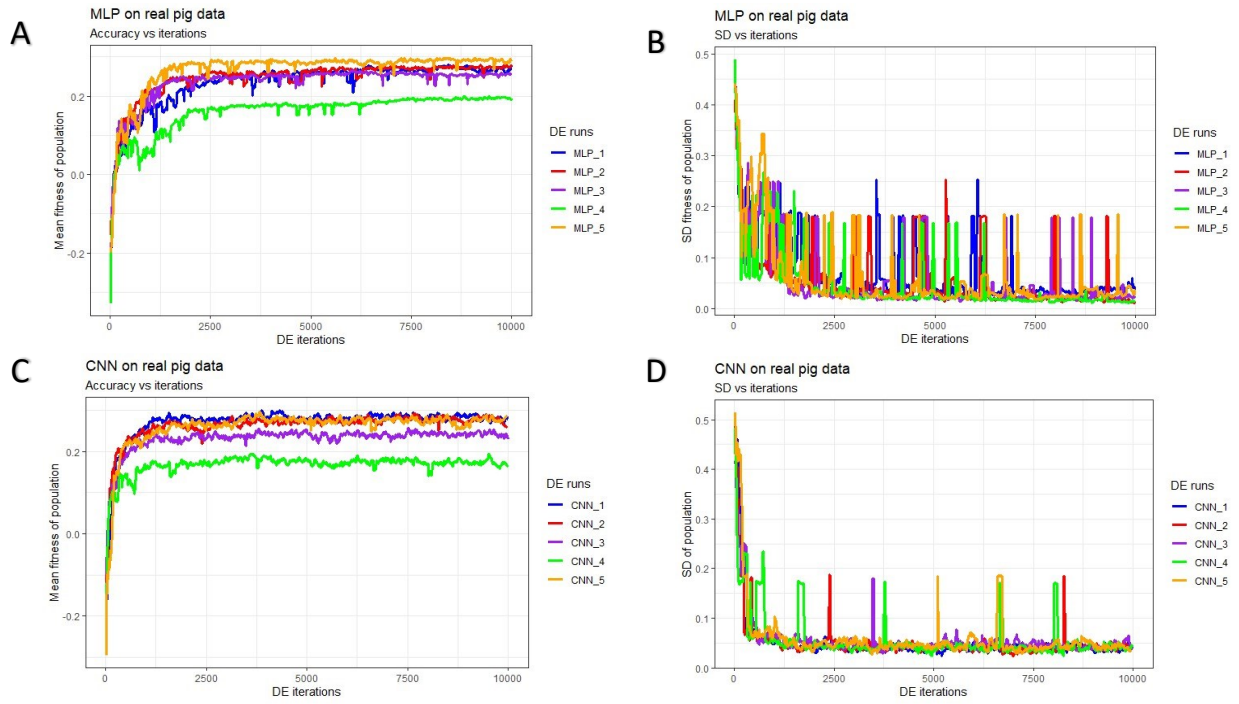

Figure 2.7 History of differential evolution by algorithm and data partition in the real pig dataset over 10,000 iterations. Mean and standard deviation were computed given each population. (A) Mean fitness of five populations by fitting multilayer perceptron (MLP) models. (B) Standard deviation of fitness within each population (MLPs). (C) Mean fitness of five populations by fitting convolutional neural network (CNN) models. (D) Standard deviation of fitness within each population (CNNs).

4.2 Characteristics of selected hyperparameters

Table 2.3 shows the top MLPs from each population (one hyperparameter solution from each population, 15 in total). Activation functions of MLPs optimized for the simulated datasets

varied in “elu”, “selu”, “relu”, “softplus” and “linear”, while in optimized MLPs for the real pig dataset, the “sigmoid” function was fixed across all selected individuals. Noteworthy: In this study, the input of the DL model was the allelic count of one of the alleles (coded as 0, 1 and 2), thus, all the input nodes were non-negative values. Interestingly, “elu”, “selu” and “relu” are almost identical when the input is a non-negative value, and the “linear” activation is very similar to those functions too (differing only in the slope). Moreover, “softplus” and “sigmoid” are the most different activation functions compared to the elu-linear family. The activation functions of the top models are described by Goodfellow et al. (2016). Our finding agrees with Bellot et al. (2018) who suggested “elu”, “softplus” and “linear”, and also “relu” recommended by Pérez-Enciso and Zingaretti (2019). Moreover, as the simulated datasets were generated by only considering additive genetic effects, we speculate that the optimized DL models for the simulated datasets unveiled the additive nature of the trait effect by selecting predominantly linear-like activation functions. For the real dataset, optimized MLPs fixed the non-linear activation function “sigmoid”. We argue that the selected non-linear activation reflects the increased complexity of polygenic inheritance in real datasets. Regarding this perspective, Zingaretti et al. (2020) indicated that in a real dataset, DL could model complex relationships by employing non-linear functions, and they also observed that sigmoid-like hyperbolic tangent (“tanh”) was a safer choice overall. In line with these assumptions, our models for the simulated datasets selected one-layer, two-layer, and three-layer MLPs, while all MLPs for the real pig data were three-layer models. The optimizers of selected MLPs for the simulated datasets focused on “adam” and “adamax”, while for the real dataset “sgd” was further included. Dropout rates of MLPs were between 0 and 0.034 for the simulated datasets and were between 0.182 and 0.617 for the real pig dataset. Compared to the model architectures selected by Bellot et al. (2018) and

Pérez-Enciso and Zingaretti (2019), we had similar hyperparameter options in number of layers and activation function. But we selected different optimizers and the dropout in our case tended to be larger in the real pig dataset. Penalty weights for L2 regularization of MLPs had a range of (0.01, 0.16) for the simulated datasets and a range of (0.03, 0.85) for the real pig dataset. We did not find any suggested L2 weight applied to genomic prediction studies.

Table 2.3 Hyperparameters of selected MLP models from each population. SP, simulated pig dataset; SC, simulated cattle dataset; RP, real pig dataset; DE No., differential evolution of different data partition; No. layer(s), number of hidden layers; No. neurons, number of neurons according to the number of hidden layers.

Dataset	DE No.	activation	No. layer(s)	No. neurons	batch	epoch	optimizer	dropout	L2
SP	1	elu	2	[446,87]	51	37	adam	0.006	0.06
SP	2	elu	2	[412,150]	41	45	adam	0.020	0.16
SP	3	elu	2	[470,155]	46	44	adam	0.015	0.06
SP	4	selu	2	[474,145]	54	45	adam	0.032	0.13
SP	5	softplus	2	[397,87]	54	45	adam	0	0.13
SC	1	elu	3	[429,330,57]	44	28	adam	0.030	0.04
SC	2	relu	2	[411,106]	48	41	adamax	0.002	0.06
SC	3	elu	3	[401,269,93]	11	27	adamax	0.001	0.01
SC	4	relu	1	409	56	21	adam	0.034	0.14
SC	5	relu	1	444	47	33	adam	0.020	0.16
RP	1	sigmoid	3	[374,192,25]	10	40	sgd	0.352	0.85
RP	2	sigmoid	3	[476,193,69]	54	42	adam	0.480	0.52
RP	3	sigmoid	3	[483,291,8]	44	46	adamax	0.182	0.12
RP	4	sigmoid	3	[457,234,79]	31	41	adamax	0.465	0.03
RP	5	sigmoid	3	[386,251,148]	8	40	sgd	0.617	0.75

Table 2.4 shows top CNNs. Optimized CNNs had three options for activation function: “linear”, “elu”, and “relu” (Goodfellow et al., 2016). Similar to our results, top CNNs selected by Bellot et al. (2018) also included “linear” and “elu”, while Pérez-Enciso and Zingaretti (2019) used “relu”. The number of convolutional layers varied from one to three while all CNNs for the simulated datasets fixed with one. Notably, Bellot et al. (2018) also selected one-layer and three-layer CNNs. The filter sizes tended to be larger in the selected models. The large filter sizes were different from other studies that suggested two or three (Bellot et al., 2018; Pérez-Enciso and Zingaretti, 2019). Optimizers of selected CNNs were “adamax”, “rmsprop” and “adam” while

CNNs for the real pig data fixed "adam", and this finding is different from the "nadam" obtained by Pérez-Enciso and Zingaretti (2019). Most CNNs across the three datasets used average pooling for the pooling layer. For genomic prediction studies, we did not find a suggested pooling option in the literature. Dropout rates of CNNs ranged from 0.008 to 0.827 and the range was smaller (0.021,0.277) for the real pig dataset. However, our finding in dropout differed from the small dropout (5-10%) recommended by Pérez-Enciso and Zingaretti (2019). Most L2 penalty weights were smaller than 0.16 while there were three exceptions (0.52, 0.75 and 0.85).

Table 2.4 Hyperparameters of selected CNN models from each population. SP, simulated pig dataset; SC, simulated cattle dataset; RP, real pig dataset; DE No., differential evolution of different data partitions; No. layers, number of convolutional layers; No. filters, number of filters applied based on No. layers; FCL, size (number of neurons) of the fully connected layer after flatten layer.

Data	DE No.	activation	No. layers	No. filters	filter size	epoch	FCL	optimizer	dropout	L2	Pooling
SP	1	linear	1	110	19	25	17	adamax	0.197	0.21	average
SP	2	elu	1	16	15	32	110	rmsprop	0.146	0.03	average
SP	3	elu	1	15	8	44	79	rmsprop	0.692	0.02	average
SP	4	linear	1	59	20	24	49	adamax	0.496	0.23	max
SP	5	linear	1	109	13	27	109	adam	0.827	0.01	average
SC	1	linear	1	116	20	30	16	adam	0.370	0.10	average
SC	2	linear	1	87	12	25	12	adam	0.086	0.13	average
SC	3	linear	1	32	8	42	24	adam	0.250	0.19	average
SC	4	linear	1	79	20	44	27	adamax	0.666	0.06	max
SC	5	linear	1	98	16	40	153	adam	0.151	0.17	average
RP	1	elu	2	[51,113]	18	22	50	adam	0.277	0.67	average
RP	2	relu	3	[24,81,121]	12	27	268	adam	0.067	0.11	average
RP	3	elu	2	[64,112]	13	45	278	adam	0.021	0.87	average
RP	4	relu	3	[44,73,106]	13	47	326	adam	0.008	0.18	average
RP	5	elu	3	[41,71,128]	5	41	238	adam	0.051	0.35	average

Despite our evolved hyperparameter sets being similar to those described in the literature (Bellot et al., 2018; Pérez-Enciso and Zingaretti, 2019), part of hyperparameter configurations e.g. number of nodes (filters), optimizer, and dropout differed from those described in the existing studies. This is likely due to that the optimal hyperparameter configuration depends on the specific genomic dataset, and a hyperparameter's relevance may depend on another

hyperparameter's value (Luo, 2016). As Bellot et al. (2018) worked on a human dataset and Pérez-Enciso and Zingaretti (2019) investigated a wheat dataset, we attribute the variation among optimized hyperparameters to the specific dataset. It is also possible that our extended hyperparameter space searched for more instances, which led to the differences in some hyperparameters compared to other studies. While other researchers optimized hyperparameters by discretizing the parameter space, we regarded number of neurons (filters), dropout, L2 regularization, batch size, epoch, and filter size as continuous values, which considerably expanded the hyperparameters search space.

4.3 Performance of optimized models under validation

The objective of this paper is to provide a framework to optimize DL hyperparameters for genomic prediction and not to compare the optimized DL with GBLUP. It is however still relevant to use GBLUP as a baseline of reference prediction methods to contextualize our results (see Figure S2.2). For the simulated datasets, GBLUP was the slightly better than the rest of the models. A similar result was presented by Abdollahi-Arpanahi et al (2020). This is not surprising in our study because GBLUP (described in File S2.1) is a model well suited for the simulated data which is entirely additive and composed of a large number of very small effects that approximate the infinitesimal model. However, for the real pig dataset, the pattern was somewhat different, and the best performing model was dependent on the data partition. As explained later in this section, we attribute this phenomenon to the small sample size of the real pig dataset. D'souza et al. (2020) argued that for a small dataset (e.g.: N<5000), the presence of substructure or even a few outliers may have a profound influence on the predictive performance under a specific data partition, skewing the overall estimate of the predictive performance and affecting the outcome of any optimization method that is used.

As DL is a methodology that relies on a learning process conditioned on the problem that it is solving (A. Montesinos-López et al., 2018), it is less likely that a DL model can achieve its best possible prediction accuracy using a hyperparameter set optimized from other independent studies. To investigate this, we trained MLPs and CNNs with hyperparameters selected for predicting human traits (Bellot et al., 2018) and for a wheat dataset (Pérez-Enciso and Zingaretti, 2019), across the three datasets in this study. Table S2.16 shows hyperparameters of MLPs and CNNs obtained from the two studies. Figures 2.8-2.10 shows the predictive performance of random DL models, optimized DL models, and top DL models selected by Pérez-Enciso and Zingaretti (2019). These models were applied to all three datasets. Randomly selected DL models and optimized DL models differed in training data partitions due to independent DE optimizations performed within each partition, while the models suggested by the two previous studies (Bellot et al., 2018; Pérez-Enciso and Zingaretti, 2019) were fixed in all partitions. Prediction accuracy of external (cross) validations was obtained by refitting each model 30 times. The panels in Figures 2.8-2.10 represent the predictive performance of competing DL models for each data partition within each dataset. Noteworthy, the optimized models using DE were consistently the best when compared to randomly chosen models or to models taken from the literature, that have been optimized for other datasets.

Models with hyperparameters chosen by Bellot et al. (2018) did not converge in any data partition and so they are not shown in Figures 2.8-2.10. This was likely due to exploding gradients or vanishing gradients (as previously discussed). Another observed case was that the model predicted every individual with the same value, making it impossible to compute the correlation between the predicted and observed response. This also confirmed the observation by Bellot et al. (2018) that convergence problems persisted after reinitializations of the algorithm.

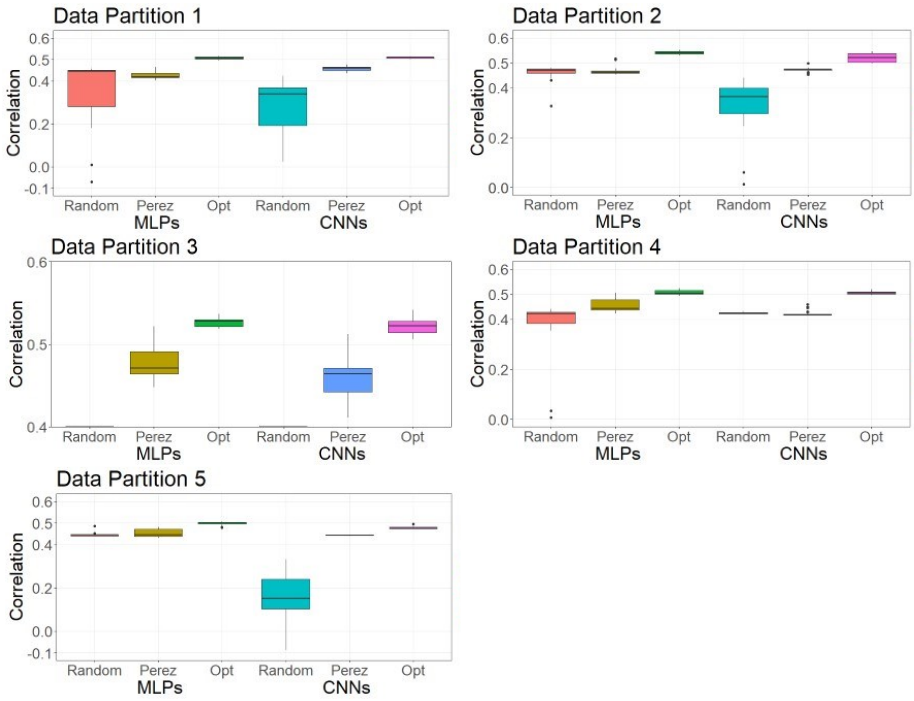

Figure 2.8 Boxplots for the predictive performance of MLPs and CNNs using different hyperparameters (simulated pig dataset). Models were tested on five data partitions of the simulated pig dataset. Statistics represent external (cross) validations by fitting the same model 30 times. The left three boxes are for MLP models and the right three boxes are for CNN models. Null box means the model did not converge. Random, random hyperparameters; Perez, hyperparameters recommended by Pérez-Enciso and Zingaretti (2019); Opt, optimized hyperparameters using DE. Abbreviations stand for the same meaning in Figure 2.9 and Figure 2.10.

For the simulated pig dataset, the MLP and the CNN suggested by Pérez-Enciso and Zingaretti (2019) was slightly worse than the optimized MLPs and CNNs that we obtained with DE. However, their performance was much worse in the simulated cattle and the real pig datasets. Again, the optimal hyperparameter configuration is problem-dependent and thus, it is important to search for the proper hyperparameters in DL genomic prediction applications given a specific dataset.

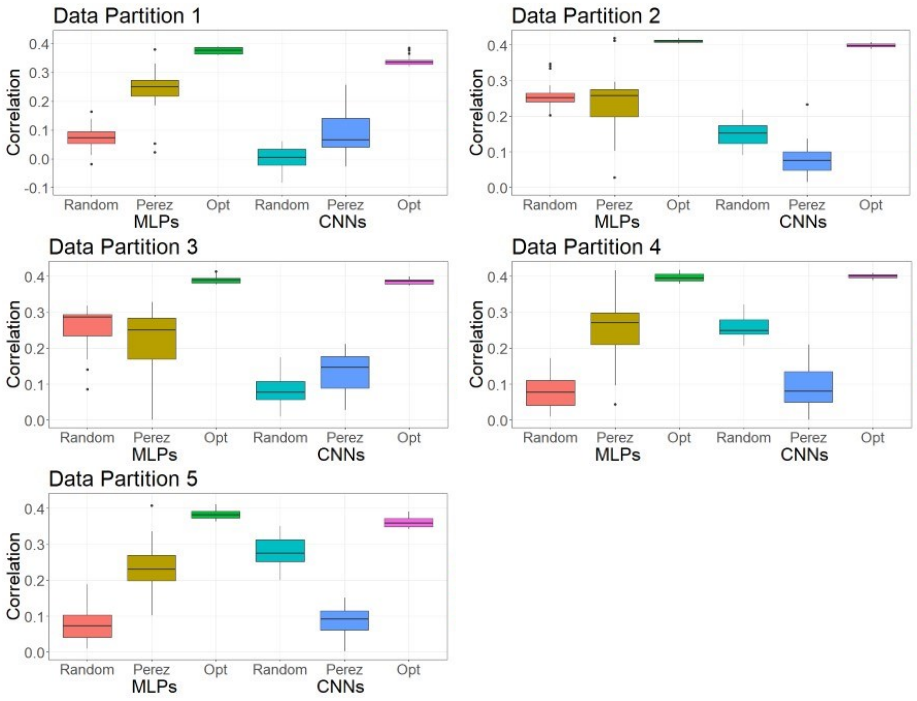

Figure 2.9 Boxplots for the predictive performance of MLPs and CNNs using different hyperparameters (simulated cattle dataset). Models were tested on five data partitions of the simulated cattle dataset. Statistics represent external (cross) validations by fitting the same model 30 times. The left three boxes are for MLP models and the right three boxes are for CNN models.

The variations in the predictive performance under re-training observed in all models indicated that DL models were likely overfitting the data. Abdollahi-Arpanahi et al. (2020) showed variance in predictive performance (in terms of accuracy and mean squared error) of MLPs and CNNs over 10 replicates of cross validation which is in agreement with our results. In general, the SD of the correlation between predicted and observed phenotypes for the optimized MLPs/CNNs and those proposed by Pérez-Enciso and Zingaretti (2019) were smaller in the simulated datasets, while the SD in the real pig dataset was larger (Figure 2.10 compared to Figures 2.8 and 2.9). We speculate that there are two possible reasons for the variation: DL models are initialized with random weights at starting points and a relatively small sample size for training. For the random weights at baseline, Bellot et al. (2018) explained that the

performance of MLPs and CNNs depended on initialization values. For the training sample size, Abdollahi-Arpanahi et al. (2020) indicated that larger sample sizes improved the predictive ability of DL methods. Furthermore, in the field of image classification, Shahinfar et al. (2020) showed increased prediction accuracy and reduced variation in the performance of DL models as the sample size grew. Based on the results in Figures 2.8-2.10, the merit in terms of less variation over replicates of external (cross) validations was clearer in the simulated datasets that had larger sample sizes (N=7,539 for both the simulated pig and the simulated cattle datasets). In the real pig dataset that had a smaller sample size (N=910), SD was larger compared to those in the simulated datasets. Therefore, we argue that both the predictive ability and variation in the same DL models are associated with training sample size. Montesinos-López *et al.* (2018) also mentioned that DL method may fail to learn a proper generalization of the knowledge contained in the data, given small datasets.

5. CONCLUSIONS

Overall, DL can be adapted to perform genomic prediction of complex traits, but it requires some effort to select appropriate hyperparameters. Any hyperparameter optimization will likely be dataset-specific and characteristics such as population structure and genetic architecture of the predicted trait may well require different DL model hyperparameters. In this study, we implemented differential evolution (DE) as a method to simultaneously identify optimal combinations of multiple hyperparameters. Compared to randomly selected models, our optimized MLPs and CNNs showed significant improvement in the predictive performance. In comparison to DL models with hyperparameters selected from other studies, optimized MLPs and CNNs also yielded better predictive accuracy. DE is an efficient and semi-automatic

algorithm that can be used to select an optimal hyperparameter set that leads to a better predictive performance. Moreover, overparameterization of DL can be mitigated by refitting models and selecting those that produce more consistent (less variable) prediction accuracies. We showed that this is more important when working with small datasets.

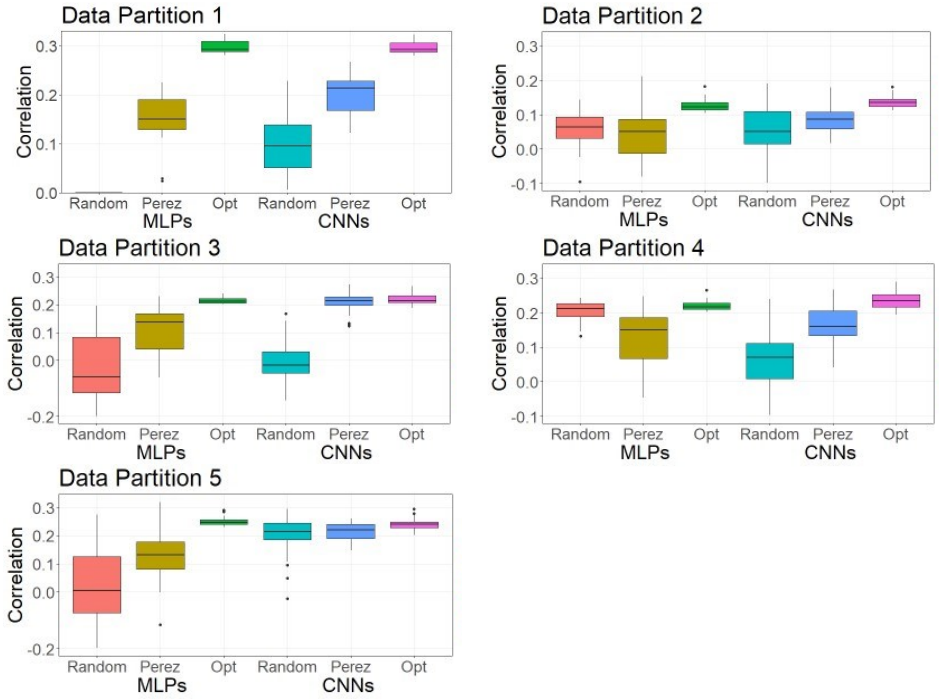

Figure 2.10 Boxplots for the predictive performance of MLPs and CNNs using different hyperparameters (real pig dataset). Models were tested on five data partitions of the real pig dataset. Statistics represent external (cross) validations by fitting the same model 30 times. The left three boxes are for MLP models and the right three boxes are for CNN models. Null box means the model did not converge.

6. ACKNOWLEDGEMENTS

This work was supported by Agriculture and Food Research Initiative Awards No. 2017-67007-26176, No. 2010-65205-20342, the National Institute of Food and Agriculture (AFRI Project No. 2019-67015-29323), and by funding from the National Pork Board Grant No. 11–042. Partial funding was also provided by the US Pig Genome Coordinator.

APPENDICES

Table S2.1 Adaptive hyperparameter space for the number of neurons. Number of neurons (nodes) given the depth of network (number of hidden layers, HL) in multilayer perceptron models.

Layer	One HL	Two HLs	Three HLs	Four HLs	Five HLs
1	[4-512]	[259-512]	[344-512]	[386-512]	[412-512]
2		[4-258]	[175-343]	[259-385]	[311-411]
3			[4-174]	[132-258]	[210-310]
4				[4-131]	[109-209]
5					[4-108]

Table S2.2 Adaptive hyperparameter space for number of filters. Number of filters (kernels) given the depth (number of convolutional layers) of convolutional neural network.

Layer	One layer	Two layers	Three layers	Four layers	Five layers
1	[4-128]	[4-65]	[4-44]	[4-34]	[4-28]
2	--	[66-128]	[45-85]	[35-65]	[29-53]
3	--	--	[86-128]	[66-96]	[54-78]
4	--	--	--	[97-128]	[79-103]
5	--	--	--	--	[104-128]

Table S2.3 Minimum length of feature maps applied to each layer of convolutional neural network. Conv: Convolutional layer.

Layer	One layer	Two layers	Three layers	Four layers	Five layers
Conv 1	4	16	64	256	1024
Pooling 1	2	8	32	128	512
Conv 2	--	4	16	64	256
Pooling 2	--	2	8	32	128
Conv 3	--	--	4	16	64
Pooling 3	--	--	2	8	32
Conv 4	--	--	--	4	16
Pooling 4	--	--	--	2	8
Conv 5	--	--	--	--	4
Pooling 5	--	--	--	--	2

Table S2.4 Distributions of optimized hyperparameters related to multilayer perceptron architectures for simulated pig data. Pop 1-5: MLP solutions to five data partitions (five differential evolution runs).

	Activation function					Number of layers			
	elu	**linear**	**selu**	**relu**	**softplus**	**One**	**Two**	**Three**	**Other**
Pop1	6	38	4	0	2	3	18	29	0
Pop2	7	11	10	9	13	3	37	6	4
Pop3	11	20	10	6	3	8	33	7	2
Pop4	13	20	8	5	4	8	15	23	4
Pop5	11	1	13	15	10	8	30	7	5

Table S2.5 Distributions of optimized hyperparameters related to CNN architectures for simulated pig data. Pop 1-5: CNN solution populations of five differential evolutions runs. Size of fully connected layer: the number of neurons applied in the fully connected layer (after flatten layer). Q0.05, 5% quantile; Q0.95, 95% quantile.

	Activation function				Number of layers				Filter size			Size of fully connected layer		
	elu	**linear**	**selu**	**tanh**	**One**	**Two**	**Three**	**Other**	**Q0.05**	**Median**	**Q0.95**	**Q0.05**	**Median**	**Q0.95**
Pop1	13	18	14	5	19	26	5	0	10	16	19	19	73	380
Pop2	20	14	16	0	31	19	0	0	5	10	20	22	149	477
Pop3	13	19	18	0	8	37	4	1	2	8	20	12	53	452
Pop4	16	27	7	0	35	15	0	0	6	11	20	9	36	308
Pop5	8	26	15	1	32	10	8	0	8	13	20	20	39	481

Table S2.6 Distributions of optimized hyperparameters related to multilayer perceptron architectures for the simulated cattle data. Pop 1-5: MLP solutions to five data partitions (five differential evolution runs).

	Activation function					Number of layers			
	linear	**relu**	**elu**	**selu**	**softplus**	**One**	**Two**	**Three**	**Four**
Pop1	43	0	4	2	1	17	17	13	3
Pop2	30	9	1	4	6	25	10	12	3
Pop3	41	1	6	2	0	6	28	14	2
Pop4	44	2	1	1	2	7	33	10	0
Pop5	49	1	0	0	0	19	20	10	1

Table S2.7 Distributions of optimized hyperparameters related to CNN architectures for simulated cattle data. Pop 1-5: CNN populations of five differential evolutions runs. Size of fully connected layer: the number of neurons applied in the fully connected layer (after flatten layer). Q0.05, 5% quantile; Q0.95, 95% quantile.

	Activation function					Number of layers				Filter size			Size of fully connected layer		
	elu	linear	selu	tanh	other	One	Two	Three	Other	Q0.05	Median	Q0.95	Q0.05	Median	Q0.95
Pop1	3	45	0	2	0	34	10	6	0	8	18	20	16	46	354
Pop2	11	27	5	7	0	8	9	33	0	10	17	20	12	195	386
Pop3	0	49	1	0	0	40	8	2	0	6	15	20	24	26	396
Pop4	13	17	19	0	1	9	9	32	0	10	18	18	27	221	485
Pop5	7	37	2	4	0	12	16	21	1	10	18	20	26	148	416

Table S2.8 Distributions of optimized hyperparameters related to multilayer perceptron architectures for the real pig data. Pop 1-5: MLP solutions to five data partitions (five differential evolution runs).

	Activation function		Number of layers		
	sigmoid	Other	Two	Three	Four
Pop1	47	3	6	44	0
Pop2	50	0	0	46	4
Pop3	50	0	0	44	6
Pop4	50	0	2	38	10
Pop5	50	0	1	46	3

Table S2.9 Distributions of optimized hyperparameters related to CNN architectures for real pig data. Pop 1-5: CNN populations of five differential evolutions runs. Size of fully connected layer: the number of neurons applied in the fully connected layer (after flatten layer). Q0.05, 5% quantile; Q0.95, 95% quantile.

	Activation function				Number of layers				Filter size			Size of fully connected layer		
	elu	linear	tanh	other	Two	Three	Four	Other	Q0.05	Median	Q0.95	Q0.05	Median	Q0.95
Pop1	19	6	25	0	16	24	10	0	12	13	18	22	367	506
Pop2	38	1	5	6	7	30	12	1	12	16	17	50	197	463
Pop3	3	4	40	3	16	26	8	0	8	13	18	60	150	463
Pop4	33	4	1	12	26	11	10	3	9	15	17	50	158	426
Pop5	20	14	16	0	2	25	22	1	5	11	18	12	195	416

Table S2.10 Distributions of optimized hyperparameters related to MLP model compilation and fitting for simulated pig data. Pop 1-5: MLP solution populations of five differential evolution runs. Q0.05, 5% quantile; Q0.95, 95% quantile.

	Optimizer			Epochs			Batch size			Dropout rate			L2		
	adam	adamax	other	Q0.05	median	Q0.95	Q0.05	median	Q0.95	Q0.05	median	Q0.95	Q0.05	median	Q0.95
Pop1	48	0	2	27	35	47	14	30	53	0.03	0.16	0.45	0.03	0.18	0.67
Pop2	43	6	1	23	43	49	12	36	56	0.01	0.05	0.41	0.01	0.16	0.75
Pop3	43	6	1	23	32	48	6	28	52	0.01	0.05	0.57	0.01	0.17	0.69
Pop4	41	8	1	23	36	48	15	40	58	0.01	0.12	0.50	0.02	0.23	0.76
Pop5	35	15	0	22	35	50	19	38	57	0	0.04	0.14	0.01	0.14	0.87

Table S2.11 Distributions of optimized hyperparameters related to CNN model compilation and fitting for simulated pig data. Pop 1-5: CNN solution populations of five differential evolution runs. Q0.05, 5% quantile; Q0.95, 95% quantile.

	Optimizer					Epochs			Dropout rate			L2			Pooling	
	adam	adamax	adadelta	nadam	rmsprop	Q0.05	median	Q0.95	Q0.05	median	Q0.95	Q0.05	median	Q0.95	Max	Average
Pop1	4	11	8	8	19	24	31	48	0.02	0.36	0.79	0.01	0.10	0.49	27	23
Pop2	10	10	20	0	10	21	34	49	0.04	0.42	0.79	<0.01	0.04	0.25	22	28
Pop3	13	13	11	7	6	22	28	44	0.11	0.56	0.77	0.01	0.11	0.49	42	8
Pop4	10	29	5	3	3	23	31	46	0.05	0.39	0.78	0.01	0.12	0.64	22	28
Pop5	22	1	4	10	13	22	31	49	0.02	0.39	0.79	0.01	0.07	0.44	29	21

Table S2.12 Distributions of optimized hyperparameters related to MLP model compilation and fitting for simulated cattle data. Pop 1-5: MLP solution populations of five differential evolution runs. Q0.05, 5% quantile; Q0.95, 95% quantile.

	Optimizer			Epochs			Batch size			Dropout rate			L2		
	adam	adamax	nadam	Q0.05	median	Q0.95	Q0.05	median	Q0.95	Q0.05	median	Q0.95	Q0.05	median	Q0.95
Pop1	47	1	2	23	41	47	11	31	57	0.01	0.15	0.52	0.04	0.27	0.81
Pop2	39	10	1	26	38	48	18	40	57	<0.01	0.09	0.45	0.06	0.26	0.85
Pop3	44	4	2	26	33	44	12	26	52	0.01	0.18	0.53	0.03	0.22	0.87
Pop4	40	0	10	21	39	50	10	23	47	0.03	0.21	0.54	0.03	0.39	0.86
Pop5	41	2	7	22	29	48	9	24	52	0.02	0.26	0.72	0.06	0.27	0.91

Table S2.13 Distributions of optimized hyperparameters related to CNN model compilation and fitting for simulated cattle data. Pop 1-5: CNN solution populations of five differential evolution runs. Q0.05, 5% quantile; Q0.95, 95% quantile.

	Optimizer					Epochs			Dropout rate			L2			Pooling	
	adadelta	adam	adamax	nadam	rmsprop	Q0.05	median	Q0.95	Q0.05	median	Q0.95	Q0.05	median	Q0.95	Max	Average
Pop1	3	28	11	1	7	23	35	50	0.01	0.30	0.65	0.03	0.24	0.72	20	30
Pop2	17	16	7	2	8	22	33	50	0.06	0.28	0.68	0.02	0.29	0.59	10	40
Pop3	3	21	22	0	4	22	40	44	0.04	0.29	0.78	0.02	0.16	0.50	12	38
Pop4	11	3	29	2	5	23	29	46	0.05	0.35	0.67	0.01	0.23	0.75	17	33
Pop5	4	19	5	1	21	21	34	46	0.02	0.24	0.60	<0.01	0.11	0.52	27	23

Table S2.14 Distributions of optimized hyperparameters related to MLP model compilation and fitting for real pig data. Pop 1-5: MLP solution populations of five differential evolution runs. Q0.05, 5% quantile; Q0.95, 95% quantile.

	Optimizer			Epochs			Batch size			Dropout rate			L2		
	adam	adamax	sgd	Q0.05	median	Q0.95	Q0.05	median	Q0.95	Q0.05	median	Q0.95	Q0.05	median	Q0.95
Pop1	1	9	40	25	40	45	7	18	64	0.03	0.38	0.85	0.12	0.66	0.95
Pop2	19	30	1	22	37	45	28	55	64	0.04	0.41	0.82	0.09	0.60	0.92
Pop3	0	47	3	30	44	49	21	44	68	0.05	0.33	0.79	0.05	0.55	0.94
Pop4	2	47	1	31	33	47	30	31	32	0.15	0.56	0.86	0.04	0.37	0.86
Pop5	1	40	9	21	24	41	8	47	63	0.05	0.38	0.84	0.05	0.48	0.90

Table S2.15 Distributions of optimized hyperparameters related to CNN model compilation and fitting for real pig data. Pop 1-5: CNN solution populations of five differential evolution runs. Q0.05, 5% quantile; Q0.95, 95% quantile.

	Optimizer			Epochs			Dropout rate			L2			Pooling	
	adam	adamax	other	Q0.05	median	Q0.95	Q0.05	median	Q0.95	Q0.05	median	Q0.95	Max	Average
Pop1	36	9	5	22	31	41	0.02	0.32	0.71	0.03	0.54	0.96	1	49
Pop2	34	13	3	25	36	50	0.08	0.41	0.78	0.04	0.51	0.97	2	48
Pop3	45	4	1	22	34	45	0.02	0.41	0.88	0.16	0.49	0.93	13	37
Pop4	40	7	3	29	31	49	0.03	0.33	0.75	0.05	0.60	0.98	8	42
Pop5	44	5	1	24	44	46	0.03	0.38	0.77	0.05	0.55	0.95	0	50

Table S2.16 Selected MLP and CNN architecture derived from other studies. No. layers, the number of fully connected layers or convolutional layers; No. neurons (filters), the number of neurons or filters adaptive based on the number of layers. In the No. layers column, 1+1 means one convolutional layer plus one fully connected layer.

Study	Model	Activation	No. layers	No. neurons (filters)	Dropout	Filter size
Bellot et al. (2018)	MLP	elu	1	32	0.0100	NA
Pérez-Enciso and Zingaretti (2019)	MLP	relu	4	[64,64,64,64]	0.0005	NA
Bellot et al. (2018)	CNN	linear	1+1	[16,32]	0.0100	3
Pérez-Enciso and Zingaretti (2019)	CNN	relu	4	[64,64,64,64]	0.0005	3

```
shape ← size of the input layer
fSize ← sampled filter size
nL ← number of Conv (and pooling) layer(s)
n ← 2×nL                                          ▷ number of feature maps
l[n] ← length(s) of the feature map(s) after Conv/pooling layer
min[n] ← minimum lengths of the feature maps
filter[nL] ← fSize                                ▷ an array with nL elements
for i = 1 to nL do
    l[2 × (i − 1) + 1] ← int(shape / (fSize^i × 2^(i−1)))      ▷ for odd indices
    l[2 × i] ← int(shape / (fSize^i × 2^i))                    ▷ for even indices
    min[2 × (i − 1) + 1] ← 2 × 4^(nL−i)
    min[2 × i] ← 4^(nL−i)
end for
flag ← the first index that satisfies l[flag] < min[flag]
if !flag==NA then
    if flag % 2==1 then                           ▷ invalid feature map after Conv layer
        index ← (flag + 1)/2
        filter[index] ← floor(nFMap[flag−2] / min[flag−1])
        if (index+1) ≤ nL then
            for j = (index+1) to nL do
                filter(j) ← 2
            end for
        end if
    else                                          ▷ invalid feature map after pooling layer
        index ← flag/2
        filter[index] ← floor(nFMap[flag−1] / min[flag])
        if (index+1) ≤ nL then
            for j = (index+1) to nL do
                filter(j) ← 2
            end for
        end if
    end if
end if
return filter
```

Figure S2.1 Pseudocode for adaptive filter size. Conv, convolutional layer; int(x), convert x into the nearest integer; floor(x), get the largest integer that is smaller or equal to x.

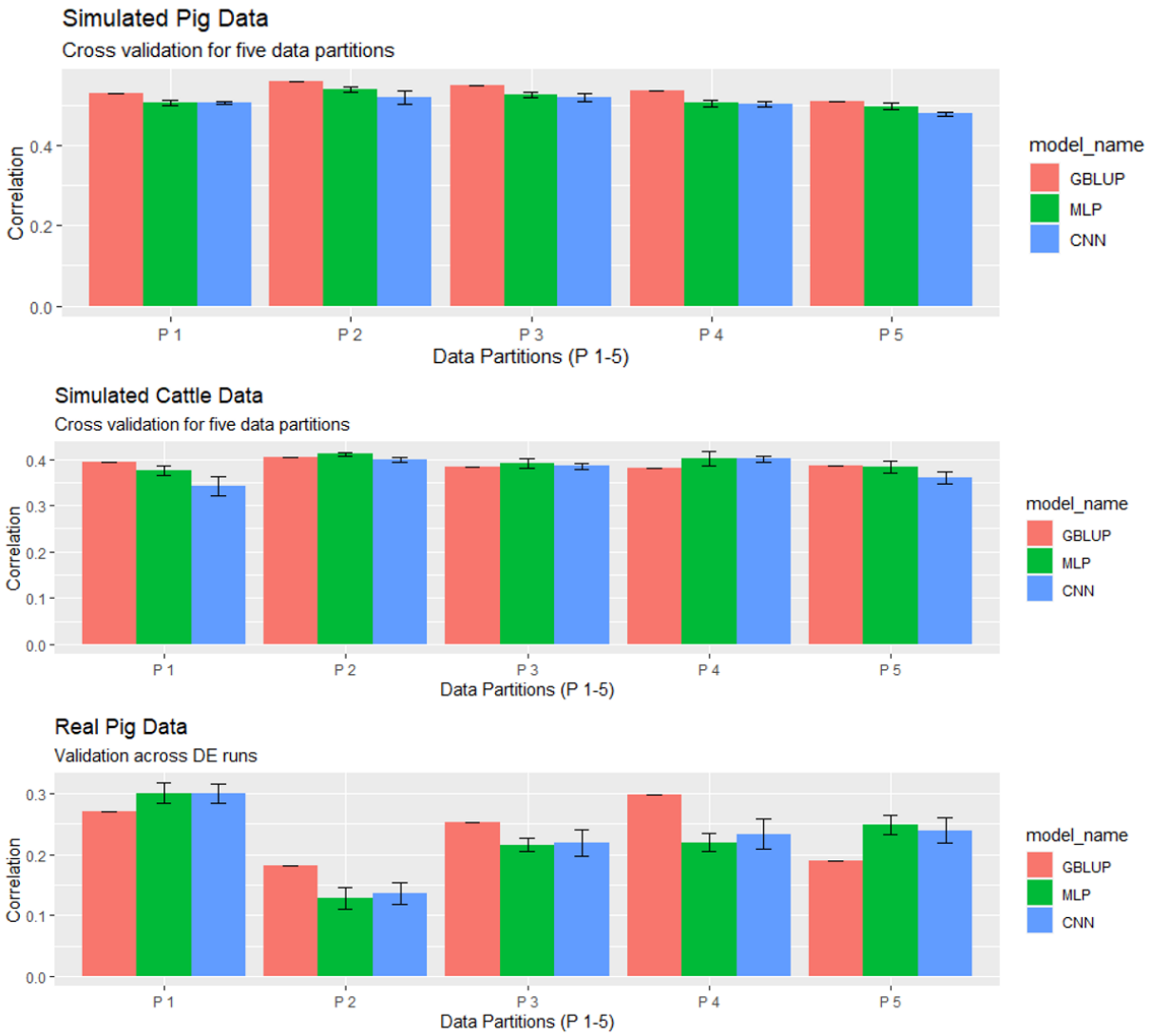

Figure S2.2 Mean predictive performance and error bars across datasets and data partitions. The error bar represents the mean ± standard deviation of cross validation by fitting the same model 30 times. Pink, green, and blue bars correspond to GBLUP, MLP, and CNN models, respectively. MLP, multilayer perceptron; CNN, convolutional neural network; GBLUP, genomic best linear unbiased prediction.

APPENDIX B: FILE S2.1

Genomic best linear unbiased prediction (GBLUP) model

We used GBLUP model to predict the response variable and its prediction accuracy as a comparison to MLPs and CNNs:

$$y = \mu + g + e,$$

where y is the response variable with phenotypes ($y = y_{adj}$ in the real dataset), μ is the population mean, $g \sim N(0, G\sigma_g^2)$ is a vector of random genomic effects, σ_g^2 is the genomic variance, and $e \sim N(0, I\sigma_e^2)$ is a vector of residuals where I is an identity matrix with 1s in the diagonal. The matrix $G = ZZ'$ is the genomic relationship matrix and Z is a matrix with standardized allelic dosages (VanRaden, 2008):

$$Z_{ij} = \frac{M_{ij} - 2p_j}{\sqrt{m(2p_j(1-p_j))}},$$

where M is a matrix of allelic dosages, p_j is the allelic frequency at SNP marker j, i is the ith animal, and m is the number of markers.

Multilayer perceptron

Typical MLP models (Figure 2.1) consist of an input layer, a variable number of hidden layer(s), and an output layer. Each layer contains several neurons (also known as nodes). Depending on the type of layer, the nature of the nodes will change. For instance, the number of nodes in the input layer is equal to the number of predictor features. In this study, the input layer represents an individual's genotype, and thus, the input layer will have as many nodes as SNP markers. In Figure 2.1, there are M nodes in the input layer, and its k^{th} node will receive input as the allelic count at the k^{th} SNP for the n^{th} individual ($x_{n,k}$). The output layer represents the prediction of the response variable produced by MLP. In this case, the output will contain the

prediction of an individual's phenotypic value ($\hat{y}_n$), which can be a continuous outcome, an ordinal outcome, or a categorical outcome.

The nodes in one layer are connected to the nodes in the previous layer by a weighted sum operator. For instance, in Figure 2.1, the input of the j^{th} node in hidden layer 1 is $z_j^1 = f(\sum_{k=1}^{M} w_{jk}^0 x_k)$, where x_k represents the k^{th} node from previous layer, the weights w_{jk}^0 are unknown and connected to x_k (SNP k) and need to be determined through a learning process. $f()$ is the activation function that is specified by the user. It is worth noting that non-linear functions can be used as $f()$ and there is no need to assume linearity as with classic genomic prediction models. Activation functions are detailed in the *Hyperparameters* section further on. Likewise, nodes between layers are fully connected, which means that the input sum of each node in a layer will contain as many terms as there are nodes in the previous layer: $z_{j'}^i = f(\sum_{j=1}^{nneuron_{i-1}} w_{j,j'}^{i-1} z_j^{i-1})$, where j represents the nodes from layer $i-1$, $nneuron_{i-1}$ is the number of nodes in hidden layer $i-1$, i is the index of hidden layer i, $w_{j,j'}^{i-1}$ is a weight connecting j^{th} node (z_j^{i-1}) in hidden layer $i-1$ and j'th node in hidden layer i ($z_{j'}^i$).

Convolutional neural network

In the context of genomic prediction, the input layer for a single observation in a CNN is a one-dimensional array. Similar to MLP, the input layer will contain an animal's (n^{th} individual's) genotype and the number of units will be equal to the number of SNP markers. In Figure 2.2, there are M units in the input layer and the k^{th} unit represents the allelic count at the k^{th} SNP for the nth individual ($x_{n,k}$). The output layer represents the predicted response value $\hat{y}_n$ for the phenotype or breeding value of the nth individual. After the input layer, a CNN contains a variable number of convolutional layers followed by pooling layers. For instance, in Convolutional Layer 1 of Figure 2.2, several filters are applied to the nodes of the input layer,

where filters are arrays containing certain number of weights to convolve the input. In this case, each filter has three weights $w_{i,1}^1$, $w_{i,2}^1$, and $w_{i,3}^1$, where i represents the i^{th} filter defined by the user. These filters are applied to every three consecutive units of the input layer (filter size equal to three). Also, the stride of the filter is equal to its length, which means that the filter is applied to non-overlapping sets of three contiguous SNP. The length of the filter (kernel) is defined by the number of weights to include i.e. the number of units to be convolved by a filter in the input data. An arbitrary number of filters i =1… I is applied in each convolution. The output of this process will be I feature maps with length equal to $\frac{M-F}{S}+1$, where M represents the number of SNP markers, F is the length of the filter, and S is the stride. In our case, because stride is equal to filter size, the length is simply $\frac{M}{3}$. Moreover, the input of the j^{th} unit in feature map 1 is $c_j^1 = f(w_{i,1}^1 x_{n,k} + w_{i,2}^1 x_{n,k+1} + w_{i,3}^1 x_{n,k+2})$, where $x_{n,k}$, $x_{n,k+1}$, and $x_{n,k+2}$ are allelic dosages of individual n at three consecutive SNP markers. The weights in the filters are unknown and need to be determined through a DL optimization process. $f()$ is the activation function. In Convolutional Layer 1, the output of each convolution is saved in feature map 1, where the length of each feature map is $a_1 = \frac{M}{3}$ and the number of feature maps (b_1) is equal to the number of filters (kernels) applied to the input layer (in this case b1=5 in Figure 2.2). A convolutional layer is followed by a pooling layer for the purposes of dimensionality reduction. In pooling layer 1 of Figure 2.2, $p^1 = (p_1^1, p_1^2, \dots, p_{a_1/2}^1)$ are elements that are summarized by every two consecutive units generated from the previous convolutional layer and the output will be b_1 feature maps with a reduced length equal to $\frac{a_1}{2}$. Likewise, feature map 2 is followed by convolutional layer 2 where filters with three weights $w_{i',1}^2$, $w_{i',2}^2$, and $w_{i',3}^2$ are applied. In feature map 3, b_2 features with a length of a_2 are summarized into feature map 4 that has b_2 features

with length $\frac{a_2}{2}$. If any value among $\frac{M}{3}$, $\frac{a_1}{2}$, $\frac{a_1/2}{3}$, or $\frac{a_2}{2}$ has a remainder, the deficit unit(s) in the input data will be padded with zero(s). The last feature map (feature map 4) is re-arranged into a single vector that has $\frac{b_2 \times a_2}{2}$ elements. Each element in the re-arranged vector $z^1 = (z_1^1, z_2^1, \dots, z_l^1, l = \frac{b_2 \times a_2}{2})$ is fully connected to a hidden layer (like the ones described in the MLP section of this paper) with $nneuron$ nodes, which are predictors for the output layer.

Hyperparameters

The number of hidden layers describes the depth of the network and DL requires at least one hidden layer, and it is known as the depth of the network. In the deep learning literature several studies found that the number of hidden layers, in similar sized problems, can often provide better results with a maximum upper bound of five (Arifin et al., 2019; Bellot et al., 2018). Thus, we optimized the number of hidden layers by selecting an integer ranging from one through five.

The number of neurons decides the number of units in a fully connected hidden layer, and it is known as the width of the network. Bellot et al. (2018) investigated the influence of neuron numbers on neural networks by varying neurons per layer in four scenarios: 16, 32, 64 and 128, and Pérez-Enciso and Zingaretti (2019) estimated the effect of the number of neurons in the first layer (8, 24, 32, 64, and 128). Both studies used discrete values for the number of neurons. In this study, we optimized the width of the neural network by selecting integers between 8 and 512. The number of neurons is optimized by the DE algorithm for every hidden layer of the MLP and the last hidden layer of the CNN.

Activation function is a function to transform the weighted sums from the previous layer. The aforementioned Pérez-Enciso and Zingaretti (2019) recommended “tanh”, “relu”, “selu” and

"sigmoid". In addition to their reccomendation, we further included "elu", "softplus" as well as "linear" as possible activation functions.

In deep learning, an optimizer is an algorithm used to alter the attributes of the model e.g. weights and learning rates, where learning rates are coefficients applied to altered weights. Optimizer options included were "sgd", "adam", "adagrad", "rmsprop", "adadelta", "adamax", and "nadam".

Dropout is used to avoid overfitting due to the large number of weights that need to be estimated. Dropout consists of and randomly sets a proportion (dropout rate) of the neurons in a layer for which their weights are not updated in the current iteration. The dropout rate may affect the predictive performance of a model and we included it in the hyperparameter optimization as a continuous parameter in a range of (0,1).

Another way to ease overfitting is to use weight regularization that adds constraints of weights to the loss function. For instance, in L2 regularization a squared penalty on the values of the weights is added to the loss function. This parameter also may have an effect on the model's predictive performance. We optimized the L2 regularization as a parameter in a range of (0,1).

L2 Regularization is defined as:

$$L(\hat{\theta}, X, y) = \frac{1}{n} \sum_i (y_i - f_{\hat{\theta}}(X_i))^2 + \lambda \sum_{j=1}^{p} \hat{\theta}^2 ,$$

where $L()$ represents loss function, X represents input data, y is the observed response variable, $\hat{\theta}$ consists of weights in the deep learning model, $f_{\hat{\theta}}()$ represents the deep learning model, and λ is the L2 regularization parameter.

Epoch refers to the number of iterations where an entire training dataset is passed through the DL model to iteratively adjust weights. Within an epoch the training dataset is further divided into an actual training set for weight adjustment and a testing set that is used for performance

evaluation. The number of epochs to be optimized was an integer between 21 and 50. We introduced an early-stopping rule when there is no improvement of the model training for ten consecutive epochs.

Batch size is used to determine the number of randomly partitioned training samples (within an epoch) utilized to update the weights. For the simulated datasets, we first optimized a continuous value α (Table 2.1) in the range of [0.001-0.01], while the range in the real pig dataset was [0.01, 0.1]. Then, the batch size was defined as the product of training sample size N and α. The number of samples utilized in each DL batch varied according to the training size (N=6539 for the simulated datasets and N=728 for the real pig dataset). This hyperparameter has a profound influence on the computing time and memory required by *TensorFlow* (Abadi *et al.* 2015, https://www.tensorflow.org/) to fit the model. We only optimized batch size in MLP while the batch size was fixed at 32 in CNN because larger batch sizes became computationally too onerous to fit CNN with the larger datasets (N=6539 and M=48,541).

The number of filters, filter (also known as kernel) size and pooling function are hyperparameters exclusive of convolutional neural networks (CNN). A filter is an array of weights used to convolve the input. Typically, multiple filters can be utilized in each layer. Pérez-Enciso and Zingaretti (2019) explored CNN architectures with 16, 32 and 64 filters while Bellot et al. (2018) varied the number of filters with 16, 32, 64 and 128. We optimized the number of filters in CNN by selecting an integer between 4 and 128.

The filter size of a 1d CNN is the number of weights in the filter. Pérez-Enciso and Zingaretti (2019) compared the predictive performance using kernel (filter) sizes of three, five and seven, while Bellot et al. (2018) used filter sizes of two, three, five and ten. In this study, we defined the sample space for filter size as an integer between two and 20.

A pooling layer is used to downsize the feature map that comes from the convolution operation by computing a summary statistical measure of several elements. The typical options for a pooling layer are average, minimum and maximum. Bellot et al. (2018) applied a 1×2 pooling to the feature maps. Similarly, we fixed the size of the pooling to two units in the feature map and optimized the pooling function by selecting between average value or maximum value of the two units. In other words, our models were optimized by selecting one of the pooling rules (average pooling or maximum pooling) to downsize the feature map that comes out of the convolutional operation by half (the two units were summarized into one).

It is necessary to point out that the hyperparameter space for sampling values or options varies across the literature, and it is up to the user to setup the adaptive architecture of the network. With differential evolution (DE) users can not only optimize the subset of hyperparameters used in this study, but can also optimize any other additional hyperparameters they deem relevant.

Predictive performance of DL models and GBLUP

Figure S2.2 shows the predictive performance (correlation between the predicted and the observed response variables in the external validation sets) for each method (optimized MLPs, optimized CNNs, and GBLUP). For the simulated pig dataset, all methods performed similarly, although GBLUP models were slightly better and CNN models were the worst. For the simulated cattle dataset, GBLUP models were better in partitions 1 and 5, while optimized MLPs and CNNs were slightly better in partitions 2, 3 and 4 (with tied performance of MLPs and CNNs). For the real pig dataset, the pattern was completely different, and the best models depended on the data partition. We did not notice a clear improvement in prediction accuracy for any of the models. Since deep learning model fitting results differ even with the same hyperparameters, we

ran 30 external (cross) validations using the same hyperparameters and validation sets for all partitions across the three datasets. Each MLP and CNN was trained 30 times independently and repeatedly. Models in both the simulated pig and cattle datasets showed little variation through repetition. However, we observed more variation in the prediction performance for the real pig dataset.

REFERENCES

REFERENCES

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jozefowicz, R., Jia, Y., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Schuster, M., Monga, R., Moore, S., Murray, D., Olah, C., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., Zheng, X., 2015. TensorFlow: Large-scale machine learning on heterogeneous systems.

Abdollahi-Arpanahi, R., Gianola, D., Peñagaricano, F., 2020. Deep learning versus parametric and ensemble methods for genomic prediction of complex phenotypes. Genet. Sel. Evol. 52, 1–15. https://doi.org/10.1186/s12711-020-00531-z

Arifin, F., Robbani, H., Annisa, T., Ma'Arof, N.N.M.I., 2019. Variations in the Number of Layers and the Number of Neurons in Artificial Neural Networks: Case Study of Pattern Recognition. J. Phys. Conf. Ser. 1413, 0–6. https://doi.org/10.1088/1742-6596/1413/1/012016

Bean, J.C., 1994. Genetic Algorithms and Random Keys for Sequencing and Optimization. ORSA J. Comput. https://doi.org/10.1287/ijoc.6.2.154

Bellot, P., de los Campos, G., Pérez-Enciso, M., 2018. Can deep learning improve genomic prediction of complex human traits? Genetics 210, 809–819. https://doi.org/10.1534/genetics.118.301298

Casiró, S., Velez-Irizarry, D., Ernst, C.W., Raney, N.E., Bates, R.O., Charles, M.G., Steibel, J.P., 2017. Genome-Wide association study in an F2 duroc x pietrain resource population for economically important meat quality and carcass traits. J. Anim. Sci. 95, 545–558. https://doi.org/10.2527/jas2016.1003

Chollet, F., Allaire, J., Falbel, D., 2017. R Interface to Keras.

Corvin, A., Craddock, N., Sullivan, P.F., 2010. Genome-Wide Association Studies: A Primer. Psychol Med. 40, 1063–1077. https://doi.org/10.1017/S0033291709991723

Crossa, J., Martini, J.W.R., Gianola, D., Pérez-Rodríguez, P., Jarquin, D., Juliana, P., Montesinos-López, O., Cuevas, J., 2019. Deep Kernel and Deep Learning for Genome-Based Prediction of Single Traits in Multienvironment Breeding Trials. Front. Genet. 10, 1–13. https://doi.org/10.3389/fgene.2019.01168

Cuyabano, B., 2020. GenEval.

D'souza, R.N., Huang, P.Y., Yeh, F.C., 2020. Structural Analysis and Optimization of Convolutional Neural Networks with a Small Sample Size. Sci. Rep. 10, 1–13. https://doi.org/10.1038/s41598-020-57866-2

Das, S., Mullick, S.S., Suganthan, P.N., 2016. Recent advances in differential evolution-An updated survey. Swarm Evol. Comput. 27, 1–30. https://doi.org/10.1016/j.swevo.2016.01.004

Edwards, D.B., Ernst, C.W., Raney, N.E., Doumit, M.E., Hoge, M.D., Bates, R.O., 2008. Quantitative trait locus mapping in an F2 Duroc x Pietrain resource population: II. Carcass and meat quality traits. J. Anim. Sci. 86, 254–266. https://doi.org/10.2527/jas.2006-626

Eraslan, G., Avsec, Ž., Gagneur, J., Theis, F.J., 2019. Deep learning: new computational modelling techniques for genomics. Nat. Rev. Genet. 20, 389–403. https://doi.org/10.1038/s41576-019-0122-6

Fragomeni, B.O., Lourenco, D.A.L., Masuda, Y., Legarra, A., Misztal, I., 2017. Incorporation of causative quantitative trait nucleotides in single-step GBLUP. Genet. Sel. Evol. 49, 1–11. https://doi.org/10.1186/s12711-017-0335-0

Gämperle, R., Müller, S.D., Koumoutsakos, P., 2002. A Parameter Study for Differential Evolution, in: Advances in Intelligent Systems, Fuzzy Systems, Evolutionary Computation. Press, pp. 293–298.

Gianola, D., 2013. Priors in whole-genome regression: The Bayesian alphabet returns. Genetics 194, 573–596. https://doi.org/10.1534/genetics.113.151753

Goodfellow, I., Bengio, Y., Courville, A., 2016. Deep Learning. Cambridge, Massachusetts : The MIT Press.

Gualdrón Duarte, J.L., Bates, R.O., Ernst, C.W., Raney, N.E., Cantet, R.J.C., Steibel, J.P., 2013. Genotype imputation accuracy in a F2 pig population using high density and low density SNP panels. BMC Genet. 14. https://doi.org/10.1186/1471-2156-14-38

Habier, D., Fernando, R.L., Kizilkaya, K., Garrick, D.J., 2011. Extension of the bayesian alphabet for genomic selection. BMC Bioinformatics 12. https://doi.org/10.1186/1471-2105-12-186

Hickey, J.M., Chiurugwi, T., Mackay, I., Powell, W., 2017. Genomic prediction unifies animal and plant breeding programs to form platforms for biological discovery. Nat. Genet. 49, 1297–1303. https://doi.org/10.1038/ng.3920

Hill, W.G., 2016. Is continued denetic improvement of livestock sustainable? Genetics 202, 877–881. https://doi.org/10.1534/genetics.115.186650

Kim, T., Lee, J.H., 2019. Effects of Hyper-Parameters for Deep Reinforcement Learning in Robotic Motion Mimicry: A Preliminary Study. 2019 16th Int. Conf. Ubiquitous Robot. UR 2019 228–235. https://doi.org/10.1109/URAI.2019.8768564

Kok, K.Y., Rajendran, P., 2016. Differential-evolution control parameter optimization for unmanned aerial vehicle path planning. PLoS One 11, 1–12. https://doi.org/10.1371/journal.pone.0150558

Koumakis, L., 2020. Deep learning models in genomics; are we there yet? Comput. Struct. Biotechnol. J. https://doi.org/10.1016/j.csbj.2020.06.017

Lecun, Y., Bengio, Y., Hinton, G., 2015. Deep learning. Nature. https://doi.org/10.1038/nature14539

Luo, G., 2016. A review of automatic selection methods for machine learning algorithms and hyper-parameter values. Netw. Model. Anal. Heal. Informatics Bioinforma. 5, 1–15. https://doi.org/10.1007/s13721-016-0125-6

Meuwissen, T.H.E., Hayes, B.J., Goddard, M.E., 2001. Prediction of total genetic value using genome-wide dense marker maps. Genetics 157, 1819–1829.

Mitchell, B., Tosun, H., Sheppard, J., 2015. Deep learning using partitioned data vectors. Proc. Int. Jt. Conf. Neural Networks 2015-Septe. https://doi.org/10.1109/IJCNN.2015.7280484

Montesinos-López, A., Montesinos-López, O.A., Gianola, D., Crossa, J., Hernández-Suárez, C.M., 2018. Multi-environment genomic prediction of plant traits using deep learners with dense architecture. G3 Genes, Genomes, Genet. 8, 3813–3828. https://doi.org/10.1534/g3.118.200740

Montesinos-López, O.A., Martín-Vallejo, J., Crossa, J., Gianola, D., Hernández-Suárez, C.M., Montesinos-López, A., Juliana, P., Singh, R., 2019. New deep learning genomic-based prediction model for multiple traits with binary, ordinal, and continuous phenotypes. G3 Genes, Genomes, Genet. 9, 1545–1556. https://doi.org/10.1534/g3.119.300585

Montesinos-López, O.A., Montesinos-López, A., Crossa, J., Gianola, D., Hernández-Suárez, C.M., Martín-Vallejo, J., 2018. Multi-trait, multi-environment deep learning modeling for genomic-enabled prediction of plant traits. G3 Genes, Genomes, Genet. https://doi.org/10.1534/g3.118.200728

Nakisa, B., Rastgoo, M.N., Rakotonirainy, A., Maire, F., Chandran, V., 2018. Long short term memory hyperparameter optimization for a neural network based emotion recognition framework. IEEE Access 6, 49325–49338. https://doi.org/10.1109/ACCESS.2018.2868361

Pérez-Enciso, M., Zingaretti, L.M., 2019. A Guide on Deep Learning for Complex Trait Genomic Prediction. Genes (Basel). 10, 19.

R Core Team, 2020. R: A Language and Environment for Statistical Computing.

Shahinfar, S., Meek, P., Falzon, G., 2020. "How many images do I need?" Understanding how sample size per class affects deep learning model performance metrics for balanced designs in autonomous wildlife monitoring. Ecol. Inform. 57, 101085. https://doi.org/10.1016/j.ecoinf.2020.101085

Slatkin, M., 2008. Linkage disequilibrium: understanding the genetic past and mapping the medical future. Nat. Rev. Genet. 9, 477–485. https://doi.org/10.1038/nrg2361.Linkage

Steibel, J.P., 2015. gwaR: Functions for performing GWA from GBLUP.

Storn, R., Price, K., 1997. Differential Evolution - A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. J. Glob. Optim. 11, 341–359. https://doi.org/10.1023/A:1008202821328

Tang, X., Sun, Y., 2019. Fast and accurate microRNA search using CNN. BMC Bioinformatics 20, 1–14. https://doi.org/10.1186/s12859-019-3279-2

VanRaden, P.M., 2008. Efficient methods to compute genomic predictions. J. Dairy Sci. 91, 4414–4423. https://doi.org/10.3168/jds.2007-0980

Yang, J., Benyamin, B., McEvoy, B.P., Gordon, S., Henders, A.K., Nyholt, D.R., Madden, P.A., Heath, A.C., Martin, N.G., Montgomery, G.W., Goddard, M.E., Visscher, P.M., 2010. Common SNPs explain a large proportion of the heritability for human height. Nat. Genet. 42, 565–569. https://doi.org/10.1038/ng.608

Yu, T., Zhu, H., 2020. Hyper-Parameter Optimization: A Review of Algorithms and Applications 1–56.

Zhang, S.X., Chan, W.S., Peng, Z.K., Zheng, S.Y., Tang, K.S., 2020. Selective-candidate framework with similarity selection rule for evolutionary optimization. Swarm Evol. Comput. 56, 2–28. https://doi.org/10.1016/j.swevo.2020.100696

Zingaretti, L.M., Gezan, S.A., Ferrão, L.F. V., Osorio, L.F., Monfort, A., Muñoz, P.R., Whitaker, V.M., Pérez-Enciso, M., 2020. Exploring Deep Learning for Complex Trait Genomic Prediction in Polyploid Outcrossing Species. Front. Plant Sci. 11, 1–14. https://doi.org/10.3389/fpls.2020.00025

CHAPTER 3: PUBLICLY AVAILABLE DATASETS FOR COMPUTER VISION IN PRECISION LIVESTOCK FARMING: A REVIEW

Junjie Han, Joao R. Dorea, Tomas Norton, Andrea Parmiggiani, Daniel Morris, Janice Siegford, and Juan P. Steibel

1. ABSTRACT

The livestock sector is increasingly using precision livestock farming (PLF) to assist in automated and real-time decision making for management purposes. Among the tools used in PLF, computer vision (CV) is a predominant approach that allows automatic feature extraction from digital images/videos. Thus, CV is useful for monitoring animals and measuring phenotypes. A key to developing CV is training models with annotated imagery data. Unlike general CV, there are limited amount of publicly available PLF imagery data. Furthermore, despite the potential of CV in PLF, most published CV applications in PLF are developed using rather small datasets, and their broader validity remains unknown. The goal of this study was to review public datasets for PLF-CV applications and the validation strategies used in the related work, which is a necessary step to create reference PLF datasets and to develop standard evaluation matrices that can be informative in practical animal farming. We focused on pig and cattle datasets as well as their CV tasks. We identified 20 public datasets, nine of which focused on pigs and 11 on cattle. The reviewed datasets spanned a wide range of CV tasks: e.g., detection, behavior recognition, identification, and tracking of animals, which are useful for developing CV algorithms without having to record and annotate new videos. Finally, we provide suggestions to improve CV applications in PLF, perspectives on data reuse, and suggestions for broader validation of results.

2. INTRODUCTION

Livestock farmers face ever-increasing farming pressure due to the growing population and demands for food. As a result, the livestock sector has become increasingly interested in the precision livestock farming technology to improve the production efficiency of the livestock industry (Norton et al., 2019). Precision livestock farming (PLF) refers to automated, continuous, and real-time monitoring technologies within animal space that assist in decision making at individual animal level, and PLF brings economic values to farmers as well as to animal breeders (Berckmans, 2017).

Among PLF technologies, computer vision is predominant (Li et al., 2021). Computer vision (CV) has advantages in animal farming as it is non-invasive and operatable in a continuous and large scale (Chen et al., 2021). In the past decade, CV has been revolutionized by deep learning (Ponti et al., 2017), which is a set of flexible representation learning methods where a machine can be fed with raw data to automatically discover the representations needed for prediction or classification (Lecun et al., 2015). Deep learning (DL) has made great contributions to CV tasks including image classification, object detection, pose estimation, behavior recognition, semantic/instance segmentation, and tracking (Ponti et al., 2017; Voulodimos et al., 2018). These CV tasks are also related to certain PLF applications. For instance, the gender of chicken can be identified through image classification (Yao et al., 2020). Moreover, for animal behavior studies, several CV models have been proposed to recognize aggressive behaviors of pigs through video analysis (Chen et al., 2020b; Li et al., 2019).

A key to developing a reliable DL-based CV model is to "feed" a large number of high-quality training samples to the model i.e., the size and quality of data are vital (Lu et al., 2017; Marcus, 2018). For general CV applications, there are reference datasets such as COCO (Lin et

al., 2014) and ImageNet (Jia Deng et al., 2009), which consist thousands of images with millions of instances annotated by experts that allow CV developers to benchmark their state-of-the-art algorithms. The datasets were extremely laborious for image collection and annotation. For instance, over 70,000 worker hours were utilized to complete the COCO Dataset (Lin et al., 2014). Unfortunately, we do not have such a dataset available for use in PLF that is specifically designed for animal farming problems.

As PLF is still an emerging technology, CV applications to animal farming are mostly at performance evaluation phase with relatively small datasets. Furthermore, a common practice in recently developed animal CV applications (Chen et al., 2020a; D. Li et al., 2019; Liu et al., 2020; Nasirahmadi et al., 2019; Zhang et al., 2020) is to use a random validation approach to evaluate model performance, where the training set (used for model development) and validation set (used for model assessment) are randomly split from the whole dataset. However, such a random split might have ignored the underlying temporal, spatial, or hierarchical structures of the data, leading to overoptimistic results (Roberts et al., 2017). In practical animal farming, there are underlying structure(s) within the data e.g., time-evolving effects (growing animals) and environmental factors (e.g., illumination, changing background environment, etc.) that may not be reflected during model development using a random validation approach. Thus, how training-validation data are split can significantly affect predictive performance of DL (Han et al., 2021).

The goals of this paper are to: 1) review publicly available datasets for PLF and their practical applications focusing on pig and cattle datasets that can be reused for future studies, 2) review the validation strategies in the mentioned applications, and 3) provide suggestions for improved CV applications in PLF focusing on data perspectives.

3. METHODOLOGY

3.1 Literature search parameters

Publications in the following databases were searched: Web of Science, Google Scholar, and Google Datasets. Search keywords were the following term combinations: “species term” + computer vision; “species term” + deep learning; “species term” + image analysis, where “species term” included pig, swine and cattle.

3.2 Eligibility criteria

To be included in this review, a publication had to fulfill the following criteria: 1) it had to belong to one of the following categories: conference proceedings, peer-reviewed journal articles, or datasets assigned with Digital Object Identifiers (DOI); 2) the publication had to be written in the English language; 3) it had to address a CV application in PLF; and 4) it had to contain a working link to an accessible dataset.

3.3 Data extraction

Literature was collected in March 2022. From the retrieved datasets and their related publications, data on several features were extracted and summarized. The full description of extracted information is given in Table 3.1. Briefly, the features included the objective of the study, a detailed description of the imagery dataset, camera specifications and settings, software and code for implementation, a detailed description of annotations tied with the imagery dataset, metadata of the observed animals, data sampling protocol, and validation strategies.

Table 3.1 Description of extracted data.

Information	Type of input	Description
		Study attributes
Author, year, and title	Text	Citation of the paper
Computer vision tasks	Categories	Choice(s) of: entire body detection, body part detection, segmentation, behavior recognition, identification, and tracking
Dataset name	Text	Name of the publicly available dataset for CV in PLF
Database link	URL	Link to the website or data repository to download data
Species	Categories	Animal species: cattle and pig
		Image and video attributes
Modality	Categories	RGB, Depth, grayscale, and thermal images/videos
Resolution	Number	Number of pixels identified by the height and width of image
Number of files	Number	Number of files included in the public dataset
		Camera attributes
Camera perspective	Categories	Choice(s) of: Angled-down view, top-down view, side view, and frontal view
		Software and code
Code availability	Yes/no	Whether computer code is available for the CV task
		Annotation attributes
Annotation types	Categories	Choice(s) of: class labels, bounding box, polygon mask, ID, and key point coordinates
Annotation labels	Text	Detailed content of annotations e.g., bounding box coordinates and list of class labels
Analysis unit	Text	The base analysis unit of the CV model e.g., a single image or a video clip
Bounding box area	Number	Area in pixel for bounding boxes (this is only available for objected detection studies)
ROI	Text	Region of interest
		Biological subject attributes
Number of animals	Number	Number of observed animals in the dataset
Number of instances per image	Number	A fixed number or a range for the count of visible animals in a single image
Coat color pattern	Text	Characteristics of the animal coat color and marks
Number of farming units	List	An exhaustive list of farming units from lower to higher levels e.g., 5 pens from 1 farm
Age or production stage	Text	Production phase or age of the animal
		Sampling protocol
Span of experiment	Text	The span of experiment for long-term image/video collection
Recording schedule during the day	Time range	The time when images/videos were collected (this is typically found in the published paper)
		Validation strategies
Validation method	Categories	The way to split the entire dataset into training, validation, and/or testing sets. Choices include: random validation, stratified random validation, blocked validation, and in-sample validation
Evaluation metric	Text	Brief explanation of the metric and the estimated value

4. RESULTS

We identified 20 public datasets, nine of which focused on pigs and 11 on cattle. For a clear overview, the authors, dataset name, species, and addressed CV task(s) of the traces are shown in Table 3.2. The full URLs to access the datasets are available in Table S3.1.

Table 3.2 Overview of public pig and cattle datasets utilized for computer vision tasks in precision livestock farming.

Authors	Dataset name	Shortcut	Species	Computer vision task(s)
Alameer et al. (2020)	Newcastle Pig Posture	D1	Pig	Entire body detection, behavior recognition, and tracking
Bergamini et al. (2021)	Edinburgh Pig Behavior	D2	Pig	Entire body detection, behavior recognition, and tracking
Riekert et al. (2020)	Pig Position and Posture	D3	Pig	Entire body detection and behavior recognition
Shirke et al. (2021b)	ISRL Multi-Camera Tracking	D4	Pig	Entire body detection and tracking
Psota et al. (2019)	Pig Detection	D5	Pig	Body part detection
Psota et al. (2020)	Pig tracking	D6	Pig	Body part detection, identification, and tracking
Shirke et al. (2021a)	Pig Novelty Preference	D7	Pig	Body part detection and behavior recognition
Wutke et al. (2021)	Pig Detection and Tracking	D8	Pig	Body part detection, tracking and behavior recognition
Tangirala et al. (2021)	PigTrace	D9	Pig	Segmentation, tracking, behavior recognition, and identification
Andrew et al. (2017)	FriesianCattle2017	D10	Cattle	Entire body detection and identification
Andrew et al. (2017)	AerialCattle2017	D11	Cattle	Entire body detection and identification
Andrew et al. (2021)	OpenCows2020	D12	Cattle	Entire body detection and identification
Shao et al. (2020)	Aerial Pasture	D13	Cattle	Entire body detection
Gao et al. (2021)	Cows2021	D14	Cattle	Entire body detection and identification
Han et al. (2019)	Aerial Livestock	D15	Cattle	Entire body detection
Li et al. (2019)	NWAFU-Cattle	D16	Cattle	Body part detection and behavior recognition
Shojaeipour et al. (2021)	300 Cattle	D17	Cattle	Body part detection and identification
Andrew et al. (2016)	FriesianCattle2015	D18	Cattle	Identification
Bhole et al. (2019)	Holstein Cattle Recognition	D19	Cattle	Identification
Pereiet et al. (2020)	Cow Behavior	D20	Cattle	Behavior recognition

In Table 3.2, the first column indicates the references for the original studies that analyzed and published the datasets. In addition to the full name of each dataset, we created a unique shortcut for naming convention that is later used in this section. Noteworthy, some datasets claimed to address multiple CV tasks in the original studies. The reviewed public animal datasets for CV focused on six tasks that are covered in Section 3.3.

4.1 Animal subjects

Table 3.3 presents the number of observed animals, number of housing units, age or production stage, and coat color pattern for the different datasets. These biological characteristics are important in terms of defining the use of the data for various CV tasks. Deep learning-based

CV algorithms are known to be data-hungry, requiring very large numbers of training samples (Marcus, 2018). Thus, explicitly stating how many images are available in a CV dataset is extremely important. However, the total image count is not enough to characterize a CV dataset in PLF. Knowing the number of animals is essential too, as a thousand images from one animal is different from one image from each of a thousand animals. Broadly valid CV applications need to be trained on a large number of images collected from many animals. Likewise, identifying the number of farming units (pens, farms, etc.) available in a CV dataset for PLF is as important as counting individual animals, as datasets comprising several farming units will support CV applications with a broader scope of validity. Specifying the age and physiological stage of the animals in the dataset are also important as there may be some ages/stages at which animals vary more in size and shape, thus introducing extra variation into the datasets (e.g.: growing pigs vs. gestating sows or milking heifers vs. mature dry cows). Finally, the performance of CV may be influenced by the coat color of animals. For example, identifying animals may be easier in breeds exhibiting natural variation in color patterns compared to animals that show a uniform coat color.

The biological sample size was not available in some datasets, while for those papers that reported it, the number of observed animals ranged from eight to 430 (Table 3.3). Most studies specified the farming units e.g., the number of experimental pens and the number of farms. Six pig datasets (D3, D4, D5, D6, D8, and D9) involved multiple pens, while all cattle datasets focused on single farms. Information about ages of animals was not available in the cattle datasets. Among the three pig datasets that have age information (D1, D5, and D6), ages of animals varied from three weeks to six months. Four studies reported production stages of the pigs (D2, D3, D4, and D8), which covered farrowing, nursery, and finisher. Most pig datasets

contained white pigs, and three datasets had back marks on white pigs. Animals with heterogeneous coat colors were presented across all cattle datasets.

Table 3.3 Characteristics of animal subjects. *: to specify an exhaustive list of units/ranges if applicable. Multiple pens mean that the number of pens is more than two while the exact number remains unknown.

Dataset	Species	# Animals	Farming Unit(s)*	Age or production stage	Coat Color of Animals
D1	Pig	15	1 pen	9-14 weeks	Heterogeneous coat colors
D2	Pig	8	1 pen	Finisher pigs	White pigs with back marks
D3	Pig	430	18 pens from 5 compartments	Fattening and rearing pigs	White pigs
D4	Pig	33	2 pens from the same facility	Finisher pigs	White pigs
D5	Pig	NA	17 pens	1.5-5.5 months	White pigs
D6	Pig	NA	Multiple pens	3-10 weeks; 11-18 weeks; 19-26 weeks	White pigs
D7	Pig	NA	1 pen	NA	White pigs with/without back marks
D8	Pig	NA	Multiple pens	Farrowing and rearing pigs	White pigs with/without back marks
D9	Pig	NA	Multiple pens from 5 farms	NA	White pigs with/without back marks
D10	Cattle	89	1 farm	NA	Heterogeneous coat colors
D11	Cattle	23	1 farm	Nursery	Heterogeneous coat colors
D12	Cattle	46	1 farm	NA	Heterogeneous coat colors
D13	Cattle	218	1 farm	NA	Heterogeneous coat colors
D14	Cattle	186	1 farm	NA	Heterogeneous coat colors
D15	Cattle	NA	NA	NA	Heterogeneous coat colors
D16	Cattle	63	1 farm	NA	Heterogeneous coat colors
D17	Cattle	300	1 farm	NA	Heterogeneous coat colors
D18	Cattle	40	1 farm	NA	Heterogeneous coat colors
D19	Cattle	136	1 farm	NA	Heterogeneous coat colors
D20	Cattle	NA	1 farm	NA	Heterogeneous coat colors

4.2 Recording setup

Camera setups and recording schedules are also known to impact data variability and system development (Li et al., 2021). Several characteristics of the recording setup were selected, extracted from all papers and the results are summarized in Table 3.4. The camera perspective is an important recording characteristic as it determines which visual component(s) of animals can be observed and used to develop CV. For instance, if an image dataset was collected using a top-down view, then a CV application would focus on extracting features from the back part of animals. Image modality indicates the type of image. Common image types

include RGB (red-green-blue), grayscale, depth, and thermal. An RGB image refers to color image and is representative of human vision. A grayscale image is a special type of digital image, which refers to gray monochrome representing light intensity. RGB images are prevalent and frequently used for artificial intelligence. Depth images consist of pixels that record the distance from the object to the camera and are useful for separating objects from background and for estimating objects' size and volume. Thermal images allow researchers to observe variations in temperature of objects. Depending on the CV task, researchers may choose the image modality that fits better in the particular context. Resolution is typically described as the number of pixels of an image and is specified as the product of width and height of the image. Higher resolution provides more details of the objects in the image but requires larger storage space. In addition, the recording schedule during the day (i.e., the time when images were recorded) is reviewed for each dataset as it reflects the illumination condition during data collection, and illumination can greatly affect image quality (Wu and Sun, 2013). Span of the experiment (long-term recording schedule) is also important, as collecting a hundred images from the same day is different from obtaining a hundred images across ten days (the latter covers a large temporal variation).

Top-down view and angled-down view (also known as tilted top-down view) are predominant camera views (Table 3.4). For datasets collected during daylight hours, RGB cameras were utilized except for two studies that introduced a depth camera and a thermal camera in addition to the RGB camera, respectively. The majority of the data were collected during daylight hours. Five datasets included night recordings (D4, D5, D6, D8, and D20), and grayscale images were introduced for night recordings. Resolution varied in the datasets. The span of experiment was unknown for nine datasets. Among the 13 datasets that have a long-term

recording schedule available, three were collected within one day, while the remaining ten datasets were collected from multiple days or even months.

Table 3.4 Recording setup and schedule of publicly available datasets for computer vision in livestock farming. RGB, red-green-blue; RGB-D, RGB and depth. Multiple weeks/days mean that the experiment lasted more than two weeks/days while the exact number remained unknown. Varying resolutions represent that more than two resolutions are involved.

Dataset	Species	Camera perspective(s)	Modality	Resolution(s)	Recording schedule during the day	Span of experiment
D1	Pig	Top-down view	RGB	640×360	11AM-3PM	8 days
D2	Pig	Angled-down view	RGB-D	1280×720	7AM-7PM	6 weeks
D3	Pig	Angled-down and top-down views	RGB	1280×720 and 640×480	Daylight hours	8 days between 2017 and 2018
D4	Pig	Angled-down and top-down views	RGB and grayscale	3840×2160	Daylight and night hours	NA
D5	Pig	Top-down view	RGB and grayscale	1920×1080 and 2688×1520	Daylight and night hours	Multiple weeks
D6	Pig	Top-down view	RGB and grayscale	2688×1520	Daylight and night hours	Multiple weeks
D7	Pig	Top-down view	RGB	1920×1080	NA	NA
D8	Pig	Top-down view	RGB and grayscale	1280×800	Daylight and night hours	3 months
D9	Pig	Angled-down and top-down views	RGB	1280×720 and 1280×960	Daylight hours	NA
D10	Cattle	Top-down view	RGB	1486×1230	Daylight hours	NA
D11	Cattle	Top-down view	RGB	Varying resolutions	Daylight hours	1 day
D12	Cattle	Top-down view	RGB	Varying resolutions	Daylight hours	NA
D13	Cattle	Top-down view	RGB	3000×4000	Daylight hours	Approximately 3 months
D14	Cattle	Top-down view	RGB	Varying resolutions	Daylight hours	Multiple days
D15	Cattle	Top-down view	RGB	3000×4000 and 3840×2160	Daylight hours	NA
D16	Cattle	Side view	RGB	1920×1080	9AM-4PM	1 day
D17	Cattle	Frontal view	RGB	4000×6000	8AM-4PM	1 day
D18	Cattle	Top-down view	RGB	1486×1230	Daylight hours	NA
D19	Cattle	Side view	RGB and thermal	640×480 and 320×240	Daylight hours	9 days
D20	Cattle	Angled-down view	RGB and grayscale	1920×1080	Daylight and night hours	3 months

4.3 Review of selected datasets by computer vision task

In this section, we organized and summarized the identified publicly available datasets into six CV tasks. A dataset was considered suitable for a certain CV task if both images and annotations were available to accomplish the task at hand. Some datasets were suitable for

accomplishing more than one CV task. Datasets with missing components (either images or key annotations) are not listed in the corresponding subsections.

4.3.1 Entire body detection

The role of object detection is to estimate concepts and locations of objects in each image (Zhao et al., 2019). Object detection can be further divided into subdomains including entire body detection and detection of body parts or key points. Entire body detection is to provide spatial location of individual animals relative to the image (Figure 3.1a). To develop models for entire body detection, at minimum, a data point includes an image displaying the object(s) of interest and the coordinates of a rectangular bounding box enclosing each object. Developers should be aware of the number of instances per image (the count of visible animals in a single image), as a large number implies a broad view, and the scene can be complex. The bounding box area (in pixels) implies the size of object relative to the image resolution, and it is informative especially to anchor-based algorithms in object detection (Liu et al., 2016). The number of annotated images is important for model development as most DL-based CV algorithms are data-hungry, and this fact applies to all CV tasks.

Eight datasets (four pig datasets and four cattle datasets) were identified to address the entire body detection problem (Table 3.5). Most datasets had varying numbers of instances (animals) per image, and there were up to 181 instances presented in a single image (in D15) while the minimum number was zero (no instance in the image; D13). For bounding box area (referred to the size of bounding boxes in pixels), the size varied in datasets. Given a fixed resolution, a small bounding box area implies that individuals/objects are small relative to the image, while a larger area means that the animal portion of the image is larger. Bounding box areas are relatively large in D1, while in D15 objects are extremely small relative to the entire

image. Interestingly, D1 has the largest number of annotated images, while D15 has the least number of images annotated. Computer code for implementation is available for three datasets (D1, D4, and D12).

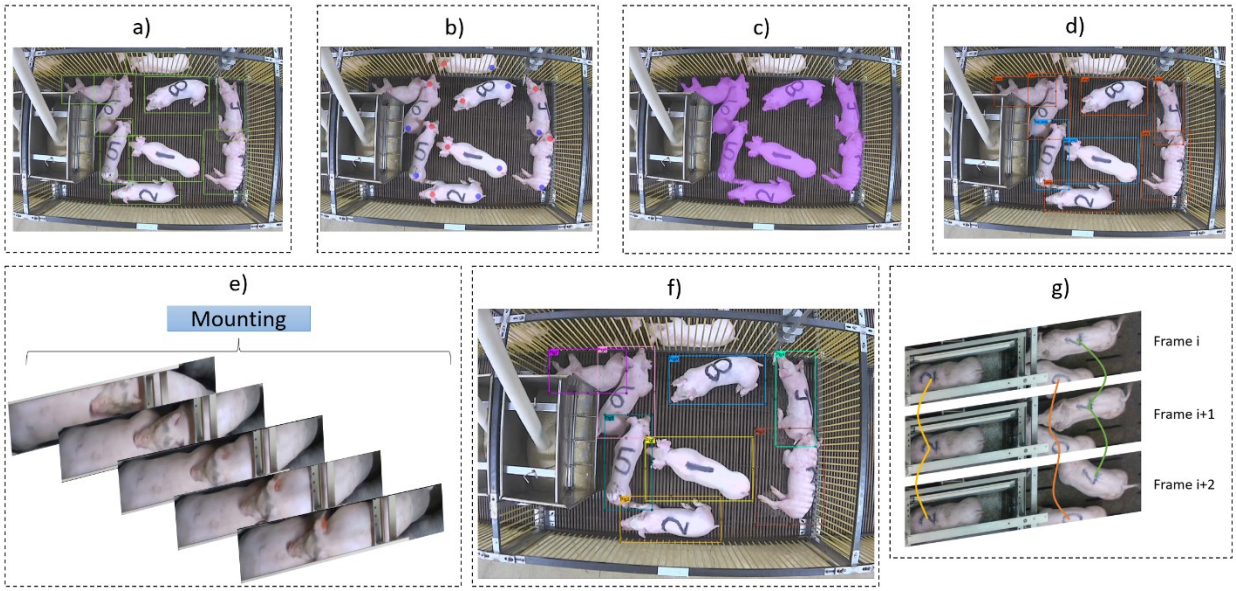

Figure 3.1 Examples of image data and key annotations for different computer vision tasks. Panel a) shows an example for entire body detection where each pig is enclosed in a bounding box. Panel b) presents an instance for body part detection, where heads of pigs are marked in red and rear parts of pigs are marked in blue. Panel c) shows an example of segmentation where each pig has a polygon mask. Panel d) presents an example of behavior recognition through an individual image, where lying pigs are enclosed in red bounding boxes and blue bounding boxes indicate pigs that are not lying. Panel e) is an example of behavior recognition by assigning a label to an image sequence. Panel f) shows an example of animal identification where each individual is assigned with a bounding box and a unique ID label. Panel g) displays an example of a tracklet across three consecutive frames.

Table 3.5 Identified public datasets for animal entire body detection via computer vision. Code availability: whether computer code is available for entire body detection. *: an annotated image is considered as an image paired with an external file that includes manually annotated bounding box coordinates. Varying resolutions mean that there are more than two resolutions in the dataset.

Dataset	Species	# Instances per image	Bounding box area (in pixels)	Image resolution	# Annotated images*	Code availability (yes/no)
D1	Pig	1-11	Approximately 0-10,000	640×360	113,079	Yes
D2	Pig	8	Approximately 60,000-90,000	1280×720	7,200	No
D3	Pig	1-48	Approximately 1-86,400	1280×720 and 640×480	305	No
D4	Pig	1-30	Approximately 2,200-508,000	3840×2160	380	Yes
D12	Cattle	1-8	Approximately 0-300,000	Varying resolutions	7,043	Yes
D13	Cattle	0-16	Approximately 5,959-7,830	3000×4000	670	No
D14	Cattle	1-5	Approximately 100,000	1280×720	10,402	No
D15	Cattle	10-181	Approximately 400-1,600	3000×4000 and 3840×2160	89	No

4.3.2 Body part detection

Another subdomain of object detection is body part detection, where key points of animals e.g., heads and hips are detected and coordinated (Figure 3.1b). This CV task is also referred to as landmark detection (Wu and Ji, 2019). Training a model for body part detection requires that each data point contains an image with the object(s) of interest and annotations have been made which specify coordinates of all possible key points or landmarks for each visible object. In general, researchers define repeatable and distinctive key points to extract reliable features across images and thus, it is important to explicitly list the body parts in a detection problem.

Four public animal datasets (three for pigs and one for cattle) were identified for body part detection (Table 3.6). All three pig datasets (D5, D6, and D7) were collected using a top-down view and thus, body parts on back of the pig were annotated (Table 3.6). The annotated key points of pigs include head, tail, shoulder, ears, and snout. Images for the cattle dataset (D16) were collected in a side view which had 16 key points annotated for each instance (Table 3.6). The number of annotated images among the four datasets ranged from 668 to 135,000. In addition, D5 and D6 have multiple pigs per image, and all visible instances were annotated, while in D7 and D16 only one animal was annotated per image.

Table 3.6 Identified public datasets for animal body part detection via computer vision. *: an annotated image is considered as an image paired with an external annotation file. Code availability: whether computer code is available for body part detection.

Dataset	Species	Camera view	Key point(s)	# Annotated images*	Code availability (yes/no)
D5	Pig	Top-down view	1. Tail, 2. Shoulder, 3. Left ear, and 4. Right ear	2,000	No
D6	Pig	Top-down view	1. Shoulder and 2. tail	135,000	No
D7	Pig	Top-down view	1. Tip of nose, 2. Head, and 3. Tail	668	Yes
D16	Cattle	Side view	1. Head, 2. Neck, 3. Spine, 4. Right front thigh root, 5. Right front knee, 6. Right front hoof, 7. Left front thigh root, 8. Left front knee, 9. Left front hoof, 10. Coccyx, 11. Right hind thigh root, 12. Right hind knee, 13. Left hind hoof, 14. Left hind thigh root, 15. Left hind knee, and 16. Left hind hoof	2,134	No

4.3.3 Segmentation

As a natural next step to object detection, segmentation makes prediction inferring labels (objects) at pixel levels to achieve fine-grained inference (Garcia-Garcia et al., 2017) i.e., the segmentation distinguishes animals from the background. Development of a segmentation model requires both images and polygon masks for all instances presented in the image set. Compared to bounding boxes, polygon masks are generally more precise (Figure 3.1c).

Currently, there is only one public animal dataset available for implementing segmentation (D9). For D9, Tangirala et al. (2021) added polygon mask annotations to RGB images, where the polygon was used to define the shape and edges of each instance (pig). Instead of annotating single image files, Tangirala et al. (2021) selected a set frames from videos with the frame indices specified. A total of 540 annotated images with multiple pigs across complex scenes were provided. The count of pigs per image in D9 ranges from 13 to 37, and each instance has its unique polygon mask. In addition, Tangirala et al. (2021) made the computer code to automatically segment instances publicly available.

4.3.4 Behavior recognition

In CV, visual components can be used to detect and recognize objects in dynamic scenes, in order to learn and describe the behavior of object (Popoola and Wang, 2012). Some basic behaviors (e.g., standing and lying) can be recognized through a single image (Figure 3.1d), while more complex behaviors (e.g., mounting) require analyzing a set of images or an image sequence (Figure 3.1e). A particularly complex behavior recognition task is the recognition of animal interactions that are behavior actions involving at least two animals. Further, regions of interest (ROI) are necessary when there are multiple objects occurring in the same image and the action recognition must focus on a specific part of the image. In some cases, the ROI can be the

whole image. Collectively, ROI, base analysis unit (single image or image sequence), and the behavior class/category for each instance are necessary for a behavior recognition dataset.

We identified six public datasets for animal behavior recognition (five for pigs and one for cattle), which cover a wide range of behaviors in pigs and cattle (Table 3.7). For the original studies that utilized these datasets, analysis units included both single images and short video clips. D2, D7, D9, and D20 datasets consist of videos, and for each dataset a fraction of video frames was selected and annotated. For D1, D2, D3, and D9datasets, the annotated behaviors include basic behaviors (e.g., standing, lying, and sitting, etc.). Further, D2 and D7 contain several complex behaviors such as moving, investigating, and exploring. For the cattle dataset (D20), images were annotated focusing on cows' activities near a feeding station. In addition, D20 is provided in video format where each frame was assigned with a label that indicated cows' behavior near the feeder. In D1, D2, D3, D7, and D9, the ROIs were specified for each instance (i.e., the instance ROI rather than the entire image was used to analyze animal behavior). However, D20 specified the entire image as the ROI. Among the behavior recognition datasets, the number of annotated images has a wide range from 305 to 1,526,473. Four datasets (D1, D7, D9, and D20) were published along with computer code to implement behavior recognition. No datasets are available for recognition of animal-animal interactions.

Table 3.7 Identified public datasets for animal behavior recognition via computer vision. [i]: an annotated file is considered as an imagery file paired with an external annotation file. [ii]: the classes were not explicitly defined. Code availability: whether computer code is available for behavior recognition.

Dataset	Species	Behavior types	ROI	Analysis unit	# Annotated images/videos[i]	Code availability (yes/no)
D1	Pig	1. Standing, 2. Lateral lying, 3. Sternal lying, 4. Sitting, and 5. Drinking	Individual	Single image	113,079 images	Yes
D2	Pig	1. Eating, 2. Drinking, 3. Lying, 4. Standing, and 5. Moving	Individual	Single image + image sequence	7,200 images from 12 videos	No
D3	Pig	1. Lying and 2. Not lying	Individual	Single image	305 images	No
D7	Pig	1. Investigating and 2. Exploring	Individual	Image sequence	20 videos	Yes
D9	Pig	1. Sitting and 2. Standing	Individual	Single image	540 images from 29 videos	Yes
D20	Cattle	1. Frontal interaction, 2. Lateral interaction, 3. Vertical interaction, 4. Crowding, 5. Drinking, 6. Exploring, 7. Queueing, and 8. Normal	Entire image	Single image	1,526,473 images from 253 videos	Yes

4.3.5 Identification

Animal identification (ID) is an important research topic, as PLF aims to monitor animal space at individual levels. Identification can be considered as a classification problem where each individual/instance is assigned with an ID label (Figure 3.1f). To develop CV for ID classification, a data point should at minimum include an image containing the relevant object and the ID label for the object. In more complex scenes, an image may include multiple individuals. In that case, ROI and ID label are required for every visible individual in a given image. Bounding boxes, body parts, or polygon masks can be used to indicate the ROI of the individual. In general, a large number of ID classes poses challenges on the predictive ability of ID models, as identifying an individual animal in a group of two is easier to identifying that animal in a group of ten. Therefore, we recorded the number of ID classes in the reviewed datasets.

We found nine public datasets (two for pigs and seven for cattle) for animal identification (Table 3.8). The two pig datasets (D6 and D9) consist of videos, and selected frames were

annotated for animal ID. Different from pig ID datasets, all cattle ID datasets provided single images rather than video frames. We need to point out that the ID applications used a single image as the base analysis unit and thus, we reviewed the number of annotated images rather than the number of videos (Table 3.8). Both D6 and D9 contain multiple pigs per image (i.e., ROIs of individuals needed to be determined), and each individual was assigned with an ID label. However, the number of ID classes in D6 and D9 do not represent the total number of observed pigs in the two datasets. For instance, 16 ID classes in D6 were annotated for all videos, but the ID classes were repeatedly used across different pig social groups (i.e., the same ID label was reused for different individuals). Furthermore, in D9, instance ID labels were used rather than unique pig IDs (i.e., the same individuals might have been assigned with new instance IDs when annotating a different video). Images in the cattle ID datasets contained one animal per image (i.e., the entire image was considered as the ROI), and each image was assigned with an animal ID. For the cattle ID datasets, the number of ID classes equaled the number of observed animals. Overall, the image sample size of the animal ID datasets ranged from 294 to 135,000. Only two of the studies (Andrew et al., 2021; Tangirala et al., 2021) published the computer code needed to reproduce the animal identification model.

Table 3.8 Public datasets for animal identification via computer vision. [i]: if an individual as the ROI means that the individual animal is first localized and then identified; otherwise, an ID class label is assigned to the entire image. [ii]: an annotated image is considered as an image assigned with an ID label. Code availability: whether computer code is available for identification.

Dataset	Species	# Classes	Unique ID for each animal? (yes/no)	ROI[i]	# Annotated images[ii]	Code availability (yes/no)
D6	Pig	16	No	Individual	135,000	No
D9	Pig	NA	No	Individual	540	Yes
D10	Cattle	89	Yes	Entire image	940	No
D11	Cattle	23	Yes	Entire image	46,340	No
D12	Cattle	46	Yes	Entire image	4,736	Yes
D14	Cattle	182	Yes	Entire image	32,020	No
D17	Cattle	300	Yes	Entire image	2,899	No
D18	Cattle	40	Yes	Entire image	294	No
D19	Cattle	136	Yes	Entire image	2,474	No

4.3.6 Tracking

In CV, tracking aims to detect and follow objects in image sequences, where a detector distinguishes the tracked object from local background (Soleimanitaleb et al., 2019; Stalder et al., 2009). Noteworthy, among all the studies that published animal tracking datasets, researchers employed tracking-by-detection methods i.e., their published datasets were prepared and annotated to develop tracking-by-detection models. Thus, in this subsection, we focus on data that are suitable for tracking-by-detection problems. In a tracking-by-detection approach, objects are first located in each frame and then formalized by frame-to-frame tracking (Özuysal et al., 2006). As the tracking-by-detection problem resorts to the object detection or segmentation problems that are based on single image analysis (see Sections 3.3.1-3.3.3), ROI and instance label (or ID) of each object are annotated across frames in a video clip. The ROI can be in the form of bounding box, body parts, or polygon mask for each instance across the images. There is no inter-frame annotation required to develop a tracking-by-detection model, but frame indices need to be explicitly specified. In short, an image sequence and the annotated tracklet make up a data point (Figure 3.1g).

We identified five public datasets (D1, D2, D4, D6, and D9) that were used for animal tracking purposes. For D1, D2, and D6, ROI and the corresponding class label were assigned to each instance, and all video frames were annotated using pre-trained object detection/segmentation models. In D1, there are 4,718 videos, where each video is approximately 9 minutes long. In D2, a total of 3,429,000 frames from 1,891 video clips are available. D6 contains 135,000 annotated frames from 15 videos. D4 and D6 are two datasets that used manual annotations. Note that Shirke et al. (2021b) annotated every 15^{th} frames in D4 instead of annotating consecutive frames, resulting in a total of 1,200 manually annotated frames from 10

videos available for tracking. In D9, 540 annotated frames from 30 video clips are available. Two studies (Shirke et al., 2021b; Tangirala et al., 2021) published computer code along with the datasets (D4 and D9) to implement animal tracking.

4.4 Validation strategy

In CV development, the whole dataset is generally split into a training set and a validation set. Once data are split, the training set is used for model fitting and the validation set is used to evaluate the performance of the trained model. Different strategies can be used to split the entire dataset. In most CV applications to animal farming, the training set and its corresponding validation set are split at random. However, dependence structures may exist in the data e.g., temporal and spatial structures, which violates the assumption of independence between the training set and validation set and leads to overoptimistic results (Roberts et al., 2017). Therefore, it is reasonable to assume that how data are split can influence algorithm evaluation (Li et al., 2021). In this section, we review validation strategies of the studies that originally used the public animal datasets to develop CV. Here, we define the validation strategy as the way of splitting the training and validation sets.

Overall, there are four types of validation strategies used to split data: random validation, stratified random validation, blocked validation, and in-sample validation. Random validation means that the training and validation sets are split at random to achieve a given ratio. An 80-20 split is the common practice (i.e., 80% of data are used for model training and the remaining 20% are used for model evaluation/validation). Stratified random validation means that researchers may randomly select training and validation samples in the same (or similar) proportion as the samples appear in the population. For instance, if a pig dataset includes three production stages: weaning, nursery, and finishing, there are three strata based on the production

stages. Within each stratum, a proportion of data are sampled for the training set and the remaining data are sampled for the validation set. The stratified random validation is different from random validation as data are stratified in the training and validation sets. Another type of validation, namely blocked validation, is less common but important to evaluate model robustness. Blocked validation means that the training and validation sets are split given a blocking factor such as time or location. For instance, a model can be trained using data collected from one farm and then tested on data from another farm. In this case, the blocking factor is farming unit. The last type of validation is in-sample validation, where the same data used for model training are also used for model validation.

Table 3.9 shows validation strategies used in the studies that originally analyzed and published the datasets reviewed in this paper. The dataset column uses shortcuts defined in Table 3.2. Note that some datasets may correspond to more than one CV task and more than one validation strategy. Most applications evaluated their models using random validation and stratified random validation. A few datasets (D1, D3, D5, D13, D14) were used for blocked validation. In-sample validation was only used in tracking tasks.

Table 3.9 Validation strategies used for reviewed datasets in their original applications.

Validation strategy	Computer vision task	Dataset
Random validation	Entire body detection	[D3, D4, D10, D11, D12, D13, D15]
	Body part detection	[D8, D16, D17]
	Segmentation	D9
	Behavior recognition	[D3, D7, D9]
	Identification	[D9, D10, D12, D18]
	Tracking	D9
Stratified random validation	Entire body detection	[D1, D2, D14]
	Behavior recognition	[D1, D2]
	Identification	[D11, D17, D19]
Blocked validation	Entire body detection	[D1, D3, D13]
	Body part detection	D5
	Behavior recognition	[D1, D3]
	Identification	D14
In-sample validation	Tracking	[D1, D2, D4, D6, D8]

Evaluation metrics reported by relative publications are summarized in Table 3.10. A few studies developed object detection models and meanwhile assigned a behavior class to the object, leading to the similar evaluation metrics for both CV tasks. According to the CV task, the authors reported different metrics to evaluate the model performance. In Table S3.2, we provide brief explanation of the evaluation metrics that are addressed in Table 3.10.

It is noted that in three studies, metrics obtained from the blocked validation strategy was compared with random validation or stratified random validation. In a behavior recognition application that used D1, Alameer et al. (2020) obtained a mean average precision (mAP; see Table S3.2) of 98.0% from stratified random validation, and the mAP of blocked by replicate (at different time points) validation decreased by 1.0% when the training and validation sets were collected at different experiments and at different time, respectively. Furthermore, Riekert et al. (2020) collected their data (D3) to a develop entire body detection model, and they used both blocked-by-pen validation and random validation. The reported mAPs for blocked-by-pen validations ranged from 76.8% to 87.4%, while mAPs obtained from random validation were between 67.7% and 87.2%. Using the same dataset, Riekert et al. (2020) also developed behavior recognition models using the two validation strategies, and the mAPs for blocked-by-pen validation were between 44.8% and 80.2%, while mAPs for random validation ranged from 49.2% to 80.9%. Moreover, Shao et al. (2020) collected D13 for cattle detection and used both blocked validation and random validation. In blocked validation of the study (Shao et al., 2020), the validation set contained data that were collected on a different day and area compared to the training set. Compared to random validation, precision, recall, and F1 obtained from blocked-by-time-and-location validation decreased by 16.7%, 28.3%, and 23.0%, respectively (Shao et al., 2020).

Table 3.10 Evaluation metrics by computer vision tasks and validation strategies. A range is provided if more than one point estimate were reported for the specific validation strategy.

Computer vision task	Validation strategy	Dataset	Evaluation metrics
Entire body detection	Random validation	D3	mAP: 67.7-87.2%
		D4	mAP: 99.5%, average IoU: 80.52%
		D10	mAP: 99.0-99.6%
		D11	mAP: 99.0-99.6%
		D12	mAP: 96.6-99.9%
		D13	precision: 94.1-95.7%, recall: 94.4-94.6%, F1: 94.3-95.2%, AUC: 86.9-92.2%
		D15	mAP: 83.0-89.3%
	Stratified random validation	D1	mAP: 98.0%
		D2	mAP: 84.6-100%, TP: 89.2-100%, FP: 0-10.8%, missing rate: 0-2.2%
		D14	mAP: 97.3-98.0%
	Blocked validation	D1	mAP: 97.0% (blocked by replicate and time)
		D3	mAP: 76.8-87.4% (blocked by pen)
		D13	precision: 77.4%, recall: 66.1%, F1: 71.3% (blocked by time and location)
Body part detection	Random validation	D8	recall: 94.2%, precision: 95.4%, F1: 95.1%
		D16	PCKh@0.5: 83.9-97.2%
		D17	TP: 99.1%, FP: 0, TN: 100%, FN: 89.0%, accuracy: 99.13%
	Blocked validation	D5	recall: 96.0%, precision: 100%, F1: 98.0% (blocked by time); recall: 66.7%, precision: 91.1%, F1: 77.1% (blocked by pen)
Segmentation	Random validation	D9	mAP: 69.0-92.0%
Behavior recognition	Random validation	D3	mAP: 49.2-80.9%
		D7	accuracy: 93.0-95.0%, mAP: 89.0-96.0%
		D9	AUC: 98.5%
	Stratified random validation	D1	mAP: 98.0%
		D2	accuracy: 63.0-89.0%
	Blocked validation	D1	mAP: 97.0% (blocked by replicate and time)
		D3	mAP: 44.8-80.2% (blocked by pen)
Identification	Random validation	D9	CMC-1: 77.1%, CMC-5: 89.5%, CMC-10: 93.9%
		D10	accuracy: 84.9-87.2%
		D12	accuracy: 90.5%-95.6%
		D18	accuracy: 97.0%
	Stratified random validation	D11	accuracy: 98.1%
		D17	accuracy: 97.3-99.1%
		D19	precision: 98.1%, recall: 97.7%, F1: 97.9%
	Blocked validation	D14	Top-1: 57.0%, Top-2: 71.8%, Top-4: 76.9%, Top-8: 79.7%, Top-16: 81.8% (blocked by time)
Tracking	Random validation	D9	cMOTSA: 75.6-77.8%
	In-sample validation	D1	MOTA: 94.0%, MOTP: 80.0%
		D2	MOTA: 76.8-100%, IDF1: 55.1-100%
		D4	IDF1: 53.2-66.1%, IDP: 49.9-61.8%, IDR: 56.9-71.0%, MOTA: 55.2-80.6%, MOTP: 44.5-61.3%
		D6	precision: 82.5%-97.2%
		D8	MOTA: 94.4%

For entire body detection, body part detection, segmentation, and behavior recognition applications, mAP is the commonly used metric, making it comparable between studies. For the

studies that reported mAP, the metric ranged from 44.8% to 99.9%. Notably, studies that used blocked validations tended to report lower mAPs, while high mAPs concentrate on those used random validations and stratified random validations. Most identification applications reported accuracies that ranged from 57.0% to 99.1%. Again, the lowest accuracy was yielded in blocked validation. For the tracking task, multiple objects tracking accuracy (MOTA) is the most frequently reported metric, and it ranged from 55.2% to 94.4% in the examined studies.

5. DISCUSSION

Deep learning-based CV algorithms have made significant progress in PLF-relevant applications. However, the scarcity of public image data for livestock is still a bottleneck, as DL applications require a large number of training samples. To the best of our knowledge, this is the first review that comprehensively investigates publicly available imagery datasets that could be used in PLF. We believe that this review contributes to the PLF community by presenting a compilation of public resources.

To date, there are several reviews for state-of-the-art CV applications in PLF (Borges Oliveira et al., 2021; Chen et al., 2021; Li et al., 2021). Their reviews focused on the perspectives of algorthims and DL methodology i.e., the literature search logic was algorithm-oriented and application-oriented. Two of the reviews (Borges Oliveira et al., 2021; Li et al., 2021) reported public imagery datasets for pigs and cattle, which were subsets of the identified datasets in this study (except one dataset that did not satisfy the inclusion criteria of this study). However, this review complement literature as we reviewed a larger collection of public datasets for pigs and cattle, compared to the studies of Li et al. (2021) and Borges Oliveira et al. (2021). Li et al. (2021) specifically reviewed convolutional neural network-based CV systems in

livestock farming and listed five public datasets for cattle and three for pigs. They specified CV tasks, resolution, number of images, and annotations for each dataset. Furthermore, Borges Oliveira et al. (2021) reviewed DL algorithms applied to CV systems in livestock farming and found seven datasets for cattle and one for pig, among which the CV tasks and image types were specified. Nevertheless, the details provided in those two reviews are limited, and there are still gaps about how data can be used in different validation strategies through data split. In this review, we consider data as the fuel of DL-based CV algorithms. Instead of algorithms, we deliberately searched public image datasets in PLF and investigated the data structure and how predictive performance of CV varies in different validation strategies using the available data.

In livestock farming, animals can be categorized based on their functional characteristics e.g., weaning pigs, growing pigs, and reproductive sows in swine farming (Puppe et al., 2008). However, few datasets are targeted at addressing the diversity of production stages of animals. Many PLF tasks such as abnormality detection generally take a time span of several weeks or months throughout a production cycle, which would require collecting images/videos over multiple production cycles to fully capture morphological features of the animals. Further, the reviewed datasets are rather small-scale in terms of environmental factors e.g., different farm conditions and sites. Therefore, more attention is needed to fill the gap when new datasets are created to account for the diversity and variation across different farms and production stages.

All the identified 20 public datasets involve RGB images/videos, which indicates the prevalence of the RGB images in CV for PLF applications. Most datasets contain images in top-down views or angled-down views, limiting the visual components or feature space to the upper body part of the animals. However, this prevalence does not necessarily mean that the top-down view or angled-down view is favorable. Some CV tasks e.g., landmark detection and behavior

recognition would require other camera views. For instance, the frontal view is the most useful view for facial landmark recognition (Shojaeipour et al., 2021). However, there is only one public dataset available for facial recognition in cattle (D17) and none that are suitable for developing a pig facial recognition model (as images in all pig datasets are of top-down view and/or angled-down view).

Across most of the focused CV tasks, ROI is the essential annotation process. In entire body detection datasets, the ratio of bounding box area relative to the entire image varies significantly between datasets i.e., the relative size of instances in the image differs. This is informative to researchers, especially when designing the architecture of anchor-based detectors e.g., YOLO and Faster R-CNN that are state-of-the-art algorithms for object detection (Liu et al., 2016; Redmon and Farhadi, 2018; Ren et al., 2015). For other CV tasks that require detecting objects e.g., simple behavior recognition through individual images and tracking-by-detection, the object size (relative to the entire image) also matters.

The reviewed datasets contribute to a wide range of CV tasks. Although a dataset might be originally collected for a given CV task, the same imagery data can be used for other purposes. As shown in Figure 3.1 (panels a, b, c, and d), the same footage can be annotated in different ways depending on the purposes. Similarly, if images from a public dataset are found to be valuable, researchers can create new annotations of the images, regardless of the CV task for which the dataset was originally collected. Reannotated images will bring additional value to the public dataset.

Most CV applications in PLF use random validation or stratified random validation for model assessment. But results from random validations can be overoptimistic, and random validation is less representative of real-life validation scenarios, as environments for capturing

images are quite complex in animal farming (Li et al., 2021). In practice, developers or researchers are interested in how CV is validated broadly in a way that examines how well the model can be generalized to other contexts (e.g., across different seasons and different farms), which is closer to blocked validation. Therefore, when there exist blocking factors e.g., time and farm unit in an imagery dataset, blocked validation can be utilized. We expect that block validation will yield a lower, but more realistic estimate, of the predictive performance of the CV application in practical animal farming contexts.

Lastly, some recommendations are provided for creating/sharing public CV datasets in PLF and the use/reuse of the datasets.

To create new public image datasets for the development of PLF, we recommend that both images and ground-truth annotation files be accessible to the community. In addition, we encourage researchers to share the rubrics or ethograms they used for the annotation process. Specifying the metadata e.g., recording schedule, long-term temporal design, farming units, and animal subject attributes (age, weight, and coat color) etc. will bring additional value to the dataset. To share data, we recommend separating the raw data from the annotated data through organizing them into different data repositories, as most researchers are more interested in the annotated set. Furthermore, timestamps or watermarks should be avoided in the shared data. Although not required, we encourage researchers to share the computer code for implementation and ethical use and approval obtained for the original experiment.

Current public datasets for animal behavior recognition only involve individual behaviors, while image datasets dedicated to animal social interactions (along with annotations) are not yet available. Researchers may revisit the existing public image datasets where groups of

animals were recorded (e.g., D2 and D6) and reannotate for animal social behaviors. Alternatively, more efforts could be made to create new datasets for animal-animal interactions.

Most importantly, blocked validation is recommended as an alternative or additional strategy to random validation when developing CV applications, as the results obtained from blocked validation tended to be lower compared to the random validation or stratified random validation. Blocking factors, that could be utilized to split the training and validation data, may include but are not are limited to: housing units or pens (Psota et al., 2019), locations (Riekert et al., 2020), replicates at different time points (Alameer et al., 2020), and/or combinations of multiple blocking factors (Shao et al., 2020). This will be helpful for the evaluation of generalizability and reproducibility of the model. Finally, this review will help researchers combine public datasets if the datasets address the same problem. Furthermore, researchers can combine multiple datasets for CV model development and perform blocked validation, where the dataset is the blocking factor.

6. CONCLUSION

In PLF, publicly available image data are valuable, and the reuse of the public datasets is important as it reduces the effort required to collect and annotate images/videos. This review fills a gap in PLF literature, as it is the first review that comprehensively investigates publicly available imagery datasets for CV development in PLF. We identified 20 public datasets, nine of which focused on pigs and 11 on cattle. The reviewed datasets are related to six CV tasks including entire body detection, body part detection, segmentation, behavior recognition, identification, and tracking. Moreover, we reviewed and classified the related CV applications by validation strategies. We observed a general trend that blocked validation yields lower (but more

realistic) performance than the commonly used validation strategies of random validation and stratified random validation.

APPENDIX

Table S3.1 Website or URLs of publicly available animal datasets for computer vision.

Dataset name	URL (Accessed on May 2022)
Newcastle Pig Posture	https://figshare.com/articles/dataset/Automated_recognition_of_postures_and_drinking_behaviour_for_the_detection_of_compromised_health_in_pigs/13042619/1
Edinburgh Pig Behavior	https://homepages.inf.ed.ac.uk/rbf/PIGDATA/#:~:text=The%20pig%20behavior%20dataset%20consisting,Most%20frames%20show%208%20pigs.
Pig Position and Posture	https://wi2.uni-hohenheim.de/analytics
ISRL Multi-Camera Tracking	https://drive.google.com/drive/folders/1E2wW2aRENgy_TqlzflCn58ahbTHVIaK6
Pig Detection	https://uofnelincoln-my.sharepoint.com/personal/epsota2_unl_edu/_layouts/15/onedrive.aspx?id=%2Fpersonal%2Fepsota2%5Funl%5Fedu%2FDocuments%2FMDPIdatasets%2FPigDetectionDataset2019%2Ezip&parent=%2Fpersonal%2Fepsota2%5Funl%5Fedu%2FDocuments%2FMDPIdatasets&ga=1
Pig tracking	https://uofnelincoln-my.sharepoint.com/personal/epsota2_unl_edu/_layouts/15/onedrive.aspx?id=%2Fpersonal%2Fepsota2%5Funl%5Fedu%2FDocuments%2FAnnotatedVideos%2Ezip&parent=%2Fpersonal%2Fepsota2%5Funl%5Fedu%2FDocuments&ga=1
Pig Novelty Preference	https://drive.google.com/drive/folders/14XUYxM15NAI-zBrntrmQofhLv5otAw5b
Pig Detection and Tracking	https://github.com/MartinWut/Supp_DetAnIn
PigTrace	https://drive.google.com/file/d/1s-bCnABh2Hef5l5OxydcY-tkPbrUGSjj/view
FriesianCattle2017	https://research-information.bris.ac.uk/en/datasets/friesiancattle2017
AerialCattle2017	https://research-information.bris.ac.uk/en/datasets/aerialcattle2017
OpenCows2020	https://data.bris.ac.uk/data/dataset/10m32xl88x2b61zlkkgz3fml17
Aerial Pasture	http://bird.nae-lab.org/cattle/
Cows2021	https://github.com/Wormgit/Cows2021
Aerial Livestock	https://github.com/hanl2010/Aerial-livestock-dataset/releases
NWAFU-Cattle	https://github.com/MicaleLee/Database
300 Cattle	https://cloud.une.edu.au/index.php/s/eMwaHAPK08dCDru
FriesianCattle2015	https://data.bris.ac.uk/data/dataset/eac634de-4b97-4dcc-ab78-66e3c9d09294
Holstein Cattle Recognition	https://dataverse.nl/dataset.xhtml?persistentId=doi:10.34894/O1ZBSA
Cow Behavior	https://zenodo.org/record/3981400#.Yq-ChHbMJD9

Table S3.2 Metrics for performance evaluation in different validation strategies.

Metric	Full name of the metric	Concise explanation
Accuracy	-	Proportion of correct predictions
TP	True positive	An outcome where the model correctly predicts the positive class (in binary classification)
TN	True negative	An outcome where the model correctly predicts the negative class (in binary classification)
FP	False positive	An outcome that a negative class is incorrectly predicted as positive
FN	False negative	An outcome that a positive class is incorrectly predicted as negative
precision (p)	-	A fraction of relevant instances among the retrieved instances
recall (r)	-	A fraction of relevant instances that were retrieved
F1	-	$F1 = \frac{2 \times precision \times recall}{precision + recall}$
AP	Average Precision	Area under the precision-recall curve. $AP = \int_0^1 p(r)dr$
mAP	mean Average Precision over N classes	$mAP = \frac{1}{N}\sum_{i=1}^{N} AP_i$
IoU	Intersection over Union	Given two areas, $IoU = \frac{Area\ of\ overlap}{Area\ of\ union}$
AUC	Area under ROC curve	AUC measures the 2d area under neath the receiver operating characteristic curve, which is a comprehensive evaluation of classification performance
missing	Missing rate	A fraction of missing detections
PCKh@0.5	Percentage of correct key-points when threshold=0.5	An evaluation metric for pose estimation that a detected joint is considered correct at the distance between the predicted and true joint is within a threshold (e.g., 0.5).
CMC-N	Cumulative Matching Characteristics	A measure of 1:N identification system performance. Detailed description is provided by Bolle et al. (2005).
Top-N accuracy	-	Model predictions with N highest probabilities. If one of N labels is a true label, it classifies the prediction as correct.
cMOTSA	Constrained multi-object tracking and segmentation accuracy	$cMOTSA = \widehat{TP}/(\lvert TP\rvert + \lvert FP\rvert)$, where $\widehat{TP}$ denotes soft TP. See the study of Tangirala et al. (2021) for details.
MOTA	Multiple objects tracking accuracy	A metric that measures the overall accuracy of both the tracker and detection. See the study of Alameer et al. (2020) for details.
MOTP	Multiple objects tracking precision	A measure to evaluate multiple object tracking. It is defined in the study of Alameer et al. (2020).
IDF1	Multi-object identification F1 score	Ratio of correctly identified detections over the average number of ground-truth and computed detections
IDP	Multi-object identification precision	Fraction of computed detections that are correct.
IDR	Multi-object identification recall	Correctly identified ground truth detections.

REFERENCES

REFERENCES

Alameer, A., Kyriazakis, I., Bacardit, J., 2020. Automated recognition of postures and drinking behaviour for the detection of compromised health in pigs. Sci. Rep. 10, 1–15. https://doi.org/10.1038/s41598-020-70688-6

Andrew, W., Gao, J., Mullan, S., Campbell, N., Dowsey, A.W., Burghardt, T., 2021. Visual identification of individual Holstein-Friesian cattle via deep metric learning. Comput. Electron. Agric. 185, 106133. https://doi.org/10.1016/j.compag.2021.106133

Andrew, W., Greatwood, C., Burghardt, T., 2017. Visual Localisation and Individual Identification of Holstein Friesian Cattle via Deep Learning. Proc. - 2017 IEEE Int. Conf. Comput. Vis. Work. ICCVW 2017 2018-Janua, 2850–2859. https://doi.org/10.1109/ICCVW.2017.336

Andrew, W., Hannuna, S., Campbell, N., Burghardt, T., 2016. Automatic individual holstein friesian cattle identification via selective local coat pattern matching in RGB-D imagery. Proc. - Int. Conf. Image Process. ICIP 2016-Augus, 484–488. https://doi.org/10.1109/ICIP.2016.7532404

Benitez Pereira, L.S., Koskela, O., Pölönen, I., Kunttu, I., 2020. Data set of labeled scenes in a barn in front of automatic milking system [WWW Document]. Zenodo. https://doi.org/10.5281/zenodo.3981400

Berckmans, D., 2017. General introduction to precision livestock farming. Anim. Front. 7, 6–11. https://doi.org/10.2527/af.2017.0102

Bergamini, L., Pini, S., Simoni, A., Vezzani, R., Calderara, S., Eath, R.B.D., Fisher, R.B., 2021. Extracting accurate long-term behavior changes from a large pig dataset. VISIGRAPP 2021 - Proc. 16th Int. Jt. Conf. Comput. Vision, Imaging Comput. Graph. Theory Appl. 5, 524–533. https://doi.org/10.5220/0010288405240533

Bhole, A., Falzon, O., Biehl, M., Azzopardi, G., 2019. A Computer Vision Pipeline that Uses Thermal and RGB Images for the Recognition of Holstein Cattle, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). Springer International Publishing. https://doi.org/10.1007/978-3-030-29891-3_10

Bolle, R.M., Connell, J.H., Pankanti, S., Ratha, N.K., Senior, A.W., 2005. The relation between the ROC curve and the CMC. Proc. - Fourth IEEE Work. Autom. Identif. Adv. Technol. AUTO ID 2005 2005, 15–20. https://doi.org/10.1109/AUTOID.2005.48

Borges Oliveira, D.A., Ribeiro Pereira, L.G., Bresolin, T., Pontes Ferreira, R.E., Reboucas Dorea, J.R., 2021. A review of deep learning algorithms for computer vision systems in livestock. Livest. Sci. 253, 104700. https://doi.org/10.1016/j.livsci.2021.104700

Chen, C., Zhu, W., Norton, T., 2021. Behaviour recognition of pigs and cattle: Journey from computer vision to deep learning. Comput. Electron. Agric. 187, 106255. https://doi.org/10.1016/j.compag.2021.106255

Chen, C., Zhu, W., Steibel, J., Siegford, J., Han, J., Norton, T., 2020a. Recognition of feeding behaviour of pigs and determination of feeding time of each pig by a video-based deep learning method. Comput. Electron. Agric. 176, 105642. https://doi.org/10.1016/j.compag.2020.105642

Chen, C., Zhu, W., Steibel, J., Siegford, J., Wurtz, K., Han, J., Norton, T., 2020b. Recognition of aggressive episodes of pigs based on convolutional neural network and long short-term memory. Comput. Electron. Agric. 169, 105166. https://doi.org/10.1016/j.compag.2019.105166

Gao, J., Burghardt, T., Andrew, W., Dowsey, A.W., Campbell, N.W., 2021. Towards Self-Supervision for Video Identification of Individual Holstein-Friesian Cattle: The Cows2021 Dataset.

Garcia-Garcia, A., Orts-Escolano, S., Oprea, S., Villena-Martinez, V., Garcia-Rodriguez, J., 2017. A Review on Deep Learning Techniques Applied to Semantic Segmentation 1–23.

Han, J., Gondro, C., Reid, K., Steibel, J.P., 2021. Heuristic hyperparameter optimization of deep learning models for genomic prediction. G3 Genes|Genomes|Genetics 11, jkab032. https://doi.org/10.1093/g3journal/jkab032

Han, L., Tao, P., Martin, R.R., 2019. Livestock detection in aerial images using a fully convolutional network. Comput. Vis. Media 5, 221–228. https://doi.org/10.1007/s41095-019-0132-5

Jia Deng, Wei Dong, Socher, R., Li-Jia Li, Kai Li, Li Fei-Fei, 2009. ImageNet: A large-scale hierarchical image database 248–255. https://doi.org/10.1109/cvprw.2009.5206848

Lecun, Y., Bengio, Y., Hinton, G., 2015. Deep learning. Nature. https://doi.org/10.1038/nature14539

Li, D., Chen, Y., Zhang, K., Li, Z., 2019. Mounting behaviour recognition for pigs based on deep learning. Sensors (Switzerland) 19. https://doi.org/10.3390/s19224924

Li, G., Huang, Y., Chen, Z., Chesser, G.D., Purswell, J.L., Linhoss, J., Zhao, Y., 2021. Practices and applications of convolutional neural network-based computer vision systems in animal farming: A review. Sensors 21, 1–42. https://doi.org/10.3390/s21041492

Li, X., Cai, C., Zhang, R., Ju, L., He, J., 2019. Deep cascaded convolutional models for cattle pose estimation. Comput. Electron. Agric. 164, 104885. https://doi.org/10.1016/j.compag.2019.104885

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L., 2014. Microsoft coco: Common objects in context, in: European Conference on Computer

Vision. Springer, pp. 740–755.

Liu, D., Oczak, M., Maschat, K., Baumgartner, J., Pletzer, B., He, D., Norton, T., 2020. A computer vision-based method for spatial-temporal action recognition of tail-biting behaviour in group-housed pigs. Biosyst. Eng. 195, 27–41. https://doi.org/10.1016/j.biosystemseng.2020.04.007

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., Berg, A.C., 2016. Ssd: Single shot multibox detector, in: European Conference on Computer Vision. Springer, pp. 21–37.

Lu, Z., Jiang, X., Kot, A., 2017. Enhance deep learning performance in face recognition. 2017 2nd Int. Conf. Image, Vis. Comput. ICIVC 2017 244–248. https://doi.org/10.1109/ICIVC.2017.7984554

Marcus, G., 2018. Deep Learning: A Critical Appraisal 1–27.

Nasirahmadi, A., Sturm, B., Edwards, S., Jeppsson, K.H., Olsson, A.C., Müller, S., Hensel, O., 2019. Deep learning and machine vision approaches for posture detection of individual pigs. Sensors (Switzerland) 19, 1–16. https://doi.org/10.3390/s19173738

Norton, T., Chen, C., Larsen, M.L.V., Berckmans, D., 2019. Review: Precision livestock farming: Building "digital representations" to bring the animals closer to the farmer. Animal 13, 3009–3017. https://doi.org/10.1017/S175173111900199X

Özuysal, M., Lepetit, V., Fleuret, F., Fua, P., 2006. Feature harvesting for tracking-by-detection. Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics) 3953 LNCS, 592–605. https://doi.org/10.1007/11744078_46

Ponti, M.A., Ribeiro, L.S.F., Nazare, T.S., Bui, T., Collomosse, J., 2017. Everything You Wanted to Know about Deep Learning for Computer Vision but Were Afraid to Ask. Proc. - 2017 30th SIBGRAPI Conf. Graph. Patterns Images Tutorials SIBGRAPI-T 2017 2018-Janua, 17–41. https://doi.org/10.1109/SIBGRAPI-T.2017.12

Popoola, O.P., Wang, K., 2012. Video-based abnormal human behavior recognition—A review. IEEE Trans. Syst. Man, Cybern. Part C (Applications Rev. 42, 865–878.

Psota, E.T., Mittek, M., Pérez, L.C., Schmidt, T., Mote, B., 2019. Multi-pig part detection and association with a fully-convolutional network. Sensors (Switzerland) 19, 1–24. https://doi.org/10.3390/s19040852

Psota, E.T., Schmidt, T., Mote, B., Pérez, L.C., 2020. Long-term tracking of group-housed livestock using keypoint detection and map estimation for individual animal identification. Sensors (Switzerland) 20, 1–25. https://doi.org/10.3390/s20133670

Puppe, B., Langbein, J., Bauer, J., Hoy, S., 2008. A comparative view on social hierarchy formation at different stages of pig production using sociometric measures. Livest. Sci. 113, 155–162. https://doi.org/10.1016/j.livsci.2007.03.004

Redmon, J., Farhadi, A., 2018. Yolov3: An incremental improvement. arXiv Prepr. arXiv1804.02767.

Ren, S., He, K., Girshick, R., Sun, J., 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. Adv. Neural Inf. Process. Syst. 28.

Riekert, M., Klein, A., Adrion, F., Hoffmann, C., Gallmann, E., 2020. Automatically detecting pig position and posture by 2D camera imaging and deep learning. Comput. Electron. Agric. 174. https://doi.org/10.1016/j.compag.2020.105391

Roberts, D.R., Bahn, V., Ciuti, S., Boyce, M.S., Elith, J., Guillera-Arroita, G., Hauenstein, S., Lahoz-Monfort, J.J., Schröder, B., Thuiller, W., Warton, D.I., Wintle, B.A., Hartig, F., Dormann, C.F., 2017. Cross-validation strategies for data with temporal, spatial, hierarchical, or phylogenetic structure. Ecography (Cop.). 40, 913–929. https://doi.org/10.1111/ecog.02881

Shao, W., Kawakami, R., Yoshihashi, R., You, S., Kawase, H., Naemura, T., 2020. Cattle detection and counting in UAV images based on convolutional neural networks. Int. J. Remote Sens. 41, 31–52. https://doi.org/10.1080/01431161.2019.1624858

Shirke, A., Golden, R., Gautam, M., Green-Miller, A., Caesar, M., Dilger, R.N., 2021a. Vision-based Behavioral Recognition of Novelty Preference in Pigs 1–5.

Shirke, A., Saifuddin, A., Luthra, A., Li, J., Williams, T., Hu, X., Kotnana, A., Kocabalkanli, O., Ahuja, N., Green-Miller, A., Condotta, I., Dilger, R.N., Caesar, M., 2021b. Tracking Grow-Finish Pigs Across Large Pens Using Multiple Cameras.

Shojaeipour, A., Falzon, G., Kwan, P., Hadavi, N., Cowley, F.C., Paul, D., 2021. Automated muzzle detection and biometric identification via few-shot deep transfer learning of mixed breed cattle. Agronomy 11. https://doi.org/10.3390/agronomy11112365

Soleimanitaleb, Z., Keyvanrad, M.A., Jafari, A., 2019. Object tracking methods: A review. 2019 9th Int. Conf. Comput. Knowl. Eng. ICCKE 2019 282–288. https://doi.org/10.1109/ICCKE48569.2019.8964761

Stalder, S., Grabner, H., Van Gool, L., 2009. Beyond semi-supervised tracking: Tracking should be as simple as detection, but not simpler than recognition. 2009 IEEE 12th Int. Conf. Comput. Vis. Work. ICCV Work. 2009 1409–1416. https://doi.org/10.1109/ICCVW.2009.5457445

Tangirala, B., Bhandari, I., Laszlo, D., Gupta, D.K., Thomas, R.M., Arya, D., 2021. Livestock Monitoring with Transformer.

Voulodimos, A., Doulamis, N., Doulamis, A., Protopapadakis, E., 2018. Deep Learning for Computer Vision: A Brief Review. Comput. Intell. Neurosci. 2018. https://doi.org/10.1155/2018/7068349

Wu, D., Sun, D.W., 2013. Colour measurements by computer vision for food quality control - A

review. Trends Food Sci. Technol. 29, 5–20. https://doi.org/10.1016/j.tifs.2012.08.004

Wu, Y., Ji, Q., 2019. Facial Landmark Detection: A Literature Survey. Int. J. Comput. Vis. 127, 115–142. https://doi.org/10.1007/s11263-018-1097-z

Wutke, M., Heinrich, F., Das, P.P., Lange, A., Gentz, M., Traulsen, I., Warns, F.K., Schmitt, A.O., Gültas, M., 2021. Detecting animal contacts—A deep learning-based pig detection and tracking approach for the quantification of social contacts. Sensors 21, 1–16. https://doi.org/10.3390/s21227512

Yao, Y., Yu, H., Mu, J., Li, J., Pu, H., 2020. Estimation of the gender ratio of chickens based on computer vision: Dataset and exploration. Entropy 22. https://doi.org/10.3390/e22070719

Zhang, K., Li, D., Huang, J., Chen, Y., 2020. Automated video behavior recognition of pigs using two-stream convolutional networks. Sensors (Switzerland) 20. https://doi.org/10.3390/s20041085

Zhao, Z.Q., Zheng, P., Xu, S.T., Wu, X., 2019. Object Detection with Deep Learning: A Review. IEEE Trans. Neural Networks Learn. Syst. 30, 3212–3232. https://doi.org/10.1109/TNNLS.2018.2876865

CHAPTER 4: EVALUATION OF COMPUTER VISION FOR DETECTING AGONISTIC BEHAVIOR OF PIGS IN A SINGLE-SPACE FEEDING STALL THROUGH BLOCKED CROSS-VALIDATION STRATEGIES

Junjie Han, Janice Siegford, Dirk Colbry, Raymond Lesiyon, Anna Bosgraaf, Chen Chen, Tomas Norton, and Juan P. Steibel

1. ABSTRACT

Agonistic behavior at feeding spaces is associated with both welfare and feed intake issues in swine farming. Studying interactive social behaviors of group-housed pigs provides valuable information to improve their production and welfare. The aims of this study were to 1) develop a deep learning pipeline based on convolutional neural network (CNN) and long short-term memory (LSTM) to classify videos depicting four types of interactive behavior between pigs in a single-space feeding stall and 2) validate the pipeline through various blocked validation strategies. Four categories of behaviors were classified in this study: head-to-body contact (including gentle nosing, casual contact between head/ears of a pig with a feeding pig, head knocking, tail biting, and pushing); levering where the feeding pig was lifted from behind by another pig; mounting in which the feeding pig was mounted by another pig; and no-contact when a second pig entered the feeding stall without physical contact with the feeding pig. Behavior at the feeding stall was filmed twice, three weeks apart, for two consecutive days each week using six groups of grow-finish pigs (10 per group) housed in pens equipped with FIRE® feeders. This resulted in a total of 15,679 30-frame video episodes for classification. The dataset presented a class-imbalance problem, and our deep learning pipeline addressed the problem by incorporating focal loss. Random cross-validation, blocking-by-time validation, and blocking-by-feeder validation were utilized for training-testing data split. The size of training sets was held constant (N=7,500) through all validation scenarios. The average testing accuracies were

0.968(±0.001), 0.860(±0.033), 0.766(±0.026), and 0.860(±0.010) for random cross-validation, blocking-by-time validation, and blocking-by-feeder validation (at Feeder 1 and Feeder 2), respectively. The results indicate that the proposed pipeline yielded acceptable predictive performance in random cross-validation. However, performance was substantially worse in blocking-by-time and blocking-by-feeder validations. More work is needed for algorithm generalization to improve its robustness across a variety of application scenarios. We provide public access to the dataset and the code.

2. INTRODUCTION

Understanding patterns of feeding behavior can be useful for pig management (Brown-Brandl et al., 2013), breeding (Ding et al., 2018) and research (Brown-Brandl et al., 2018; Salgado et al., 2021). In pig farming, animals are typically housed in groups and animals often have to compete for access to feeder space (Georgsson and Svendsen, 2002). Competition for feeder space may be especially intense with the single-space automatic feeders that are typically used in pig feed efficiency studies in grow-finish pigs. Moreover, the way pigs interact at the feeder with their group mates may affect growth and feed intake due to differential competition for feeder access (Georgsson and Svendsen, 2002; Nielsen et al., 1995). We have demonstrated previously that accounting for interactions between pigs during feeding events brings important information into pig research and breeding, because it allows more accurate estimation of social genetic effects of competition for feeder space (Angarita et al., 2021). Also, quantifying interactions at the feeder may eventually be used to improve pig's feeding performance as well as their welfare (Angarita et al., 2021; Rodenburg and Turner, 2012).

The traditional method of analyzing animal behavior is through direct observation or by filming and later manual decoding of videos (Agha et al., 2020; Csermely and Wood-Gush, 1990; Machado et al., 2017; Nielsen et al., 1995). Direct observation by a human of many pigs simultaneously and for the length of time needed to generate useful data is not possible on a commercial farm environment (Martínez-Avilés et al., 2017). On the other hand, manual decoding of video footage can be laborious, time-consuming, and subject to annotator error (Chen et al., 2021). Computer vision (Forsyth and Ponce, 2011) applications, where artificial intelligence is used to process images, are now being developed to detect animal behaviors. Compared to the traditional approach that involves human effort, computer vision (CV) has advantages of being low-cost, objective, and non-interventional, and to generate information continuously (Chen et al., 2021; Li et al., 2021). In most animal farming applications, CV for behavioral phenotyping is at the performance-evaluation phase. Most studies have primarily concentrated on the predictive ability of CV while less attention has been paid to validation of CV algorithms. Livestock farms continue to produce growing amount of CV datasets, reflecting a variety of information (Bahlo et al., 2019). However, validation studies on the predictive ability of CV are lacking for an important percentage (Gómez et al., 2021). An assessment of model generalization is still needed in practical animal farming contexts (Li et al., 2021).

In CV applications for detecting pig posture and behavior where the training set and its corresponding testing set were randomly split from the whole dataset, accurate results have been obtained (Chen et al., 2020a; Li et al., 2019; Liu et al., 2020; Nasirahmadi et al., 2019; Zhang et al., 2020). Most of these studies trained CV models using balanced datasets, manually constructed to obtain an equal sample size within each category of the classification problem (Chen et al., 2020a; Liu et al., 2020; Nasirahmadi et al., 2019; Zhang et al., 2020). However, a

strategy using balanced training sets sometimes overlooks the long-tail distribution (Zhou et al., 2018) of the categories under real-world conditions, where the sample sizes within categories of behavior vary.

The aims of this study were to 1) develop a CV approach to classify pigs' interactive behaviors in single-space feeding stalls, and 2) test the algorithm through random cross-validation and two blocked cross-validation strategies (Roberts et al., 2017), where the data are split temporally and spatially. We also present the importance of algorithm evaluation as well as diagnostics through multiple training-testing scenarios that are more practical in animal farming.

3. MATERIAL AND METHODS

3.1 Experimental design

3.1.1 Recording schedule and specifications

The behavior of grow-finish pigs in a single-space feeding stall was observed through video recordings. Videos were collected from the Swine Teaching and Research Center at Michigan State University (East Lansing, MI 48824, USA). All animal protocols were approved by the Michigan State Institutional Animal Care and Use Committee (Animal Use Form number 01/17-007-00). A total of six social groups (SGs) with ten crossbred pigs per group were used for this study. These groups were rotated through two test pens (Table 4.1). Pig weight at the start of data collection was 32±3.57 kg and final weight was 72.6±6.6 kg. No remixing of pigs was performed during the experiment, and thus all pigs remained in the same social group during the study. We observed six SGs for six consecutive weeks beginning immediately after the pigs were introduced to grow-finish pens. Each group was observed for a total of four days (7 hours per

day) and this took place on two different weeks (three weeks apart) with two consecutive days of observation being carried out during each selected week.

Table 4.1 Rotation schedule of social groups for the two experimental pens. SG, social group.

	Week 1	**Week 2**	**Week 3**	**Week 4**	**Week 5**	**Week 6**
Pen 1	SG 3	SG 4	SG 6	SG 3	SG 4	SG 6
Pen 2	SG 5	SG 2	SG 1	SG 5	SG 2	SG 1

Each SG spent seven days in a test pen before being replaced by another group. Within each seven-day period, the first five days included no recordings to allow for pigs to (re)acclimate to the pen. On the fifth day, test recordings were used to check and calibrate equipment as needed. On the sixth day and seventh day, video recording occurred from 9AM-4PM (this was typically the time that the barn lights remained with some minor variation).

Both pens were 3.88 m × 2.44 m, and each pen was equipped with a nipple drinker and a single-space automatic feeder (FIRE® Osborne Industries, KS, USA) with a dimension of 1.78 m × 0.74 m. Figure 4.1 shows the center views of the pens and the views of the single-space feeding stalls, respectively. We used Intel® RealSense™ D435 cameras for RGB video recording, which were installed on top of the feeders at a height of 2.44 m relative to the floor. Each pen was equipped with one camera to collect top-down-view videos of the feeding stall. Cameras were managed through MATLAB (R2018b, The MathWorks Inc., MA), and video recordings were saved in MP4 format at 30 frames per second. Raw videos were cropped to create a fixed top-down view of the feeding stall region (Figure 4.1, Panels C and D). In Pen 1, the cropped videos had a resolution of 982 × 238 pixels, while the resolution for Pen 2 was 964 × 248 pixels. Each camera was connected to an on-site microcomputer that had an Intel® Core™ i5-7500T CPU @ 2.7 GHz with 16GB DDR4 RAM and with Microsoft Windows 10 Pro operating system.

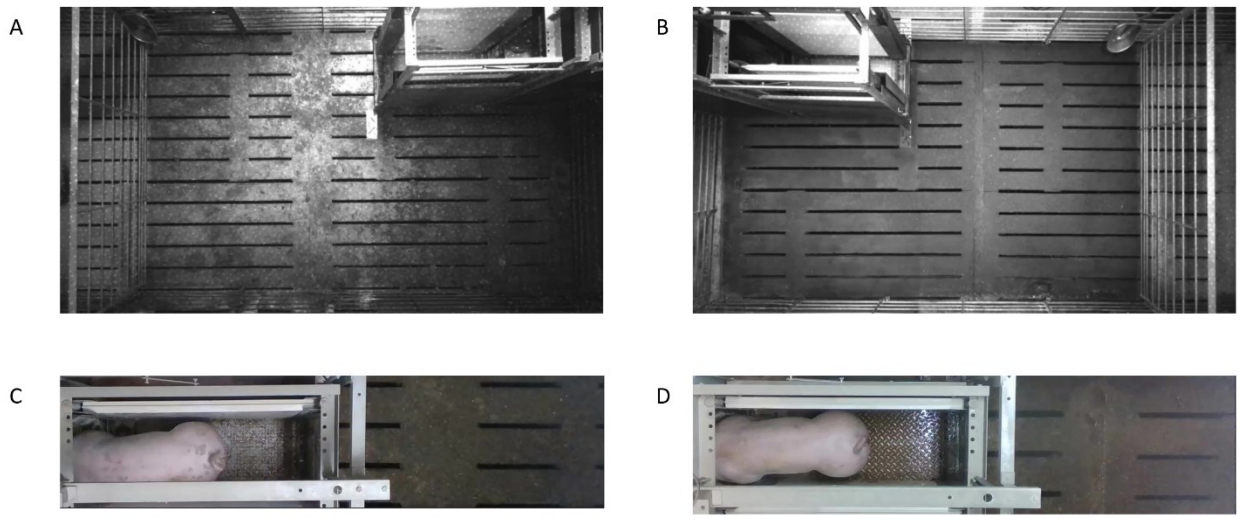

Figure 4.1 Top-down views of pens and feeding stalls. Panels A and B (infrared images) show the center views of Pen 1 and Pen 2, respectively. Panels C and D are top-down views of the feeding stalls for Pen 1 and Pen 2, respectively.

3.1.2 Behavior ethogram and dataset

Five classes of agonistic behaviors in the feeding stall were observed and defined, including no-contact (NC), ear-to-body (EB), head-to-body (HB), levering (L), and mounting (M) between pigs. Behaviors were annotated by trained observers according to the ethogram described in Table 4.2. After a first analysis, HB and EB were merged into a single category, HB, for two reasons. First, the two classes shared considerable visual and dynamic similarity. Second, preliminary results indicated that our prototype CV model could not distinguish EB and HB.

Table 4.2 Ethogram for the agonistic behaviors in pigs. *: ear-to-body was merged into head-to-body.

Behavior	Description	Code
No contact	Two pigs were in view at the feeding stall. The behind pig had at least both ears in the feeding stall but there was no physical contact between the behind pig and the body of front pig.	NC
Ear-to-body*	The behind pig had at least both ears in the feeding stall and unintentional contact was made. The behind pig might be nosing the floor or eating displaced feed and making slight, non-forceful contact with the front pig. This often appeared as the behind pig's ears grazing the front pig or the behind pig's nose bumping the rear legs of the front pig while investigating the floor.	EB*
Head-to-body*	The behind pig used its head to make intentional contact (greater than 1 second) with the body of the front pig. Quick (less than 1 second) bumps/run-ins by the behind pig were not recorded.	HB*
Levering	The behind pig's snout was under the body of the front pig and the front pig was lifted from the ground vertically. Any lifting of the front pig that involved a behind pig was considered levering. Typically, only the back half of the front pig was lifted. This often manifested as the behind pig pushing forward under the front pig, but it could also appear as the front pig backing up and over the head of the behind pig.	L
Mounting	The behind pig lifted its two front legs and put the two legs or its breast on the rear part of the front pig. The mounting pig may sit down during the mounting. Mounting commenced when the two front legs of or the breast of the behind pig contacted the front pig and terminated as soon as the mounting pig was no longer on top of the front pig even some contact was still maintained.	M

We only focused on video segments when there were at least two pigs present in the feeding stall. Such events with two or more pigs were passed to the observers. After a preliminary review of the videos, observers indicated that videos less than 30 frames (approximately 1 second) long tended to lack information (not enough frames) to make a classification decision for the video, while longer videos might include more than one behavior.

Therefore, 30 consecutive frames were set as a video episode (the base processing unit) to classify agonistic interactions of pigs at the feeding stall. Prior to further processing, each segment of video (when there were two or more pigs in the feeding stall) was cut into 30-frame video episodes labelled with one of the four behavior classes (NC, HB, L or M) following annotation by a trained human observer. Some behavioral classes (i.e., L and M) were less common than others (i.e., HB and NC) as they occurred less often or for shorter periods of time, thus yielding fewer episodes of these behavioral classes. Several attempts were applied to video data augmentation using temporal perturbation e.g., sub-sampling short video clips from the whole event sequence (Ji et al., 2019; Kim et al., 2020; Yun et al., 2020). To augment the instances of minority classes, specifically L and M, we up-sampled episodes by overlapping 25 frames when generating consecutive episodes from a whole video segment (Figure 4.2). Episodes labelled as HB and NC were cut from the whole video segments without overlapping frames in consecutive episodes (Figure 4.2). Episodes were created in this way to mitigate the class imbalance in our dataset.

We obtained a total of 15,679 30-frame episodes. Among them, NC, HB, L, and M activities made up 3,398 (22%), 10,114 (66%), 925 (6%), and 1,242 (8%) of the dataset, respectively. The video dataset was then ready to be split into a training set and a testing set according to various validation strategies, as explained in Section 2.2.5.

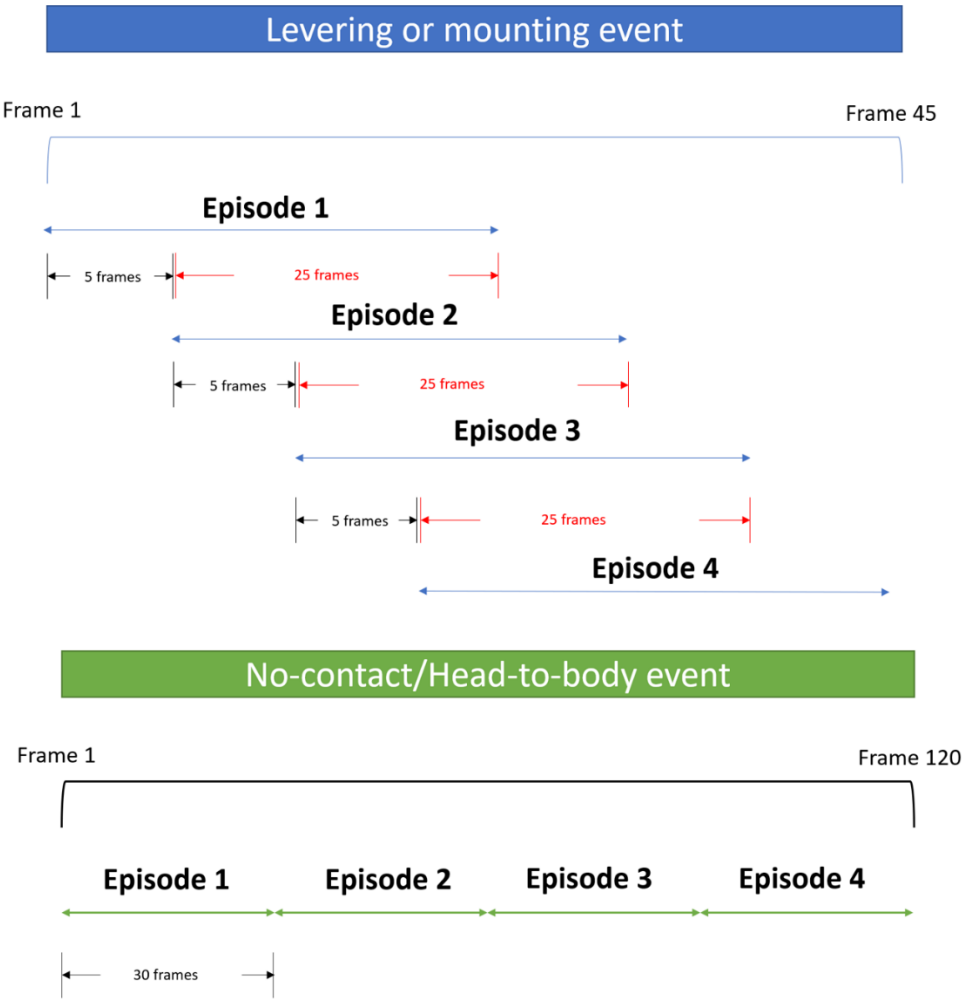

Figure 4.2 Examples for generating episodes for no-contact, head-to-body, levering, and mounting events.

3.1.3 Validation strategies

In modelling, dependence structures in the data, such as underlying temporal, spatial, and hierarchical structures, violate the assumption of independence between the training set and testing set and leads to overoptimistic results (Roberts et al., 2017). To tackle the temporal structure e.g. growing sizes of pigs and the spatial structure e.g. varying conditions of the two pens, we used blocked cross-validation, as proposed by Roberts et al. (2017), to split the whole dataset into a training set and a testing set given different blocking factors. Specifically, we split the entire data according to temporal characteristics (blocking by time) and spatial characteristics

(blocking by feeder). In addition, we used random cross-validation as a reference for comparison. The three validation strategies are as followed:

1. Five replicates of random cross-validation were used to evaluate predictive performance of the CV model. In each replicate, a random subset of the data was used for model training, while the remaining instances were for model training.
2. A blocking-by-time dataset was then created to study whether a model could be trained using the footage of younger pigs and then applied to older pigs with acceptable predictive performance. In this scenario, episodes from the first three weeks were defined as the training set, while episodes from the last three weeks were for testing purposes.
3. A blocking-by-feeder dataset was generated, where episodes from one of the two feeders were used for training and episodes from the other feeder was used for testing. This was done to study whether slight changes in experiment setup including different illumination, camera position/angle, and social groups affected the predictive performance of DL.

3.2 Computer vision algorithm

3.2.1 Deep learning pipeline for video classification

Deep learning (DL), a predominant analytical tool used in CV, is a set of representation learning methods, where a machine can be trained with raw data to discover the representations needed for prediction or classification without requiring extensive background knowledge (Lecun et al., 2015). Such advantages have made DL the preferred tool for behavior recognition applications use videos and images from different animal farming contexts (Chen et al., 2021; Li et al., 2021). A commonly used pipeline for video segment classification to detect behavior of pigs consists of coupling a convolutional neural network (CNN) (LeCun and Bengio, 1995) with

a long short-term memory (LSTM) model (Hochreiter, 1997). This allows the CNN to extract relevant spatial features from each individual frame and the LSTM to classify the whole set of frames while accounting for the temporal dependence in the video. For example, a CNN + LSTM pipeline has been successfully applied to recognize aggressive/non-aggressive episodes and tail-biting behavior in group-housed pigs (Chen et al., 2020b; Liu et al., 2020). In this study, we employed a CNN + LSTM pipeline (Figure 4.3) to classify the 30-frame short videos.

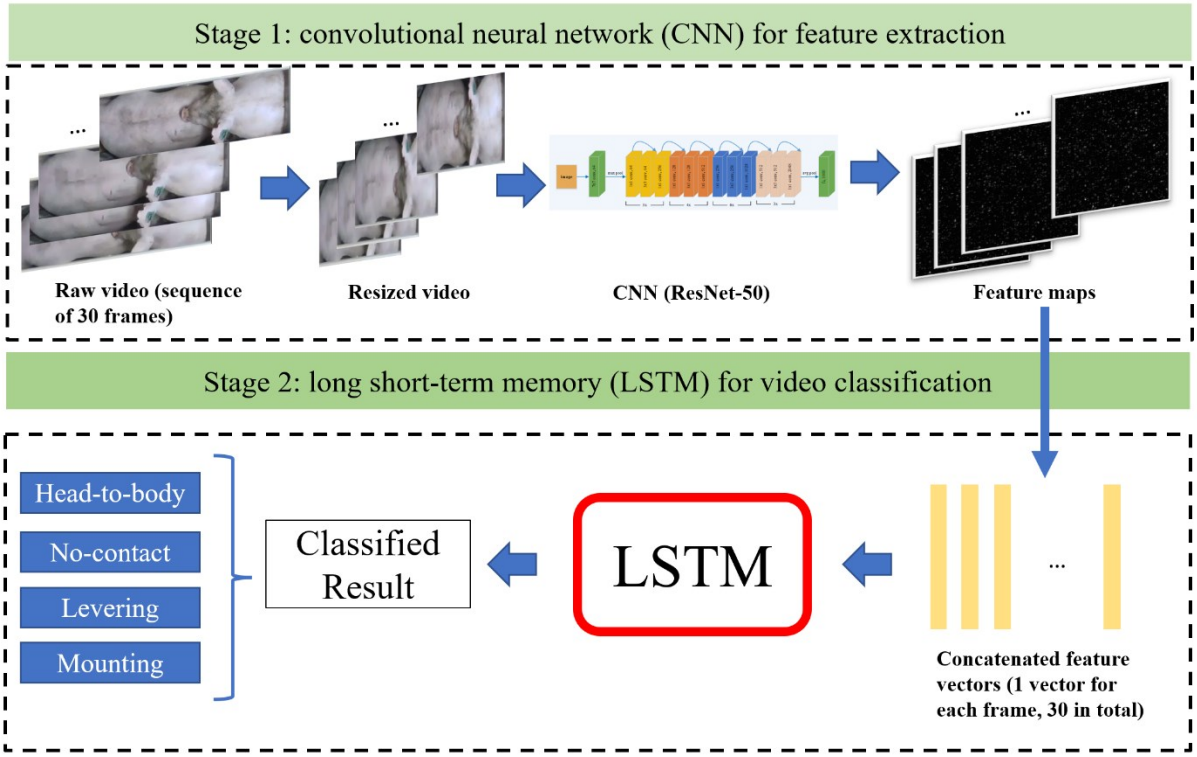

Figure 4.3 Deep learning pipeline for pig's aggressive behavior detection based on videos. Graph for ResNet-50 Architecture was obtained from Talo (2019).

3.2.2 Feature extraction with convolutional neural network

In our work, a CNN served as a feature extractor that received an individual frame as input and generated numerical features as output that were considered spatial representations. CNNs were designed to process data that has a spatial structure, for example using 2-D images for object detection (Lecun et al., 2015). We used transfer learning in Stage 1 (Figure 4.3) for

spatial feature extraction, where existing knowledge from a related CV application is transferred to a new context (Torrey and Shavlik, 2010). This is a common practice with CV in livestock applications (Chen et al., 2020b; Wu et al., 2021; Yin et al., 2020). We compared three pre-trained CNN models that were well established for computer vision tasks: ResNet-50 (He et al., 2016), GoogleNet (Szegedy et al., 2015), and VGG-16 (Simonyan and Zisserman, 2014). All three models required an image input size of 224×224 pixels and thus, we resized the raw episodes before passing them to CNNs. Through transfer learning, for each video frame we obtained $1 \times 1 \times 2{,}048$, $1 \times 1 \times 1{,}024$, and $7 \times 7 \times 512$ feature matrices from ResNet-50, GoogleNet, and VGG-16, respectively. Note that CNNs deal with individual frames, so each episode would result in 30 feature sets extracted by the CNN.

3.2.3 Long short-term memory

LSTM, one of the recurrent neural network architectures, was designed to handle data presenting a sequential or temporal structure, such as texts and videos (Lecun et al., 2015). In this study, the LSTM consisted of 30 modules, which were the same as the number of frames in each episode. Each module (except the first module) received two 1-D vectors, C_{t-1} and h_{t-1}, as its input and generated two vectors, C_t and h_t, as its output, where t means t-th frame of the episode and $1 < t < 30$, C_t is the cell state of t-th LSTM module, and h_{t-1} represents the hidden status of Frame $t-1$ (Figure 4.4). h_{t-1} was concatenated with x_{t-1}, the 1-D feature vector extracted from CNN given Frame $t-1$, and the concatenated vector had four copies with the length of $d_h + d_x$, where d_h is the number of hidden units in a LSTM module and d_x represents the output length from CNN. The four sets of weights (to be optimized through DL model fitting) were applied to the four copies and then the results were activated by sigmoid (Han and Moraga, 1995), sigmoid, hyperbolic tangent (Lecun et al., 2015), and sigmoid

functions, respectively. Structurally, an LSTM module includes a forget gate, an input gate, and an output gate (Figure 4.4). Furthermore, an LSTM model could be specified as a unidirectional LSTM or a bidirectional LSTM (Schuster and Paliwal, 1997). Each module produced an output h_t $(1 \leq t \leq 30)$ as the hidden status, whereas h_{30} (hidden status of the last frame given an episode) was used for classification of the episode. We trained all the LSTM parameters ourselves as there was no available pre-trained LSTM to use in this application.

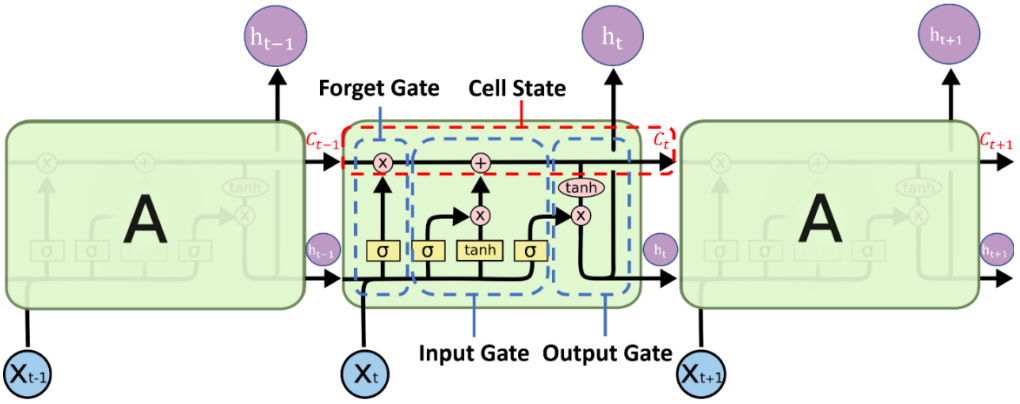

Figure 4.4 Diagram of long short-term memory. The figure was redrawn, and the original figure was obtained from https://colah.github.io/posts/2015-08-Understanding-LSTMs/

3.2.4 Hyperparameters

Hyperparameters are a set of values and options of the DL algorithm that are typically specified by data analysts before training a DL model. Selected hyperparameter values have an impact on the performance of DL (Han et al., 2021; Luo, 2016). A search of proper hyperparameter set(s) is recommended (Wu et al., 2021). We explored $3 \times 3 \times 2$ hyperparameter combinations by considering different CNNs for transfer learning and different LSTM architectures. The searching space referred to existing literature and is described in Table 4.3. The remaining hyperparameters were fixed following suggestions from the literature and are

described in Table S4.1. The average accuracy (Eq. 1) of five replicates of random cross-validation was used to select the best hyperparameter solution:

$$Accuracy = \frac{Number\ of\ correctly\ classified\ episodes}{Total\ number\ of\ episodes\ in\ testing\ set} \quad \text{(Eq. 1)}$$

Table 4.3 Explored hyperparameters and related work. CNN, convolutional neural network; LSTM, long short-term memory.

Hyperparameter	Reference	Options
CNN for transfer learning	He et al. (2016), Simonyan and Zisserman (2014), and Szegedy et al. (2015)	[VGG-16, ResNet-50, GoogleNet]
Dimension of hidden status (LSTM)	Saurabh (2021), Wu et al. (2021), Xiao et al. (2020), and Yin et al. (2020)	[50, 256, 512]
LSTM architecture	Hochreiter (1997) and Ullah et al. (2017)	[Single LSTM, Bidirectional LSTM]

The selected solution with highest average accuracy was used to train CNN-LSTM models for all validation strategies described in Section 2.1.3. Predictive performance of all hyperparameter combinations is presented in Figure S4.1. The selected model configuration is described in Table S4.1.

3.2.5 Region of interest

Once hyperparameters were selected, three different regions of interest (ROI) were further investigated to improve performance of the classification algorithm. Raw videos were cropped given the three ROIs: 1) extended feeding stall that included the whole feeding stall and the area immediate to its entrance, 2) feeding stall region only, and 3) truncated feeding stall region consisting of half the stall closer to the entrance, where most interactive behaviors were initialized (Figure 4.5). It is worth mentioning that 1) was the default ROI when searching for hyperparameters. Table S4.2 lists the average accuracy of the three ROIs, which suggested that

the model utilized the truncated feeding stall region (3) yielded the best performance. Therefore, we selected the truncated feeding stall region as the ROI in this study.

Figure 4.5 Explored regions of interest.

3.2.6 Deep learning training accounting for class-imbalance

Training a DL model is an optimization process that calculates and minimizes an objective function, also known as loss function, which measures prediction/classification errors. Categorical cross entropy is a commonly used loss function for classification problems with DL (Zhang and Sabuncu, 2018). Related work has implied that tuning the loss function helps address the class-imbalance problem during DL model fitting (Hossain et al., 2021). Lin et al. (2017) proposed focal loss function to deal with the imbalance problem for binary classification. Further, Liu et al. (2018) extended focal loss for multi-class classification. In this study, we used the multi-class focal loss as the objective function (Eq. 2):

$$loss_{FL} = -\sum_{i=1}^{c} \alpha(1-y_i)^{\gamma} t_i log(y_i), \quad \text{(Eq. 2)}$$

where c is the number of behavior categories, t_i represents the true probability distribution, y_i denotes the probability of behavior class i from *Softmax* activation (Goodfellow et al., 2016), while α and γ are the balancing parameter and the focusing parameter respectively. Lin et al. (2017) reported that the best-performing model was obtained when $\alpha = 0.25$ and $\gamma = 2$, and Oksuz et al. (2019) further indicated that the values performed well in practice. Therefore, we adopted the recommended values. Further, t_i is defined as:

$$t_i = \begin{cases} 1, & i = true\ label \\ 0, & i \neq true\ label \end{cases}$$

The performance of DL is sensitive to the size of the training set. In this study, the number of available training episodes differed depending on the blocking strategy (Table S4.3). A constant training set size was suggested for DL when comparing predictive performance under different validation scenarios (Fernandes et al., 2020). Thus, we restricted the training set size to be 7,500 30-frame episodes across all three validation strategies, and for those scenarios with more training samples, we randomly selected which 7,500 episodes to include in the training set. For testing, all available episodes were utilized for evaluation.

The training process was executed by MATLAB (R2021a, The MathWorks Inc., MA) with GPU computing activated. The computer used for DL model training had Intel® Core™ i7-8750H CPU @ 2.2 GHz with 16GB RAM, NVIDIA® GTX 1070 GPU with 8GB GDDR5 memory and a Microsoft Windows 10 operating system.

3.2.7 Evaluation matrices

Predictive performance of the DL model given behavior class i was evaluated through three measurements including overall accuracy (Eq. 1), recall (Eq. 3), and precision (Eq. 4):

$$recall_i = \frac{Number\ of\ correctly\ predicted\ episodes\ for\ class\ i}{Total\ number\ of\ annotated\ episodes\ for\ class\ i} \quad \text{(Eq. 3)}$$

$$precision_i = \frac{Number\ of\ correctly\ predicted\ episodes\ for\ class\ i}{Total\ number\ of\ episodes\ predicted\ for\ class\ i} \quad \text{(Eq. 4)}$$

4. RESULTS AND DISCUSSION

The model training took 1.8 hours (the training size was N=7,500 episodes), and the average computing time to classify an episode in the testing set was 1.9 seconds. Figure S4.2 presents the model training history of the four validation scenarios in terms of prediction accuracy over time (five sets of curves per scenario). After 100 epochs of DL model fitting process, final testing accuracy was 0.968(±0.001), 0.860(±0.033), 0.766(±0.026), and

0.860(±0.010) for random cross-validation, blocking-by-time validation, and blocking-by-feeder validation (at Feeder 1 and Feeder 2), respectively. In random cross-validation, the testing accuracy almost agreed with the training accuracy. The remaining three validation scenarios showed different levels of overfitting. Compared to training accuracy, testing accuracy decreased by 0.140, 0.214, and 0.140 in blocking-by-time, blocking-by-feeder (Feeder 1 for testing), and blocking-by-feeder (Feeder 2 for testing) validations, respectively. The result of random cross-validation indicated that a CNN + LSTM pipeline could be utilized to classify accurately the four types of agonistic behaviors. However, significant decreases in testing accuracy as well as overfitting were observed in blocking-by-time and blocking-by-feeder validations. In a previous study, Li et al. (2020) utilized DL to classify five categories of pigs' behaviors: feeding, lying, motoring, scratching, and mounting. They reported an accuracy of 96.35% in random cross-validation while the prediction accuracy on an independent testing set (a different pigsty) was 84.47%. In another study, Fernandes et al. (Fernandes et al., 2020) applied DL to predict pig body composition traits and indicated that when testing the trained model on independent genetic lines of pigs, the accuracy was systematically lower compared to 5-fold and 3-fold random cross-validations. Our results further confirmed the finding that the predictive performance of DL in blocked/independent testing sets was worse compared to random cross-validation.

Figure 4.6 shows the average recall and precision of each behavioral category in the four validation scenarios. As a reference, random CV yielded recall of 0.983, 0.849, 0.964, and 0.956 for HB, L, M, and NC behaviors, respectively. On the other hand, precisions of random CV were 0.971, 0.932, 0.968, and 0.969 for head-to-body, levering, mounting, and no-contact behaviors, respectively. The encouraging result of random CV implies that the DL pipeline was suitable for multi-class classification of pig's aggressive interactions. This may be especially useful in

retrospective studies, such as in research applications, where a whole dataset is collected, but annotated and analyzed after all the video recordings happened. In a study conducted by Li et al. (2020), they trained a DL model for multi-behavior recognition of pigs that had precisions ranging from 0.946 to 1 for five categories: feeding, scratching, mounting, lying, and motoring. According to a more recent study, Wu et al. (2021) fitted a CNN-LSTM model with recalls ranging from 0.950 to 0.985 and precisions ranging from 0.958 to 0.995 for drinking, ruminating, walking, standing, and lying behaviors of a single dairy cow. Both studies utilized random cross-validation, and we obtained similar results in random cross-validation in the present study. The obvious unique characteristic of our work is to classify interactive behaviors in pigs (rather than behaviors by a single animal), which is more complicated than the recognition of basic behaviors that do not necessarily involve two animals.

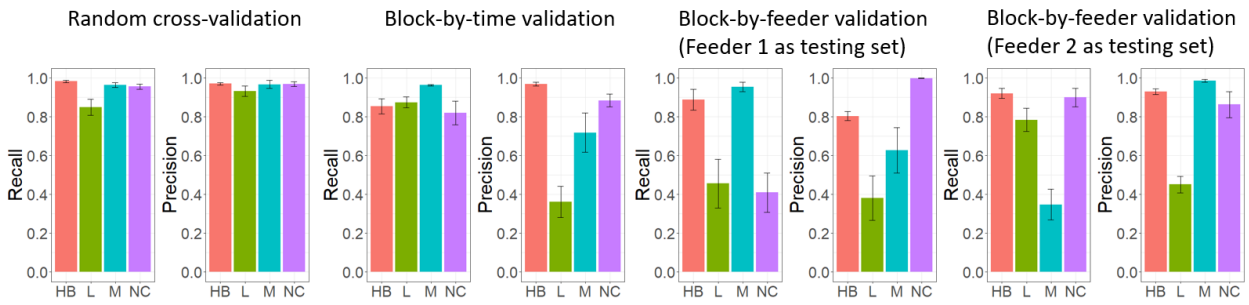

Figure 4.6 Bar plots of recall and precision for random, block-by-time, and block-by-feeder cross-validations. HB, head-to-body; L, levering; M, mounting; NC, no-contact.

In blocking-by-time validation, decrease in both recall and precision was observed. Recall for blocking-by-time validation was 0.855 (HB), 0.875 (L), 0.963 (M), and 0.821 (NC), while precision was 0.968, 0.362, 0.719, and 0.884, respectively (Figure 4.6). Using episodes collected from the first three weeks to classify episodes recorded during the final three weeks resulted in a significant decrease in predictive performance, especially in terms of precision for levering and mounting. Regarding blocking-by-time validation, Bergmeir and Benítez (2012)

raised concerns about data containing time-evolving effects. They indicated that last block validation tended to cause a less robust predictive performance, where the last block was a subset taken from the end of a time series. Thus, the decreased predictive performance in our blocking-by-time validation was possibly the result of time-evolving features i.e. the growing pigs that could not be picked up or were not present in the training set (episodes from the first three weeks). This type of validation is potentially useful for time-sensitive applications where model training is done after minimal or limited initial data collection and then the trained model is used on the same individuals, but at later time points.

Predictive performance also dropped in blocking-by-feeder validation. Notably, the model trained with episodes from Feeder 1 showed better performance than the model trained with Feeder 2. When training with Feeder 1 data and testing on Feeder 2, recall for the four categories was 0.921 (HB), 0.785 (L), 0.348 (M), and 0.900 (NC), whereas precision for the four classes was 0.930, 0.452, 0.987, and 0.863, respectively. Meanwhile, training with Feeder 2 and testing on Feeder 1 led to different results. In this scenario, recall for the four categories was 0.889 (HB), 0.456 (L), 0.955 (M), and 0.411 (NC), while precision rates were 0.804, 0.381, 0.628, and 0.999, respectively. Additionally, patterns of misclassifications differed between the two scenarios of blocking-by-feeder validation. Compared to random cross-validation, the model trained with Feeder 1 resulted in significantly lower recall for mounting and precision for levering, while worse recall for levering and no-contact and precision for levering and mounting were observed when validating the model trained with Feeder 2. Roberts et al. (2017) argued that ignoring data structures e.g., spatial and grouping structures may lead to over-optimistic estimation of the model performance. Furthermore, they reported notably worse predictive performance when the testing sets were blocked by spatial group and by space, compared to

random cross-validation. In another study that employed DL for classification (Lopez-Del Rio et al., 2019), the authors reported a worse performance when the training/testing sets were split by a clustering factor. As episodes recorded from Feeder 1 were composed of different social groups (Table 4.1) and a slightly different experiment setup (e.g., camera angle and illumination) compared to episodes recorded from Feeder 2, we speculate that the nested structures in the two feeders led to divergent variabilities between the two datasets. To further investigate this, we performed a principal component analysis (PCA) of the feature vectors extracted from each episode that were inputs of LSTM (Figure 4.3). Results of PCA implied that frames from Feeder 1 were more variable than frames of Feeder 2. We included the relationship of the first six principal components in Figure S4.3.

Figure 4.7 shows the cumulative confusion matrix over five replicates of each validation scenario. Validation sets and their corresponding misclassified episodes were further clustered by social group, week, feeder, pig's back mark (Arabic numerals from 0 to 9), and behavior category. For random cross-validation, misclassified episodes maintained the same clustering patterns as in the testing sets, but the pattern was different when accounting for the behavior category (Figure S4.4). For blocking-by-time validation, misclassified episodes presented different clustering structure in terms of social group, feeder, mark, and week compared to the testing set, while the breakdown by behavior category was similar to the testing set (Figure S4.5). Misclassification of both blocking-by-feeder validation scenarios showed similar patterns as in the testing set when divided by social group, mark, and week, but the misclassification breakdown by behavior category disagreed with the corresponding testing sets (Figures S4.6 and S4.7).

Moreover, when studying blocking-by-time validation and blocking-by-feeder validation, we ranked episodes based on the frequency that they occurred in the misclassification and selected the top 50 misclassified episodes for each off-diagonal element in of the confusion table. If the total number of episodes for the misclassification category was fewer than 50, we selected all episodes for diagnosis. Both blocking-by-time and blocking-by-feeder validations showed similar patterns in three misclassification categories: HB misclassified as NC, NC misclassified as HB, and HB confused with L. For HB misclassified as NC and NC episodes confused with HB, the following pig presented mild contact or almost no contact with the front pig (Figure 4.8 a.). For HB misclassified as L, almost all episodes presented a pattern where the following pig buried its head underneath the rear part of the front pig and attempted to push/lift the front pig but did not succeed in doing so (Figure 4.8 b.).

Figure 4.7 Confusion tables of three validation strategies. Tables were based on the result by merging statistics over 5 replicates. Each validation strategy has five reps. Prediction means classified result from our model and Target means ground-truth labels. Panel A, random validation; Panel B, block-by-time validation; Panel C, block-by-feeder validation (Feeder 1 as testing set); Panel D, block-by-feeder validation (Feeder 2 as testing set). NC, no-contact; M, mounting; L, levering; HB, head-to-body.

In blocking-by-time validation, the major sources of misclassification were HB predicted as NC, HB predicted as M, HB predicted as L, and L predicted as M. For HB classified as M, it was frequently observed that the following pig pushed/head-knocked the front pig, and we also found that the front pig tended to retreat from the feeder which caused contact with the following pig (Figure 4.8 c. and d.). For L misclassified as M, a common structure across all episodes was that the two pigs overlapped considerably and their heads were barely visible (Figure 4.8 e.).

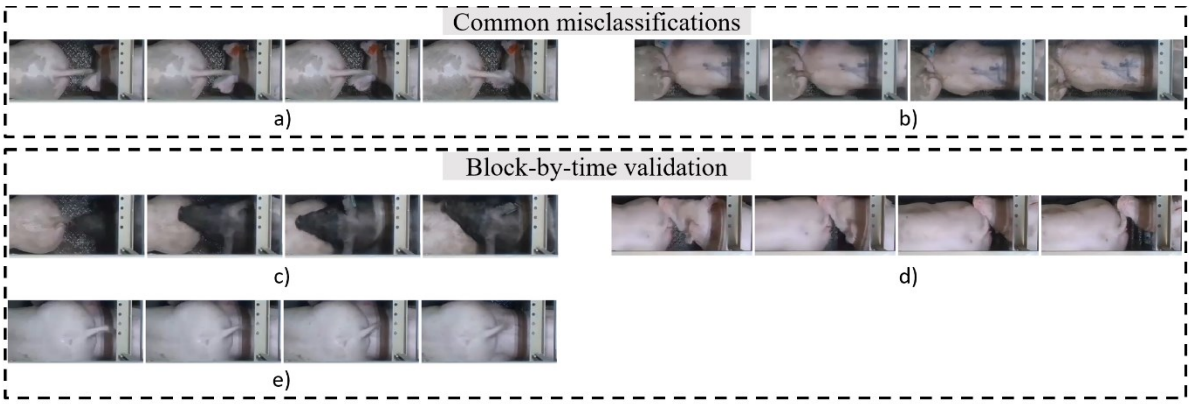

Figure 4.8 Error patterns for misclassification. The 1st, 10th, 20th, and 30th frames of example episodes were selected for display purpose. a), head-to-body and no-contact confused by no-contact and head-to-body, respectively; b), head-to-body misclassified as levering; c-d), head-to-body false predicted as mounting; e), levering confused by mounting. Panels a) and b) were common misclassification patterns across all three validation scenarios. Panels c-e) only represent block-by-time validation.

When testing on Feeder 1 (Feeder 2 as the training set), the main misclassified categories were: HB predicted as M, HB predicted as L, L predicted as HB, and L predicted as M. For HB confused with M, no specific behavioral patterns were observed, but most misclassified episodes involved back-marked pigs (with Arabic numerals) or dirty pigs (Figure 4.9 a.). Episodes that were labelled as L but classified as HB showed two types of features. In one case, the two pigs were in the middle of levering behavior, and they were relatively motionless (Figure 4.9 b.). In the second instance, some episodes contained more than one behavior category; for example, the following pig was in the transition period between levering and performing another behavior (Figure 4.9 c.). For L misclassified as M, only a small proportion of the front pig's body (mostly the rear part of the pig) was included in the episodes (Figure 4.9 d.), as we set the truncated feeder area as the ROI.

In the last validation scenario, we trained our model with the data from Feeder 1 and tested on video episodes from Feeder 2. We then watched misclassified episodes in the testing set (Feeder 2). Most misclassifications focused on five error categories: HB predicted as L, M

predicted as L, M predicted as HB, and L predicted as HB. When M was erroneously confused with L, commonly the view was almost entirely filled with body of the mounting pig and only a very small proportion of the front pig was visible (Figure 4.9 e.). For M misclassified as HB, all episodes presented clear views of the following pig, and the following pig was at very early stage of mounting (minor overlap with the front pig; Figure 4.9 f.). For L misclassified as HB, there were two patterns. On the one hand, many episodes included three pigs and both L and HB were occurring at the same time among the animals. Although we manually labeled the episodes by prioritizing the interaction related to the front pig (nearest the food trough), there were rare events that involved a secondary interaction between the 2 follower pigs at the other end of the feeding stall near the entrance (Figure 4.9 g.). On the other hand, over half of the L misclassified as HB errors included drastic levering with the follower pig's head/ears visible (Figure 4.9 h.).

Figure 4.9 Error patterns for misclassification in block-by-feeder validation. The 1st, 10th, 20th, and 30th frames of example episodes were selected for display purpose. a), head-to-body misclassified as mounting, respectively; b-c), levering misclassified as head-to-body; d), levering false predicted as mounting; e), mounting confused by levering; f), mounting confused by head-to-body; g-h), levering false classified as mounting.

This paper classified multiple interactive behaviors of grow-finish pigs in single-space feeding stalls, which is an original application of computer vision for studying livestock systems.

In addition, we employed a state-of-the-art CNN+LSTM pipeline that explicitly accounted for a class-imbalance problem in the training set. Furthermore, our results suggest that the data structure matters in the predictive performance of the proposed DL pipeline. Compared to previous studies that classified aggressive/non-aggressive behaviors of pigs (Chen et al., 2020a, 2019), our study took a further step to distinguish among different interactive agonistic behaviors in grow-finish pigs in the confines of a single-space feeding stall. Agonistic behavior in pigs involves complicated motion structure, and thus, it is difficult to handle challenging instances especially those occurring during the transition between two different activity types. The results from random cross-validation suggest that the proposed model could be used for classification of multiple types of pigs' agonistic behavior. However, such a validation strategy may overlook the effect of confounds within the dataset itself. Blocked validation strategies are closer to real-world applications, and results of these validation strategies should be considered when evaluating models for use on farm. As we observed more errors in blocked validation, it is essential to identify challenging cases through model diagnosis, which has also been proposed by many other researchers utilizing DL to study animal behavior (Chen et al., 2020b; Liu et al., 2020; Wu et al., 2021). Detailed diagnostics such as those we described underlying various misclassifications and advanced CV algorithms will be helpful to improve the model as well as predict its potential robustness in real-world situations.

5. CONCLUSION

Our results illustrate the importance of matching validation with application when evaluating DL models for behavioral classification of videos. We used a state-of-the-art CNN+LSTM pipeline trained with an imbalanced video dataset to classify four interactive

behaviors in grow-finish pigs. While random cross-validation produced an acceptable accuracy of 96.8%, using validation strategies that blocked data over time or by pen/feeder location had poorer performance. In the future, more datasets with known structures should be added to existing datasets to train video classification models under various real-world conditions that will be relevant to animal phenomics and precision livestock farming uses.

6. DECLARATION OF COMPETING INTEREST

We declare that we have no financial and personal relationships with other people or organizations which can inappropriately influence our work. There is no professional or other personal interest of any nature or kind in any product, service, and/or company that could be construed as influencing the position presented in, or the review of, the manuscript entitled.

7. ACKNOWLEDGEMENTS

This work was funded by NIFA Awards 2017-67007-26176 and 2021-67021-34150 and the National Natural Science Foundation of China (32102598).

8. DATA AVAILABILITY

Custom MATLAB Code used for (CNN) transfer learning, LSTM model fitting, and blocked validation strategies are available at GitHub: https://github.com/jun-jieh/AgonisticPigBehav/. Raw video episodes, video metadata, and extracted features from transfer learning are available at OSF data repository: https://osf.io/wa732/.

APPENDIX

Table S4.1 Hyperparameters configuration.

Hyperparameter	Option/Value	Reference
CNN for transfer learning	ResNet-50	Hyperparameter search
LSTM architecture	Bi-directional with 50 units	Hyperparameter search
α (Balancing parameter)	0.25	Lin et al. (2017)
γ (Focusing parameter)	2	Lin et al. (2017)
Optimizer	Adam	Wu et al. (2021)
Initial learning rate	0.0001	Wu et al. (2021)
Batch size	20	Wu et al. (2021)
Dropout rate	0.5	Wu et al. (2021)
Epochs	100	Chen et al. (2020b)

Table S4.2 Overall accuracy for different regions of interest.

Region of interest	Mean accuracy (5 reps)	Standard deviation (5 reps)
Extended feeder	0.868	0.004
Feeder only	0.875	0.001
Truncated feeder	0.887	0.004

Table S4.3 Available sample size by validation strategies. *: the training set and the testing set were interchangeable depending on which feeder was used for training/testing. #: once a training set size N1 was determined, the remaining N2=15,679 - N1 samples were considered as the testing set.

	Random cross-validation#	Blocking-by-time	Blocking-by-feeder*
Available for Training	15,679	9,907 (Weeks 1-3)	7,700 or 7,979
Available for Testing	15,679	5,772 (Weeks 4-6)	7,979 or 7,700

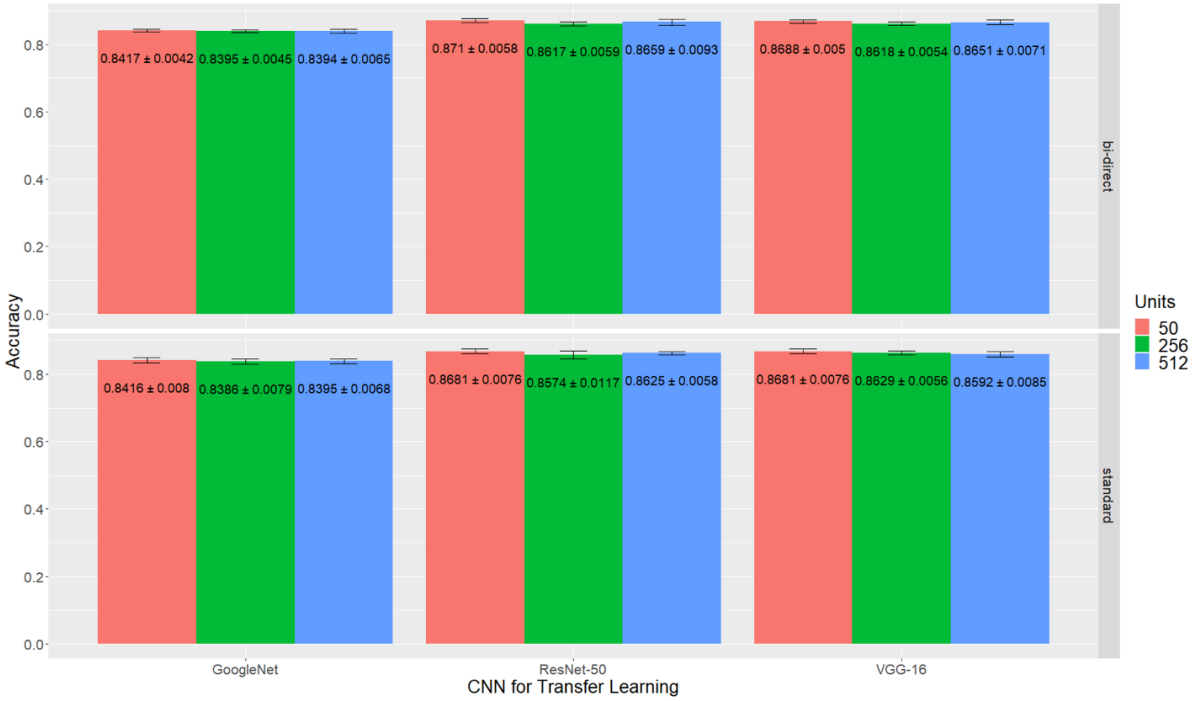

Figure S4.1 Average accuracy of different hyperparameter sets. Units, number of hidden units in long short-term memory module; bi-direct, bi-directional long short-term memory; standard, standard one-way long short-term memory.

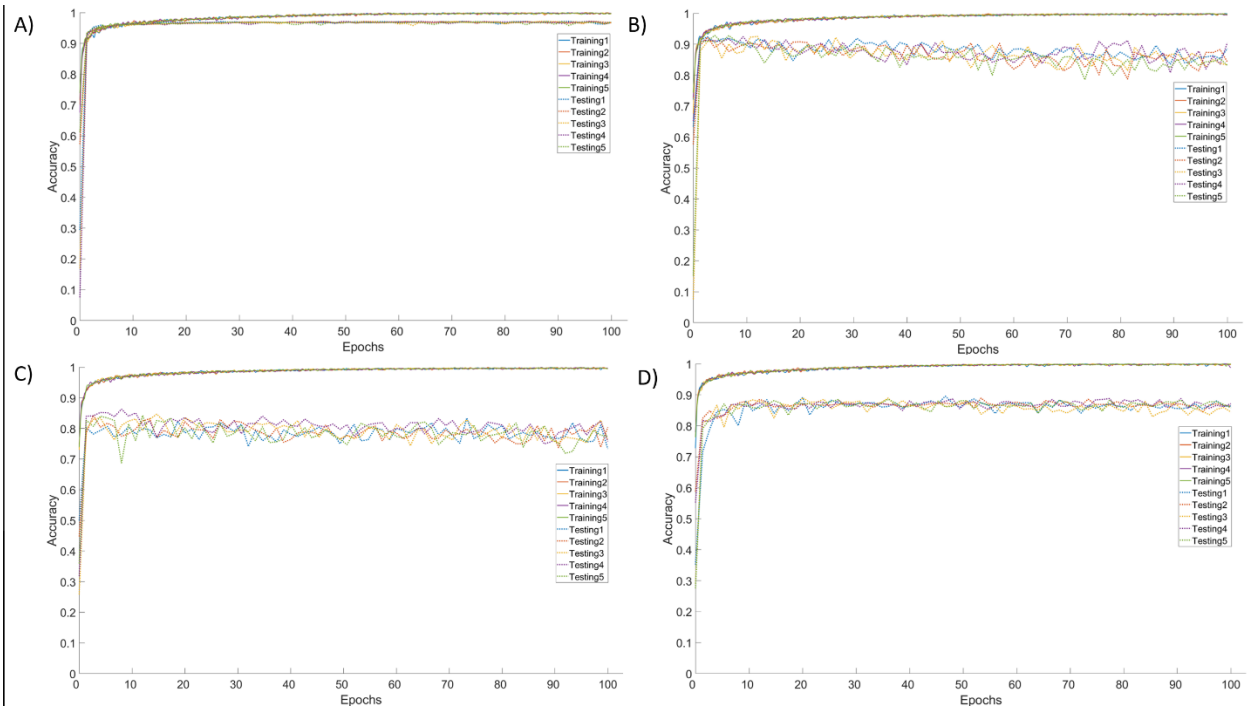

Figure S4.2 Training history of three validation strategies. Solid lines are for training curves and dashed lines show the testing curve. Each validation strategy has five reps. Panel A, random validation; Panel B, block-by-time validation; Panel C, block-by-feeder validation (Feeder 1 as testing set); Panel D, block-by-feeder validation (Feeder 2 as testing set).

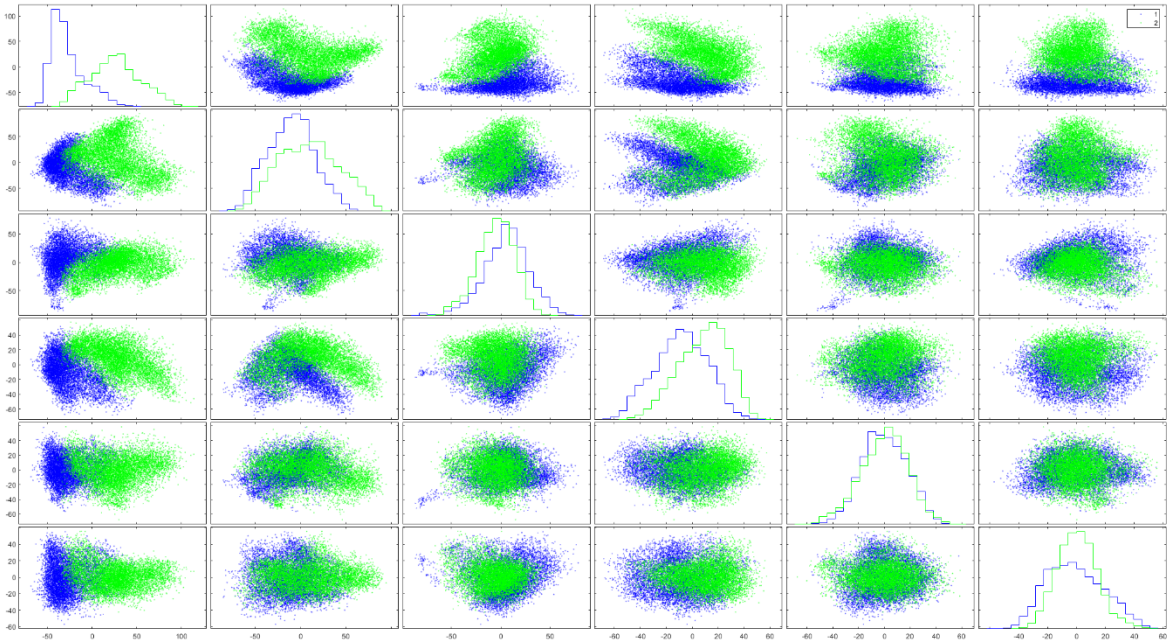

Figure S4.3 Scatter plots of individual score for each episode given the first six principal components (grouped by feeder). Blue dots represent Feeder 1 and green dots are for Feeder 2. Principle component analysis was done using feature vectors extracted from ResNet-50.

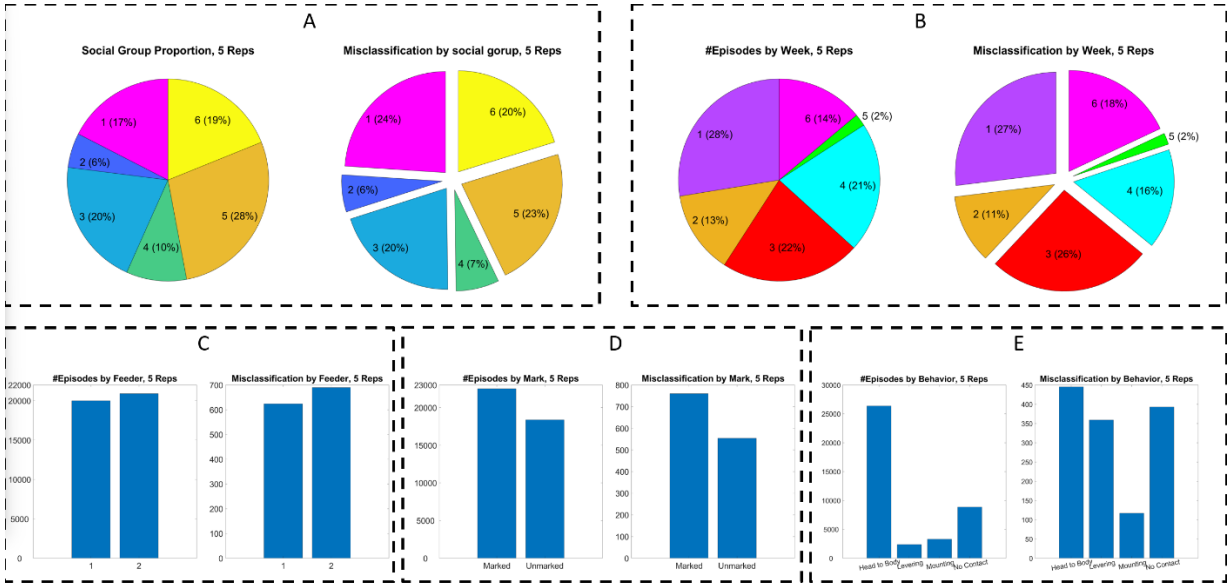

Figure S4.4 Testing sets breakdown (on the left of panels) and misclassification breakdown (on the right of panels) by social group (A), week (B), feeder (C), mark (D), and behavior category (E) in random cross-validation (five replicates). Marked pigs meant back-marked pigs with Arabic numerals; Unmarked pigs were pigs without artifactual marks on their backs.

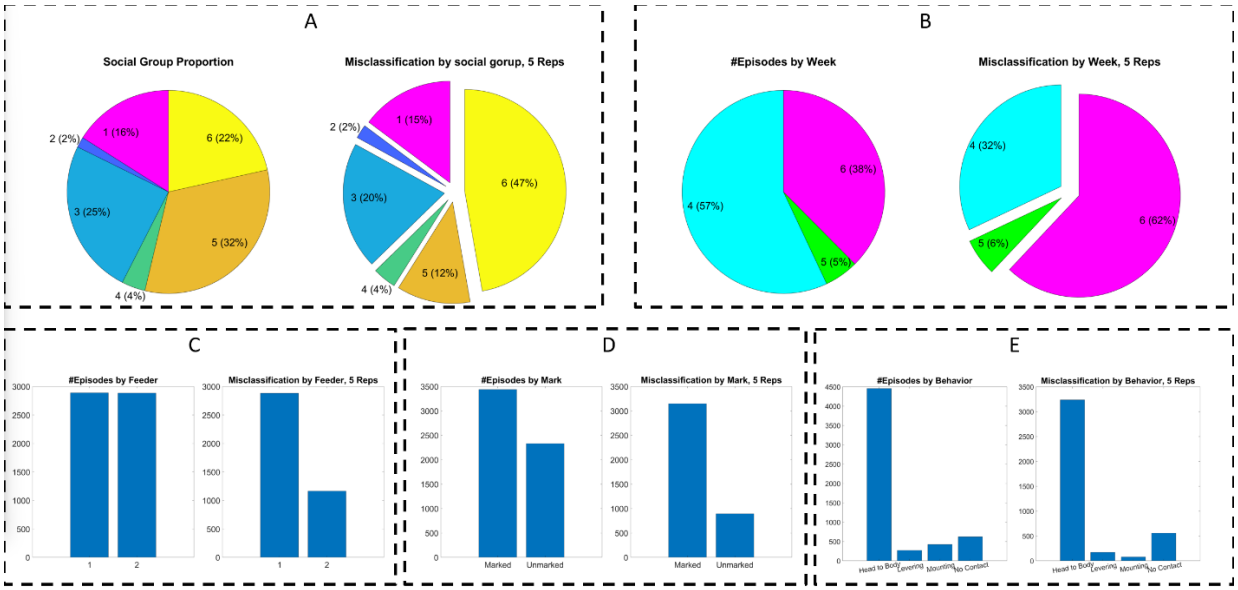

Figure S4.5 Testing set breakdown (on the left of panels) and misclassification breakdown (on the right of panels) by social group (A), week (B), feeder (C), mark (D), and behavior category (E) in block-by-time validation. Plots on the left of panels show the proportion/count for a single dataset, while plots on the right of panels stand for the statistics across 5 replicates. Marked pigs meant back-marked pigs with Arabic numerals; Unmarked pigs were pigs without artifactual marks on their backs.

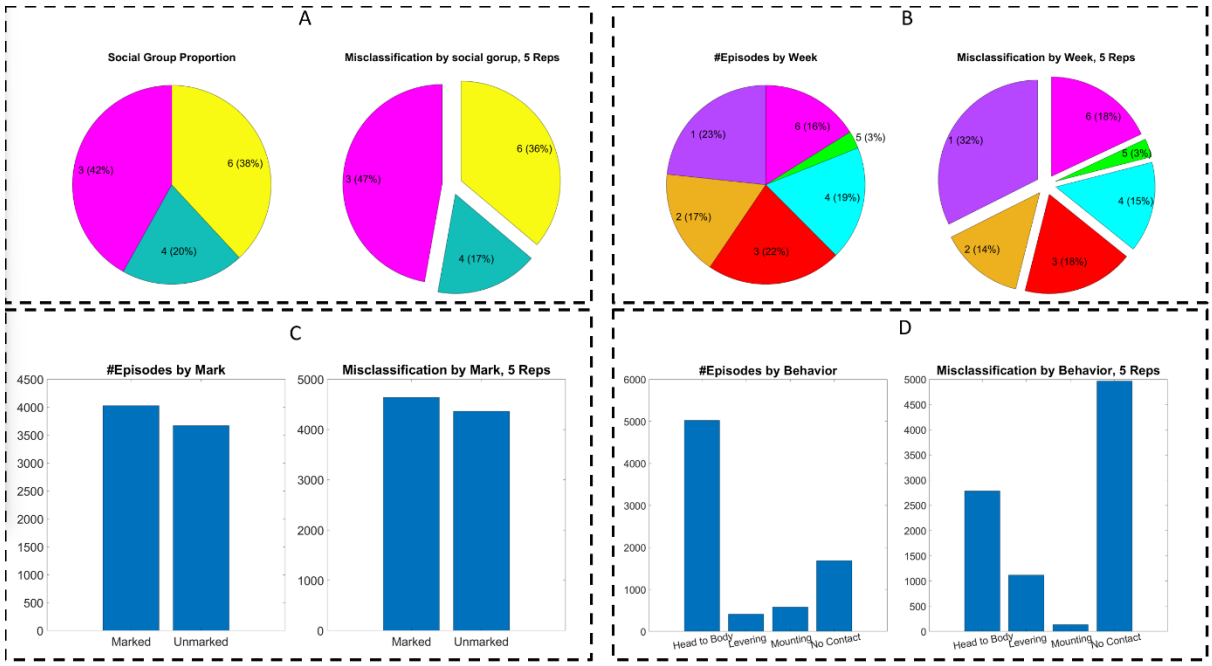

Figure S4.6 Testing set breakdown (on the left of panels) and misclassification breakdown (on the right of panels) by social group (A), week (B), mark (C), and behavior category (D) in block-by-feeder validation, whereas Feeder 1 was the testing set. Plots on the left of panels show the proportion/count for a single dataset, while plots on the right of panels stand for the statistics across 5 replicates. Marked pigs meant back-marked pigs with Arabic numerals; Unmarked pigs were pigs without artifactual marks on their backs.

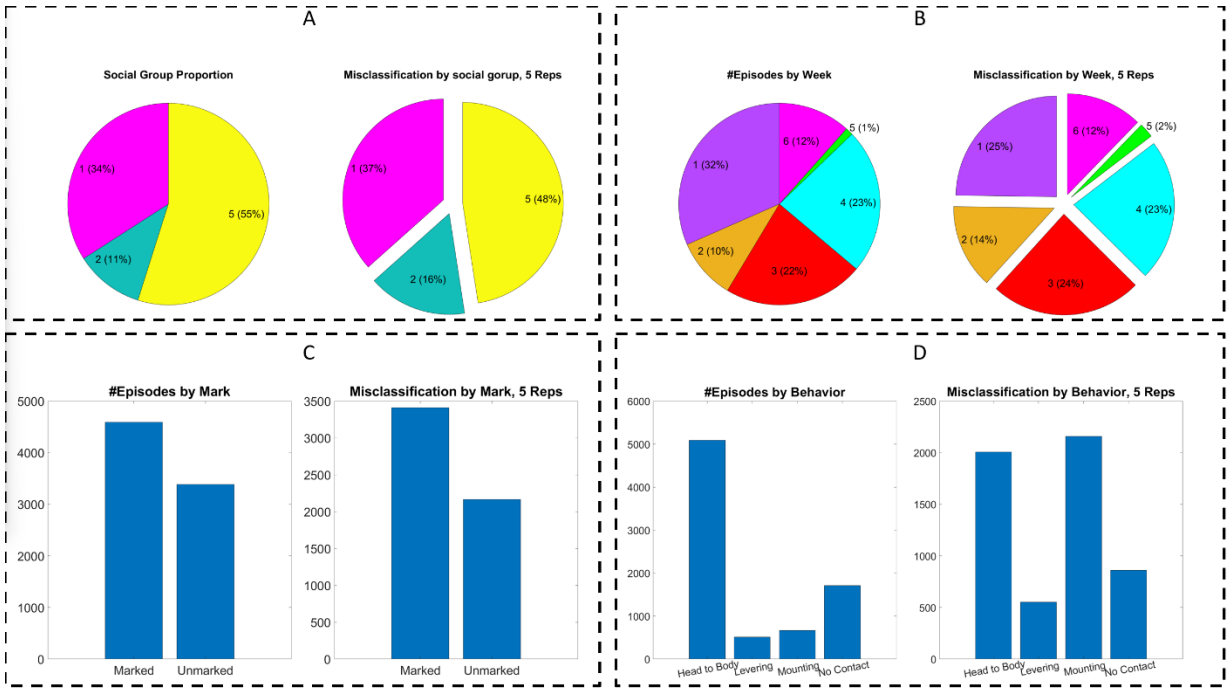

Figure S4.7 Testing set breakdown (on the left of panels) and misclassification breakdown (on the right of panels) by social group (A), week (B), mark (C), and behavior category (D) in block-by-feeder validation, whereas Feeder 2 was the testing set. Plots on the left of panels show the proportion/count for a single dataset, while plots on the right of panels stand for the statistics across 5 replicates. Marked pigs meant back-marked pigs with Arabic numerals; Unmarked pigs were pigs without artifactual marks on their backs.

REFERENCES

REFERENCES

Agha, S., Fàbrega, E., Quintanilla, R., Sánchez, J.P., 2020. Social network analysis of agonistic behaviour and its association with economically important traits in pigs. Animals 10, 1–13. https://doi.org/10.3390/ani10112123

Angarita, Belcy K., Han, J., Cantet, R.J.C., Chewning, S.K., Wurtz, K.E., Siegford, J.M., Ernst, C.W., Steibel, J.P., 2021. Estimation of direct and social effects of feeding duration in growing pigs using records from automatic feeding stations. J. Anim. Sci. 99, 1–8. https://doi.org/10.1093/jas/skab042

Bahlo, C., Dahlhaus, P., Thompson, H., Trotter, M., 2019. The role of interoperable data standards in precision livestock farming in extensive livestock systems: A review. Comput. Electron. Agric. 156, 459–466. https://doi.org/10.1016/j.compag.2018.12.007

Bergmeir, C., Benítez, J.M., 2012. On the use of cross-validation for time series predictor evaluation. Inf. Sci. (Ny). 191, 192–213. https://doi.org/10.1016/j.ins.2011.12.028

Brown-Brandl, T.M., Adrion, F., Gallmann, E., Eigenberg, R., 2018. Development and Validation of a Low-Frequency RFID System for Monitoring Grow-Finish Pig Feeding and Drinking Behavior 1–9. https://doi.org/10.13031/iles.18-041

Brown-Brandl, T.M., Rohrer, G.A., Eigenberg, R.A., 2013. Analysis of feeding behavior of group housed growing-finishing pigs. Comput. Electron. Agric. 96, 246–252. https://doi.org/10.1016/j.compag.2013.06.002

Chen, C., Zhu, W., Liu, D., Steibel, J., Siegford, J., Wurtz, K., Han, J., Norton, T., 2019. Detection of aggressive behaviours in pigs using a RealSence depth sensor. Comput. Electron. Agric. 166, 105003. https://doi.org/10.1016/j.compag.2019.105003

Chen, C., Zhu, W., Norton, T., 2021. Behaviour recognition of pigs and cattle: Journey from computer vision to deep learning. Comput. Electron. Agric. 187, 106255. https://doi.org/10.1016/j.compag.2021.106255

Chen, C., Zhu, W., Steibel, J., Siegford, J., Han, J., Norton, T., 2020a. Recognition of feeding behaviour of pigs and determination of feeding time of each pig by a video-based deep learning method. Comput. Electron. Agric. 176, 105642. https://doi.org/10.1016/j.compag.2020.105642

Chen, C., Zhu, W., Steibel, J., Siegford, J., Wurtz, K., Han, J., Norton, T., 2020b. Recognition of aggressive episodes of pigs based on convolutional neural network and long short-term memory. Comput. Electron. Agric. 169, 105166. https://doi.org/10.1016/j.compag.2019.105166

Csermely, D., Wood-Gush, D.G.M., 1990. Agonistic behaviour in grouped sows. Ii. how social rank affects feeding and drinking behaviour. Bolletino di Zool. 57, 55–58.

https://doi.org/10.1080/11250009009355674

Ding, R., Yang, M., Wang, X., Quan, J., Zhuang, Z., Zhou, S., Li, S., Xu, Z., Zheng, E., Cai, G., Liu, D., Huang, W., Yang, J., Wu, Z., 2018. Genetic architecture of feeding behavior and feed efficiency in a Duroc pig population. Front. Genet. 9, 1–11. https://doi.org/10.3389/fgene.2018.00220

Fernandes, A.F.A., Dórea, J.R.R., Valente, B.D., Fitzgerald, R., Herring, W., Rosa, G.J.M., 2020. Comparison of data analytics strategies in computer vision systems to predict pig body composition traits from 3D images. J. Anim. Sci. 98, skaa250. https://doi.org/10.1093/jas/skaa250

Forsyth, D., Ponce, J., 2011. Computer vision: A modern approach. Prentice hall.

Georgsson, L., Svendsen, J., 2002. Degree of competition at feeding differentially affects behavior and performance of group-housed growing-finishing pigs of different relative weights. J. Anim. Sci. 80, 376–383. https://doi.org/10.2527/2002.802376x

Gómez, Y., Stygar, A.H., Boumans, I.J.M.M., Bokkers, E.A.M., Pedersen, L.J., Niemi, J.K., Pastell, M., Manteca, X., Llonch, P., 2021. A Systematic Review on Validated Precision Livestock Farming Technologies for Pig Production and Its Potential to Assess Animal Welfare. Front. Vet. Sci. 8, 1–20. https://doi.org/10.3389/fvets.2021.660565

Goodfellow, I., Bengio, Y., Courville, A., 2016. Deep Learning. Cambridge, Massachusetts : The MIT Press.

Han, J., Gondro, C., Reid, K., Steibel, J.P., 2021. Heuristic hyperparameter optimization of deep learning models for genomic prediction. G3 Genes|Genomes|Genetics 11. https://doi.org/10.1093/g3journal/jkab032

Han, J., Moraga, C., 1995. The influence of the sigmoid function parameters on the speed of backpropagation learning, in: International Workshop on Artificial Neural Networks. Springer, pp. 195–201.

He, K., Zhang, X., Ren, S., Sun, J., 2016. Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 770–778.

Hochreiter, S., 1997. Long Short-Term Memory 1780, 1735–1780.

Hossain, M.S., Betts, J.M., Paplinski, A.P., 2021. Dual Focal Loss to address class imbalance in semantic segmentation. Neurocomputing 462, 69–87. https://doi.org/10.1016/j.neucom.2021.07.055

Ji, J., Cao, K., Niebles, J.C., 2019. Learning temporal action proposals with fewer labels. Proc. IEEE Int. Conf. Comput. Vis. 2019-Octob, 7072–7081. https://doi.org/10.1109/ICCV.2019.00717

Kim, T., Lee, H., Cho, M.A., Lee, H.S., Cho, D.H., Lee, S., 2020. Learning Temporally Invariant and Localizable Features via Data Augmentation for Video Recognition. Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics) 12536 LNCS, 386–403. https://doi.org/10.1007/978-3-030-66096-3_27

LeCun, Y., Bengio, Y., 1995. Convolutional networks for images, speech, and time series. Handb. brain theory neural networks 3361, 1995.

Lecun, Y., Bengio, Y., Hinton, G., 2015. Deep learning. Nature. https://doi.org/10.1038/nature14539

Li, D., Chen, Y., Zhang, K., Li, Z., 2019. Mounting behaviour recognition for pigs based on deep learning. Sensors (Switzerland) 19. https://doi.org/10.3390/s19224924

Li, D., Zhang, K., Li, Z., Chen, Y., 2020. A spatiotemporal convolutional network for multi-behavior recognition of pigs. Sensors (Switzerland) 20. https://doi.org/10.3390/s20082381

Li, G., Huang, Y., Chen, Z., Chesser, G.D., Purswell, J.L., Linhoss, J., Zhao, Y., 2021. Practices and applications of convolutional neural network-based computer vision systems in animal farming: A review. Sensors 21, 1–42. https://doi.org/10.3390/s21041492

Lin, T.-Y., Goyal, P., Girshick, R., He, K., Dollár, P., 2017. Focal loss for dense object detection, in: Proceedings of the IEEE International Conference on Computer Vision. pp. 2980–2988.

Liu, D., Oczak, M., Maschat, K., Baumgartner, J., Pletzer, B., He, D., Norton, T., 2020. A computer vision-based method for spatial-temporal action recognition of tail-biting behaviour in group-housed pigs. Biosyst. Eng. 195, 27–41. https://doi.org/10.1016/j.biosystemseng.2020.04.007

Liu, W., Chen, L., Chen, Y., 2018. Age Classification Using Convolutional Neural Networks with the Multi-class Focal Loss. IOP Conf. Ser. Mater. Sci. Eng. 428. https://doi.org/10.1088/1757-899X/428/1/012043

Lopez-Del Rio, A., Nonell-Canals, A., Vidal, D., Perera-Lluna, A., 2019. Evaluation of Cross-Validation Strategies in Sequence-Based Binding Prediction Using Deep Learning. J. Chem. Inf. Model. 59, 1645–1657. https://doi.org/10.1021/acs.jcim.8b00663

Luo, G., 2016. A review of automatic selection methods for machine learning algorithms and hyper-parameter values. Netw. Model. Anal. Heal. Informatics Bioinforma. 5, 1–15. https://doi.org/10.1007/s13721-016-0125-6

Machado, S.P., Caldara, F.R., Foppa, L., De Moura, R., Gonçalves, L.M.P., Garcia, R.G., De Alencar Nääs, I., Dos Santos Nieto, V.M.O., De Oliveira, G.F., 2017. Behavior of pigs reared in enriched environment: Alternatives to extend pigs attention. PLoS One 12, 1–18. https://doi.org/10.1371/journal.pone.0168427

Martínez-Avilés, M., Fernández-Carrión, E., López García-Baones, J.M., Sánchez-Vizcaíno, J.M., 2017. Early Detection of Infection in Pigs through an Online Monitoring System.

Transbound. Emerg. Dis. 64, 364–373. https://doi.org/10.1111/tbed.12372

Nasirahmadi, A., Sturm, B., Edwards, S., Jeppsson, K.H., Olsson, A.C., Müller, S., Hensel, O., 2019. Deep learning and machine vision approaches for posture detection of individual pigs. Sensors (Switzerland) 19, 1–16. https://doi.org/10.3390/s19173738

Nielsen, B.L., Lawrence, A.B., Whittemore, C.T., 1995. Effect of group size on feeding behaviour, social behaviour, and performance of growing pigs using single-space feeders. Anim. Sci. 61, 575–579. https://doi.org/10.1017/S1357729800014168

Oksuz, I., Clough, J.R., Schnabel, J.A., 2019. Artefact detection in video endoscopy using retinanet and focal loss function, in: CEUR Workshop Proceedings. CEUR-WS.

Roberts, D.R., Bahn, V., Ciuti, S., Boyce, M.S., Elith, J., Guillera-Arroita, G., Hauenstein, S., Lahoz-Monfort, J.J., Schröder, B., Thuiller, W., Warton, D.I., Wintle, B.A., Hartig, F., Dormann, C.F., 2017. Cross-validation strategies for data with temporal, spatial, hierarchical, or phylogenetic structure. Ecography (Cop.). 40, 913–929. https://doi.org/10.1111/ecog.02881

Rodenburg, T.B., Turner, S.P., 2012. The role of breeding and genetics in the welfare of farm animals. Anim. Front. 2, 16–21. https://doi.org/10.2527/af.2012-0044

Salgado, H.H., Méthot, S., Remus, A., Létourneau-Montminy, M.P., Pomar, C., 2021. A novel feeding behavior index integrating several components of the feeding behavior of finishing pigs. Animal 15, 100251. https://doi.org/10.1016/j.animal.2021.100251

Saurabh, N., 2021. LSTM -RNN Model to Predict Future Stock Prices using an Efficient Optimizer LSTM -RNN Model to Predict Future Stock Prices using an Efficient Optimizer 672–677.

Schuster, M., Paliwal, K.K., 1997. Bidirectional recurrent neural networks. IEEE Trans. Signal Process. 45, 2673–2681. https://doi.org/10.1109/78.650093

Simonyan, K., Zisserman, A., 2014. Very deep convolutional networks for large-scale image recognition. arXiv Prepr. arXiv1409.1556.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A., 2015. Going deeper with convolutions, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1–9.

Talo, M., 2019. Convolutional Neural Networks for Multi-class Histopathology Image Classification.

Torrey, L., Shavlik, J., 2010. Transfer learning, in: Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods, and Techniques. IGI global, pp. 242–264.

Ullah, A., Ahmad, J., Muhammad, K., Sajjad, M., Baik, S.W., 2017. Action Recognition in Video Sequences using Deep Bi-Directional LSTM with CNN Features. IEEE Access 6,

1155–1166. https://doi.org/10.1109/ACCESS.2017.2778011

Wu, D., Wang, Y., Han, M., Song, L., Shang, Y., Zhang, X., Song, H., 2021. Using a CNN-LSTM for basic behaviors detection of a single dairy cow in a complex environment. Comput. Electron. Agric. 182, 106016. https://doi.org/10.1016/j.compag.2021.106016

Xiao, H., Wang, C., Li, Z., Wang, R., Bo, C., Sotelo, M.A., Xu, Y., 2020. UB-LSTM: A Trajectory Prediction Method Combined with Vehicle Behavior Recognition. J. Adv. Transp. 2020. https://doi.org/10.1155/2020/8859689

Yin, X., Wu, D., Shang, Y., Jiang, B., Song, H., 2020. Using an EfficientNet-LSTM for the recognition of single Cow's motion behaviours in a complicated environment. Comput. Electron. Agric. 177, 105707. https://doi.org/10.1016/j.compag.2020.105707

Yun, S., Oh, S.J., Heo, B., Han, D., Kim, J., 2020. VideoMix: Rethinking Data Augmentation for Video Classification.

Zhang, K., Li, D., Huang, J., Chen, Y., 2020. Automated video behavior recognition of pigs using two-stream convolutional networks. Sensors (Switzerland) 20. https://doi.org/10.3390/s20041085

Zhang, Z., Sabuncu, M.R., 2018. Generalized cross entropy loss for training deep neural networks with noisy labels. Adv. Neural Inf. Process. Syst. 2018-Decem, 8778–8788.

Zhou, Y., Hu, Q., Wang, Y., 2018. Deep super-class learning for long-tail distributed image classification. Pattern Recognit. 80, 118–128. https://doi.org/10.1016/j.patcog.2018.03.003

CHAPTER 5: ANALYSIS OF SOCIAL INTERACTIONS IN GROUP-HOUSED ANIMALS USING DYADIC LINEAR MODELS

Junjie Han, Janice Siegford, Gustavo de los Campos, Robert J. Tempelman, Cedric Gondro, and Juan P. Steibel

1. ABSTRACT

Understanding factors affecting social interactions among animals is important for applied animal behavior research. Thus, there is a need to elicit statistical models to analyze data collected from pairwise behavioral interactions. In this study, we propose treating social interaction data as dyadic observations and propose a statistical model for their analysis. We performed posterior predictive checks of the model through different validation strategies: stratified 5-fold random cross-validation, block-by-social-group cross-validation, and block-by-focal-animals validation. The proposed model was applied to a pig behavior dataset collected from 797 growing pigs freshly remixed into 59 social groups that resulted in 10,032 records of directional dyadic interactions. The response variable was the duration in seconds that each animal spent delivering attacks on another group mate. Generalized linear mixed models were fitted. Fixed effects included sex, individual weight, prior nursery mate experience, and prior littermate experience of the two pigs in the dyad. Random effects included aggression giver, aggression receiver, dyad, and social group. A Bayesian framework was utilized for parameter estimation and posterior predictive model checking. Prior nursery mate experience was the only significant fixed effect. In addition, a weak but significant correlation between the random giver effect and the random receiver effect was obtained when analyzing the attacking duration. The predictive performance of the model varied depending on the validation strategy, with substantially lower performance from the block-by-social-group strategy than other validation

strategies. Collectively, this paper demonstrates a statistical model to analyze interactive animal behaviors, particularly dyadic interactions.

2. INTRODUCTION

The study of social interactions is of paramount importance in applied animal behavior research (Rodenburg et al., 2010; Silk et al., 2018). Researchers are interested in elucidating the basis for the observed variation in the intensity and frequency of interactions among pairs of individuals that are part of a social group. Some of the applications of such study include mate choice (Andersson and Simmons, 2006; Bierbach et al., 2013), aggression and other damaging behaviors (Angarita et al., 2019; Oczak et al., 2013; Peden et al., 2018), and competition for access to feeding space (Angarita et al., 2021; Lu et al., 2017), etc. Thus, given data on pairwise behavioral interactions recorded from an experimental or observational study, it is necessary to quantify the effects of various individual- and group-level factors on social interactions.

Data from pairwise social interactions are considered dyadic (Kenny et al., 2020). This is, the unit of observation is not the individual, but a pair of individuals. In general, dyadic interaction data can be arranged in square matrices. It can be further re-arranged in the form of a response vector, which is generally accomplished in two different ways (Figure 5.1): a) The data are summed row-wise/column-wise to represent an individual level-observation (total duration that each animal is engaged in a particular behavior regardless of whom the animal interacted with), or b) the matrix elements are stacked keeping intact their dyadic nature. In the first case, there is loss of information, and it should be avoided if the aim is to study the dyadic nature of social interactions. In the second case, however, it is of utmost importance that all sources of variations are modeled to properly account for group means, variances and covariances between

subsets of the data; otherwise, if important factors are ignored, this can adversely affect estimates and predictions.

a)

Columns: aggression receiver

Marginal giver intensity

Rows: aggression giver	1058	1084	1094	1100	1102	1106	1112	1114	1116	1118	1120	1121	
1058	0	0.00	5.00	13.00	4.00	1.00	0.00	9.00	0.00	0.00	0.00	43.00	75.00
1084	2.00	0	0.00	1.00	2.00	0.00	1.00	0.00	2.00	3.00	0.00	1.00	12.00
1094	0.00	7.00	0	4.00	0.00	2.00	0.00	4.00	3.00	10.00	20.00	2.00	52.00
1100	5.00	0.00	33.00	0	10.00	0.00	0.00	0.00	4.00	8.00	5.00	6.00	71.00
1102	0.00	0.00	0.00	10.00	0	0.00	0.00	2.00	0.00	12.00	0.00	1.00	25.00
1106	0.00	0.00	0.00	3.00	0.00	0	0.00	5.00	0.00	1.00	3.00	10.00	22.00
1112	12.00	12.00	16.00	4.00	3.00	12.00	0	8.00	17.00	0.00	7.00	6.00	97.00
1114	0.00	7.00	0.00	0.00	4.00	9.00	0.00	0	1.00	0.00	13.00	0.00	34.00
1116	3.00	1.00	10.00	6.00	4.00	5.00	0.00	0.00	0	0.00	19.00	0.00	48.00
1118	1.00	0.00	0.00	0.00	3.00	0.00	0.00	4.00	0.00	0	1.00	1.00	10.00
1120	0.00	1.00	1.00	1.00	0.00	0.00	4.00	3.00	8.00	2.00	0	0.00	20.00
1121	0.00	0.00	0.00	4.00	0.00	0.00	1.00	0.00	0.00	0.00	45.00	0	50.00
Marginal receiver intensity →	23.00	28.00	65.00	46.00	30.00	29.00	6.00	35.00	35.00	36.00	113.00	70.00	

b)

observation	y	giver	receiver
1	0.00	1058	1084
2	5.00	1058	1094
3	13.00	1058	1100
4	4.00	1058	1102
5	1.00	1058	1106
6	0.00	1058	1112
7	9.00	1058	1114
8	0.00	1058	1116
9	0.00	1058	1118
10	0.00	1058	1120
11	43.00	1058	1121
12	2.00	1084	1058
13	0.00	1084	1094
14	1.00	1084	1100
15	2.00	1084	1102
16	0.00	1084	1106
17	1.00	1084	1112
18	0.00	1084	1114
.			
.			
.			

Figure 5.1 Panel a), directional dyadic interaction intensity matrix (elements in the matrix represent attacking duration); row sums and column sums are shown in the margins of the matrix. Panel b), a truncated long-format table that is re-arranged from the interaction matrix; each row represents a record that is the attacking duration in seconds from a giver animal to a receiver animal. 0.00 means observed zero while 0 means structural zero that we do not consider as an actual interaction.

A proper way to model dyadic data is to fit generalized linear mixed models (GLMM) that include fixed and random effects to account for means and covariances depending on the actual design of the experiment (Kenny et al., 2020). In this study we describe how GLMM can be used to analyze dyadic data and illustrate how to use this approach to analyze a pig behavior dataset. First, we defined the type of social interaction data and how to properly model variation in the response using GLMM. Second, we applied the proposed GLMM to the experimental data and illustrated how to elicit, fit, and check the models and how to interpret the results. Finally, we performed posterior predictive checks of the models through several validation strategies. The GLMM presented in this paper can be used by applied animal behaviorists to analyze other pairwise social interaction data to obtain statistically valid and biologically meaningful results,

which can be helpful to understand interactive behaviors of animals for practical purposes of management or improved welfare.

3. METHODS AND MATERIALS

3.1 Data from social interactions should be analyzed as dyadic data

For a social interaction to occur, at least two animals need to be involved. Although behavioral interactions may involve more than two animals at a time, in this paper we assume that the data on social interactions is obtained through observations of pairwise/dyadic behaviors and that it can be arranged in an interaction matrix (Figure 5.1a).

The data may be obtained within a single large social group, in which all the potential pairwise interactions have been monitored and quantified. Alternatively, the dyadic data can be collected from several social groups of variable sizes, within which all potential pairwise interactions have been monitored and quantified, but no between-group relations are possible.

We also assume that in addition to the social interactions per se other variables have been observed. These variables may be individual-specific or dyad-specific. Examples of individual-specific variables are those related to each individual's age, sex, size, and past life experiences (e.g., early-life social or nutritional stress) and they can be continuous, discrete or categorical in nature. Dyad-level variables are those that only pertain to the pair of individuals. For instance, a dyad-level variable can be described as whether they have met each other before the interaction is observed. It is important to notice that sometimes individual-level variables may be coded as dyad specific, for instance, the difference in live weight between two animals can be viewed as a dyadic-level observation, but in fact it arises from a linear combination of two individual-level variables. In that case, we prefer to keep individual level observations separate.

3.1.1 Social interaction data

Social interaction data can be of different types. From a mathematical point of view, the social interactions could be represented by a binary outcome (0/1=it occurred/it did not occur), by a discrete outcome (frequency of occurrence of an interaction), by an ordinal outcome (intensity or severity of interaction on an arbitrary scale), or by a continuous outcome (intensity of the interaction on a continuous scale, duration of the interaction, etc.). The practical implications of the different types of responses pertain to the statistical distribution that is used to model the stochasticity in social interaction data.

From the point of view of the directionality of the behavior, in most cases, we can assume that the behavior is directional i.e., there is a giver and a receiver. For instance, in the study of animal aggression, in many cases there is a clear attacker and a victim. In studies of feather pecking in group-caged chickens (Savory and Mann, 1997) and tail-biting in group-housed pigs (Angarita et al., 2019; Wurtz et al., 2017), there was one animal that was delivering the behavior (we will call this animal the giver animal) and another one which was clearly receiving the behavior (the receiver). In the following subsections we lay out these concepts with the directional interaction data, and we summarize the model parameterization in a generalized form.

3.1.2 Analysis of dyadic data from directional social interactions

When the social interactions are directional, the data collected from each social group can be arranged in a matrix as represented in Figure 5.1a. If there are *n* animals in a certain group, *n(n-1)* interactions will be observed within the group. We assume that there is no measurement error implying that if an interaction was recorded then this interaction did indeed happen as recorded, and, perhaps more relevant, that a zero entry in the matrix implies that no interaction occurred for this specific dyad.

In the analysis of dyadic interaction, the model is necessarily componential, where the interaction consists of three major components: a main effect of the giver (giver effect), a main effect of the receiver (receiver effect), and the relation of the two individuals that is independent of the giver and receiver effects, referred to as the dyad (Back and Kenny, 2010; Kenny et al., 2006).

3.2 Experimental data analysis: attacking time in group-housed pigs

3.2.1 Experiment setup

In this study, the experimental data was collected from 797 Yorkshire pigs (409 gilts and 388 barrows) that were strategically mixed into 59 single-sex social groups and housed in grow-finish pens with 10-15 pigs per pen. In terms of prior social acquaintances, each social group included pairs or trios of animals that had shared a common nursery pen for seven weeks immediately before moving into the grow-finish pens. Prior social acquaintances also existed for some animals that had shared the same litter after farrowing (10 weeks before mixed into the grow-finish pens; these pigs were previously housed together as a litter before weaning). No prior social acquaintance was assumed to exist for animals that were housed together for the first time after being mixed into the grow-finish pens. At the beginning of the experiment the average weight of the animals was 27.09 kg (SD±4.07). The experiment has been described in detail in previous studies (Angarita et al., 2019; Wurtz et al., 2017).

Pigs were video recorded five hours after mixing and four hours on the following morning (no overnight recording was performed). Videos were decoded manually by trained observers who recorded all attacks, their duration, and the identity of giver and receiver. After decoding, the total amount of time for each dyadic interaction was computed as described by Angarita et al. (2019). The directional aggression duration y_{ijk} was defined as the total time in

seconds that animal i spent attacking another group mate animal j within social group k during the 9-hour post-mixing period. The final dataset contained 10,032 records consisting of total attacking duration for all possible dyads. Among those records, 1,100 pairs of animals (2,200 records) shared the same nursery pen prior to being remixed into the grow-finish pens, and 367 pairs of animals (734 records) were from the same litter.

3.2.2 Analysis model

After extensive model assessment and comparison, a hurdle Bernoulli-lognormal model was adopted. To keep things simple, in this paper it was assumed that a positive continuous response could be adequately modeled using a lognormal distribution and that a Bernoulli distribution could model the response when it is zero. However, the general principles presented here can be easily extended to other types of distributions as mentioned in the discussion. Thus, in this application, there are two sub-models (Equation 5.1). One sub-model estimates the probability of observing a zero (no attacks) while the other sub-model represents the duration of attacks conditional on its occurrence:

$$\begin{cases} y_{ijk} \sim Bernoulli(\theta_{ijk}), & if\ y_{ijk} = 0 \\ y_{ijk} \sim Lognormal(\mu_{ijk}, \sigma^2), & if\ y_{ijk} > 0 \end{cases} \qquad [5.1]$$

where, y_{ijk} is the total duration of the behavioral interactions between animal *i* and animal *j* in social group *k* (in y_{ijk}, the first subindex corresponds to the aggression giver, the second subindex corresponds to the aggression receiver, and the third subindex indicates the social group), θ_{ijk} is the expected probability of the total attacking duration being zero for animals *i* and *j* in social group *k*. Further, μ_{ijk} is the expected value (mean) of natural logarithm of y_{ijk}, and σ^2 is the variance of natural logarithm of y_{ijk}.

The transformed θ_{ijk} and μ_{ijk} have linear relationships with the explanatory variables (Equation 5.2):

$$\begin{cases} log\left(\frac{\theta_{ijk}}{1-\theta_{ijk}}\right) = \mu'_{ijk} = b'_0 + FE'_{ijk} + g'_i + r'_j + d'_{ij} + sg'_k, & if\ y_{ijk} = 0 \\ \mu_{ijk} = b_0 + FE_{ijk} + g_i + r_j + d_{ij} + sg_k, & if\ y_{ijk} > 0 \end{cases} \quad [5.2]$$

where μ'_{ijk} and μ_{ijk} are expected values (mean) on an underlying linked scale that can be modeled as linear combinations of individual-level and dyad-level systematic effects (described below), b'_0 and b_0 are overall intercepts. Note that b'_0, FE'_{ijk}, g'_i, r'_j, d'_{ij}, and sg'_k (notations with the superscript) represent effects to model the probability of presenting no attacks while b_0, FE_{ijk}, g_i, r_j, d_{ij}, and sg_k (effects without the superscripts) model the attacking duration if the attack occurred. $d'_{ijk} \sim N(0, \sigma'^2_d)$ and $d_{ijk} \sim N(0, \sigma^2_d)$ represent random dyad effects. $sg'_k \sim N(0, \sigma'^2_{sg})$ and $sg_k \sim N(0, \sigma^2_{sg})$ are random social group effects. The parameters g'_i, g_i, r'_j, and r_j are explained in the following section. Further, the fixed effects (overall means) FE'_{ijk} and FE_{ijk} in Equation 5.2 are defined as:

$$\begin{cases} FE'_{ijk} = sex'_k + \alpha' x_{jk} + \beta' w_{ik} + \delta'_1 z^1_{ijk} + \delta'_2 z^2_{ijk}, & if\ y_{ijk} = 0 \\ FE_{ijk} = sex_k + \alpha x_{jk} + \beta w_{ik} + \delta_1 z^1_{ijk} + \delta_2 z^2_{ijk}, & if\ y_{ijk} > 0 \end{cases} \quad [5.3]$$

where sex'_k and sex_k are sex effects in social group k, x_{jk} is the (within-group) centered weight of the receiver animal j from social group k, w_{ik} indicates the (within-group) centered weight of the giver animal i from social group k, z^1_{ijk} represents whether animal i and animal j from social group k shared the same nursery group previously ($z^1_{ijk} = 0$ if they did not; otherwise, $z^1_{ijk} = 1$), and z^2_{ijk} indicates whether animal i and animal j from social group k were previously housed together in the same litter before weaning ($z^2_{ijk} = 0$ if they were not; otherwise, $z^2_{ijk} = 1$). Finally, α', α, β', β, δ'_1, δ_1, δ'_2, and δ_2 denote the corresponding coefficients of the exploratory

variables. Without losing generality, we illustrate a linear model where the response can be simply decomposed into giver effects, receiver effects, and dyad-specific effects in Figure 5.2.

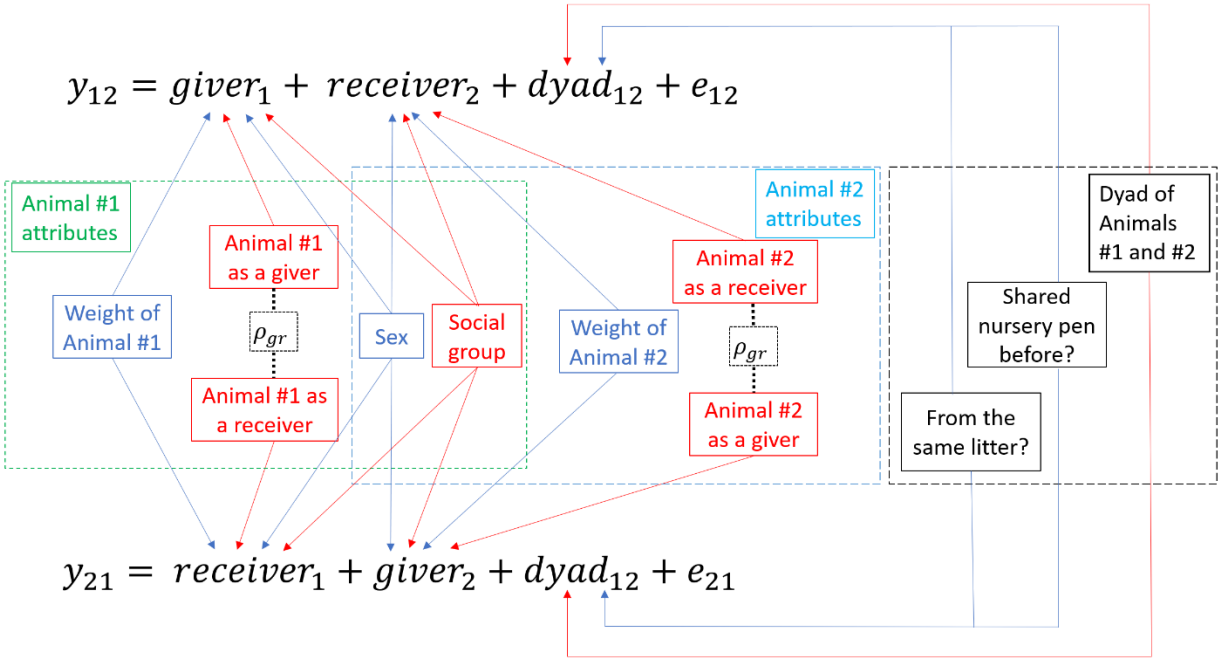

Figure 5.2 Illustration of a dyadic interaction model as an example that partitions the response into giver effects, receiver effects, and dyad-specific effects. Blue lines/arrows mean fixed effects, and red represents random effects, *e* stands for the residual term.

3.2.3 Modeling of (co)variances

Under the model in Equation 2, effects of the giver and the receiver are modeled for each animal. Those two effects covary, assuming:

$$\begin{pmatrix} g'_j \\ r'_j \end{pmatrix} \sim N\left(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} \sigma'^2_g & \sigma'_{gr} \\ \sigma'_{gr} & \sigma'^2_r \end{pmatrix}\right) \quad [4]$$

$$\begin{pmatrix} g_j \\ r_j \end{pmatrix} \sim N\left(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} \sigma^2_g & \sigma_{gr} \\ \sigma_{gr} & \sigma^2_r \end{pmatrix}\right) \quad [5]$$

where σ'_{gr} and σ_{gr} represent the covariance between the receiver and the giver effects of the same animal. Moreover, σ'_{gr} or σ_{gr} could take negative values. For example, $\sigma_{gr} < 0$ if an

animal spends more time attacking other animals but receives less aggression (in terms of duration) from other animals. Contrarily, σ_{gr} will be positive in those cases where animals that deliver more aggression also receive more aggression from other animals. A similar analysis could be done for σ'_{gr} but related to the probability of not delivering attacks. The magnitude and sign of these parameters are of importance to behaviorists.

We also derive the estimated giver-receiver correlation as this is easily interpretable to applied scientists:

$$\rho'_{gr} = \frac{\sigma'_{gr}}{\sigma'_g \sigma'_r}, \rho_{gr} = \frac{\sigma_{gr}}{\sigma_g \sigma_r} \qquad [6]$$

The relative magnitude of σ'^2_g, σ'^2_r, σ^2_g, and σ^2_r are also important. A relatively small value for a specific source of variation means that the process is mostly driven by other random sources.

3.2.4 Estimation

For statistical analysis, the model represented in Equations [1-3] could be fitted using restricted maximum likelihood or using Bayesian methods. We chose to use a Bayesian approach (Box and Tiao, 2011). Details for the implementation of model fitting are provided in Appendix. In the companion GitHub (https://github.com/jun-jieh/DyadAnalysis) we provide examples for implementation. A total of 4,000 Markov chain Monte Carlo (MCMC) samples were generated for parameter estimates. The parameter θ for a fixed effect given the observed data y was considered significant (P<0.05) if

$$1 - max\,(p(\theta < 0|y), p(\theta > 0|y)) < 0.05 \qquad [7]$$

where $p(\theta < 0|y)$ means that the probability of the parameter θ is smaller than zero while $p(\theta > 0|y)$ represents the probability of θ being larger than zero, and the function *max*() returns

the maximum value of the two elements. In practice, these probabilities are estimated based on the relative frequencies obtained from the MCMC samples.

3.2.5 Validation strategies and posterior predictive checks: how well does the model fit the data?

Posterior predictive checking is an important part of model evaluation. For this checking, new data are simulated conditional on the fitted model and their distribution is compared to observed data (Gabry et al., 2019). Moreover, this posterior predictive checking can be done with internal validation (all the data are used for model fitting and for validation) or using external validations (also known as out-of-sample or hold-out validations) (Vehtari et al., 2017; Vehtari and Ojanen, 2012), where the data are split into a training set (used for model fitting) and a validation set (used for the validation/checking). In the case of external validation, the way data are split is very important. Specifically, we split the entire dataset into a training set and a validation set using three different strategies:

1. A stratified 5-fold cross-validation (Vehtari and Ojanen, 2012) was used where in each fold, a random subset of each social group (80% of the data) was utilized for model training, while the remaining records were for testing purpose. It maintains the same social group ratio throughout the five folds as the ratio in the original (entire) dataset.

2. A block-by-social-group (5-fold) cross-validation was performed. In each fold, all records from randomly selected social groups that make up approximately 80% of the entire data were pooled and used for training purpose, while the remaining (validation data comprising 20% of the observations) set was from the left-out social groups that were not part of the training data.

3. A block-by-focal-animals validation was proposed and run for five replicates. In this validation scenario, we selected seven animals from each group and used all their aggression

records as both aggression givers and aggression receivers for the training set. The validation set contained only interactions between non-focal animals. This resembles a common way in which videos could be decoded; only some animals are followed and all their interactions with everyone else are decoded. Furthermore, by selecting seven focal animals per group, the resulting training set size was 78% of the entire data, while the remaining data (approximately 22% of all records) was used for testing. This led to a similar set size for comparison to the other two model checking strategies.

In each validation strategy, the model was fitted five times (for the five folds or replicates), and as part of the Bayesian model fit procedure, 500 MCMC samples were generated from the posterior predictive density. The posterior distribution of the generated samples was compared to the distribution of the observed dataset, where the response variables were transformed into a logarithm scale. To evaluate the predictive performance, Pearson correlation and root mean square error (RMSE) (Chai and Draxler, 2014) between the log-transformed (observed) response i.e. log(response+1) and the mean of log-transformed predicted response (linear predictors in Equation 2) in the validation set was computed across all validation scenarios. In addition, area under the ROC (Receiver Operating Characteristics) curve (Ling et al., 2003), also known as AUC, was computed to evaluate performance on the prediction of attack presence/absence.

3.3 Ethical approval

All animal protocols were approved by the Institutional Animal Care and Use Committee (Animal Use Form number 01/14-003-00).

4. RESULTS

4.1 Estimation of animal-specific effects, dyad-specific effects, and (co)variance components

Table 5.1 shows the posterior distributions of the individual animal effects and dyadic effects. The random dyad effect was not estimable (the model including the dyad effect did not converge; see Appendix for details). The nursery mate experience (animals in a dyad knew each other from sharing the same nursery pen prior to being remixed into grow-finish pens) exhibited significant effects, reducing the probability of presenting attacks and, if attacks happened, reducing duration of them ($\delta'_1 = 0.505, P < 0.05$; and $\delta_1 = -0.501, P < 0.05$; Table 5.1). The estimates indicated that for the dyads where the pigs had nursery mate experience, we would expect to see 65.7% increase of the odds of presenting no attacks; on the other hand, if the dyad did present attacks, pigs with nursery mate experience exhibited a 39.4% decrease in attacking duration. This means that if animal i and animal j were housed in the same nursery pen previously and then are remixed into a grow-finish pen, as might be expected under production conditions, they are less likely to attack each other and if they do attack each other, the length of attacking duration will be significantly shorter than the average attacking time of two animals who had not recently been housed together. The remaining animal-specific properties (weight of giver, weight of receiver, and sex) and dyad-specific attributes (whether the giver and the receiver were from the same litter) were not significant.

The giver-receiver correlation was not significant for the Bernoulli sub-model (when estimating the probability of animal i not presenting attacks to animal j; Table 5.1). On the other hand, a weak but significant correlation was obtained in the lognormal sub-model to analyze the attacking duration ($\rho_{gr} = 0.203, P < 0.05$). This means that when one pig spends more time delivering aggression this same pig will also receive longer attacks.

Table 5.1 Estimated posterior statistics for fixed effects and (co)variance components explained on total attacking duration between the giver animal and the receiver animal. Q: quantile.

		$y_{ijk} = 0$				$y_{ijk} > 0$			
Parameter		Mean	Q 2.5%	Q 50%	Q 97.5%	Mean	Q 2.5%	Q 50%	Q 97.5%
sex', sex		-0.083	-0.483	-0.085	0.312	-0.039	-0.162	-0.038	0.085
α', α	Receiver weight	0.001	-0.011	0.001	0.013	-0.001	-0.008	-0.001	0.006
β', β	Giver weight	-0.008	-0.027	-0.008	0.012	-0.007	-0.017	-0.007	0.003
$\boldsymbol{\delta'_1, \delta_1}$	Nursery mate	**0.505**	**0.392**	**0.504**	**0.618**	**-0.501**	**-0.573**	**-0.501**	**-0.424**
δ'_2, δ_2	Litter mate	-0.170	-0.357	-0.171	0.015	-0.023	-0.139	-0.023	0.087
σ'^2_g, σ^2_g	Giver variance	1.050	0.737	1.035	1.443	0.063	0.044	0.062	0.089
σ'^2_r, σ^2_r	Receiver variance	0.023	0.008	0.021	0.045	0.002	0.001	0.002	0.005
$\rho'_{gr}, \boldsymbol{\rho_{gr}}$	Giver-receiver correlation	0.155	-0.015	0.155	0.328	**0.203**	**0.009**	**0.202**	**0.397**
$\sigma'^2_{sg}, \sigma^2_{sg}$	Social group variance	0.473	0.282	0.460	0.735	0.020	0.001	0.019	0.052
σ^2	Error variance	-	-	-	-	1.076	1.031	1.075	1.123

4.2 Predictive performance in different validation strategies

We assessed the fitted model through posterior predictive model checks by inspecting two important aspects of: 1) how well it predicted the probability of not having an attack, and 2) how well it predicted the mean duration of the attacks when they occur.

Figure 5.3 presents the posterior predictive distribution of the probability of observing no attacks between animals (relative frequency of zeros). The distribution of the proportion of predicted zeros across multiple replicates of simulated data (lighter bins in Figure 5.3) was centered around the proportion of zeros in the observed response (dark solid lines in Figure 5.3). That is, regardless of training-testing data partitions, the estimated validation proportion fell well within the posterior predictive density in all validation strategies.

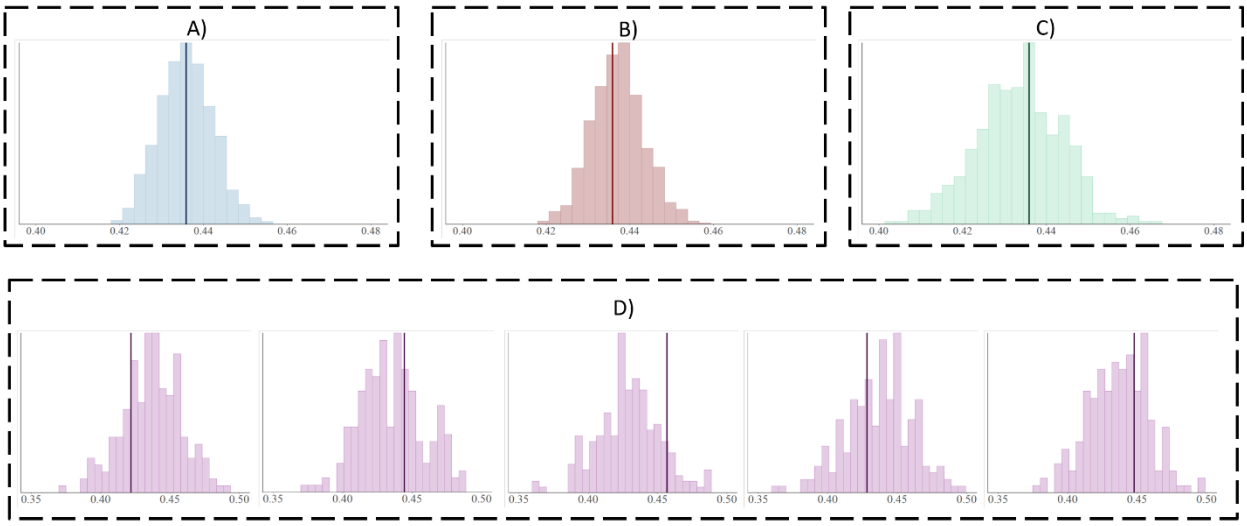

Figure 5.3 Proportion of zeros of validation set y (dark lines), with proportions of zeros for 500 simulated datasets $\tilde{y}$ drawn from the posterior predictive distribution (lighter bins). A), the model that used all data points for model training to predict the same dataset; B) 5-fold cross-validation; C) Block-by-social-group cross-validation; D) 5 replicates of block-by-focal-animal validation.

For each of the validation strategies, Figure 5.4 shows the distribution for the means of the simulated data (light bins) and the observed data (dark solid line). The response variables were transformed into a logarithm scale i.e., log(response+1). In general, the simulated data were consistent with the observed data (no systematic lack-of-fit was observed). However, in the internal validation and block-by-focal-animals validation, the mean duration of attacks was better approximated compared to when the stratified 5-fold and block-by-social-group cross-validation approaches were used.

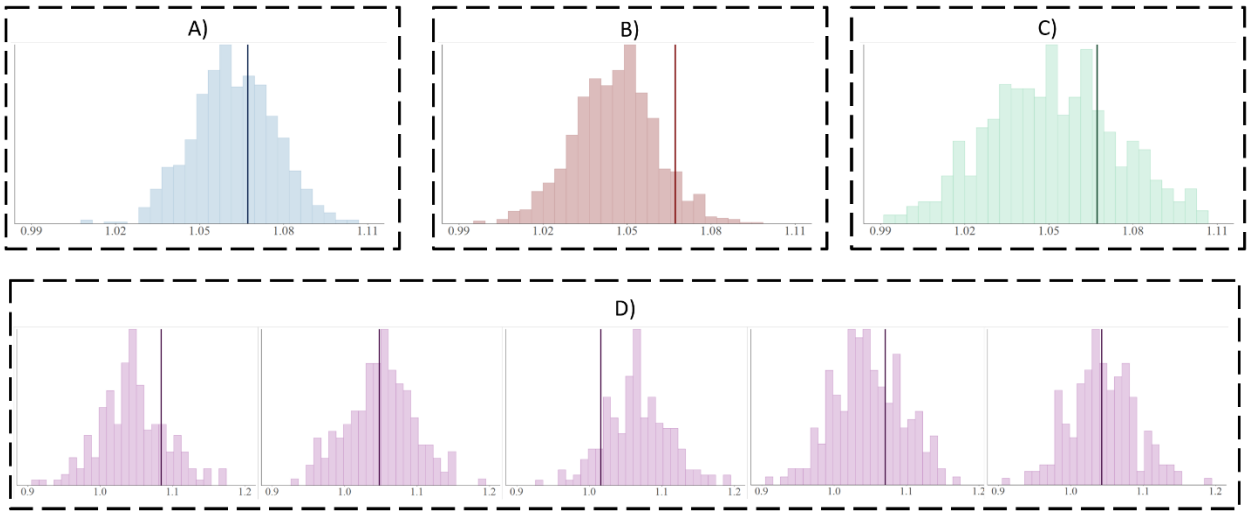

Figure 5.4 Distribution for the mean value of all observations across replicates. Mean of the validation set y (dark solid line) is compared with the means of 500 simulated datasets $\tilde{y}$ drawn from the posterior predictive distribution (lighter bins). We compared the logarithm of the observed and the simulated variables i.e. log(y+1) and log($\tilde{y}$ +1). A), the model that used all data points for model training to predict the same dataset; B) 5-fold cross-validation; C) Block-by-social-group cross-validation; D) 5 replicates of block-by-focal-animal validation.

Correlation and RMSE of the log-transformed response and AUC for the prediction of presence/absence of attacks in the validation set were computed across all validation scenarios (Table 5.2). The metrics allow for comparison between different validation strategies. Compared to the internal validation (i.e., all the data were used for model fitting and for validation), the predictive performance of the stratified 5-fold cross-validation, block-by-social-group cross-validation, and block-by-focal-animals validation showed much lower correlation and AUC, and larger RMSE (Table 5.2). Notably, the predictive performance in block-by-social-group validation was consistently worse than that of the other validation strategies.

Table 5.2 Metrics for evaluating predictive performance of the model under different validation strategies. AUC, area under ROC (Receiver Operating Characteristics) curve; RMSE, root mean square error; CV, cross-validation.

	Pearson correlation	**AUC**	**RMSE**
In-sample validation	0.595	0.826	1.173
Stratified 5-fold CV	0.227 (SD±0.014)	0.653 (SD±0.017)	1.391 (SD±0.020)
Block-by-social-group CV	0.115 (SD±0.023)	0.523 (SD±0.017)	1.422 (SD±0.023)
Block-by-focal-animals validation	0.286 (SD±0.020)	0.532 (SD±0.013)	1.362 (SD±0.014)

5. DISCUSSION

In this study, we have illustrated how to use GLMMs to analyze dyadic data from animal behavior studies that record interactions between animals in social groups. Through changing distributional assumptions and link functions, this approach can be easily adapted to analyses of categorical, ordinal, count, and continuous response types. Instead of modeling an individual animal's response, the proposed model exhibits advantages of analyzing interactive behaviors of pairs of animals in terms of flexibility and interpretability. Furthermore, the inclusion of random and fixed effects specific to each giver, receiver, and dyad (when possible) contributes to partitioning the observed variance into interpretable components.

Several approaches have been used in the analysis of animal behavioral interactions. A commonly used approach ignores the dyadic nature of the data and sums over rows or columns of an interaction matrix to simply obtain total time spent by each individual engaged in the behavior of interest (Figure 5.1a). Following this summation, linear models are used to study several sources of individual-level effects on the behavior of interest. We call this a 'marginal analysis' as it operates on the margin of the interaction matrix. For example, Savory and Mann (1997) studied the effects of genetic strain, age, and feeding pattern on aggressive pecking behavior of pullets, where for each individual the proportion of the aggressive behavior was computed (i.e., the total time of aggression was summed and divided by the length of observation period). Similarly, in two other studies (Turner et al., 2009, 2008), the authors recorded and treated the total duration of nonreciprocal aggression delivered and received by each individual pig as response variables, and they fitted linear mixed models to estimate additive genetic effects on the marginal response. In addition, Verdon et al. (2018) studied aggressive behavior of sows where the unit of analysis was a group of sows. They counted the frequency of aggressive

interactions from all possible pairs of animals within each group and fitted generalized mixed models to analyze the marginal response. A shortcoming of these analyses is that the effect of dyadic factors cannot be investigated. The proposed approach of this study allows the inclusion of both animal-level effects (marginal effects) as well as dyadic effects relevant to that particular pair of animals. For instance, we can add previous group- or littermate experience into the model for each dyad. In addition, genetics/genomics information of the giver and the receiver, as well as their genetic relationship, can be further included as an extensive form of our proposed model.

Another common approach analyzes dyadic interactions as independent observations. Oldham et al. (2020) fitted a linear mixed model to investigate effects of characteristics of both pigs on the initiating pig's latency to initiate agonistic behavior in a dyadic contest. This study was carefully designed and analyzed such that only one observation per animal and per contest (dyad) was available. This allowed the use of a simple linear model for the analysis. However, more precisely, dyads refer to relation of two individuals embedded in a social context (Kenny et al., 2020), while Oldham et al. (2020) manually selected paired pigs for contests instead of selecting dyads from a social context. In dyadic data extracted from multiple social groups with more than 2 individuals per group where each pig is exposed to multiple group mates, the assumption of independence between observations does hold i.e., the dyad is the fundamental unit of analysis (Kenny et al., 2020), and the proposed approach allows modeling the variances and co-variances of social groups, dyads, and individuals in a very straightforward way. For instance: we include the giver and receiver effects and account for their correlation. Thus, our proposed model is particularly useful for studying social interactions where animals are housed in multiple social groups over time.

In addition to introducing a model for the analysis of dyadic data in studies of social animal behavior, this study yields valuable results for understanding factors that affect post-mixing aggression in growing pigs. The results indicate that the giver explained more variation of the dyadic interaction than the receiver (Table 5.1). To the best of our knowledge, only one previous publication has used a GLMM to dissect the giver and receiver effects in animal behavior data (Wang et al., 2022), however, they did not consider the inclusion of dyadic fixed effects or the inclusion of a dyad-level random effect. A related line of research used bivariate marginal models to study delivery and reception of non-reciprocal aggression (Turner et al., 2009, 2008). Interestingly, both studies found that delivery of aggression was more heritable than reception of aggression. This encourages further analyses with the dyadic model to tease apart genetic effects from environmental effects.

One application in human behavioral ecology (Koster et al., 2015) proposed using GLMMs to perform dyadic analysis of food sharing between households and reported that the meal giver explained 75% of the variance components while the variance ratio of the meal receiver was 6%, which showed results quantitatively similar to ours. Given more complete datasets (with more observed variables of the interacting individuals), behaviorists could further use the proposed dyadic model to dissect factors that may influence delivery of the behavior as well as the characteristics of the receiver that attract the behavior.

In the context of post-mixing aggression in finishing pigs, we estimated the correlation between the random giver effect and the random receiver effect of the same individuals (Table 5.1). In a previously published marginal analysis of post-mixing aggression in pigs, the correlation between delivering and receiving non-reciprocal aggression did not differ significantly from zero (Turner et al., 2008). However, our model revealed a weak but significant

correlation between the giver and receiver effects on the duration of the attacks. This means that animals which attack for longer duration also receive longer attacks themselves, and this could be a result of receiver animals defending themselves and striking back (i.e., receivers may use attacks as a form of defense) (Oldham et al., 2020). The aggregated data used in this study did not allow investigation of the sequences of attacks (as our dyadic data was defined as the total aggression duration from a giver to a receiver), thus further work is needed to analyze heterogeneous and repeated measures of dyadic interactions over time.

Interestingly, our model did not yield a significant effect of the bodyweight of giver or receiver on the occurrence or duration of attacks. It is worth mentioning that the goal of this study was not to investigate bodyweight effects of the giver and receiver on attacking duration. Our result for the bodyweight effect might be due to the limited variation in body size within the social groups of our study as we had deliberately mixed together animals of similar body size in the finishing groups. This could result in a non-significant effect of the animal weight given limited variation in those covariates. The literature on this matter (bodyweight effect) does not offer a definite conclusion regarding the effect of bodyweight on aggressive behaviors of pigs. In one study of aggressive contests between pigs (Oldham et al., 2020), neither the weight of the contest initiating pig nor the weight difference between the contestants significantly influenced the latency to initiate the aggression. This agrees with our findings on bodyweight effects. However, in another study of dyadic contests in pigs, the winner pigs were significantly heavier than the loser pigs (Camerlink et al., 2019). We need to point out that initiating an attack and winning a contest are different. Our result suggests that pigs might not be good at telling whether they were going to win or not when they decided to attack. Camerlink et al. (2015) also showed

that between pairs of size-matched pigs, pigs which were more likely to be attackers were not more likely to be winners.

The variance component of random dyad effect (the effect of giver-receiver relation; see Sections 2.1.2 and 2.2.2) was not estimable in this study; however, we found that having shared nursery pens immediately before being mixed into grow-finish pens (a dyad-level covariate) showed a significant effect ($P<0.05$; Table 5.1). This finding is confirmed in the literature, for instance, Li and Wang (2011) reported that unfamiliar pigs fought for longer durations and fought more frequently than familiar pigs when pigs were remixed into new social groups. However, another dyadic level predictor (whether the two pigs were previously housed together as a litter before weaning; the pigs were housed as a litter for approximately three weeks before introduced to nursery pens) was not significantly associated with delivery or duration of aggression. This hints at the fact that animals who once shared a social group several weeks prior to the mixing, even if they are related, are unlikely to remember each other. It is unclear how long pigs remember each other though a possible time range could be three to six weeks (Mendl et al., 2010). Since pigs in this study spent approximately seven weeks in nursery pens immediately before being remixed into grow-finish pens, it was possible that pigs did not recognize their initial littermates when re-introduced to them in grow-finish pens.

In addition to the models presented in this study, we also evaluated other GLMMs, including log-Poisson, zero-inflated log-Poisson, Gaussian, and zero-inflated Gaussian. The posterior predictive checks conducted (results not shown) showed that the hurdle-Bernoulli model was the one that fitted the data better. The hurdle model did so by dissecting the trait into two components: the tendency of not delivering attacks and a second component of the attacking duration. The complexity of the model may limit its practical use as there are two correlated

traits rather than one per animal that can be used for decision making. However, it is worth mentioning that other GLMMs may be adequate for other settings. In the companion GitHub (https://github.com/jun-jieh/DyadAnalysis), we provide simpler GLMMs that are more general and can be easily adapted. In addition, to check model fitting, the in-sample posterior predictive checking (predicting the data used for model fitting) suffices, but for studying the model's ability to predict future data, out-of-sample validation (predicting observations left out of the model fitting process) should be used.

Social interaction data has been recently used as predictors of other traits. For example, Turner et al. (2020) constructed play fighting social networks of pigs using dyadic interaction data and extracted individual level and network level traits to build prediction models for lesion score counts. In a different application, Angarita et al. (2019) proposed using the dyadic matrices of aggression duration between pigs to parametrize social genetic effects of lesion scores. In these cases, the dyadic data (or its derived social network features) were used as a predictor rather than as a response variable. Nevertheless, the proposed predictive modeling of dyadic data could be used to add uncertainty to these applications. For instance, the internal and external validation (see Section 2.2.5 and Figures 5.3 and 5.4) used for model checking provides a natural way to resample plausible social interaction matrices that could be then subject to social network analysis or included in social genetic effects modeling. Moreover, obtaining the summation of all dyadic interaction of an animal as a giver (or receiver) allows for predicting individual-level aggressiveness (or vulnerability), for instance, the marginal intensity as shown in Figure 5.1a. Such individual level phenotypes can be used for management and as traits in genetic evaluations. Further experiments could be designed to validate the early prediction of animal social behavior that can be further related to animal welfare and production traits.

In this study, we considered several ways of splitting data for training (model fitting) and validation. Different validation scenarios (see Section 2.2.5) could be related to possible situations in real-life applications or relevant prediction problems (Burgueño et al., 2012). The stratified 5-fold cross-validation was designed for evaluation when the model was used for predicting unobserved (directional) social behaviors between two animals. The block-by-social-group mimicked a situation where the effects of giver, receiver, and dyad were evaluated in some social groups but not in others. Similarly, the block-by-focal-animals validation mimicked a situation when the giver, receiver, and dyad effects were modeled given records related to the focal animals but not for non-focal animals.

The predictive performance of the fitted models varies depending on the validation strategies (Table 5.2). The block-by-social-group cross-validation yielded the lowest correlation. This could be the result of not accounting for factors affecting social group composition. In fact, Samarakone and Gonyou (2009) have suggested that pigs may shift their aggressive behaviors accordingly to the composition of their social groups. Consequently, this could be revisited in further analyses and experimental setups where more group-specific variables are recorded and included in the dyadic model. In short, animal behavioral studies may consider introducing group-specific effects into the proposed model and exploring how these effects influence interactive behaviors.

The block-by-focal-animal validation yielded a slightly higher correlation and smaller RMSE compared to the stratified 5-fold cross-validation (Table 5.2). The result suggests that selecting focal animals and decoding their interactions with all other animals in the group may be a more efficient way to build predictive models of dyadic interactions than randomly selecting snippets of video for decoding. This idea has also been suggested by ethologists (Bosholn and

Anciães, 2018). Furthermore, a dyadic model could be fitted using preliminary data to determine which factors better predict animal interactions, and then focal animals could be selected based on the significant factors to cover a large variation in responses. Such sampling strategies could be useful for improved manual video decoding efficiency.

6. CONCLUSION

We proposed an approach for the analysis of animals' social interactions based on modeling dyadic data. We illustrated its use through fitting a generalized linear model to total attacking time post-mixing between pairs of grow-finish pigs. Taking advantage of the flexibility and interpretability of the proposed model, we found that if two pigs had shared a common nursery pen immediately before being remixed into new social groups, they tended to spend less time engaging in the agonistic behavior. In addition, the positive correlation between the giver and receiver suggested that a pig that spent more time attacking was also more likely to be attacked for more time. The proposed model can be easily extended to incorporate additional giver-specific, receiver-specific, and dyad-specific effects. Moreover, we pursued alternative cross-validations and found that overlooking group-specific factors worsened the predictive performance of the proposed model. We also demonstrated that focusing on a fraction of all animals and decoding all their interactions with the remaining animals in the group is an effective way to perform inference and predictions on social interactions in the group while limiting the amount of time and effort dedicated to decoding video.

7. ACKNOWLEDGEMENTS

This work was funded by NIFA Awards 2017-67007-26176 and 2021-67021-34150.

APPENDIX

1. Implementation detail of the Bayesian approach

In parameter estimation, marginal posterior distributions of the parameters were obtained using Markov chain Monte Carlo method through Stan program. We ran four chains of 15,000 iterations, where we set the burn-in (warmup) to 5,000 iterations and every 10th samples were saved in each chain. Convergence diagnostics and graphical posterior predictive checks were performed using *rstan* and *bayesplot* packages in R (R Core Team, 2020). In the following parts of Appendix, we first present the prior distributions that were used in this study. We then show trace plots and autocorrelation plots of the fitted model. In addition, the summary of convergence diagnostics for the model with random dyad effect are included at the end. In the companion GitHub (https://github.com/jun-jieh/DyadAnalysis) we provide examples for implementations of multiple GLMMs and their model fitting that utilized a Bayesian method through *rstan* package in R (Carpenter et al., 2017; R Core Team, 2020).

2. Prior distributions of model parameters that are described in Section 2.2

$$\begin{pmatrix} \sigma'^2_g & \sigma'_{gr} \\ \sigma'_{gr} & \sigma'^2_r \end{pmatrix} \sim Wishart\left(4, \begin{pmatrix} 5 & 0 \\ 0 & 5 \end{pmatrix}\right)$$

$$\begin{pmatrix} \sigma^2_g & \sigma_{gr} \\ \sigma_{gr} & \sigma^2_r \end{pmatrix} \sim Wishart\left(4, \begin{pmatrix} 5 & 0 \\ 0 & 5 \end{pmatrix}\right)$$

$$sg'_k \sim Uniform(0,10)$$

$$sg_k \sim Uniform(0,10)$$

$$\sigma \sim Uniform(0,100)$$

$$sex'_k \sim Uniform(-10,10)$$

$$sex_k \sim Uniform(-10,10)$$

$$\alpha' \sim Uniform(-100,100)$$

$$\alpha \sim Uniform(-100,100)$$

$$\beta' \sim Uniform(-100,100)$$

$$\beta \sim Uniform(-100,100)$$

$$\delta'_1 \sim Uniform(-10,10)$$

$$\delta_1 \sim Uniform(-10,10)$$

$$\delta'_2 \sim Uniform(-10,10)$$

$$\delta_2 \sim Uniform(-10,10)$$

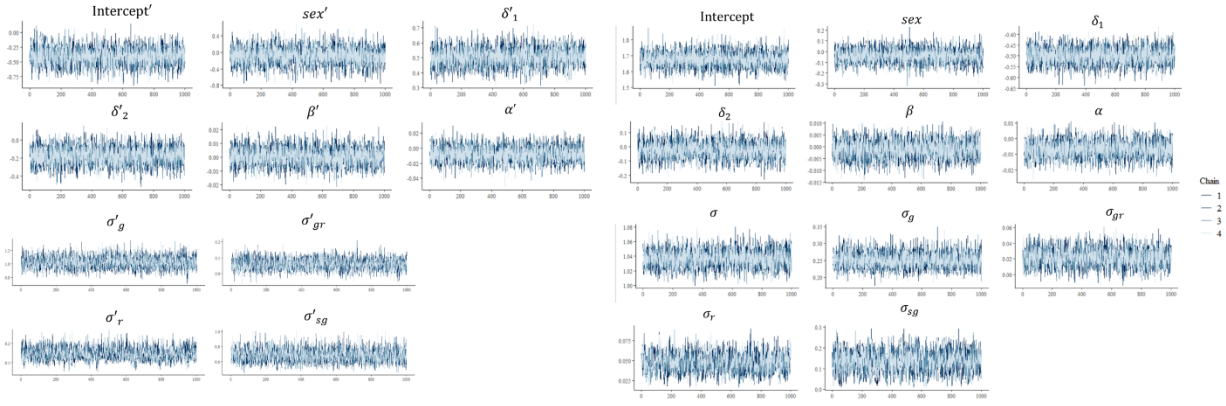

Figure S5.1 Trace plots of posterior estimates of effects and variance components.

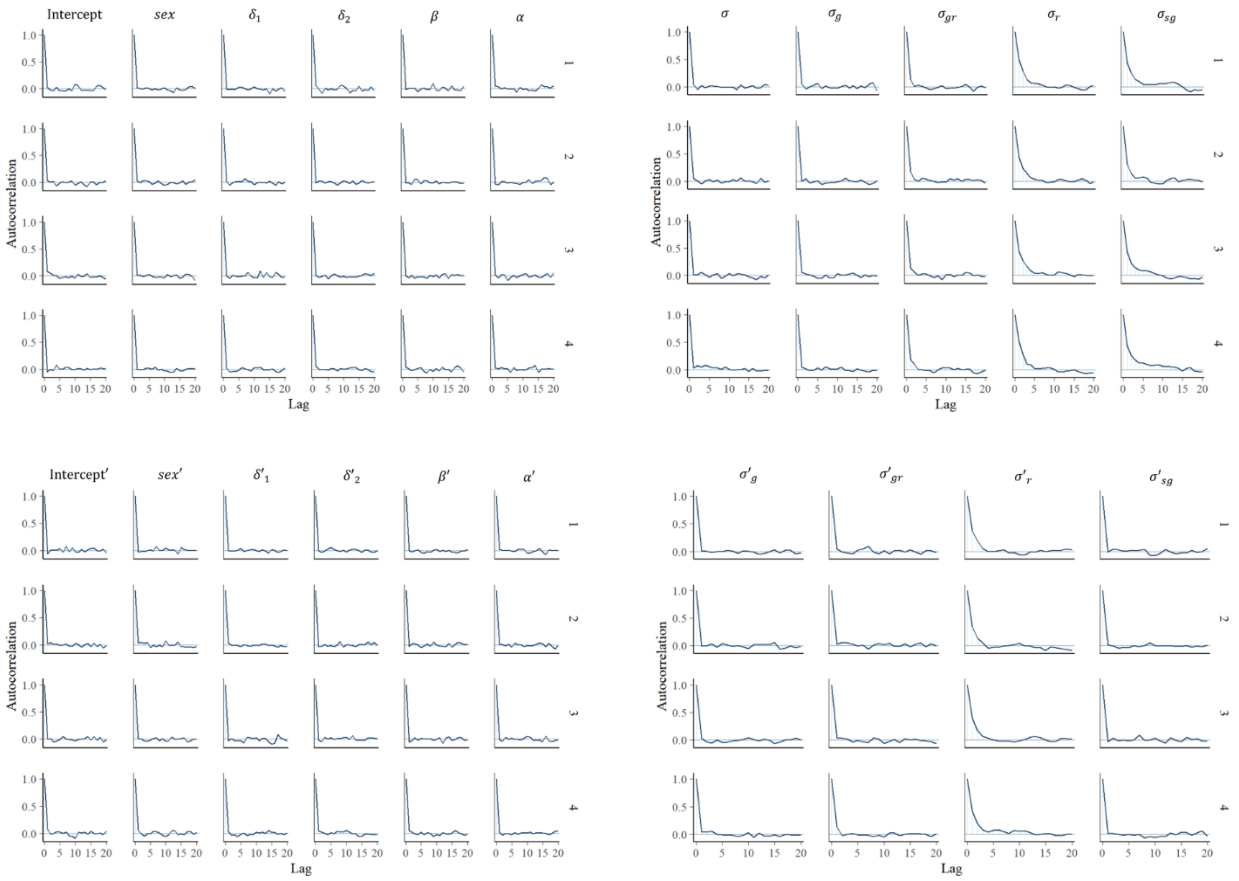

Figure S5.2. Autocorrelation plots by chain and by parameters.

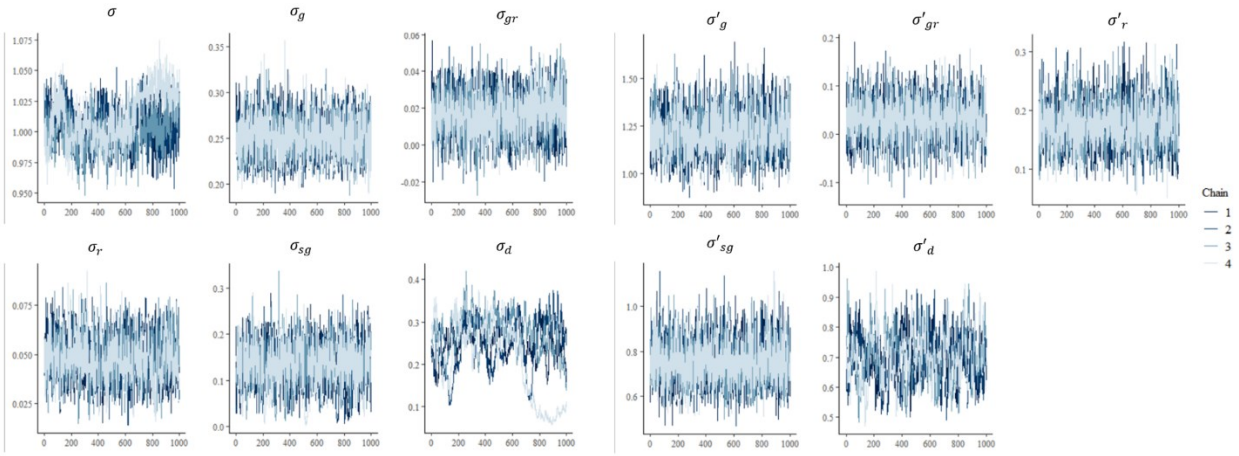

Figure S5.3 Trace plots of variance components when the random dyad effect is included in the model

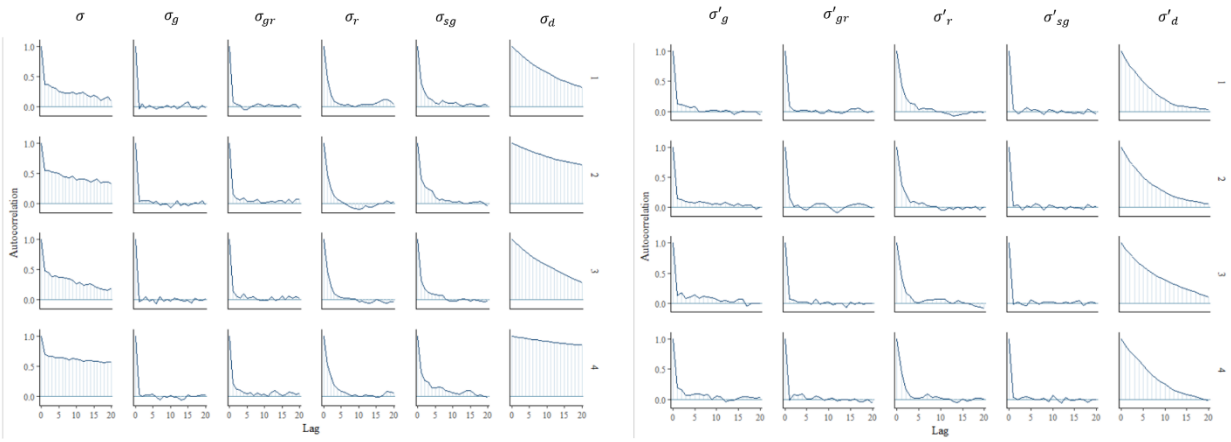

Figure S5.4 Autocorrelation plots by chain and by variance components when the random dyad effect is included in the model

Table S5.1 Summary of MCMC samples. Q, quantile; n_eff, effective sample size.

Parameter	mean	sd	Q2.5%	Q50%	Q97.5%	n_eff	Rhat
sex	-0.038	0.063	-0.159	-0.038	0.088	3711.37	1.002
δ_1	-0.502	0.038	-0.574	-0.501	-0.427	4127.503	1
δ_2	-0.023	0.059	-0.14	-0.023	0.095	3887.644	1
α	-0.001	0.004	-0.009	-0.001	0.006	3316.333	1
β	-0.007	0.005	-0.018	-0.007	0.004	3820.173	1
σ	1.006	0.018	0.973	1.005	1.042	35.164	1.102
σ_{sg}	0.134	0.048	0.030	0.135	0.223	954.933	1
σ_d	0.245	0.066	0.076	0.258	0.340	17.19	1.232
sex'	-0.096	0.218	-0.521	-0.094	0.337	3921.573	1
δ'_1	0.552	0.068	0.420	0.552	0.684	3572.37	1
δ'_2	-0.189	0.109	-0.402	-0.19	0.027	3438.372	1.001
α'	0.001	0.007	-0.012	0.001	0.014	4097.341	1
β'	-0.009	0.011	-0.029	-0.009	0.012	3935.423	1.002
σ'_{sg}	0.749	0.094	0.584	0.744	0.954	3759.123	1
σ'_d	0.697	0.079	0.548	0.697	0.855	266.248	1.004
σ_r	0.047	0.011	0.026	0.047	0.071	1281.134	1.001
σ'_r	0.173	0.039	0.103	0.172	0.253	1341.962	0.999
σ_g	0.251	0.022	0.212	0.250	0.295	3503.677	0.999
σ'_g	1.216	0.113	1.011	1.209	1.450	1264.305	1.001
ρ_{gr}	0.147	0.107	-0.066	0.145	0.360	1322.434	1.008
ρ'_{gr}	0.068	0.093	-0.12	0.071	0.244	2482.093	0.999
lp__	-17317.6	1792.383	-19481.7	-17766.3	-11893.2	15.225	1.27

REFERENCES

REFERENCES

Andersson, M., Simmons, L.W., 2006. Sexual selection and mate choice. Trends Ecol. Evol. 21, 296–302. https://doi.org/10.1016/j.tree.2006.03.015

Angarita, B.K., Cantet, R.J.C., Wurtz, K.E., O'Malley, C.I., Siegford, J.M., Ernst, C.W., Turner, S.P., Steibel, J.P., 2019. Estimation of indirect social genetic effects for skin lesion count in group-housed pigs by quantifying behavioral interactions. J. Anim. Sci. 97, 3658–3668. https://doi.org/10.1093/jas/skz244

Angarita, B.K., Han, J., Cantet, R.J.C., Chewning, S.K., Wurtz, K.E., Siegford, J.M., Ernst, C.W., Steibel, J.P., 2021. Estimation of direct and social effects of feeding duration in growing pigs using records from automatic feeding stations. J. Anim. Sci. 99, 1–8. https://doi.org/10.1093/jas/skab042

Back, M.D., Kenny, D.A., 2010. The Social Relations Model: How to Understand Dyadic Processes. Soc. Personal. Psychol. Compass 4, 855–870. https://doi.org/10.1111/j.1751-9004.2010.00303.x

Bierbach, D., Sassmannshausen, V., Streit, B., Arias-Rodriguez, L., Plath, M., 2013. Females prefer males with superior fighting abilities but avoid sexually harassing winners when eavesdropping on male fights. Behav. Ecol. Sociobiol. 67, 675–683. https://doi.org/10.1007/s00265-013-1487-8

Bosholn, M., Anciães, M., 2018. Focal Animal Sampling. Encycl. Anim. Cogn. Behav. 1–3. https://doi.org/10.1007/978-3-319-47829-6_262-1

Box, G.E.P., Tiao, G.C., 2011. Bayesian inference in statistical analysis. John Wiley & Sons.

Burgueño, J., de los Campos, G., Weigel, K., Crossa, J., 2012. Genomic prediction of breeding values when modeling genotype × environment interaction using pedigree and dense molecular markers. Crop Sci. 52, 707–719. https://doi.org/10.2135/cropsci2011.06.0299

Camerlink, I., Turner, S.P., Farish, M., Arnott, G., 2019. Advantages of social skills for contest resolution. R. Soc. Open Sci. 6, 1–8. https://doi.org/10.1098/rsos.181456

Camerlink, I., Turner, S.P., Farish, M., Arnott, G., 2015. Aggressiveness as a component of fighting ability in pigs using a game-theoretical framework. Anim. Behav. 108, 183–191. https://doi.org/10.1016/j.anbehav.2015.07.032

Carpenter, B., Gelman, A., Hoffman, M.D., Lee, D., Goodrich, B., Betancourt, M., Brubaker, M., Guo, J., Li, P., Riddell, A., 2017. Stan: A probabilistic programming language. J. Stat. Softw. 76.

Chai, T., Draxler, R.R., 2014. Root mean square error (RMSE) or mean absolute error (MAE)? - Arguments against avoiding RMSE in the literature. Geosci. Model Dev. 7, 1247–1250.

https://doi.org/10.5194/gmd-7-1247-2014

Gabry, J., Simpson, D., Vehtari, A., Betancourt, M., Gelman, A., 2019. Visualization in Bayesian workflow. J. R. Stat. Soc. Ser. A Stat. Soc. 182, 389–402. https://doi.org/10.1111/rssa.12378

Kenny, D.A., Kashy, D.A., Cook, W.L., 2020. Dyadic data analysis. Guilford Publications.

Kenny, D.A., West, T. V., Malloy, T.E., Albright, L., 2006. Componential analysis of interpersonal perception data. Personal. Soc. Psychol. Rev. 10, 282–294. https://doi.org/10.1207/s15327957pspr1004_1

Koster, J., Leckie, G., Miller, A., Hames, R., 2015. Multilevel modeling analysis of dyadic network data with an application to Ye'kwana food sharing. Am. J. Phys. Anthropol. 157, 507–512. https://doi.org/10.1002/ajpa.22721

Li, Y., Wang, L., 2011. Effects of previous housing system on agonistic behaviors of growing pigs at mixing. Appl. Anim. Behav. Sci. 132, 20–26. https://doi.org/10.1016/j.applanim.2011.03.009

Ling, C.X., Huang, J., Zhang, H., 2003. AUC: A statistically consistent and more discriminating measure than accuracy. IJCAI Int. Jt. Conf. Artif. Intell. 519–524.

Lu, D., Jiao, S., Tiezzi, F., Knauer, M., Huang, Y., Gray, K.A., Maltecca, C., 2017. The relationship between different measures of feed efficiency and feeding behavior traits in Duroc pigs. J. Anim. Sci. 95, 3370. https://doi.org/10.2527/jas2017.1509

Mendl, M., Held, S., Byrne, R.W., 2010. Pig cognition. Curr. Biol. 20, 796–798. https://doi.org/10.1016/j.cub.2010.07.018

Oczak, M., Ismayilova, G., Costa, A., Viazzi, S., Sonoda, L.T., Fels, M., Bahr, C., Hartung, J., Guarino, M., Berckmans, D., Vranken, E., 2013. Analysis of aggressive behaviours of pigs by automatic video recordings. Comput. Electron. Agric. 99, 209–217. https://doi.org/10.1016/j.compag.2013.09.015

Oldham, L., Camerlink, I., Arnott, G., Doeschl-Wilson, A., Farish, M., Turner, S.P., 2020. Winner–loser effects overrule aggressiveness during the early stages of contests between pigs. Sci. Rep. 10, 1–13. https://doi.org/10.1038/s41598-020-69664-x

Peden, R.S.E., Turner, S.P., Boyle, L.A., Camerlink, I., 2018. The translation of animal welfare research into practice: The case of mixing aggression between pigs. Appl. Anim. Behav. Sci. 204, 1–9. https://doi.org/10.1016/j.applanim.2018.03.003

R Core Team, 2020. R: A Language and Environment for Statistical Computing.

Rodenburg, T.B., Bijma, P., Ellen, E.D., Bergsma, R., De Vries, S., Bolhuis, J.E., Kemp, B., Van Arendonk, J.A.M., 2010. Breeding amiable animals? Improving farm animal welfare by including social effects in breeding programmes. Anim. Welf. 19, 77–82.

Samarakone, T.S., Gonyou, H.W., 2009. Domestic pigs alter their social strategy in response to social group size. Appl. Anim. Behav. Sci. 121, 8–15. https://doi.org/10.1016/j.applanim.2009.08.006

Savory, C.J., Mann, J.S., 1997. Behavioural development in groups of pen-housed pullets in relation to genetic strain, age and food form. Br. Poult. Sci. 38, 38–47. https://doi.org/10.1080/00071669708417938

Silk, M.J., Finn, K.R., Porter, M.A., Pinter-Wollman, N., 2018. Can Multilayer Networks Advance Animal Behavior Research? Trends Ecol. Evol. 33, 376–378. https://doi.org/10.1016/j.tree.2018.03.008

Turner, S.P., Roehe, R., D'Eath, R.B., Ison, S.H., Farish, M., Jack, M.C., Lundeheim, N., Rydhmer, L., Lawrence, A.B., 2009. Genetic validation of postmixing skin injuries in pigs as an indicator of aggressiveness and the relationship with injuries under more stable social conditions. J. Anim. Sci. 87, 3076–3082. https://doi.org/10.2527/jas.2008-1558

Turner, S.P., Roehe, R., Mekkawy, W., Farnworth, M.J., Knap, P.W., Lawrence, A.B., 2008. Bayesian analysis of genetic associations of skin lesions and behavioural traits to identify genetic components of individual aggressiveness in pigs. Behav. Genet. 38, 67–75. https://doi.org/10.1007/s10519-007-9171-2

Turner, S.P., Weller, J.E., Camerlink, I., Arnott, G., Choi, T., Doeschl-Wilson, A., Farish, M., Foister, S., 2020. Play fighting social networks do not predict injuries from later aggression. Sci. Rep. 10, 1–16. https://doi.org/10.1038/s41598-020-72477-7

Vehtari, A., Gelman, A., Gabry, J., 2017. Practical Bayesian model evaluation using leave-one-out cross-validation and WAIC. Stat. Comput. 27, 1413–1432. https://doi.org/10.1007/s11222-016-9696-4

Vehtari, A., Ojanen, J., 2012. A survey of Bayesian predictive methods for model assessment, selection and comparison. Stat. Surv. 6, 142–228. https://doi.org/10.1214/12-ss102

Verdon, M., Morrison, R.S., Hemsworth, P.H., 2018. Forming groups of aggressive sows based on a predictive test of aggression does not affect overall sow aggression or welfare. Behav. Processes 150, 17–24. https://doi.org/10.1016/j.beproc.2018.02.016

Wang, Z, Doekes, H.P. and Bijma P., 2022. Analysis of social behaviors in large groups: simulation and genetic evaluation. Proceedings of the WCGALP. Rotterdam, July 2022.

Wurtz, K.E., Siegford, J.M., Bates, R.O., Ernst, C.W., Steibel, J.P., 2017. Estimation of genetic parameters for lesion scores and growth traits in group-housed pigs. J. Anim. Sci. 95, 4310–4317. https://doi.org/10.2527/jas2017.1757

CHAPTER 6: GENERAL DICUSSION

1. DISCUSSION

Predictive modeling has great potential to improve swine farming efficiency in various contexts. Despite the success of predictive modeling methods (Putka et al., 2018), many of them are not yet applied to swine farming. Furthermore, the validity of those state-of-the-art prediction models remains unknown in pig genomic prediction and behavior recognition. Genomic prediction refers to the prediction of an animal's measurable trait or genetic value, and it has the potential for improved animal selection and reduced costs (Hickey et al., 2017). However, measuring animals' traits is costly in both time and labor and thus, predictive models for automated phenotyping (through video analysis) are helpful to obtain more rapid results. In this dissertation, I explored and adapted deep learning (DL) and generalized linear mixed model (GLMM) for the studies of animal breeding and behavior. To validate the models, several strategies were investigated to split data into the training data and validation data, where the training data was used for model development and the validation data was used for model evaluation.

Hyperparameter tuning is non-trivial and is a bottleneck for adapting DL into pig genomic prediction. The common practice is to optimize hyperparameters through grid search or exhaustive search, but they are costly in terms of time and computational power. In Chapter 2, I utilized differential evolution to search for hyperparameters that saved considerable time compared to the traditional approach. During the model development, I found that hyperparameter tuning was not the only factor that influenced predictive performance of DL. Different training-validation data splits as well as training dataset size also led to varying

performance. Compared to random hyperparameter configurations and the hyperparameters recommended in literature, the optimized DL models in this study showed significant performance gains. For the comparison of different genomic prediction methods, the prediction accuracy of the optimized DL tied to the standard genomic prediction method, genomic best linear unbiased prediction (VanRaden, 2008), suggesting that DL can be used as an alternative method for genomic prediction.

As the livestock sector is undergoing data revolution, computer vision (CV) is emerging as a powerful solution for phenotyping and behavioral studies (Borges Oliveira et al., 2021). However, to date, there is not yet a reference CV dataset in livestock farming, which poses a challenge to develop video-based automatic phenotyping systems for animals. In Chapter 3, I investigated the small number of public imagery datasets that have been used to develop CV systems in livestock farming, and I reviewed the validation strategies utilized in the related work. In this review, I considered data as the fuel of DL-based CV algorithms. Most CV applications in livestock farming used random validation for model assessment, in which the training and validation sets were split at random. However, results from random validations could be overoptimistic, and random validation is less representative of real-life validation scenarios, as environments for capturing images are quite complex in animal farming (Li et al., 2021). I also found that in the studies which fitted the same model through different validation strategies (Alameer et al., 2020; Riekert et al., 2020; Shao et al., 2020), the evaluation metrics obtained from blocked validation strategies (where the training and validation sets are split by a blocking factor) were lower than the metrics computed for random validations. These results are relevant to researchers as they are more interested in how CV is validated in a way that examines how

well the model can be generalized to other contexts (e.g., across different seasons and different farms), which is closer to blocked validation strategies.

The traditional method for analyzing pigs' activities at feeders is through direct observation or by filming and later manual decoding of videos (Agha et al., 2020; Csermely and Wood-Gush, 1990; Machado et al., 2017; Nielsen et al., 1995). However, such approaches are not possible on a commercial farming setup (Martínez-Avilés et al., 2017). In Chapter 4, I employed a state-of-the-art DL model for behavior recognition that learned both spatial and temporal features of pigs from videos. Hyperparameter tuning was performed, but little improvement was achieved in the optimization process. However, I found a major factor that greatly influenced the predictive performance of the model, which was validation strategies that split the dataset differently for training and validation purposes. For random validation (the standard validation approach of CV applications to animal farming), the proposed model yielded encouraging results. In addition to the random validation, I proposed blocked-by-time validation and blocked-by-feeder validation to evaluate the same model. As a result, the blocked validations yieled much lower performance compared to random validation. Through this finding, I demonstrated that the random validation strategy might neglect temporal structure and spatial structure, which were also concerned by other researchers (Bergmeir and Benítez, 2012; Roberts et al., 2017). These results suggest that blocked-by-time and blocked-by-feeder validation shows much lower yet more reliable estimates of DL model performance.

In Chapter 5, I emphasized that the unit of observation in dyadic interaction was not an individual, but a pair of individuals. I showed conceptualization, parameterization, and implementation of GLMMs that can be used to analyze dyadic data. The proposed model exhibited advantages of analyzing dyadic interactions of pairs of animals in terms of flexibility

(as it can be easily adapted to analyses different types of responses) and interpretability (as it decomposes the dyadic interaction into the giver animal effect, receiver animal effect, and dyad effect). As expected, predictive performance of the model varied in different validation strategies. In this study, different validation strategies could be related to possible situations in real-life applications or relevant prediction problems (Burgueño et al., 2012). The stratified 5-fold cross-validation was designed for evaluation when the model was used for predicting unobserved social behaviors between two animals. For block-by-social-group validation, it mimicked a situation where the effects of giver, receiver, and dyad were evaluated in some social groups but not in others. Similarly, the block-by-focal-animals validation mimicked a situation when the giver, receiver, and dyad effects were modeled given records related to the focal animals but not in non-focal animals. Interestingly, block-by-focal-animal validation yield a slightly better performance than random validation. This result suggested that focusing on a fraction of all animals and decoding all their interactions with the remaining animals in the group was an effective way to perform inference and predictions on social interactions in the group while limiting the amount of time and effort dedicated to decoding video. This is informative to ethologists and breeders.

2. FUTURE DIRECTIONS

Deep learning-based CV models are powerful predictive tools in swine farming. Nevertheless, most published CV applications into animal farming are developed using rather small datasets, and their broader validity remains unknown. There are needs to create reference image datasets and standard validation methods depending on the livestock species and prediction problems, which allows CV developers to benchmark their state-of-the-art algorithms.

To date, CV applications are shown to be promising tools to assist in animal behavioral studies for many challenging tasks e.g., detecting anomalous behaviors and phenotyping complex traits. However, those applications are currently "technical islands" as each CV model was developed for very specific problems, while in livestock farming, we are more interested in a versatile model that address multiple problems simultaneously e.g., behavior recognition and bodyweight estimation through a single CV system. Thus, there may be a need for developing integrated systems that pool information for multiple purposes.

Lastly, a key step in livestock farming is animal identification that is useful for both management and phenotyping. In commercial pig farming, most pigs have white coat color, and their graphical morphologies vary in images as they move, which poses challenge to CV systems to identify individuals. To address this problem, interdisciplinary work might be required between animal scientists and computer scientists to extract reliable visual components of pigs that contributes to animal identification through CV. Future work could be focused on robust CV models for pig identification.

REFERENCES

REFERENCES

Agha, S., Fàbrega, E., Quintanilla, R., Sánchez, J.P., 2020. Social network analysis of agonistic behaviour and its association with economically important traits in pigs. Animals 10, 1–13. https://doi.org/10.3390/ani10112123

Alameer, A., Kyriazakis, I., Bacardit, J., 2020. Automated recognition of postures and drinking behaviour for the detection of compromised health in pigs. Sci. Rep. 10, 1–15. https://doi.org/10.1038/s41598-020-70688-6

Bergmeir, C., Benítez, J.M., 2012. On the use of cross-validation for time series predictor evaluation. Inf. Sci. (Ny). 191, 192–213. https://doi.org/10.1016/j.ins.2011.12.028

Borges Oliveira, D.A., Ribeiro Pereira, L.G., Bresolin, T., Pontes Ferreira, R.E., Reboucas Dorea, J.R., 2021. A review of deep learning algorithms for computer vision systems in livestock. Livest. Sci. 253, 104700. https://doi.org/10.1016/j.livsci.2021.104700

Burgueño, J., de los Campos, G., Weigel, K., Crossa, J., 2012. Genomic prediction of breeding values when modeling genotype × environment interaction using pedigree and dense molecular markers. Crop Sci. 52, 707–719. https://doi.org/10.2135/cropsci2011.06.0299

Csermely, D., Wood-Gush, D.G.M., 1990. Agonistic behaviour in grouped sows. Ii. how social rank affects feeding and drinking behaviour. Bolletino di Zool. 57, 55–58. https://doi.org/10.1080/11250009009355674

Hickey, J.M., Chiurugwi, T., Mackay, I., Powell, W., 2017. Genomic prediction unifies animal and plant breeding programs to form platforms for biological discovery. Nat. Genet. 49, 1297–1303. https://doi.org/10.1038/ng.3920

Li, G., Huang, Y., Chen, Z., Chesser, G.D., Purswell, J.L., Linhoss, J., Zhao, Y., 2021. Practices and applications of convolutional neural network-based computer vision systems in animal farming: A review. Sensors 21, 1–42. https://doi.org/10.3390/s21041492

Machado, S.P., Caldara, F.R., Foppa, L., De Moura, R., Gonçalves, L.M.P., Garcia, R.G., De Alencar Nääs, I., Dos Santos Nieto, V.M.O., De Oliveira, G.F., 2017. Behavior of pigs reared in enriched environment: Alternatives to extend pigs attention. PLoS One 12, 1–18. https://doi.org/10.1371/journal.pone.0168427

Martínez-Avilés, M., Fernández-Carrión, E., López García-Baones, J.M., Sánchez-Vizcaíno, J.M., 2017. Early Detection of Infection in Pigs through an Online Monitoring System. Transbound. Emerg. Dis. 64, 364–373. https://doi.org/10.1111/tbed.12372

Nielsen, B.L., Lawrence, A.B., Whittemore, C.T., 1995. Effect of group size on feeding behaviour, social behaviour, and performance of growing pigs using single-space feeders. Anim. Sci. 61, 575–579. https://doi.org/10.1017/S1357729800014168

Psota, E.T., Mittek, M., Pérez, L.C., Schmidt, T., Mote, B., 2019. Multi-pig part detection and association with a fully-convolutional network. Sensors (Switzerland) 19, 1–24. https://doi.org/10.3390/s19040852

Putka, D.J., Beatty, A.S., Reeder, M.C., 2018. Modern Prediction Methods: New Perspectives on a Common Problem. Organ. Res. Methods 21, 689–732. https://doi.org/10.1177/1094428117697041

Riekert, M., Klein, A., Adrion, F., Hoffmann, C., Gallmann, E., 2020. Automatically detecting pig position and posture by 2D camera imaging and deep learning. Comput. Electron. Agric. 174. https://doi.org/10.1016/j.compag.2020.105391

Roberts, D.R., Bahn, V., Ciuti, S., Boyce, M.S., Elith, J., Guillera-Arroita, G., Hauenstein, S., Lahoz-Monfort, J.J., Schröder, B., Thuiller, W., Warton, D.I., Wintle, B.A., Hartig, F., Dormann, C.F., 2017. Cross-validation strategies for data with temporal, spatial, hierarchical, or phylogenetic structure. Ecography (Cop.). 40, 913–929. https://doi.org/10.1111/ecog.02881

Shao, W., Kawakami, R., Yoshihashi, R., You, S., Kawase, H., Naemura, T., 2020. Cattle detection and counting in UAV images based on convolutional neural networks. Int. J. Remote Sens. 41, 31–52. https://doi.org/10.1080/01431161.2019.1624858

VanRaden, P.M., 2008. Efficient methods to compute genomic predictions. J. Dairy Sci. 91, 4414–4423. https://doi.org/10.3168/jds.2007-0980