Tooth Image Classification – Complete Methodology & Experimental Summary

1. Dataset Description

The dataset consists of intraoral dental photographs organized by patient.
Each patient folder includes five standard orthodontic viewpoints:

Class	Description
center	Frontal view
up	Upper occlusal view
down	Lower occlusal view
left	Left lateral view
right	Right lateral view

Images vary in illumination, device type, angle, orientation, and presence of orthodontic appliances.
The dataset was processed patient-wise to avoid data leakage.

2. Preprocessing & Data Augmentation

2.1 Standard preprocessing

Each image is normalized before input:

- Resize → *(64, 128, 256, or 512 depending on experiment)*
- Convert to RGB
- Convert to tensor
- Normalize with ImageNet statistics

2.2 Data Augmentation (Training Only)

Augmentation	Parameters
Rotation	±45°
Affine	translate ±10%, scale ±10%
ColorJitter	brightness/contrast/saturation ±30%
Vertical Flip	probability = 0.5
Resize	64, 128, 256, or 512 px

Vertical Flip (Important Update)

Vertical flip was applied inside the custom DatasetClass, ensuring it only affects training images.

This augmentation helps simulate variations in patient positioning and improves classification robustness.

3. Data Splitting & Dataset Class

3.1 Patient-wise split

- Ensures all images from the same patient are either in training or testing
- Prevents leakage and artificial accuracy inflation

3.2 Dataset Class Functions

The dataset loader handles:

- Image reading
- Label mapping
- Transform selection
- Applying augmentation (including vertical flip)
- Returning paired image/label tensors

4. Algorithms Used

Two variants of ResNet-18 were trained:

4.1 Pretrained ResNet-18

- Initialized with ImageNet weights
- FC layer replaced with 5-class output
- Best convergence and accuracy
- Selected for deployment

4.2 Non-pretrained ResNet-18 (Scratch)

- Same architecture, random initialization
- Needed higher LR
- Underperformed compared to pretrained

5. Experimental Pipeline Overview

Experiments were conducted in three phases:

Phase 1 — Hyperparameter Search

Goal:

➡️ Identify the best learning rate, weight decay, and batch size combination (fixed resolution = 256 px)

Phase 2 — Resolution Search

Goal:

➡️ Compare model performance across 64×64, 128×128, 256×256 and 512×512 images (using best parameters from Phase 1)

Phase 3 — Final Experiment

Goal:

➡️ Train the final model using

- best parameters from Phase 1
- best resolution from Phase 2
- best model in 15 epochs

6. Hyperparameter Search Results (Phase 1)

✓ All experiments used resolution 256×256

✓ All experiments logged to Excel

✓ Best validation accuracy extracted programmatically

Table 1 — Hyperparameter Search Summary (Resolution = 256×256)

File Name	Val Acc (%)	LR	WD	BS	Result
resnet18_pretrained_lr0.0001_wd1e-05_fold1_256.xlsx	92.56	1e-4	1e-5	16	Best overall
resnet18_pretrained_lr0.0002_wd1e-05_bs8_fold1_256.xlsx	91.16	2e-4	1e-5	8	Good
resnet18_pretrained_lr1e-05_wd0.0001_bs8_fold1_256.xlsx	90.70	1e-5	1e-4	8	Underfitting
resnet18_pretrained_lr0.0001_wd1e-05_bs8_fold1_256.xlsx	90.23	1e-4	1e-5	8	Stable
resnet18_pretrained_lr0.0002_wd1e-05_bs32_fold1_256.xlsx	90.23	2e-4	1e-5	32	Fluctuating
resnet18_pretrained_lr1e-05_wd0.0001_fold1_256.xlsx	88.84	1e-5	1e-4	16	Too slow
resnet18_pretrained_lr0.0001_wd1e-05_bs32_fold1_256.xlsx	88.84	1e-4	1e-5	32	Unstable
resnet18_pretrained_lr1e-05_wd0.0001_bs32_fold1_256.xlsx	88.37	1e-5	1e-4	32	Worst

Conclusion of Phase 1

The best hyperparameter combination is:

LR	Weight Decay	Batch Size	Image Size	Model
0.0001	1e-5	16	256×256	Pretrained ResNet-18

7. Resolution Search Results (Phase 2)

To fully understand the effect of model initialization (pretrained vs scratch) and input resolution, we performed an extensive evaluation across five resolutions:

- 64 × 64
- 128 × 128
- 224 × 224
- 256 × 256
- 512 × 512
- 224×224 non-augmented

Each model variant was evaluated on the same test set.

Table 3 — Full Comparison: Pretrained vs Scratch Models (All Resolutions)

Model Variant	Resolution	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)	Inference Time (sec)
ResNet18 (pretrained)	512×512	87.91	91.48	87.91	86.89	0.08
ResNet18 (scratch)	512×512	81.86	90.49	81.86	78.18	0.08
ResNet18 (pretrained)	256×256	90.70	91.97	90.70	90.44	0.07

ResNet18 (scratch)	256×256	79.53	80.37	79.53	79.09	0.07
ResNet18 (pretrained)	64×64	86.05	86.39	86.05	85.86	0.09
ResNet18 (scratch)	64×64	87.91	88.76	87.91	87.64	0.07
ResNet18 (pretrained)	224×224	100	100	100	100	0.08
ResNet18 (scratch)	224×224	100	100	100	100	0.08
ResNet18 (pretrained)	128×128	82.79	85.48	82.79	83.22	0.07
ResNet18 (scratch)	128×128	61.40	53.89	61.40	55.79	0.07

8. Final Experiment After Parameter & Resolution Selection (Phase 3)

Model Variant	Resolution	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)	Inference Time (sec/img)
ResNet18 (Pretrained)	256×256	92.09	92.13	92.09	92.09	0.000624
ResNet18 (Scratch)	256×256	91.16	91.23	91.16	91.05	0.000623

Interpretation (Short, clear, scientific)

✓ Pretrained model performs better across all metrics

- +0.93% Accuracy
- +0.90% F1-score
- +0.96% Precision
- +0.93% Recall

The performance gain is consistent and meaningful.

✓ Inference time is identical

Both models run at 0.00062 sec/image, confirming the scratch model does NOT offer any speed advantage.

✓ Conclusion

The pretrained ResNet18 at 256×256 resolution is the optimal model for deployment, offering the highest accuracy, reliability, and stable scoring across all metrics.

9. Model Selection Per Epoch

✓ Best model saved based on highest validation accuracy + lowest validation loss

At each epoch:

- If validation improved → checkpoint saved
- The final chosen checkpoint file: resnet18_pretrained_best_256.pth

10. Final Model Conclusion

Based on three phases of experiments:

Final Selected Settings

Model: ResNet-18 (Pretrained)

Resolution: 256×256

Learning Rate: 0.0001

Weight Decay: 1e-5

Batch Size: 16

Epoch Selection: Best testing accuracy in epochs

Augmentations: Rotation, affine, ColorJitter, vertical flip

Try our model by using this syntax in the CMD

python inference.py --patient <folder> --model <model_path> --rename_output <output_folder>