ACCSCIENCE PUBLISHING

International Journal of AI for Materials and Design

ORIGINAL RESEARCH ARTICLE

Computer vision and deep learning-based prediction for inkjet-printed electrodes

Supplementary Files

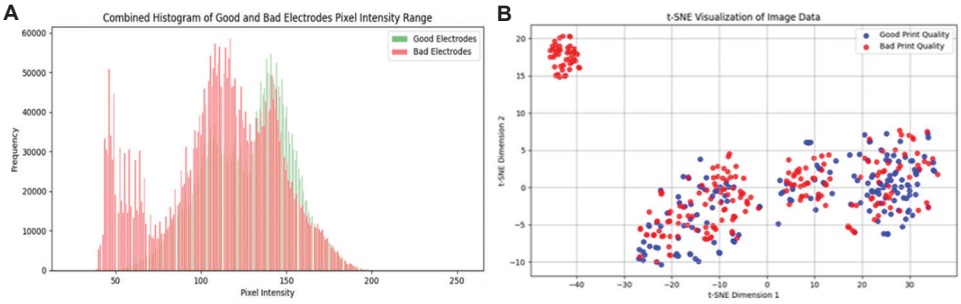

Figure S1. Results from exploration of dataset. (A) Histogram of pixel values; (B) t-SNE plot.

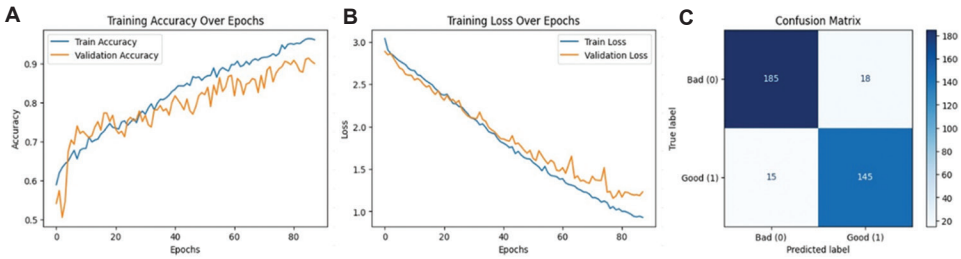

Figure S2. Performance evaluation of the optimized CNN model. (A) Training and validation accuracy trends across epochs; (B) Training and validation loss progression indicating model convergence; (C) Confusion matrix demonstrating the CNN's classification accuracy for good- and bad-print-quality electrodes.
Abbreviation: CNN: Convolutional Neural Network.

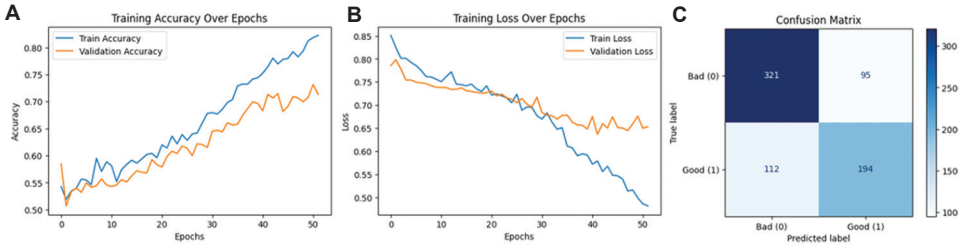

Figure S3. Performance evaluation of the optimized FNN model. (A) Training and validation accuracy trends across epochs; (B) Training and validation loss trends across epochs, indicating model convergence; (C) Confusion matrix demonstrating the CNN's classification accuracy for good- and bad-print-quality electrodes.
Abbreviations: CNN: Convolutional Neural Network; FNN: Feedforward Neural Network.

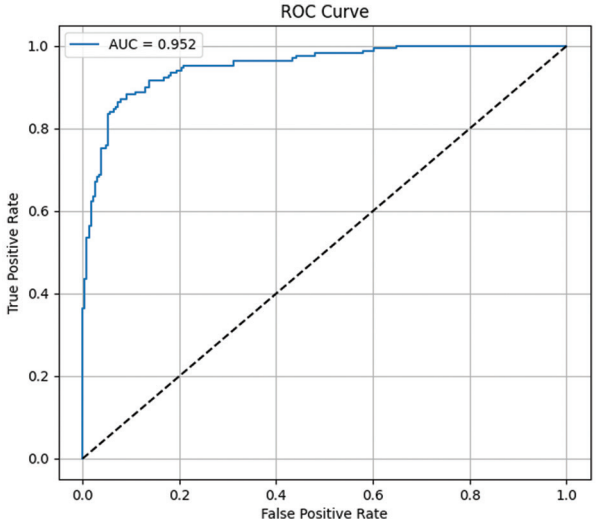

Figure S4. Receiver operating characteristic curve illustrating the performance of the optimized Convolutional Neural Network model on the original dataset of 401 images

The receiver operating characteristic (ROC) curve shows how well a classifier can separate classes by plotting the true positive rate (sensitivity) against the false-positive rate (1 − specificity) at different decision thresholds. The area under the curve (AUC) provides a single measure of the model's overall performance, where a value of 1 indicates perfect classification and 0.5 corresponds to random guessing.

The ROC and AUC results were generated using an evaluation script that loaded the trained neural networks, processed the test images, and computed prediction probabilities. The ROC curve illustrates the model's performance across different decision thresholds, and the thresholds were not set manually. The roc_curve() function from Scikit-Learn was used to compute the ROC curve, generating FPR, TPR, and thresholds from the predicted probabilities. The auc() function was then applied to calculate the AUC score. For the dataset of 401 images, this resulted in 73 thresholds for the CNN and 150 thresholds for the FNN, corresponding to the number of unique probability values produced by each model. The AUC score was calculated from these ROC points to provide a quantitative measure of each model's ability to discriminate between good- and bad-print-quality electrodes.

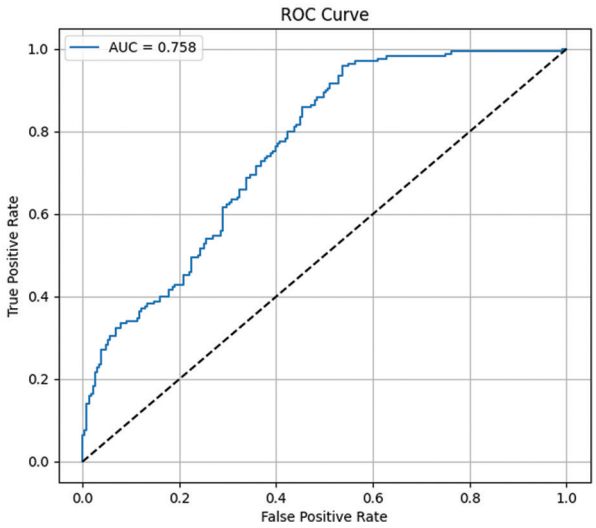

Figure S5. Receiver operating characteristic curve illustrating the performance of the optimized Feedforward Neural Network model on the original dataset of 401 images

The AUC for optimized CNN is 0.952, which indicates excellent discrimination performance compared to the diagonal line representing a random classifier (AUC = 0.5).

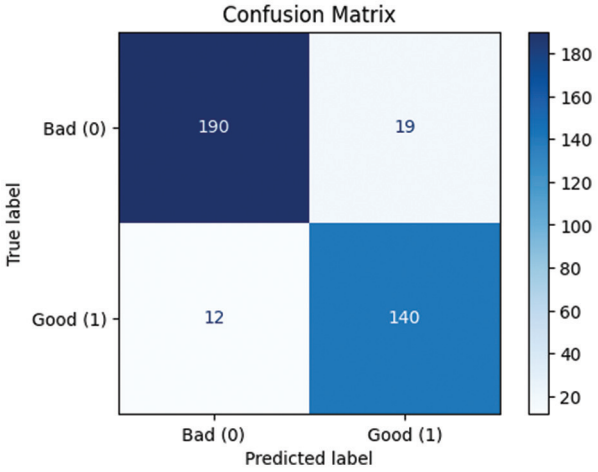

Figure S6. Visual representation of the Neural Architecture Search model's testing performance, shown through a confusion matrix.

The AUC for optimized FNN is 0.758, which indicates acceptable discrimination performance compared to the diagonal line representing a random classifier (AUC = 0.5).

Table S1. Electrode sample data

Sample	No. of layers	Frequency (Hz)	Temperature (°C)	No. of electrodes in sample	No. of defective electrodes
1	10	40	30	15	8
2	10	60	30	15	8
3	10	80	30	15	8
4	15	80	30	15	8
5	15	60	30	15	2
6	15	40	30	15	8
7	30	80	30	14	14
8	30	60	30	15	15
9	30	40	30	15	11
10	10	80	35	15	7
11	10	60	35	15	5
12	10	40	35	15	11
13	15	80	35	16	1
14	15	60	35	15	5
15	15	40	35	15	6
16	30	80	35	15	5
17	30	60	35	15	8
18	30	40	35	15	9
19	10	80	28	15	2
20	10	60	28	15	10
21	10	40	28	15	2
22	15	80	28	15	15

(Cont'd...)

Table S1. ***(Continued)***

Sample	No. of layers	Frequency (Hz)	Temperature (°C)	No. of electrodes in sample	No. of defective electrodes
23	15	60	28	15	9
24	15	40	28	15	15
25	30	80	28	15	15
26	30	60	28	11	9
27	30	40	28	15	15

Table S2. Basic CNN structure

Layer type	Output shape
Input layer	(128,128,1)
Convolutional layer (Kernel size: 3×3; activation: ReLU)	(126,126,32)
Max pooling (2×2)	(63,63,32)
Convolutional layer (Kernel size: 3×3; activation: ReLU)	(61,61,64)
Max pooling (2×2)	(30,30,64)
Convolutional Layer (Kernel size: 3×3; activation: ReLU)	(28,28,128)
Max pooling (2×2)	(14,14,128)
Flatten	(25088)
Fully connected layer (Activation: ReLU)	(128)
Dropout (30%)	(128)
Fully connected layer (Activation: ReLU)	(64)
Dropout (30%)	(64)
Output layer	(1)

Abbreviations: CNN: Convolutional Neural Network; ReLU: Rectified Linear Unit.

Table S3. Basic CNN performance metrics

Dataset	Accuracy	Precision	Recall	F1 Score
Training	0.685714	0.583893	0.769912	0.664122
Testing	0.672131	0.615385	0.827586	0.705882
% Difference	1.98	5.39	7.49	6.29

Abbreviation: CNN: Convolutional Neural Network.

Table S4. Deeper CNN structure

Layer type	Output shape
Input layer	(128,128,3)
Convolutional layer (Kernel size: 3×3; activation: ReLU)	(126,126,32)
Batch normalization	(126,126,32)
Max pooling (2×2)	(63,63,32)
Convolutional layer (Kernel size: 3×3; activation: ReLU)	(61,61,64)
Batch normalization	(61,61,64)
Max Pooling (2×2)	(30,30,64)
Convolutional Layer (Kernel size: 3×3; activation: ReLU)	(28,28,128)
Batch normalization	(28,28,128)
Max Pooling (2×2)	(14,14,128)

(Cont'd...)

Table S4. ***(Continued)***

Layer type	Output shape
Convolutional layer (Kernel size: 3×3; activation: ReLU)	(12,12,256)
Batch normalization	(12,12,256)
Max pooling (2×2)	(6,6,256)
Convolutional layer (Kernel size: 3×3; activation: ReLU)	(4,4,512)
Batch normalization	(4,4,512)
Max pooling (2×2)	(4,4,512)
Flatten	(9216)
Fully connected layer (Activation: ReLU)	(512)
Dropout (30%)	(512)
Fully connected layer (Activation: ReLU)	(256)
Dropout (30%)	(256)
Fully connected layer (Activation: ReLU)	(128)
Dropout (30%)	(128)
Fully connected layer (Activation: ReLU)	(64)
Dropout (30%)	(64)
Output layer	(1)

Abbreviations: CNN: Convolutional Neural Network; ReLU: Rectified Linear Unit.

Table S5. Basic FNN architecture

Layer type	Output shape
Input layer	(16384)
Hidden layer (Activation: ReLU)	(128)
Dropout (20%)	(128)
Hidden layer (Activation: ReLU)	(64)
Hidden layer (Activation: ReLU)	(16)
Output layer (Activation: Sigmoid)	(1)

Abbreviations: FNN: Feedforward Neural Network; ReLU: Rectified Linear Unit.

Table S6. Basic FNN architecture with regularization

Layer type	Output shape
Input layer	(16384)
Hidden layer (Activation: ReLU; λ: 0.001)	(128)
Batch normalization	(128)
Dropout (30%)	(128)
Hidden layer (Activation: ReLU; λ: 0.001)	(64)
Batch normalization	(64)
Dropout (30%)	(64)
Hidden layer (Activation: ReLU; λ: 0.001)	(16)
Output layer (Activation: Sigmoid)	(1)

Abbreviations: FNN: Feedforward Neural Network; ReLU: Rectified Linear Unit.