19th Australasian Tunnelling Conference – Acosta-Martínez, Stewart and Hudson-Smith (Eds.)

Applying convolutional and artificial neural networks for image-based numerical modelling analysis in tunnelling

N. Mirsepassi
WSP, Sydney, Australia

A. Mastan
Sydney, Australia

ABSTRACT: Tunnel engineering relies on numerical modelling to generate complex outputs essential for design and maintenance. However, manual interpretation of these outputs can be time-consuming and inconsistent. This study presents a Convolutional Neural Network (CNN) model to automate the interpretation process by identifying the number of tunnels in an image and extracting vertical displacement values from colour contour plots. Using a dataset of 2,500 images from softwares including RS2, FLAC2D, FLAC3D, UDEC, and 3DEC, the model standardises image preprocessing and applies convolutional and pooling layers to extract spatial features. It performs two tasks: classifying tunnel count and regressing displacement values, targeting 90% accuracy for tunnel count and 85% of displacement CNN model interpretation within 5 mm of actual numerical outputs. This framework significantly reduces processing time and variability while maintaining accuracy. Initially trained on model images, it demonstrates potential for broader applications. By automating feature extraction, the CNN enhances efficiency in tunnel analysis and supports faster, more consistent decision-making in infrastructure projects.

1 INTRODUCTION

Tunnels are integral to Australian infrastructure, supporting transport corridors, mining operations, and urban underground utilities. Numerical modelling generates detailed visual outputs—ground deformation, stress and strain distributions, and imposed structural actions—that engineers rely on for tunnel ground support design. These outputs, often presented as colour-coded images, encode critical data such as deformation at the tunnel crown, sidewalls, surface movement, and displacements at utility levels, as shown in Figure 1.

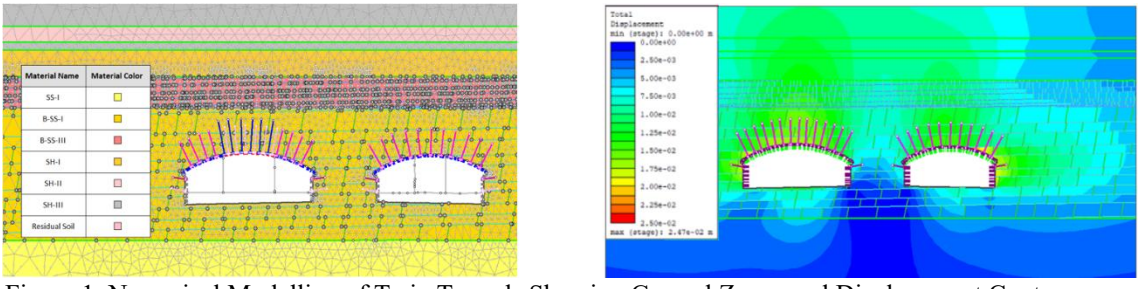

Figure 1. Numerical Modelling of Twin Tunnels Showing Ground Zones and Displacement Contours

In Figure 1 (left), the numerical model depicts subsurface geological zones and material boundaries around a twin arched-tunnel excavation, while the right-hand image shows the corresponding total displacement contours. Together, these plots highlight the spatial richness of numerical out-

puts and the challenge engineers face in interpreting them to assess tunnel stability. Manual interpretation especially 3D models can be time-consuming and subjective, introducing variability and risking delays in design delivery.

Deep learning, particularly CNNs, offers a solution. CNNs excel at image analysis by automatically detecting spatial patterns without manual feature engineering (Guo et al., 2018). While proven in fields such as medical imaging and autonomous driving, their application to tunnel engineering remains limited but promising. This study presents a CNN model designed to interpret numerical outputs, focusing on two key tasks: classifying the number of tunnels and interpreting vertical displacement at the crown. These tasks reflect broader capabilities, including stress concentration mapping and deformation detection.

The dataset comprises approximately 2,500 images from numerical models, representing varied tunnel configurations (i.e. single, twin, and multi-tunnel clusters). While vertical displacement is a key stability indicator, the model is structured to allow extension to other outputs with minimal redesign. This flexibility supports rapid interpretation across tunnelling metrics. The CNN targets 90% accuracy for tunnel count classification and 85% of displacement interpretations within 5 mm, benchmarks reflecting efficiency gains over manual analysis.

The motivation is twofold: tunnelling projects are becoming more complex, and conventional tools struggle with growing data volumes. The following sections describe the theoretical background, methodology, and future potential of CNN-driven tunnel analysis.

2 THEORITICAL BACKGROUND

2.1 *Artificial neural networks*

Artificial Neural Networks (ANNs) emulate biological systems using layers of interconnected nodes—input, hidden, and output (Goodfellow et al., 2016). Each neuron processes weighted inputs, adds a bias, and applies an activation function (e.g. Rectified Linear Unit (ReLU) to produce an output. During training, weights are adjusted via backpropagation to minimise prediction errors. While effective for simple tasks, ANNs are inefficient for images, as flattening pixel data inflates parameters and increases overfitting risk.

An ANN resembles a tunnelling design team: each engineer (neuron) evaluates multiple inputs, applies judgement (weights and biases), and passes recommendations forward. This process repeats until the final solution meets design requirements, mirroring how a network minimises prediction error.

ANNs typically follow four steps: forward propagation, output computation, error calculation, and backpropagation. In forward propagation, weighted inputs flow through hidden layers where non-linearity is introduced (Mirsepassi, 1997). Outputs are computed, compared against target values using a loss function, and errors are backpropagated using gradient descent to update weights. This iterative cycle allows the network to learn from data and refine predictions over time.

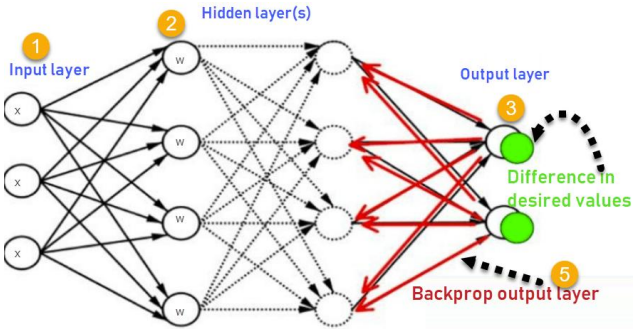

Figure 2. Schematic of an Artificial Neural Network Architecture with Forward and Backward Propagation (Ramzan et al., 2023)

2.2 *Convolutional Neural Networks*

Convolutional Neural Networks (CNNs) extend ANNs for spatial data, using convolution, pooling, and feature extraction to process images efficiently (Chollet, 2021). Convolutional layers apply filters to detect patterns like edges and textures, gradually learning more complex features. These operations produce feature maps that capture essential spatial information. Pooling layers then reduce dimensionality, typically via min/max/average-pooling while preserving critical features, improving efficiency and reducing overfitting. The resulting maps are flattened into a vector and passed to a Fully Connected Network (FCN), where hidden layers interpret higher-level patterns for classification or regression. This flow convolution → pooling → flattening → FCN → output converts raw images into meaningful predictions.

This process parallels tunnel inspections during construction: initial scans detect broad signs of movement or stress (convolutions), followed by focused investigation of critical areas (pooling), leading to detailed assessments and decisions (fully connected layers). This staged approach enables efficient interpretation without exhaustive manual review.

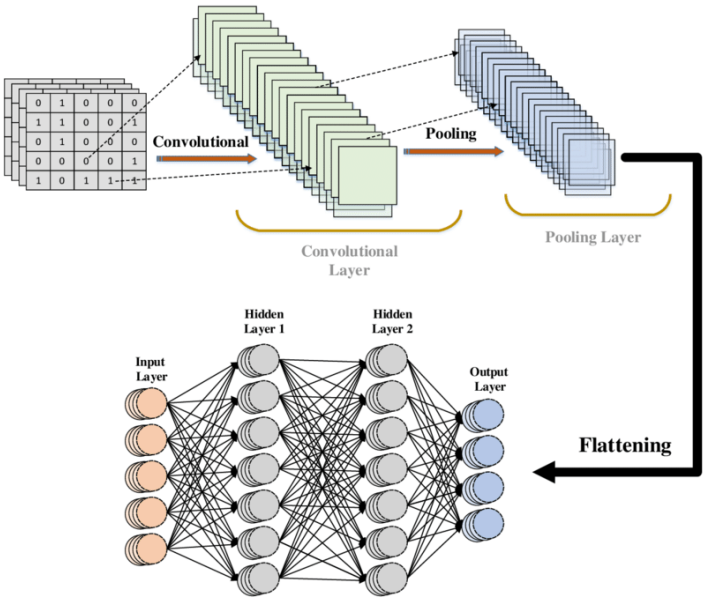

Figure 3. Overview of a Convolutional Neural Network Architecture Showing Convolution, Pooling, and Flattening Before Fully Connected Layers (Aqel and Marji, 2021)

3 METHODOLOGY

This study develops a dual-task CNN to automate the interpretation of numerical modelling outputs in tunnel engineering, focusing on tunnel count classification and CNN-based interpretation of vertical convergence at tunnel crowns. The methodology includes data collection, preprocessing, model architecture, and training.

Data Collection: The dataset comprises approximately 2,500 images generated using finite and discrete element methods from software such as RS2, UDEC, 3DEC, FLAC2D, and FLAC3D. These represent cross-sections under varied geological settings, depths, and tunnel geometries. RS2 accounts for ~1,300 images, followed by UDEC and 3DEC (~200 each), with FLAC3D contributing fewer. Tunnel counts range from 1 to 4, with a skew toward 2-tunnel images (~950). Vertical convergence is limited to images with up to two tunnels at the crown (Crown_T1 and Crown_T2), ranging from –5 mm to –55 mm. Additional parameters (e.g., stress, strain, structural actions) can be incorporated as training expands.

Preprocessing: Images are standardised to 200 by 200 pixels and normalised (0–1 range) for consistency across software outputs (Chollet, 2021). Tunnel counts are one-hot encoded; displacement values remain continuous. To address class imbalance—especially for 3- and 4-tunnel

images—oversampling is applied during training (Geron, 2022). This duplicates minority samples within batches to improve class representation and model generalisation.

The dataset is split 80/20 into training and validation subsets. Class distribution is preserved to ensure unbiased performance and reduce overfitting risk.

CNN Architecture: The architecture (Figure 4) consists of convolutional layers followed by a fully connected classifier. The tunnel classification model uses 200 × 200 × 3 inputs with filters of 32, 64, 128, and 256. The displacement model uses 256 × 256 × 3 inputs with 32, 64, and 128 filters (Guo et al., 2018). Max pooling reduces dimensionality while retaining key features.

In the regression model, features are flattened into 115,200 neurons, passed through dense layers with ~14.8M parameters, and output 2 crown values with a linear activation. The classification model flattens into 25,600 neurons, with ~3.67M parameters and a softmax output layer for 4-class classification.

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 254, 254, 32)	320
max_pooling2d (MaxPooling2D)	(None, 127, 127, 32)	0
conv2d_1 (Conv2D)	(None, 125, 125, 64)	18,496
max_pooling2d_1 (MaxPooling2D)	(None, 62, 62, 64)	0
conv2d_2 (Conv2D)	(None, 60, 60, 128)	73,856
max_pooling2d_2 (MaxPooling2D)	(None, 30, 30, 128)	0
flatten (Flatten)	(None, 115200)	0
dense (Dense)	(None, 128)	14,745,728
dropout (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 64)	8,256
dense_2 (Dense)	(None, 2)	130

Total params: 14,846,788 (56.64 MB)

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 198, 198, 32)	896
max_pooling2d (MaxPooling2D)	(None, 99, 99, 32)	0
conv2d_1 (Conv2D)	(None, 97, 97, 64)	18,496
max_pooling2d_1 (MaxPooling2D)	(None, 48, 48, 64)	0
conv2d_2 (Conv2D)	(None, 46, 46, 128)	73,856
max_pooling2d_2 (MaxPooling2D)	(None, 23, 23, 128)	0
conv2d_3 (Conv2D)	(None, 21, 21, 256)	295,168
max_pooling2d_3 (MaxPooling2D)	(None, 10, 10, 256)	0
flatten (Flatten)	(None, 25600)	0
dense (Dense)	(None, 128)	3,276,928
dense_1 (Dense)	(None, 4)	516

Total params: 3,665,862 (13.98 MB)

Figure 4. CNN Architecture Summaries for Image Recognition (right) and Vertical Displacement Interpretation (left), Including Layer Types, Shapes, and Parameters (TensorFlow/Keras v2.19)

The Python code used for this study includes Jupyter notebooks for model development and training, supported by custom scripts for data preparation, augmentation, and labelling. Image preprocessing and dataset handling were implemented using libraries such as NumPy, OpenCV, and Pandas. Model definition and training utilised TensorFlow/Keras frameworks, incorporating convolutional architectures tailored to classification and regression tasks. For model evaluation and presentation, Matplotlib and Seaborn were used to generate loss curves, interpretation-error plots, and confidence metrics, enabling visual interpretation of performance across tunnel types and displacement levels.

Training: The model is trained using a combined loss function: cross-entropy for tunnel count classification and mean squared error (MSE) for vertical displacement regression (Goodfellow et al., 2016). The Adaptive Moment Estimation (ADAM) optimiser (learning rate 0.001) is employed over 20 to 90 epochs with a batch size of 8 to 32, executed on a Graphical Processing Unit (GPU) in approximately 1 hour.

This methodology establishes a scalable framework for automated interpretation of the numerical outputs, with the potential to incorporate additional outputs through iterative data collection and model tuning.

4 RESULTS AND DISCUSSION

4.1 *Tunnel Count Classification*

The classification branch identifies the number of tunnels (1 to 4) by analysing spatial patterns in cross-sectional images, including boundaries, spacing, and layout. CNN convolutional layers detect tunnel outlines through edge and shape features, while max-pooling preserves spatial information, leading to a softmax layer that outputs class probabilities. Figure 5 - "Number of Tunnels

by Software" shows tunnel count distribution across various softwares, highlighting dataset diversity. The target accuracy of 90% reflects the model's ability to reliably classify tunnel groups.

Training dynamics (Figure 5 – "Model Loss and Accuracy") show training loss steadily decreases, while validation loss fluctuates slightly after 10 epochs, indicating minor overfitting. Accuracy stabilises around 0.95 for training and ~0.85 for validation by 15 epochs.

The CNN achieved 90% overall accuracy, correctly classifying 307 of 347 samples, with peak performance on 2-tunnel images (169 correct). Most misclassifications occurred between adjacent tunnel counts, particularly for 3-tunnel cases (Figure 5 – "Confusion Matrix").

Precision and recall (Figure 5 – " Precision and Recall by Class") exceed 0.9 for 1- and 2-tunnel classes but drop to 0.6–0.7 for underrepresented 3- and 4-tunnel classes. The Precision-Recall Curve (Figure 5 – "Precision and Recall Curve") shows sharper separation for dominant classes and flatter curves for minority ones, highlighting imbalance effects.

Figure 5 – "CNN Confidence" peaks near 1.0 for ~250 classifications, suggesting low uncertainty in dominant classes. The class imbalance shows two-tunnel images dominate, with few samples for 3- and 4-tunnel cases, particularly in FLAC3D and UDEC potentially affecting generalisation.

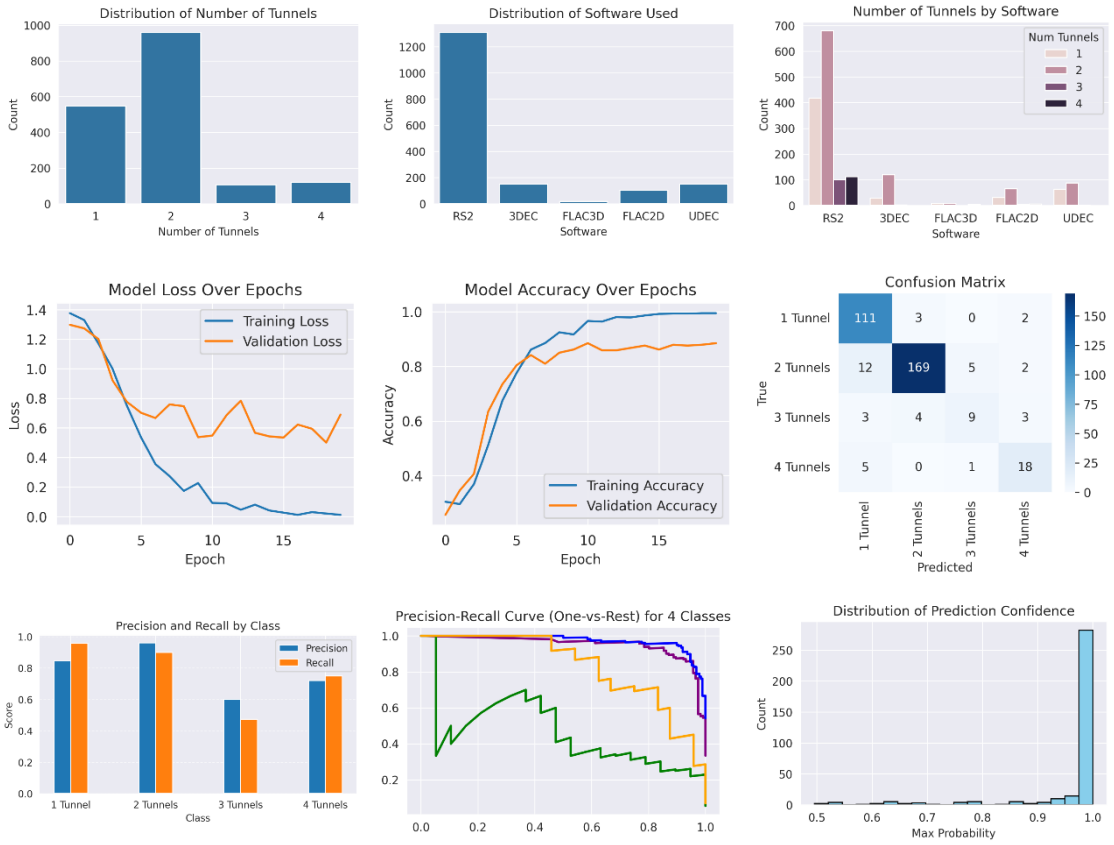

Figure 5. Model Performance Metrics for Tunnel Count Classification Including Data Distributions, Training Trends, and Evaluation Results

4.2 *Vertical Convergence Interpretation*

The regression branch interprets vertical settlement at the crown for images with a maximum of two tunnels (Crown_T1 and Crown_T2), a key indicator of tunnel stability. By analysing deformation gradients—pixel intensity shifts in displacement maps—the model aims for 85% of interpretations to be within 5 mm of the actual values, with the expectation of achieving higher accuracy and lower deviation as the dataset grows. Figure 6 – "Sample Test Interpretations" visually demonstrate the model's ability to interpret settlements (e.g., Interpreted: –0.025, –0.025 m vs.

True: –0.028, –0.025 m), with discrepancies within acceptable limits considering the data available for training. This precision supports the model's adaptability and allows retraining for other displacement metrics, such as sidewall movement or ground surface displacement.

For the vertical displacement model Figure 6 – "Loss Curve" indicates a steady decline in training loss, whereas validation loss stabilises around epoch 40, with minor fluctuations. This suggests that while the model generalises well, further regularisation techniques may improve performance. The objective is to ensure that 85% of settlement interpretations remain within 5 mm of the numerically predicted values.

The model accurately interprets settlements at Crown_T1 and Crown_T2, with 85% of absolute errors falling below 5 mm, as shown in Figure 6 – "Cumulative Distribution of Absolute Errors".

Figure 6 – "Interpreted vs Actual Vertical Settlement" demonstrates a strong linear correlation, with points closely aligned along the diagonal line, indicating that the model is effectively learning the true displacement trend.

Figure 6 – "Distribution of Interpretation Errors" shows that errors are centred around –0.001 m with a standard deviation of approximately 0.01 m, suggesting the model is relatively unbiased and its interpretations are tightly distributed around the ground truth.

Figure 6 – "Residual Plot" illustrates how interpretation errors vary across the range of interpreted settlement values. It shows the difference between interpreted and actual settlement values across the full range. Most residuals fall within ±0.02 m (±20 mm) but considering the target accuracy of ±0.005 m (5 mm), this spread suggests the model still produces a number of cases with higher-than-acceptable error, particularly for Crown_T2, which shows a slight positive bias meaning it tends to under-interpret larger (more negative) settlements. This may reflect limitations in the training data or challenges in capturing subtle deformation patterns near the crown.

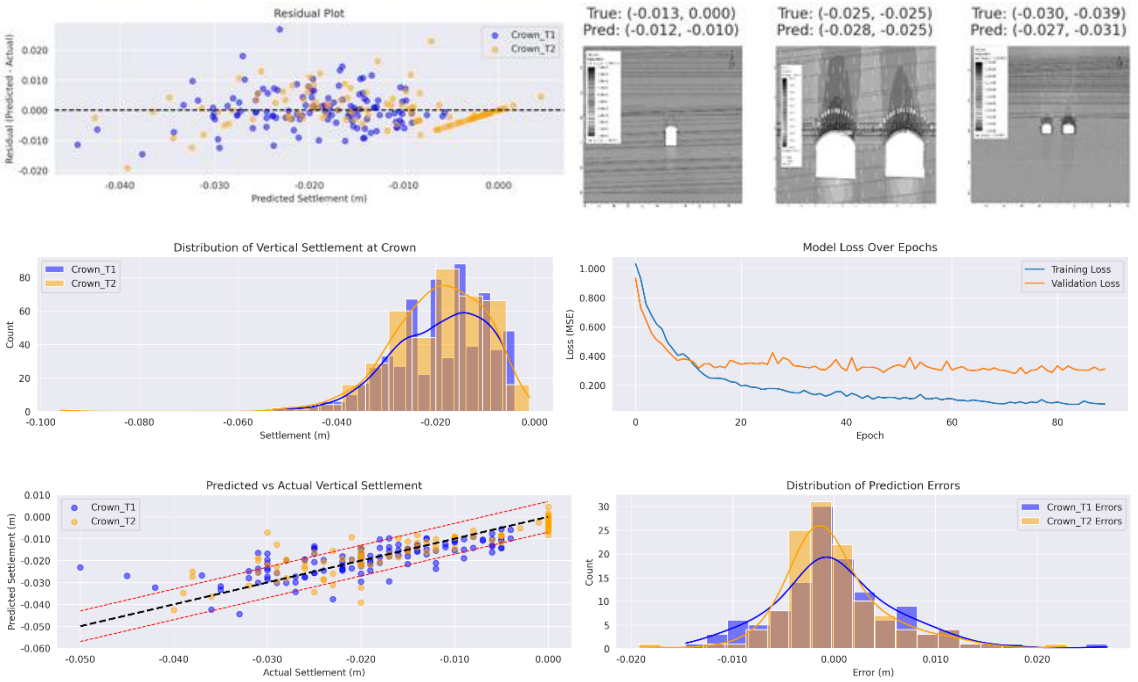

Figure 6. Model Evaluation for Vertical Settlement Interpretation Including Data Distribution, Training Curves, Error Metrics, and Sample Outputs

Sensitivity Analysis: The model's performance is influenced by data quality, class imbalance, and the diversity of geometrical scenarios. Figure 5 – " The Distribution of Number of Tunnels " highlights the significant imbalance, particularly underrepresentation of 3- and 4-tunnel cases, which correlates with the lower precision and recall observed for those classes. The Confusion Matrix reveals notable misclassifications between adjacent classes — for example, several 1-tunnel images misclassified as 2, indicating that subtle differences in geometry or colour contours may be similar without sufficient training examples.

For vertical settlement, Figure 6 – "Distribution of Vertical Settlement at Crown" shows a strong concentration around –0.025 m, but limited representation at extreme values (e.g., beyond –0.05 m). This sparsity may reduce model accuracy in regions where larger deformations occur, which are often critical for tunnel safety assessment. Such limitations suggest that current model interpretations may be less reliable under edge-case conditions.

Additionally, the model's performance may be sensitive to software-specific image styles, as the dataset is dominated by RS2 outputs, while contributions from FLAC3D, UDEC, and 3DEC remain minimal. This could introduce a form of model bias toward dominant styling or formatting, reducing generalisation to broader software ecosystems unless addressed with cross-platform augmentation.

Expanding the dataset to 5,000+ images and including greater geometric variability, different software sources, and more balanced class representation could significantly improve accuracy across both classification and regression tasks.

Comparison with Traditional Methods: Manual interpretation typically requires 5–10 minutes per image, resulting in 125–250 hours for 2,000 images, with subjective bias contributing to error rates of up to 15–20%. The CNN processes the same dataset in under one minute, producing repeatable, unbiased interpretations. Compared to traditional ML methods like Support Vector Machine (SVM), the CNN delivers a 10–15% gain in accuracy, attributed to its ability to learn complex spatial features directly from raw images.

Discussion: These results demonstrate that even with a moderately sized dataset and inherent class imbalance, a dual-task CNN can deliver reliable, automated interpretation for tunnel analysis. The model's strength lies in its ability to interpret numerical outputs rapidly and consistently, positioning it as a promising tool for design checks, model review, and quality assurance workflows.

However, limitations remain. The underperformance in 3- and 4-tunnel classification and wider interpretation error at settlement extremes indicate that data-driven blind spots exist. Performance is currently strongest in high-frequency classes, with reduced precision where data is limited or more complex. These gaps must be addressed before wider deployment.

Future Potential: Improvements could be achieved by applying synthetic oversampling or targeted data augmentation to address underrepresented classes, particularly 3- and 4-tunnel cases. Incorporating additional edge-case deformation patterns, such as soft ground behaviour or large displacement scenarios, may also improve regression performance in critical ranges. Expanding the dataset to include more complex 3D tunnel configurations, such as junctions, bifurcations, or crossover sections, would allow the model to generalise across a wider range of real-world geometries. Exploring ensemble CNN architectures could enhance interpretation stability, while the inclusion of additional outputs such as stress, strain, or plastic zone boundaries would support more comprehensive numerical output interpretation. Collectively, these improvements could raise model accuracy to 95% or higher, positioning ML-driven interpretation as a scalable, reliable tool for future tunnelling workflows.

5 CONCLUSION

This study presents a dual-task CNN for interpreting tunnel engineering numerical modelling outputs, achieving 90% accuracy in tunnel count classification (1 to 4) and interpreting vertical settlements with 85% of interpretations within 5 mm of the actual values. Trained on a dataset of 2,500 images, the model automates a time-consuming manual process, reducing processing time from minutes to seconds per image while providing consistent and repeatable outputs.

The model offers two key contributions. First, it streamlines tunnel count classification, supporting faster inventory checks and design validation. Second, it accurately derives vertical displacement values from colour contour plots, improving interpretation without manual input. The CNN's ability to extract and learn from spatial features enables detection of tunnel outlines, deformation gradients, and other key indicators, making it a practical tool for both early-stage modelling and ongoing project workflows.

Limitations include reduced performance on underrepresented classes (3- and 4-tunnel scenarios), some interpretation variability at displacement extremes, and reliance on a synthetic dataset

dominated by a single modelling platform (RS2). While generalisation was strong, model confidence was notably higher on dominant classes, and settlement interpretations, though unbiased, occasionally exceeded desired tolerances in edge cases.

Future work will focus on expanding the dataset with more diverse tunnel geometries, edge conditions, and software outputs. Additional outputs such as stress, strain, and plastic zones will be integrated, and ensemble methods will be explored to improve robustness across varying input conditions. These steps are expected to enhance interpretation accuracy and generalisation, positioning the model for broader application in digital tunnel engineering.

6 REFERENCES

Ahmed, A. et al. 2003. Neuronet prediction of tunnelling-induced settlements. Proc. 10th Int. Colloquium on Structural and Geotechnical Engineering, Ain Shams University, Cairo, Egypt, 22–24 April.

Aqel, M. & Marji, M. 2021. Convolutional neural networks for visual recognition. In: Proc. Int. Conf. on Data Science, E-learning and Information Systems 2020, 229–238. Cham: Springer.

Chollet, F. 2021. Deep learning with Python, 2nd ed. New York: Manning Publications.

Couture, H. 2023. How to design a roadmap for a machine learning project. Towards Data Science.

Géron, A. 2022. Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow, 3rd ed. Sebastopol: O'Reilly Media.

Goodfellow, I., Bengio, Y. & Courville, A. 2016. Deep learning. Cambridge: MIT Press.

Guo, Y., Liu, Y., Oerlemans, A., Lao, S., Wu, S. & Lew, M.S. 2018. Deep learning for visual understanding: A review of recent advances in convolutional neural networks. Pattern Reconition 77: 701–721. https://doi.org/10.1016/j.patcog.2017.10.013

Li, G., Xie, Y. & Lin, L. 2024. A review of convolutional neural networks in computer vision. Artificial Intelligence Review. https://doi.org/10.1007/s10462-024-10721-6

Liu, X., Chen, Y. & Wang, Z. 2024. A review of deep learning applications in tunneling and underground engineering. Applied Sciences.

Mirsepassi, M.A. 1997. Application of artificial neural networks to water treatment plant operation. PhD thesis, University of Wollongong.

Mohamadnejad, M., Diederichs, M. & Carvalho, R. 2021. A convolutional neural network for predicting tunnel support and liner performance: A case study. Rock Mechanics and Rock Engineering.

Nye, T. 2023. Reinforcement learning: Tunnel design/construction – is it possible? Paper presented at the 18th Australasian Tunnelling Conference, Auckland.

Ramzan, A., Ehsan, N. & Alvi, R. 2023. Application of artificial neural network for construction and demolition waste quantification: A parametric study. Heliyon 9(7): e16148.

Zhang, Y., Wang, J. & Li, L. 2024. Deep learning–based prediction of tunnel face stability in layered soils using images of random fields. Journal of Geotechnical and Geoenvironmental Engineering.