

“Surface Roughness Classification using Neural Networks”

Classification of microscopic surface images into roughness categories, comparing increasing model complexity.

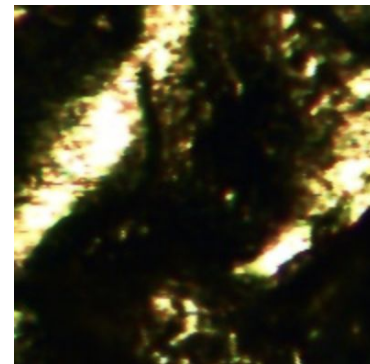
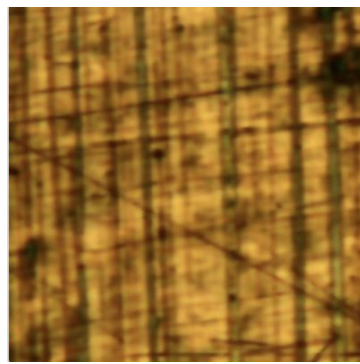
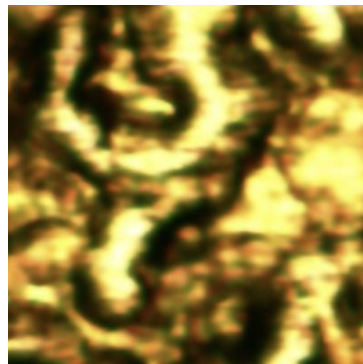
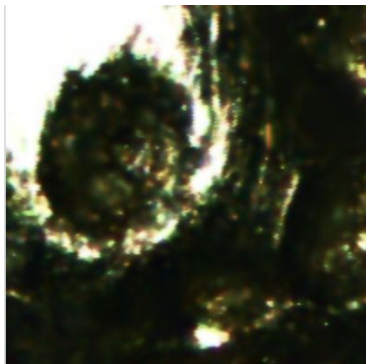
Problem & Dataset

Task: classify microscopy images into **16 roughness categories**

Dataset: Kaggle “Surface Roughness Classification”

The dataset was collected using an optical microscope with high-resolution imaging and expanded via systematic cropping of raw microscope images to obtain 3,840 balanced texture samples across 16 surface roughness classes.

Size: 3,840 .jpeg images, balanced (240 per class)



Data Preparation Pipeline

1. Label Encoding

- Class names sorted and mapped to integers 0-15

2. Train / Validation / Test Split

- Stratified sampling to preserve class distribution

3. Image Preprocessing Pipeline

1. Grayscale conversion — color discarded (texture is primary feature)
2. Resize to 64×64 — fixed resolution for batching
3. Normalize to [0, 1] — pixel values from [0, 255] to [0, 1]
4. Add channel dimension — shape: (1, 64, 64) for PyTorch

Subset	Proportion	Images	Per class
Train	70%	2688	168
Validation	15%	576	36
Test	15%	576	36

Model 1: Softmax Regression

Motivation

- Simplest discriminative model for multi-class classification
- Assumes classes are linearly separable in pixel space
- Establishes a lower bound on performance

Forward Pass

- Input: flattened image $\mathbf{x} \in \mathbb{R}^{4096}$ (64×64 pixels)
- Logits: $\mathbf{z} = \mathbf{W}^\top \mathbf{x} + \mathbf{b}$
- Parameters: $\mathbf{W} \in \mathbb{R}^{4096 \times 16}$, $\mathbf{b} \in \mathbb{R}^{16}$

Softmax Function

$$p(y = c \mid \mathbf{x}) = \frac{e^{z_c}}{\sum_{j=1}^{16} e^{z_j}}$$

Cross-Entropy Loss

$$\mathcal{L} = -\frac{1}{B} \sum_{n=1}^B \log p(y_n \mid \mathbf{x}_n)$$

Gradient (Manual Backpropagation)

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}} = \mathbf{X}^\top (\mathbf{P} - \mathbf{Y})$$

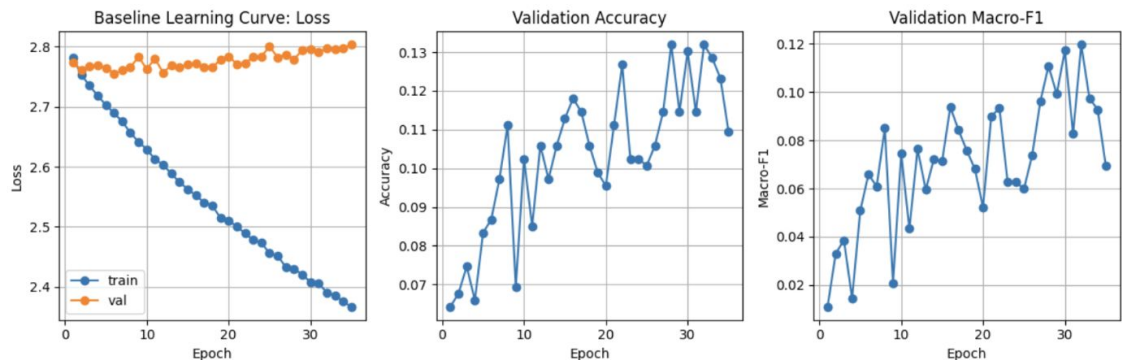
Model 1: Softmax Regression

Hyperparameter Search

- 36 configurations tested
- Grid: epochs {5, 10, 20, 35}, lr {0.01, 0.003, 0.001}, weight decay {0, 1e-4, 1e-3}

Best config: epochs=35, lr=0.01, weight_decay=1e-4

Learning Curves



Final Performance

Metric	Validation	Test
Accuracy	~0.11	0.13
Macro-F1	~0.08	0.10

Conclusion

Fast convergence but limited capacity

Linear decision boundaries are insufficient for texture classification

Model 2: MLP with Sigmoid

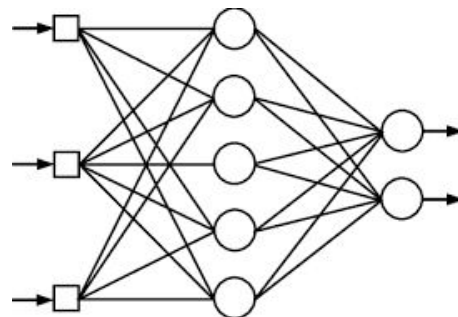
Motivation

- Adds non-linear capacity through hidden layers
- Can learn non-linear decision boundaries
- Still treats image as flat vector (no spatial structure)

Architecture

Input (4096) \rightarrow Linear(1024) \rightarrow Sigmoid \rightarrow Linear(512) \rightarrow Sigmoid \rightarrow Linear(16) \rightarrow Softmax

Sigmoid Activation $\sigma(z) = \frac{1}{1 + e^{-z}}$



Backpropagation (Manual)

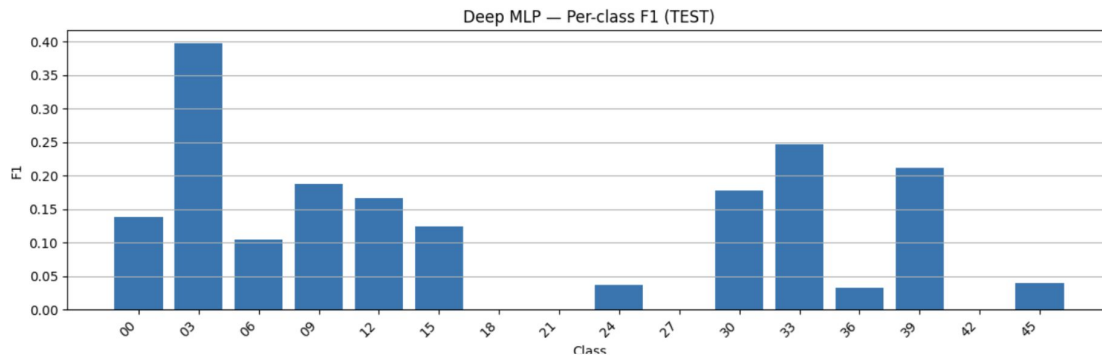
- Linear layer gradient: $\frac{\partial \mathcal{L}}{\partial \mathbf{W}} = \mathbf{X}^\top \mathbf{G}$
- Sigmoid gradient: $\frac{\partial \mathcal{L}}{\partial \mathbf{Z}} = \frac{\partial \mathcal{L}}{\partial \mathbf{S}} \odot \mathbf{S}(1 - \mathbf{S})$

Model 2: MLP with Sigmoid

Hyperparameter Search

- 96 configurations tested
- Grid: hidden layouts, dropout $\{0, 0.3, 0.5\}$, lr $\{0.001, 0.0003\}$, epochs $\{40, 60\}$

Best config: hidden=(1024, 512), dropout=0.0, lr=0.001, epochs=60



Final Performance

Metric	Validation	Test
Accuracy	0.160	0.130
Macro-F1	0.127	0.104

Why MLP Struggles

1. Flattening destroys spatial structure
2. No translation invariance — patterns position-dependent
3. Sigmoid saturation — vanishing gradients

Model 3: CNN

Motivation

CNNs exploit image-specific properties:

- Spatial locality — nearby pixels are related
- Translation invariance — detect patterns regardless of position
- Hierarchical features — edges → textures → patterns

Convolution Operation

$$z_k(i, j) = \sum_c \sum_{u, v} w_{k, c}(u, v) \cdot x_c(i + u, j + v) + b_k$$

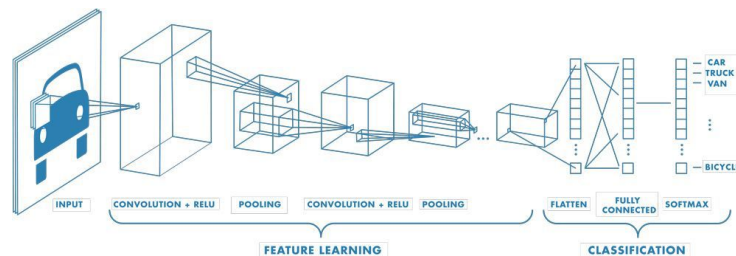
ReLU Activation (vs Sigmoid)

$$\text{ReLU}(z) = \max(0, z)$$

- Avoids vanishing gradient problem
- Allows training deeper networks

Spatial Dimension Evolution

$$64 \times 64 \xrightarrow{\text{pool}} 32 \times 32 \xrightarrow{\text{pool}} 16 \times 16 \xrightarrow{\text{pool}} 8 \times 8$$



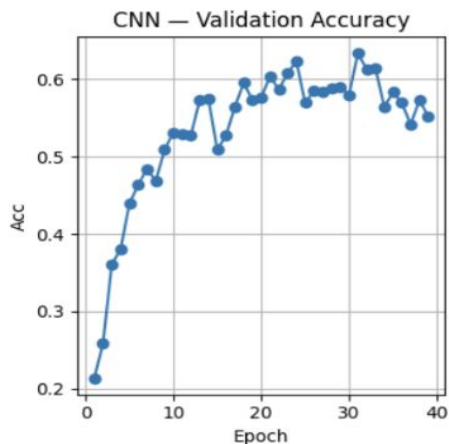
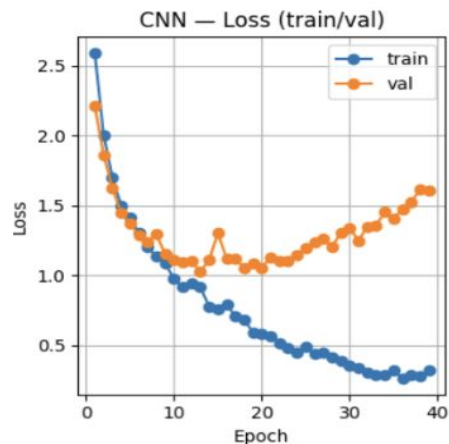
Architecture

Conv2d(1→32, 3×3) → ReLU → MaxPool(2)
Conv2d(32→64, 3×3) → ReLU → MaxPool(2)
Conv2d(64→128, 3×3) → ReLU → MaxPool(2)
Flatten → Linear(8192→256) → ReLU →
Dropout(0.5) → Linear(256→16)

Model 3: CNN

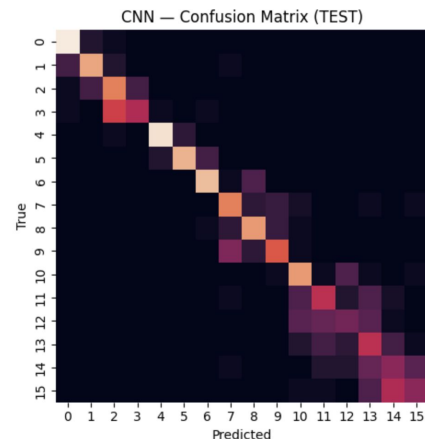
Training Setup

- Optimizer: Adam (adaptive learning rates)
- Regularization: Dropout (0.5), Weight decay ($1e-4$)
- Early stopping: patience=8 epochs
- Training stopped at epoch 39 (best: epoch 31)



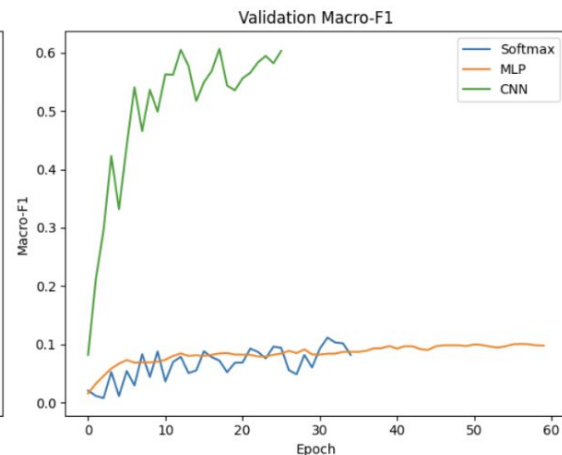
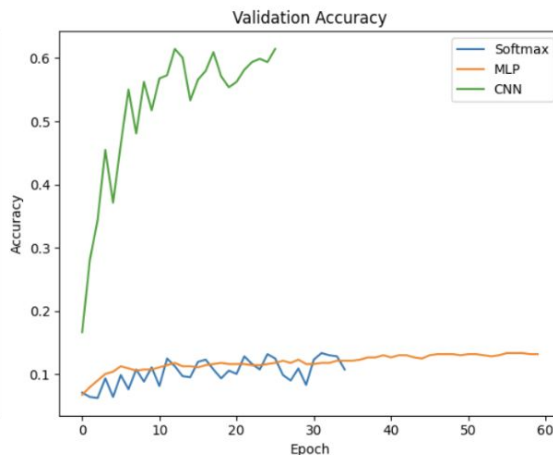
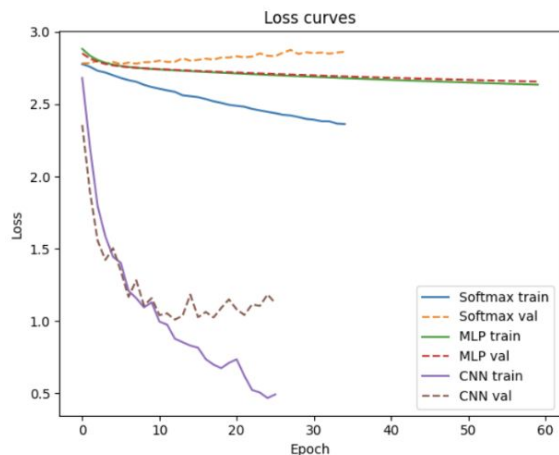
Final Performance

Metric	Validation	Test
Accuracy	0.597	0.592
Macro-F1	0.594	0.590
Top-2 Accuracy	0.837	0.825



Results: Models Comparison

Model	Test Accuracy	Test Macro-F1	Parameters	Latency
Softmax Regression	0.13	0.10	~65K	<1 ms
MLP (1024, 512)	0.16	0.12	~5M	1.2 ms
CNN (32-64-128)	0.59	0.59	~2.2M	4 ms



Conclusion

1. Linear models are insufficient for complex visual tasks requiring texture discrimination
2. Fully connected networks benefit from non-linearity but struggle without spatial inductive bias
3. Convolutional architectures provide the best balance between representation power and generalization for image data

CNN is the most appropriate model for surface roughness classification:

- $\sim 6\times$ better accuracy than linear/MLP baselines
- Exploits spatial structure through local filters
- Achieves $\sim 60\%$ accuracy on 16-class problem

Softmax & MLP serve as valuable baselines demonstrating the importance of architectural inductive biases in deep learning.