Power System Protection and Control
ISSN:1674-3415

EARLY PREDICTION OF 28-DAY COMPRESSIVE STRENGTH OF RECYCLED-AGGREGATE CONCRETE FROM FRESH-MIX SMARTPHONE IMAGES: COMPARATIVE EVALUATION OF TEN DEEP VISION ARCHITECTURES

Mehrdad Izadseresht[1], Seyed Amir Hossein Hashemi[2], Ghasem Akbari[3]

[1]Department of Civil Engineering , Qa .c., Isalamic Azad university , Qazvin , Iran
m.izadseresht@iau.ac.ir
[2]Department of Civil Engineering , Qa .c., Isalamic Azad university , Qazvin , Iran
sa.hashemi1360@iau.ac.ir
[3]Department of Mechanical Engineering , Qa .c., Isalamic Azad university , Qazvin , Iran
gh.akbari@iau.ac.ir
Corresponding author: Seyed Amir Hossein Hashemi

Received: 08/06/2026 Revised: 16/07/2026 Accepted: 10/08/2026

***ABSTRACT*:**

Early estimation of 28-day concrete compressive strength from the visual state of a fresh mixture could provide a rapid screening tool before conventional curing and destructive testing are completed. This study investigates an image-based prediction framework for concrete containing recycled aggregates derived from crushed pavement blocks that originally had a silica-rich wearing surface. Fifty-four concrete mixtures and 108 cube specimens were produced. Fresh concrete was photographed less than one minute after mixing with a Samsung Galaxy S23 Ultra fixed 30 cm above a tray under ring-light illumination. One hundred and fifty photographs were acquired for each mixture, yielding 8,100 raw images. The final benchmark comprised 108 selected representative images associated with the specimen records. Ten ImageNet-pretrained architectures—ConvNeXt-Small, ConvNeXt-Tiny, DenseNet121, EfficientNet-B0, MobileNetV3-Large, MobileNetV3-Small, ResNet18, ResNet34, ResNet50, and ViT-B/16—were evaluated using transfer learning, two-stage fine-tuning, and three random seeds (42, 123, and 2024). ConvNeXt-Small achieved the lowest mean validation RMSE (3.299 ± 0.039 MPa), followed by ConvNeXt-Tiny (3.448 ± 0.168 MPa). ViT-B/16 showed the highest seed sensitivity and the greatest computational demand. The findings support the feasibility of extracting strength-related information from standardized fresh-concrete smartphone images. The limited number of independent mixtures, the absence of a documented reproducible rule for selecting the final 108 representative images from the raw image pool, and the lack of external validation currently restrict generalization beyond the studied materials.

***Keywords**: Fresh Concrete; Recycled Aggregate Concrete; Compressive Strength; Computer Vision; Convnext; Transfer Learning; Smartphone Imaging; Deep Learning*

INTRODUCTION

Compressive strength is a central performance indicator in concrete engineering, yet the conventional 28-day test inherently delays definitive strength information until curing has progressed for several weeks. This limitation has encouraged the development of predictive approaches that can provide earlier information from mixture composition, material properties, or other measurable signals. The challenge is particularly relevant for recycled-aggregate concrete because the residual mortar, higher absorption, surface heterogeneity, and more complex interfacial transition zones associated with recycled particles can increase response variability [1–4].

Recycled-aggregate concrete has therefore become an active area for data-driven modelling. An earlier companion publication from the same experimental program evaluated eight classical machine-learning algorithms—ANN, SVR, GBM, RF, Lasso regression, Ridge regression, Decision Tree, and KNN—using tabular mixture-design inputs [22]. The present article addresses a different research question and does not re-evaluate those classical models: it investigates whether the visual state of freshly mixed concrete, captured by a smartphone, can provide

DOI: 10.46121/pspc.54.3.43

a predictive signal for later compressive strength. The input modality, model families, processing pipeline, and central comparison are therefore image-based.

Image-based analysis provides a distinct source of information because the appearance of freshly mixed concrete can encode the spatial distribution of aggregate particles, paste-rich regions, local heterogeneity, visible voids, and moisture-related patterns. A second modelling path was therefore developed in which RGB images, rather than tabular mixture-design variables, were supplied directly to deep neural networks. This image-regression path is the exclusive focus of the present article.

The study was designed around contemporary transfer-learning architectures representing several generations of computer vision. Residual networks (ResNet18, ResNet34, and ResNet50) provide a classical deep convolutional baseline [9]. DenseNet121 introduces dense feature reuse [10]. EfficientNet-B0 and MobileNetV3 investigate lightweight scaling and mobile-oriented design [11,12]. ConvNeXt-Tiny and ConvNeXt-Small represent modernized convolutional networks that incorporate design ideas developed during the transformer era while retaining convolutional inductive biases [13]. ViT-B/16 represents the transformer family and models image patches through self-attention [14].

A methodological difficulty in small image datasets is information leakage. When several observations originate from the same concrete mixture, random image-level splitting can place highly related samples in both training and evaluation sets and inflate apparent performance. To reduce this risk, data were organized by mixture identifier (mix_id), with all records linked to one mixture kept within the same subset. This grouped strategy is consistent with established guidance for validation of structured and limited datasets [15–17].

Accordingly, the objectives of this article are to: (i) document a standardized fresh-concrete smartphone imaging protocol; (ii) describe the image-processing and leakage-control pipeline; (iii) compare ten pretrained deep vision architectures under a common regression framework; (iv) quantify predictive accuracy, seed-to-seed stability, and computational cost; and (v) identify the architecture that provides the strongest accuracy–efficiency balance for the present limited dataset.

MATERIALS AND METHODS

2.1 Experimental program and recycled aggregates

The laboratory program comprised 54 independent concrete mixtures and 108 cube specimens, with two parallel specimens produced for each mixture. The 28-day compressive strength of each specimen was measured, and the paired results were used to characterize the corresponding mixture. The water-to-cement ratio varied within the range 0.40–0.60.

The recycled aggregate was produced by crushing cementitious pavement or flooring elements with a silica-containing surface layer. After crushing and grading, the recycled material was separated into three particle-size groups. Large recycled aggregate was retained on the 19 mm sieve. Medium recycled aggregate was retained on the 9.5 and 4.75 mm sieves. Small recycled aggregate passed through the 4.75 mm sieve. This size classification was retained in the experimental dataset because particle size and the amount of adhered mortar can influence the resulting concrete microstructure and mechanical response [1–4].

The measured 28-day compressive strengths ranged approximately from 26.8 to 37.0 MPa. The image-regression task therefore mapped a standardized fresh-concrete RGB image to a continuous 28-day compressive-strength value within this experimental range.

Experimental item	Dissertation specification
Concrete mixtures	54
Cube specimens	108 (two per mixture)
Water-to-cement ratio	0.40–0.60
Large recycled aggregate	Retained on 19 mm sieve
Medium recycled aggregate	Retained on 9.5 and 4.75 mm sieves
Small recycled aggregate	Passed 4.75 mm sieve
Target variable	28-day compressive strength (MPa)

DOI: 10.46121/pspc.54.3.43

Power System Protection and Control
ISSN:1674-3415

Reported strength range	~26.8–37.0 MPa

Table 1. Experimental design and target variable used in the image-based study.

2.2 Fresh-concrete image acquisition

Image capture was performed immediately after concrete mixing. For each mixture, freshly mixed concrete was poured onto a tray less than one minute after completion of mixing. A Samsung Galaxy S23 Ultra smartphone was mounted on a fixed support at a nominal distance of 30 cm from the tray. A ring light positioned behind the camera was used to provide controlled illumination and reduce local shadows.

For each of the 54 mixtures, 150 photographs were recorded from different viewing angles, producing 8,100 raw images. Different ring-light colour-temperature conditions were considered during acquisition while camera-to-sample geometry remained fixed. The controlled protocol was intended to reduce nuisance variation caused by distance, illumination, background, and camera position, thereby directing the learning process toward concrete texture and aggregate distribution.

Figure 1. Freshly mixed concrete photographed under the standardized smartphone image-acquisition setup.

2.3 Image-processing pipeline

A multistage image-processing pipeline was used before regression modelling. The stages comprised standardized image acquisition, semantic segmentation, geometric and textural filtering, data augmentation, and final quality control.

Semantic segmentation was performed with a U-Net architecture. Training masks identified the concrete region as foreground and the surrounding scene as background. The purpose of segmentation was to prevent the downstream regression models from learning spurious associations with the tray, table, sample edges, or other visually irrelevant regions. After segmentation, additional geometric and colour/texture filtering was used to remove regions inconsistent with the main concrete surface. Quality-control measures included image sharpness and edge-detail checks.

The raw acquisition contained 8,100 photographs. For the final ten-model benchmark, 108 selected representative images were retained, described in the source experimental record as one representative image associated with each specimen record. A reproducible algorithmic rule for selecting these 108 representatives from the raw image pool was not preserved in the available study documentation. The reported 108-image benchmark is therefore retained without inventing an undocumented selection procedure, and this point is treated explicitly as a limitation.

DOI: 10.46121/pspc.54.3.43

2.4 Data organization and leakage prevention

All images and specimens were organized using a mixture identifier (mix_id). The central rule was that observations associated with one concrete mixture could appear in only one of the training, validation, or test subsets. This group-wise organization was adopted to prevent highly related observations from the same mixture from being shared across subsets.

The final experiment suite used a fixed mixture-wise 70/15/15 train/validation/test partition. Mixtures were stratified by compressive-strength range before splitting so that the target distribution remained reasonably balanced across subsets. All records linked to a given mix_id remained in one subset, and preprocessing parameters were determined from the training data before application to validation and test data. The development documentation also discusses K-fold training as a methodological option; however, the final execution loop used for the reported ten-model benchmark iterated over architecture and random seed with preconstructed train, validation, and test loaders. Consequently, the values reported in Table 3 are presented as validation RMSE, consistent with the final results table.

2.5 Image preprocessing and augmentation

The final deep-learning configuration resized images to 224 × 224 pixels. Pixel values were normalized using the standard ImageNet channel statistics: mean = [0.485, 0.456, 0.406] and standard deviation = [0.229, 0.224, 0.225]. This preprocessing was aligned with the pretrained torchvision weights used for transfer learning.

Data augmentation was applied only to the training set. The final configuration included random horizontal flipping, while deterministic resizing and normalization were used for validation and test images. Controlled geometric and illumination transformations were also considered during development, but the final reported transform pipeline retained only transformations compatible with the physical appearance of the fresh-concrete surface.

2.6 Deep-learning architectures

Ten pretrained image architectures were evaluated under the same regression objective: ConvNeXt-Small, ConvNeXt-Tiny, DenseNet121, EfficientNet-B0, MobileNetV3-Large, MobileNetV3-Small, ResNet18, ResNet34, ResNet50, and ViT-B/16. The original classification heads were replaced with single-output regression heads that predicted 28-day compressive strength in MPa.

The architecture set was selected to compare conventional residual CNNs, densely connected networks, lightweight mobile networks, compound-scaled CNNs, modern ConvNeXt models, and a vision transformer. All models used pretrained ImageNet weights when available.

Family	Architectures included	Role in benchmark
Residual CNN	ResNet18, ResNet34, ResNet50	Depth and residual-connection baseline
Dense CNN	DenseNet121	Feature reuse through dense connectivity
Efficient/lightweight CNN	EfficientNet-B0	Compound-scaled efficient model
Mobile CNN	MobileNetV3-Small, MobileNetV3-Large	Low-cost deployment-oriented models
Modern CNN	ConvNeXt-Tiny, ConvNeXt-Small	Modernized convolutional design
Vision transformer	ViT-B/16	Patch-based self-attention model

Table 2. Ten deep vision architectures evaluated under the common regression framework.

2.7 Transfer learning and optimization

Because the number of independent concrete mixtures was limited, transfer learning and a two-stage training strategy were used. In the first stage, the pretrained backbone was frozen while the newly initialized regression head and normalization parameters were trained. After six epochs, the backbone was unfrozen and the complete network was fine-tuned using a lower learning rate for the pretrained layers.

DOI: 10.46121/pspc.54.3.43

AdamW was used as the optimizer. The head learning rate was 1e-4, the backbone learning rate was 1e-5, and weight decay was 1e-2 during full fine-tuning. The final DataLoader configuration used a batch size of 16. Early stopping used a patience of 10 epochs and a minimum validation-RMSE improvement threshold of 0.001 MPa. MSELoss was used for most architectures. For ConvNeXt, HuberLoss was adopted after preliminary MSE experiments produced stronger validation-loss oscillations. A 5 MPa transition threshold was used in the Huber formulation. Automatic mixed precision was disabled for ConvNeXt because of numerical-stability concerns in the available hardware environment.

2.8 Hardware, reproducibility, and evaluation

The deep-learning experiments were conducted on an NVIDIA GTX 1660 Ti GPU with 6 GB of VRAM. Each architecture was trained with three independent random seeds: 42, 123, and 2024. Validation RMSE was recorded for each run and summarized using the mean and standard deviation across seeds.

The principal error metric was root mean squared error (RMSE):

$$RMSE = \sqrt{[(1/N)\ \Sigma_{i=1}^{N}\ (y_i - \hat{y}_i)^2]}$$

where y_i is the measured 28-day compressive strength and $\hat{y}_i$ is the model prediction. Lower RMSE indicates better predictive accuracy. The coefficient of determination (R^2) was also implemented as a complementary evaluation metric, but the complete ten-model seed-wise R^2 values were not retained in the final summary table. To avoid reconstructing unreported values, the comparative results in this article are therefore based on the fully available validation RMSE records.

RESULTS

3.1 Predictive accuracy across ten architectures

The repeated-validation results show a clear separation between the two ConvNeXt models and the remaining architectures. ConvNeXt-Small achieved the lowest mean validation RMSE, 3.299 MPa, and also showed low run-to-run variation (SD = 0.039 MPa). ConvNeXt-Tiny ranked second with a mean RMSE of 3.448 MPa and a larger standard deviation of 0.168 MPa.

MobileNetV3-Large produced the strongest performance among the lightweight architectures, with a mean RMSE of 4.962 MPa. DenseNet121, ResNet34, ResNet50, MobileNetV3-Small, ViT-B/16, ResNet18, and EfficientNet-B0 all yielded mean RMSE values above 5 MPa. The ViT-B/16 model showed the largest standard deviation (0.349 MPa), indicating the greatest sensitivity to random initialization among the architectures tested.

Architecture	Seed 42	Seed 123	Seed 2024	Mean RMSE	SD
ConvNeXt-Small	3.299	3.272	3.327	3.299	0.039
ConvNeXt-Tiny	3.597	3.480	3.268	3.448	0.168
DenseNet-121	4.972	5.439	5.199	5.203	0.234
EfficientNet-B0	5.486	5.720	5.345	5.517	0.191
MobileNet-V3-Large	4.826	4.868	5.191	4.962	0.200
MobileNet-V3-Small	5.320	5.353	5.380	5.351	0.030
ResNet-18	5.441	5.416	5.511	5.456	0.049
ResNet-34	5.262	5.330	5.117	5.236	0.109
ResNet-50	5.372	5.272	5.334	5.326	0.050
ViT-B/16	5.836	5.160	5.299	5.432	0.349

Table 3. Validation RMSE (MPa) for the ten deep-learning architectures across three independent random seeds.

DOI: 10.46121/pspc.54.3.43

Power System Protection and Control
ISSN:1674-3415

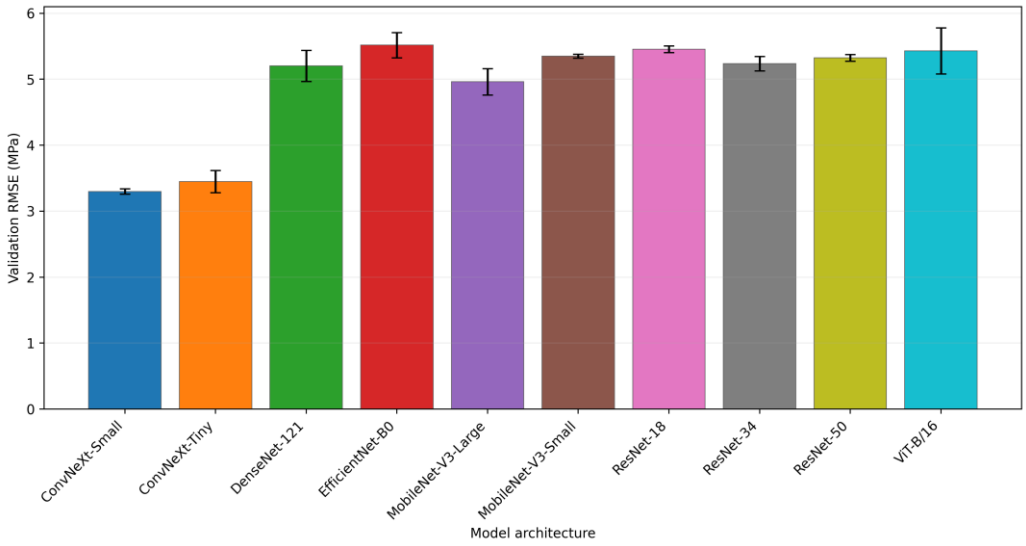

Figure note. Figures 2–5 were regenerated directly from the numerical values in Tables 3 and 4 so that the visual comparisons correspond exactly to the ten architectures included in the final benchmark.

Figure 2. Mean validation RMSE (±SD) across three independent random seeds for the ten evaluated architectures.

3.2 Performance stability across random seeds

The seed analysis reinforces the accuracy ranking. ConvNeXt-Small remained within a narrow error interval across the three runs, while MobileNetV3-Small also showed very low variability despite its higher average error. By contrast, ViT-B/16 produced a wide RMSE range across seeds. DenseNet121 also showed substantial run-to-run variability relative to most convolutional models.

Seed-to-seed variability reinforces the accuracy ranking. ConvNeXt-Small had an RMSE standard deviation of 0.039 MPa, while MobileNetV3-Small showed an even smaller standard deviation of 0.030 MPa but a higher mean error. ViT-B/16 had the largest standard deviation (0.349 MPa), followed by DenseNet121 (0.234 MPa). Based on the three seed-wise values in Table 3, the maximum–minimum RMSE range was 0.055 MPa for ConvNeXt-Small and 0.676 MPa for ViT-B/16, illustrating the substantially greater initialization sensitivity of the transformer model.

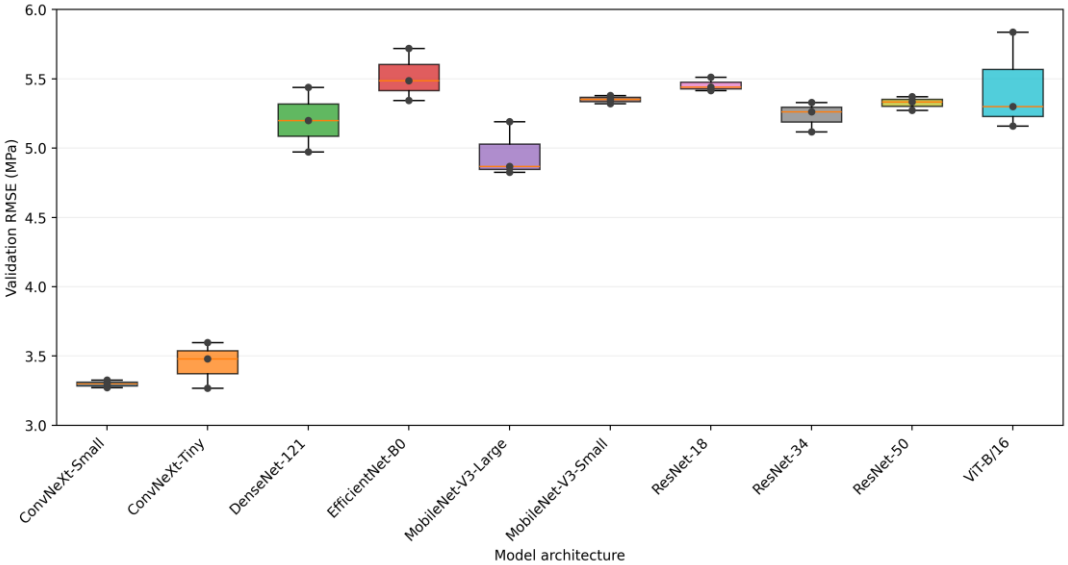

Figure 3. Seed-wise validation RMSE distribution for the ten architectures using the three independent runs reported in Table 3.

DOI: 10.46121/pspc.54.3.43

Power System Protection and Control
ISSN:1674-3415

3.3 Computational complexity and training time

The computational analysis demonstrates that a larger or more expensive architecture did not automatically provide a lower error. ViT-B/16 was the computationally heaviest model in the reported benchmark, with 57.30 million parameters, 11.285 GFLOPs, and a mean training time of 288.2 minutes, yet it did not approach the accuracy of the ConvNeXt models. ConvNeXt-Small required 49.44 million parameters, 8.697 GFLOPs, and 89.9 minutes of training, but it achieved the best RMSE.

At the opposite end of the resource spectrum, MobileNetV3-Small required only 1.52 million parameters and 0.061 GFLOPs, with a mean training time of 12.3 minutes. Its prediction error was higher (5.351 MPa), but its RMSE standard deviation was only 0.030 MPa. ConvNeXt-Small and MobileNetV3-Small therefore represent different operating points: the former prioritizes predictive accuracy, whereas the latter minimizes computational demand.

Architecture	Mean RMSE (MPa)	Parameters (M)	GFLOPs	Training time (min)
ConvNeXt-Small	3.299	49.44	8.697	89.9
ConvNeXt-Tiny	3.448	27.81	4.463	51.2
DenseNet-121	5.203	6.95	2.896	67.6
EfficientNet-B0	5.517	4.01	0.414	57.0
MobileNet-V3-Large	4.962	4.20	0.234	23.6
MobileNet-V3-Small	5.351	1.52	0.061	12.3
ResNet-18	5.456	11.18	1.824	39.9
ResNet-34	5.236	21.29	3.678	67.1
ResNet-50	5.326	23.51	4.132	162.2
ViT-B/16	5.432	57.30	11.285	288.2

Table 4. Predictive accuracy and computational characteristics of the ten evaluated architectures.

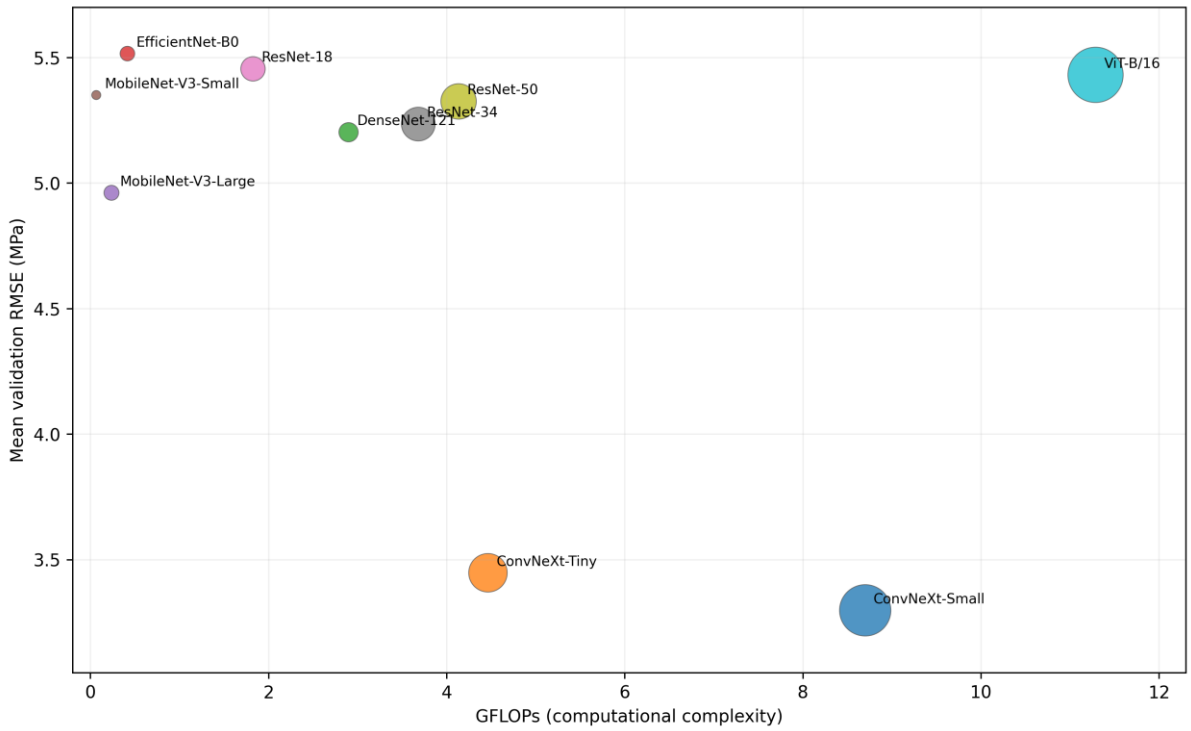

Figure 4. Trade-off between mean validation RMSE and computational complexity (GFLOPs); bubble size is proportional to parameter count.

DOI: 10.46121/pspc.54.3.43

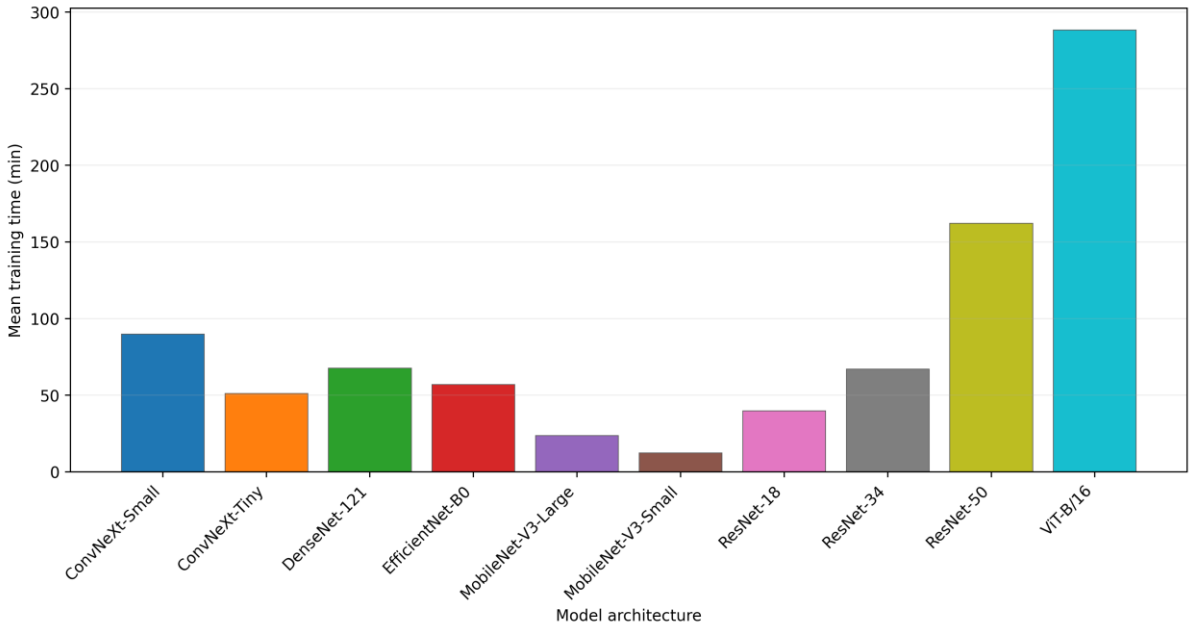

Figure 5. Mean training-time comparison for the ten evaluated architectures.

3.4 Architecture-specific observations

ConvNeXt family: ConvNeXt-Small and ConvNeXt-Tiny formed the two most accurate models. Their performance is consistent with the suitability of modern convolutional design for texture analysis, including large-kernel depthwise convolutions and Layer Normalization. ConvNeXt-Small combined the lowest mean error with low seed sensitivity.

ResNet family: ResNet18, ResNet34, and ResNet50 produced intermediate-to-high errors. Increasing depth did not yield a monotonic improvement: ResNet34 slightly outperformed both ResNet18 and ResNet50 in mean RMSE. This result indicates that extra depth alone was not sufficient to improve performance in the limited dataset.

DenseNet121: DenseNet121 achieved a mean RMSE of 5.203 MPa, but its standard deviation of 0.234 MPa indicated greater sensitivity to initialization than the ResNet models and ConvNeXt-Small.

EfficientNet-B0: EfficientNet-B0 produced the highest mean RMSE among the principal convolutional models (5.517 MPa). In this limited dataset, compound scaling did not translate into superior regression performance, suggesting that architectural efficiency on large-scale classification tasks does not automatically transfer to a small concrete-image regression problem.

MobileNetV3: MobileNetV3-Large achieved a mean RMSE below 5 MPa and performed better than the Small version. MobileNetV3-Small, however, was among the most stable architectures and required the least computational cost.

ViT-B/16: ViT-B/16 yielded a mean RMSE of 5.432 MPa and the largest standard deviation, 0.349 MPa. Its higher seed sensitivity is consistent with the weaker convolutional locality bias of a vision transformer and the greater data demand typically associated with learning spatial relationships from image patches.

DISCUSSION

The principal finding of the image-based branch is that a standardized RGB photograph of freshly mixed recycled-aggregate concrete can contain sufficient visual information to support a measurable association with 28-day compressive strength. The model does not receive the water-to-cement ratio, cement content, or replacement percentage directly. Instead, the deep network learns visual representations from the material surface. The result

DOI: 10.46121/pspc.54.3.43

does not imply that an image uniquely determines concrete strength; rather, it demonstrates that the visual state contains predictive cues that can be exploited statistically within the studied mixtures.

This finding is physically plausible from a materials perspective. Recycled aggregate introduces variability through adhered mortar, absorption, particle morphology, and complex interfacial regions [1–4]. These factors alter aggregate–paste distribution, visible void structure, surface moisture patterns, and fresh-mixture heterogeneity. An RGB photograph cannot resolve nanoscale interfacial chemistry directly, but it can encode macroscopic texture and spatial organization that are influenced by the underlying material state.

The superiority of ConvNeXt-Small is also consistent with the structure of the input. Fresh-concrete photographs are highly textural and contain information at several spatial scales: individual particles, paste-rich zones, voids, local clustering, and broader distribution patterns. ConvNeXt preserves convolutional locality while using architectural updates such as larger kernels and Layer Normalization [13]. Within the small dataset, these inductive biases appear to have been more advantageous than the patch-based self-attention formulation of ViT-B/16.

The seed results are particularly important. A single successful run can overstate model reliability when data are limited. Repeating every architecture with seeds 42, 123, and 2024 showed that model selection should consider both mean error and run-to-run variability. ConvNeXt-Small remained accurate across repeated runs, whereas ViT-B/16 and DenseNet121 varied more strongly. Small-sample validation literature similarly emphasizes that performance uncertainty can be substantial when the number of independent observations is limited [15–17].

The computational comparison also prevents an overly simple interpretation of model capacity. ViT-B/16 had the largest reported parameter count and highest GFLOPs, but did not provide the best prediction. ResNet50 was substantially slower to train than several alternatives without achieving lower RMSE. Conversely, MobileNetV3-Small was extremely inexpensive computationally and highly stable, though less accurate. These results support application-dependent model selection rather than selection by parameter count alone.

The acquisition protocol is a second major contribution of the study. The use of a consumer smartphone, fixed 30 cm distance, controlled ring lighting, a tray, and imaging within one minute of mixing defines a reproducible low-cost procedure. Standardization is important because a deep network can otherwise learn differences in lighting, background, or camera geometry instead of concrete characteristics. The segmentation stage was included for the same reason: U-Net masks isolate the concrete region and reduce the opportunity for background shortcuts [18].
The present results should be interpreted as proof-of-concept evidence. Although 8,100 raw photographs were recorded, independent experimental diversity remained limited to 54 mixtures and 108 specimen records, and the final benchmark used only 108 selected representative images. The available experimental record does not preserve a reproducible rule for selecting those representatives from the raw image pool. This limitation prevents a fully independent reconstruction of the selection step and should be corrected in future studies by retaining image-level identifiers, selection criteria, and complete experiment logs.

A further limitation is the controlled imaging environment. The fixed camera distance and ring-light setup reduce nuisance variation and are appropriate for algorithm comparison, but field conditions may involve different illumination, camera devices, vibration, background, moisture condition, and concrete placement geometry. Independent validation on additional projects, material sources, devices, and imaging conditions is therefore required before the method is used for engineering decision-making.

Finally, the reported mean validation RMSE of approximately 3.3 MPa for ConvNeXt-Small should be interpreted relative to the measured strength range. The result is useful for experimental comparison and early screening within the present dataset, but it does not establish that image-based prediction can replace standardized compressive-strength testing. The appropriate current interpretation is an image-assisted early prediction framework that may complement established testing after broader validation.

DOI: 10.46121/pspc.54.3.43

CONCLUSIONS

1. The results support the feasibility of predicting later compressive strength from standardized images captured from freshly mixed recycled-aggregate concrete.
2. The imaging protocol used a Samsung Galaxy S23 Ultra fixed 30 cm above a tray, ring-light illumination, imaging within one minute of mixing, and 150 images for each of 54 mixtures, yielding 8,100 raw photographs.
3. The recycled aggregate was separated into large (retained on the 19 mm sieve), medium (retained on the 9.5 and 4.75 mm sieves), and small (passing the 4.75 mm sieve) particle-size groups.
4. The image-analysis workflow included U-Net segmentation, image filtering and quality control, group-wise data organization by mix_id, ImageNet-compatible preprocessing, and transfer learning.
5. Among ten tested architectures, ConvNeXt-Small achieved the best mean validation RMSE (3.299 ± 0.039 MPa), followed by ConvNeXt-Tiny (3.448 ± 0.168 MPa).
6. ViT-B/16 was the most seed-sensitive model and also the most computationally demanding architecture in the reported benchmark.
7. MobileNetV3-Small provided the lowest computational cost and very high repeatability, illustrating a possible low-resource alternative when maximum accuracy is not the only objective.
8. The current evidence remains limited by the small number of independent mixtures, the specific recycled-aggregate source, controlled imaging conditions, the undocumented selection rule for the 108-image final benchmark, and the absence of external-project validation. Larger multi-source datasets, a fully traceable image-selection protocol, and independent test datasets are required before field application.

DECLARATIONS

Funding: To be confirmed by the authors before journal submission.
Conflict of interest: To be confirmed by the authors before journal submission.
Data availability: The study generated raw images, tabular records, model checkpoints, and experiment logs. Repository access and a permanent accession identifier should be supplied by the authors before journal submission.
Author contributions: To be completed by the authors according to the target journal's CRediT requirements.

REFERENCES

1. Wang, B., Yan, L.-B., Fu, Q., & Kasal, B. (2021). A comprehensive review on recycled aggregate and recycled aggregate concrete. Resources, Conservation and Recycling, 171, 105565. https://doi.org/10.1016/j.resconrec.2021.105565
2. Memon, S. A., Bekzhanova, Z., & Murzakarimova, A. (2022). A review of improvement of interfacial transition zone and adherent mortar in recycled concrete aggregate. Buildings, 12(10), 1600. https://doi.org/10.3390/buildings12101600
3. Muhammad, F., Harun, M., Ahmed, A., Kabir, N., Khalid, H. R., & Hanif, A. (2024). Influence of bonded mortar on recycled aggregate concrete properties: A review. Construction and Building Materials, 432, 136564. https://doi.org/10.1016/j.conbuildmat.2024.136564
4. Chen, Q., Zhang, J., Wang, Z., Zhao, T., & Wang, Z. (2024). A review of the interfacial transition zones in concrete: Identification, physical characteristics, and mechanical properties. Engineering Fracture Mechanics, 300, 109979. https://doi.org/10.1016/j.engfracmech.2024.109979
5. Tran, V. Q., Nguyen, H. L., Dao, V. D., Hilloulin, B., Nguyen, L. K., Nguyen, Q. H., & Le, T. T. (2022). Evaluating compressive strength of concrete made with recycled concrete aggregates

using machine learning approaches. Construction and Building Materials, 323, 126578. https://doi.org/10.1016/j.conbuildmat.2022.126578

6. Yuan, X., Chen, W., Pham, T. M., & Hao, H. (2022). Machine learning prediction models to evaluate the strength of recycled aggregate concrete. Materials, 15(8), 2823. https://doi.org/10.3390/ma15082823
7. Zhang, X., Dai, C., Li, W., & Chen, Y. (2023). Prediction of compressive strength of recycled aggregate concrete using machine learning and Bayesian optimization methods. Frontiers in Earth Science, 11, 1112105. https://doi.org/10.3389/feart.2023.1112105
8. Zhao, Z., Liu, Y., Lu, Y., Ji, C., Lin, C., Yao, L., Pu, Z., & de Brito, J. (2024). Prediction of properties of recycled aggregate concrete using machine learning models: A critical review. Journal of Building Engineering, 90, 109516. https://doi.org/10.1016/j.jobe.2024.109516
9. He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 770–778. https://doi.org/10.1109/CVPR.2016.90
10. Huang, G., Liu, Z., van der Maaten, L., & Weinberger, K. Q. (2017). Densely connected convolutional networks. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 4700–4708. https://doi.org/10.1109/CVPR.2017.243
11. Tan, M., & Le, Q. V. (2019). EfficientNet: Rethinking model scaling for convolutional neural networks. Proceedings of the 36th International Conference on Machine Learning, 97, 6105–6114.
12. Howard, A., Sandler, M., Chu, G., Chen, L.-C., Chen, B., Tan, M., Wang, W., Zhu, Y., Pang, R., Vasudevan, V., Le, Q. V., & Adam, H. (2019). Searching for MobileNetV3. Proceedings of the IEEE/CVF International Conference on Computer Vision, 1314–1324. https://doi.org/10.1109/ICCV.2019.00140
13. Liu, Z., Mao, H., Wu, C.-Y., Feichtenhofer, C., Darrell, T., & Xie, S. (2022). A ConvNet for the 2020s. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 11976–11986. https://doi.org/10.1109/CVPR52688.2022.01167
14. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., & Houlsby, N. (2021). An image is worth 16 × 16 words: Transformers for image recognition at scale. International Conference on Learning Representations.
15. Vabalas, A., Gowen, E., Poliakoff, E., & Casson, A. J. (2019). Machine learning algorithm validation with a limited sample size. PLOS ONE, 14(11), e0224365. https://doi.org/10.1371/journal.pone.0224365
16. Cawley, G. C., & Talbot, N. L. C. (2010). On over-fitting in model selection and subsequent selection bias in performance evaluation. Journal of Machine Learning Research, 11, 2079–2107.
17. 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, 40(8), 913–929. https://doi.org/10.1111/ecog.02881

DOI: 10.46121/pspc.54.3.43

18. Ronneberger, O., Fischer, P., & Brox, T. (2015). U-Net: Convolutional networks for biomedical image segmentation. In Medical Image Computing and Computer-Assisted Intervention—MICCAI 2015 (pp. 234–241). Springer. https://doi.org/10.1007/978-3-319-24574-4_28
19. Poon, C.-S., Shui, Z.-H., Lam, L., Fok, H., & Kou, S.-C. (2004). Influence of moisture states of natural and recycled aggregates on the slump and compressive strength of concrete. Cement and Concrete Research, 34(1), 31–36. https://doi.org/10.1016/S0008-8846(03)00186-8
20. Tam, V. W. Y., Soomro, M., & Evangelista, A. C. J. (2021). Quality improvement of recycled concrete aggregate by removal of residual mortar: A comprehensive review of approaches adopted to increase the quality of recycled concrete aggregate. Construction and Building Materials, 288, 123066. https://doi.org/10.1016/j.conbuildmat.2021.123066
21. Jagadesh, P., Khan, A. H., Priya, B. S., Asheeka, A., Zoubir, Z., Magbool, H. M., Alam, S., & Bakather, O. Y. (2024). Artificial neural network, machine learning modelling of compressive strength of recycled coarse aggregate based self-compacting concrete. PLOS ONE, 19(5), e0303101. https://doi.org/10.1371/journal.pone.0303101
22. Izadseresht, M., Hashemi, S. A. H., & Akbari, G. (2025). Performance Evaluation and Machine Learning Predictions for Recycled Concrete with Silica-Coated Pavement Aggregates. Sciences of Conservation and Archaeology, 37(4), 850–865. https://doi.org/10.48141/sci-arch-37.3.25.73

SUPPLEMENTARY APPENDIX A. IMPLEMENTATION EXCERPTS

The following code excerpts preserve the implementation logic and parameter values available in the study record. Formatting has been normalized for readability. Project-specific helper functions, local paths, and code not present in the source record have not been reconstructed; consequently, this appendix documents the implementation rather than serving as a standalone executable repository.

A.1 Image transforms and DataLoader

```
from torchvision import transforms
from torch.utils.data import DataLoader

train_transform = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.RandomHorizontalFlip(),
    transforms.ToTensor(),
    transforms.Normalize(
        mean=[0.485, 0.456, 0.406],
        std=[0.229, 0.224, 0.225]
    ),
])

val_test_transform = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.ToTensor(),
    transforms.Normalize(
        mean=[0.485, 0.456, 0.406],
        std=[0.229, 0.224, 0.225]
    ),
])

NUM_WORKERS = 0
BATCH_SIZE = 16
```


DOI: 10.46121/pspc.54.3.43

```
train_loader = DataLoader(
    train_dataset,
    batch_size=BATCH_SIZE,
    shuffle=True,
    num_workers=NUM_WORKERS,
    pin_memory=True,
    persistent_workers=NUM_WORKERS > 0,
)

val_loader = DataLoader(
    val_dataset,
    batch_size=BATCH_SIZE,
    shuffle=False,
    num_workers=NUM_WORKERS,
    pin_memory=True,
    persistent_workers=NUM_WORKERS > 0,
)

test_loader = DataLoader(
    test_dataset,
    batch_size=BATCH_SIZE,
    shuffle=False,
    num_workers=NUM_WORKERS,
    pin_memory=True,
    persistent_workers=NUM_WORKERS > 0,
)
```

A. A.2 Ten-model registry

```
def _build_convnext(variant: str) -> nn.Module:
    builder, weights = {
        "convnext_tiny": (
            models.convnext_tiny,
            models.ConvNeXt_Tiny_Weights.DEFAULT
        ),
        "convnext_small": (
            models.convnext_small,
            models.ConvNeXt_Small_Weights.DEFAULT
        ),
    }[variant]
    m = builder(weights=weights)
    m.classifier[-1] = _regression_head(
        m.classifier[-1].in_features
    )
    return m


def _build_vit(variant: str) -> nn.Module:
    builder, weights = {
        "vit_b_16": (
            models.vit_b_16,
            models.ViT_B_16_Weights.DEFAULT
        ),
    }[variant]
    m = builder(weights=weights)
    m.heads.head = _regression_head(
        m.heads.head.in_features
    )
```


DOI: 10.46121/pspc.54.3.43

```
    return m


MODEL_REGISTRY: dict[str, Callable[[], nn.Module]] = {
    "resnet18": lambda: _build_resnet("resnet18"),
    "resnet34": lambda: _build_resnet("resnet34"),
    "resnet50": lambda: _build_resnet("resnet50"),
    "densenet121": _build_densenet121,
    "efficientnet_b0": lambda: _build_efficientnet("efficientnet_b0"),
    "mobilenet_v3_small": lambda: _build_mobilenet("mobilenet_v3_small"),
    "mobilenet_v3_large": lambda: _build_mobilenet("mobilenet_v3_large"),
    "convnext_tiny": lambda: _build_convnext("convnext_tiny"),
    "convnext_small": lambda: _build_convnext("convnext_small"),
    "vit_b_16": lambda: _build_vit("vit_b_16"),
}
```

B. A.3 Reproducibility and seed-aware DataLoader

```
def seed_worker(worker_id: int) -> None:
    worker_seed = torch.initial_seed() % (2**32)
    np.random.seed(worker_seed)
    random.seed(worker_seed)


def make_loader(dataset, shuffle: bool, seed: int,
                batch_size: int = 16) -> DataLoader:
    g = torch.Generator()
    g.manual_seed(seed)
    return DataLoader(
        dataset,
        batch_size=batch_size,
        shuffle=shuffle,
        num_workers=0,
        pin_memory=True,
        persistent_workers=False,
        worker_init_fn=seed_worker,
        generator=g,
    )

SEED_LIST = [42, 123, 2024]
```

C. A.4 AdamW with discriminative learning rates

```
def build_optimizer(model: nn.Module, head_attr: str,
                    head_lr: float = 1e-4,
                    backbone_lr: float = 1e-5,
                    weight_decay: float = 1e-2) -> torch.optim.AdamW:
    head_params, backbone_params = [], []

    for name, param in model.named_parameters():
        if not param.requires_grad:
            continue
        (head_params if head_attr in name else backbone_params).append(param)

    return torch.optim.AdamW([
        {"params": backbone_params, "lr": backbone_lr},
        {"params": head_params, "lr": head_lr},
    ], weight_decay=weight_decay)
```

D. A.5 Mixed precision, output clamping and gradient clipping

```
scaler = torch.cuda.amp.GradScaler(enabled=use_amp)
```


DOI: 10.46121/pspc.54.3.43

```
with torch.autocast(device_type="cuda", enabled=use_amp):
    outputs = model(images).squeeze(1).clamp(-1e3, 1e3)
    loss = criterion(outputs, targets)

scaler.scale(loss).backward()
scaler.unscale_(optimizer)
torch.nn.utils.clip_grad_norm_(model.parameters(), 5.0)
scaler.step(optimizer)
scaler.update()

# Implementation note:
# use_amp = False for ConvNeXt in the final configuration.
```

E. A.6 Early stopping

```
early_stopping_patience = 10
early_stopping_delta = 0.001

class EarlyStopping:
    def __init__(self, patience=10, delta=0.001):
        self.patience = patience
        self.delta = delta
        self.best_score = None
        self.counter = 0
        self.best_weights = None

    def __call__(self, val_rmse, model):
        if (self.best_score is None or
                val_rmse < self.best_score - self.delta):
            self.best_score = val_rmse
            self.best_weights = {
                k: v.clone()
                for k, v in model.state_dict().items()
            }
            self.counter = 0
        else:
            self.counter += 1

        return self.counter >= self.patience
```

F. A.7 Training-suite loop across architectures and seeds

```
results_summary = []

for arch_name in MODEL_REGISTRY.keys():
    for seed in SEED_LIST:  # [42, 123, 2024]
        run_dir = f"checkpoints/{arch_name}/seed_{seed}"
        os.makedirs(run_dir, exist_ok=True)

        result = train_one_model(
            arch_name=arch_name,
            seed=seed,
            run_dir=run_dir,
            train_loader=train_loader,
            val_loader=val_loader,
            test_loader=test_loader,
            max_epochs=MAX_EPOCHS,
            patience=PATIENCE,
        )

        results_summary.append(result)
```


DOI: 10.46121/pspc.54.3.43

```
        save_results_log(results_summary, "experiment_log.csv")
```

G. A.8 Final evaluation and result record

```
model.load_state_dict(
    torch.load(
        f"{run_dir}/best_model.pth",
        map_location=device
    )
)
model.eval()

test_rmse, test_r2 = evaluate(model, test_loader)

result = {
    "architecture": arch_name,
    "seed": seed,
    "best_val_rmse": best_rmse,
    "test_rmse": test_rmse,
    "test_r2": test_r2,
    "total_epochs_run": epoch + 1,
}

with open(f"{run_dir}/DONE.flag", "w") as f:
    f.write("completed")

return result
```

H. A.9 Export of aggregated results

```
def export_to_excel(results_df, env_info,
                    output_path="results.xlsx"):
    with pd.ExcelWriter(output_path, engine="openpyxl") as writer:
        results_df.to_excel(
            writer,
            sheet_name="All_Runs",
            index=False
        )

        summary = results_df.groupby("architecture").agg(
            mean_test_rmse=("test_rmse", "mean"),
            std_test_rmse=("test_rmse", "std"),
            mean_r2=("test_r2", "mean"),
        ).reset_index()

        summary.to_excel(
            writer,
            sheet_name="Summary_Stats",
            index=False
        )

        summary.sort_values("mean_test_rmse").reset_index(drop=True).to_excel(
            writer,
            sheet_name="Model_Ranking"
        )

        pd.DataFrame([env_info]).to_excel(
            writer,
            sheet_name="Environment",
            index=False
        )
```


DOI: 10.46121/pspc.54.3.43