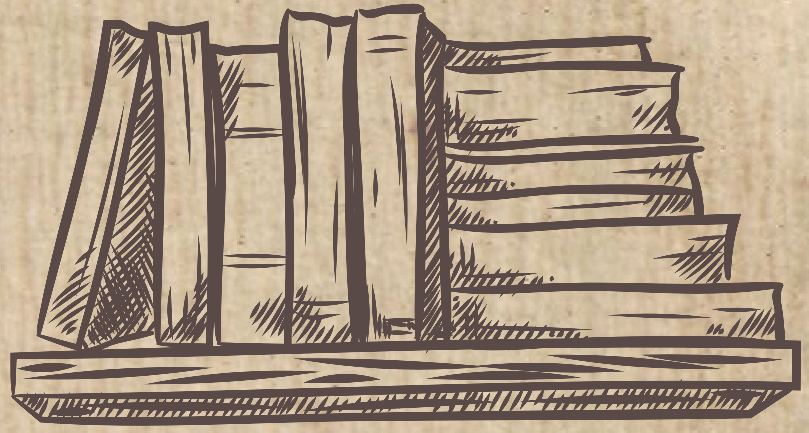
LUISS

The librarian from Alexandria

Gabriele De Ieso, Denise Di Franza, Alessia Tonicello

Problem and Context



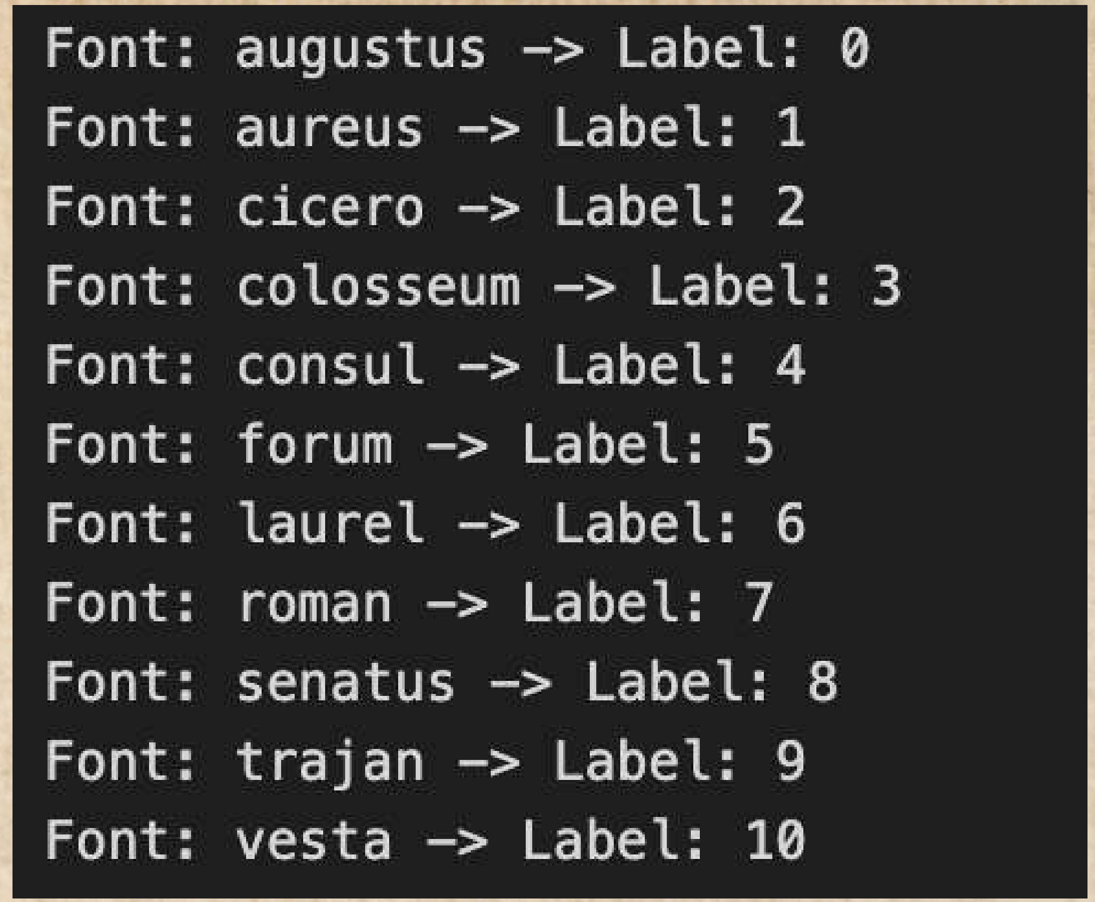
Objective: classify ancient digitized texts based on their fonts

Creation of a Neural Network model to categorize these different wrtiting styles

Dataset

The dataset contains 1,256 scanned historical texts, that originate from various years and contain different writing styles and fonts, such as 'Cicero' or 'Vesta'.

Each font was mapped into labels, though a dictionary, resulting in the image on the right.



```
Font: augustus -> Label: 0
Font: aureus -> Label: 1
Font: cicero -> Label: 2
Font: colosseum -> Label: 3
Font: consul -> Label: 4
Font: forum -> Label: 5
Font: laurel -> Label: 6
Font: roman -> Label: 7
Font: senatus -> Label: 8
Font: trajan -> Label: 9
Font: vesta -> Label: 10
```

The steps of our work:

1. Phase 0: **Imports & Setup**
2. Phase 1: **Exploratory Data Analysis (EDA)**
3. Phases 1.1 & 1.2: **Pre-processing & Data Augmentation**
4. Phase 1.3 & 1.4 : **Defining Dataset Classes and Splitting Training and Test**
5. Phase 1.5: **Creating DataLoaders**
6. Phase 2: **Model Development**
7. Phase 3: **Evaluation & Insights**

Phase 0:
Imports & Setup

Import & Setup

Imported essential libraries for:

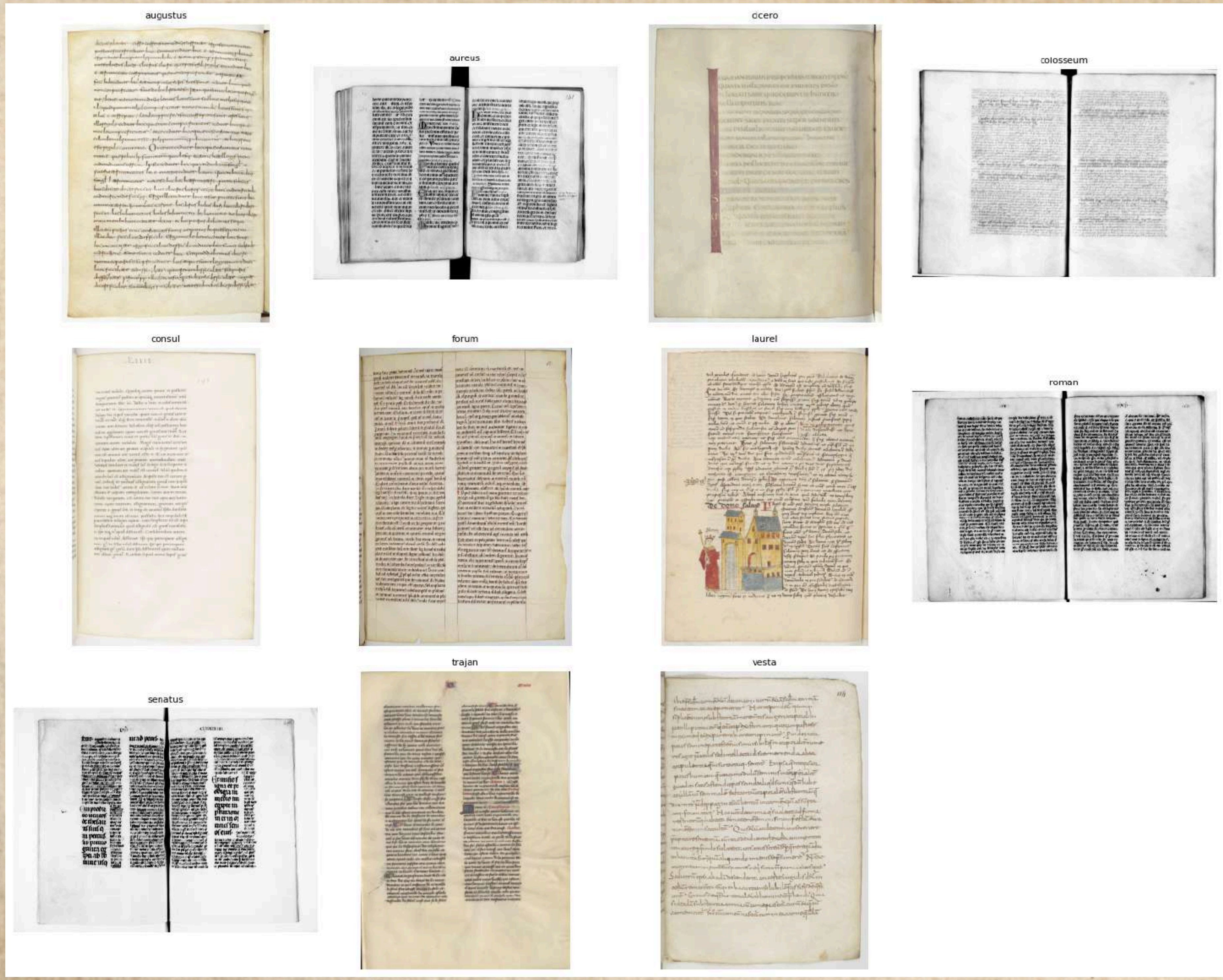
- Deep learning: torch, torchvision
- Data handling: pandas, numpy
- Visualization: matplotlib, seaborn

Configured global settings:

- Set device to CUDA if available, otherwise CPU
- Initialized random seed for reproducibility
- Defined file paths for dataset and model checkpoints

Phase 1:
Exploratory Data Analysis
(E.D.A)

Explanatory Data Analysis

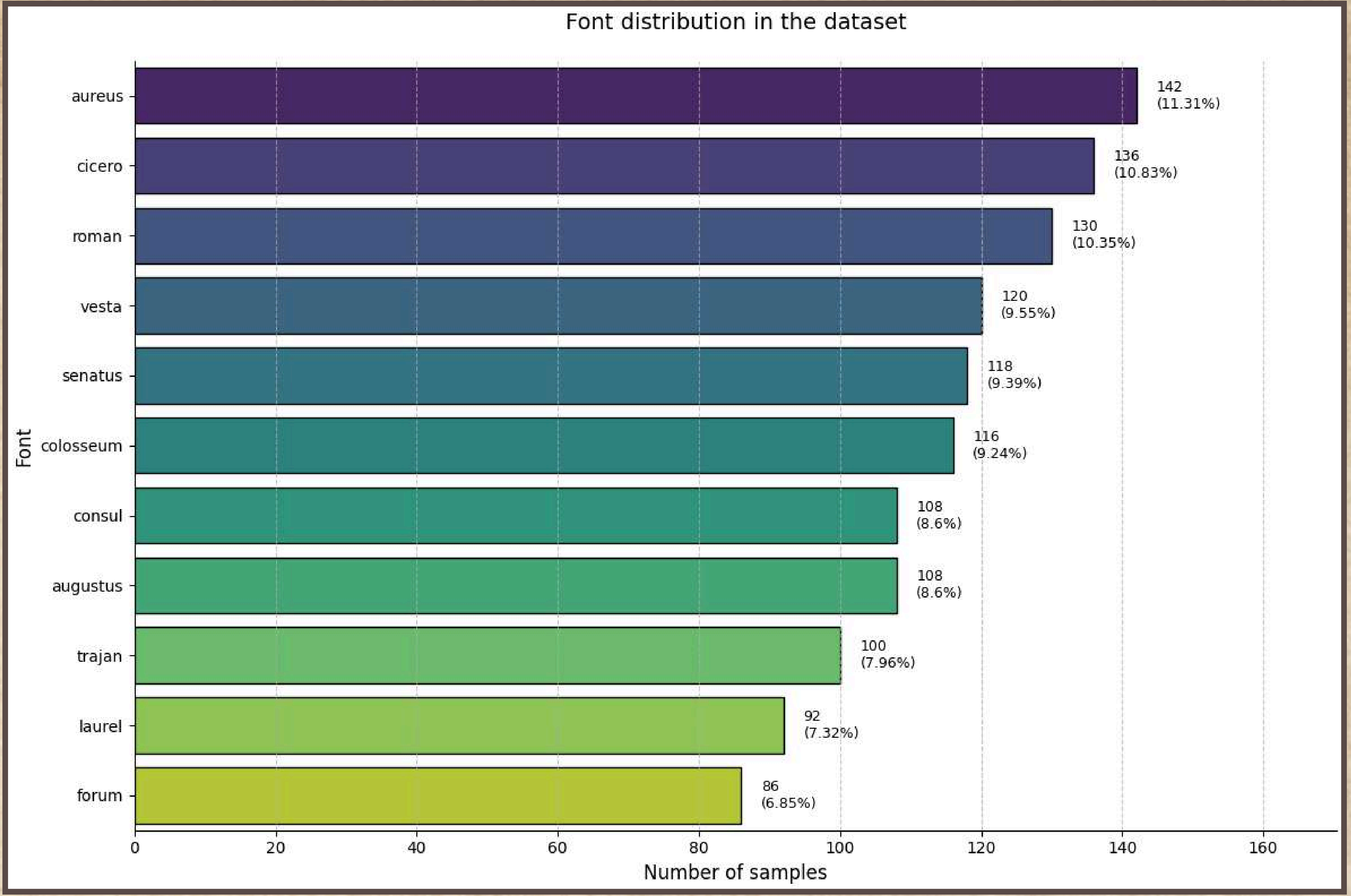


Thanks to our initial visual exploration, we observed that:

- Some pages are in black and white, others in color
- Pages vary in layout: some are single, others double-page spreads
- Several pages include illustrations or drawings
- Many images are affected by noise or visual artifacts
- Text is not always centered within the page
- The scanning quality is inconsistent across the dataset

Distribution of each font:

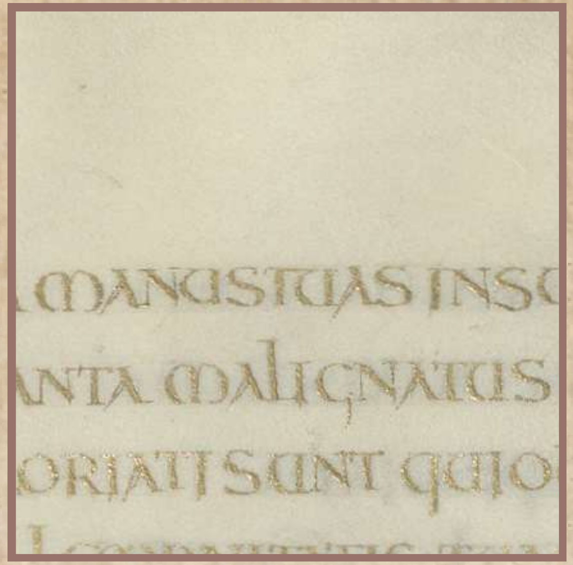
Font Name	# of Samples	Percentage
Aureus	142	11.31%
Cicero	136	10.83%
Roman	130	10.35%
Vesta	120	9.55%
Senatus	118	9.39%
Colosseum	116	9.24%
Consul	108	8.60%
Augustus	108	8.60%
Trajan	100	7.96%
Laurel	92	7.32%
Forum	86	6.85%



Phases 1.1 & 1.2: Pre-processing & Data Augmentation

Preprocessing

- **Grayscale Conversion**: convert images to grayscale to reduce complexity, color isn't needed for font recognition.
- **Contrast Enhancement (CLAHE)**: improve local contrast, making text more visible even in uneven lighting conditions.
- **Double-Page Splitting**: If an image is too wide (double-page spread), we split it into two pages based on aspect ratio.
- **Text Patch Extraction**: Using adaptive thresholding and a sliding window, we extract 255×255 patches rich in text. If no valid patches are found, we fall back to the center of the image.



Data Augmentation (training set only)

- **Random Horizontal Flip**: simulates mirrored layouts to improve generalization.
- **Random Affine Transformations: Includes small rotations, shifts, and scaling to mimic real-world misalignments.**
- **Random Perspective Distortion:** applies slight angular distortions to simulate different viewing angles.

After preprocessing and augmentation, images are converted into tensors and normalized for training. The validation set is kept clean, with only essential transforms to maintain consistency.

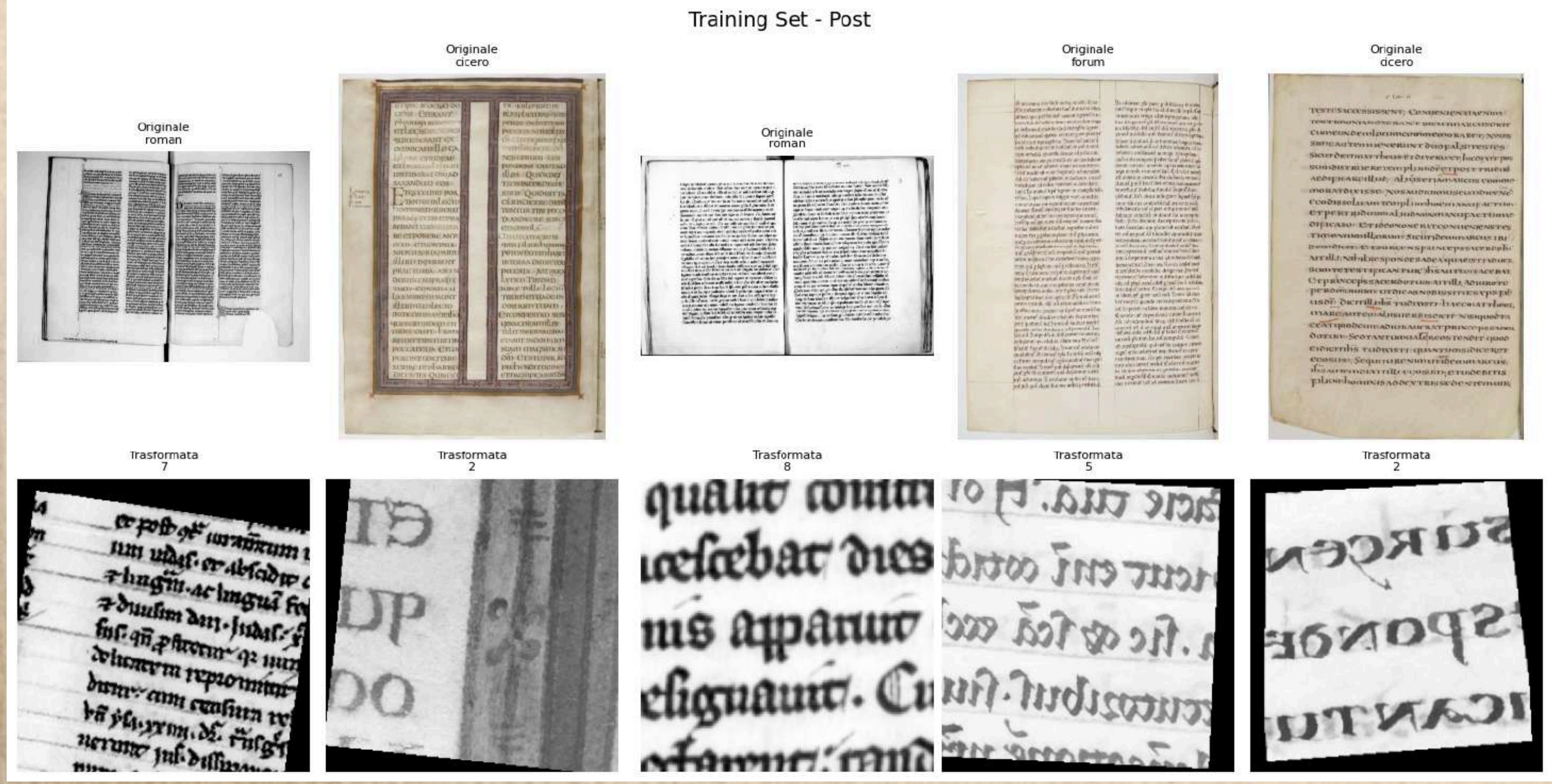
Phases 1.3 & 1.4 & 1.5 :

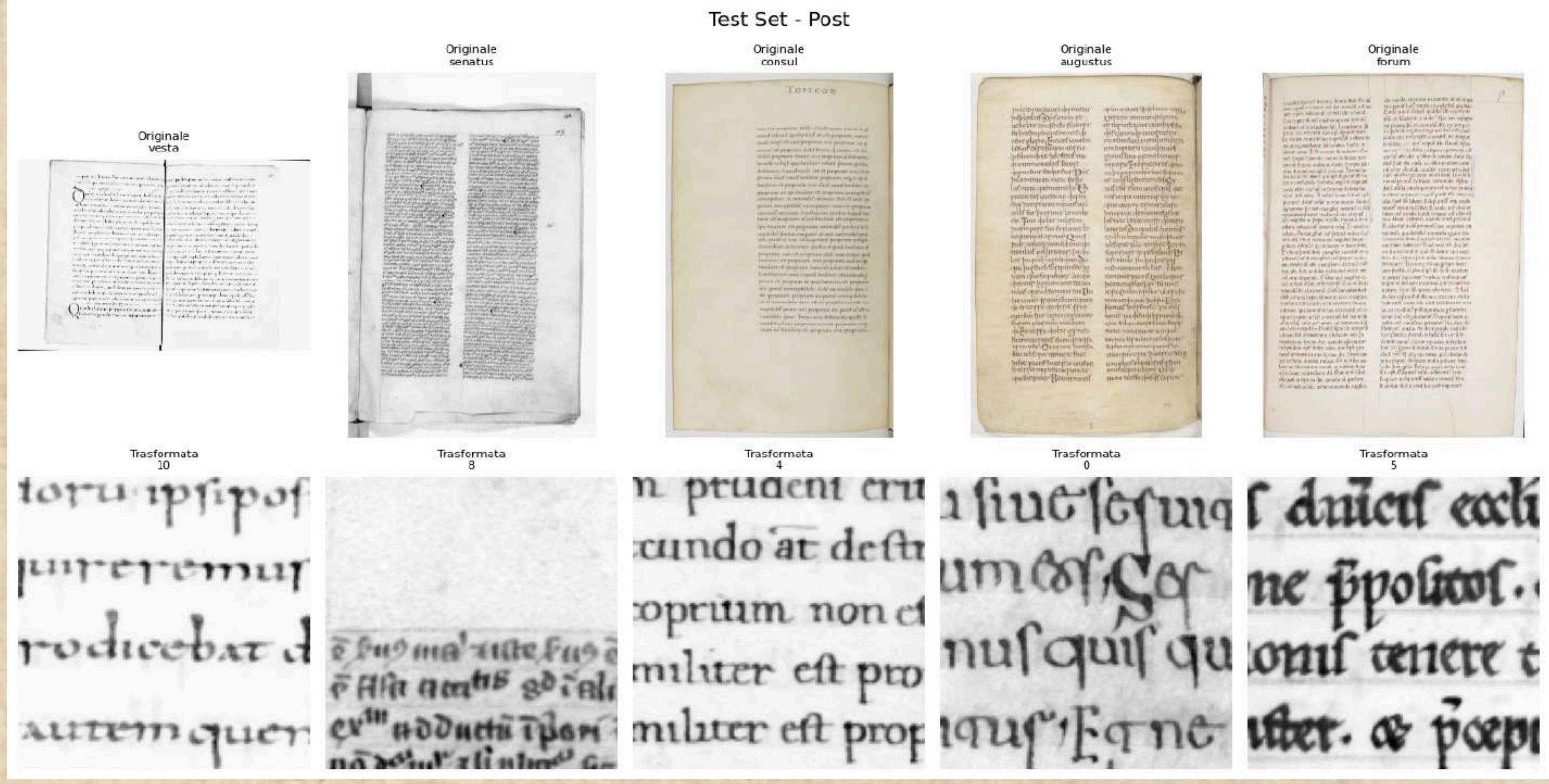
- Defining Dataset Classes
- Splitting Training and Test
- Creating the data Loaders

Dataset Splitting

Instead of randomly dividing the dataset, we used **stratified splitting**. → keeps font distribution balanced across train/test sets

To help the model generalize, we applied augmentations to the training set, but kept the test set clean, even if it still needs basic preprocessing like resizing and normalization.





Font	N° Observations Training Set (Test Set)	% x font	% x Split
cicero	109 (27)	80.1 (19.9)	10.9 (10.7)
vesta	96 (24)	80.0 (20.0)	9.6 (9.5)
senatus	94 (24)	79.7 (20.3)	9.4 (9.5)
trajan	80 (20)	80.0 (20.0)	8.0 (7.9)
colosseum	93 (23)	80.2 (19.8)	9.3 (9.1)
forum	69 (17)	80.2 (19.8)	6.9 (6.7)
aureus	113 (29)	79.6 (20.4)	11.3 (11.5)
consul	86 (22)	79.6 (20.4)	8.6 (8.7)
roman	104 (26)	80.0 (20.0)	10.4 (10.3)
augustus	86 (22)	79.6 (20.4)	8.6 (8.7)
laurel	74 (18)	80.4 (19.6)	7.4 (7.1)

Phases 2 & 3: Model Development and Evaluation

Architecture of the models

We experimented with several CNN architectures to classify historical fonts from scanned pages:

- EnhancedFontCNN: a deep CNN with four convolutional blocks, batch normalization, ReLU activations, and adaptive average pooling. The classifier includes dropout regularization and two dense layers.

The various attempts have seen the application of different initial learning rates, number of epoch and different scheduler inputs, however the results have not convinced us, leading us to search for other solutions

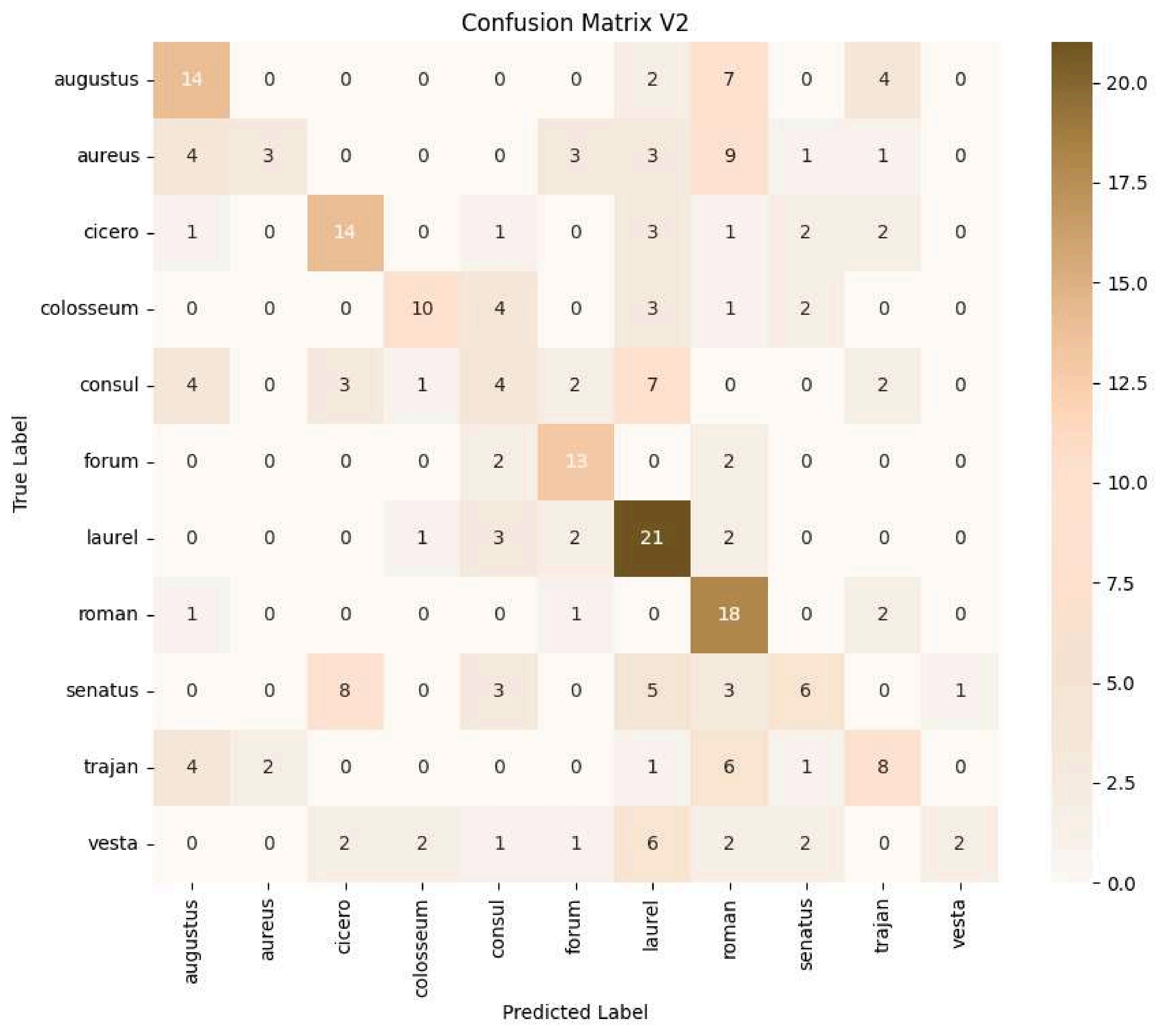
Plus: We also experimented with some pretrained models, such as ResNet and MobileNetV2, to explore transfer learning approaches.

Optimization

To improve generalization and performance, we also implemented:

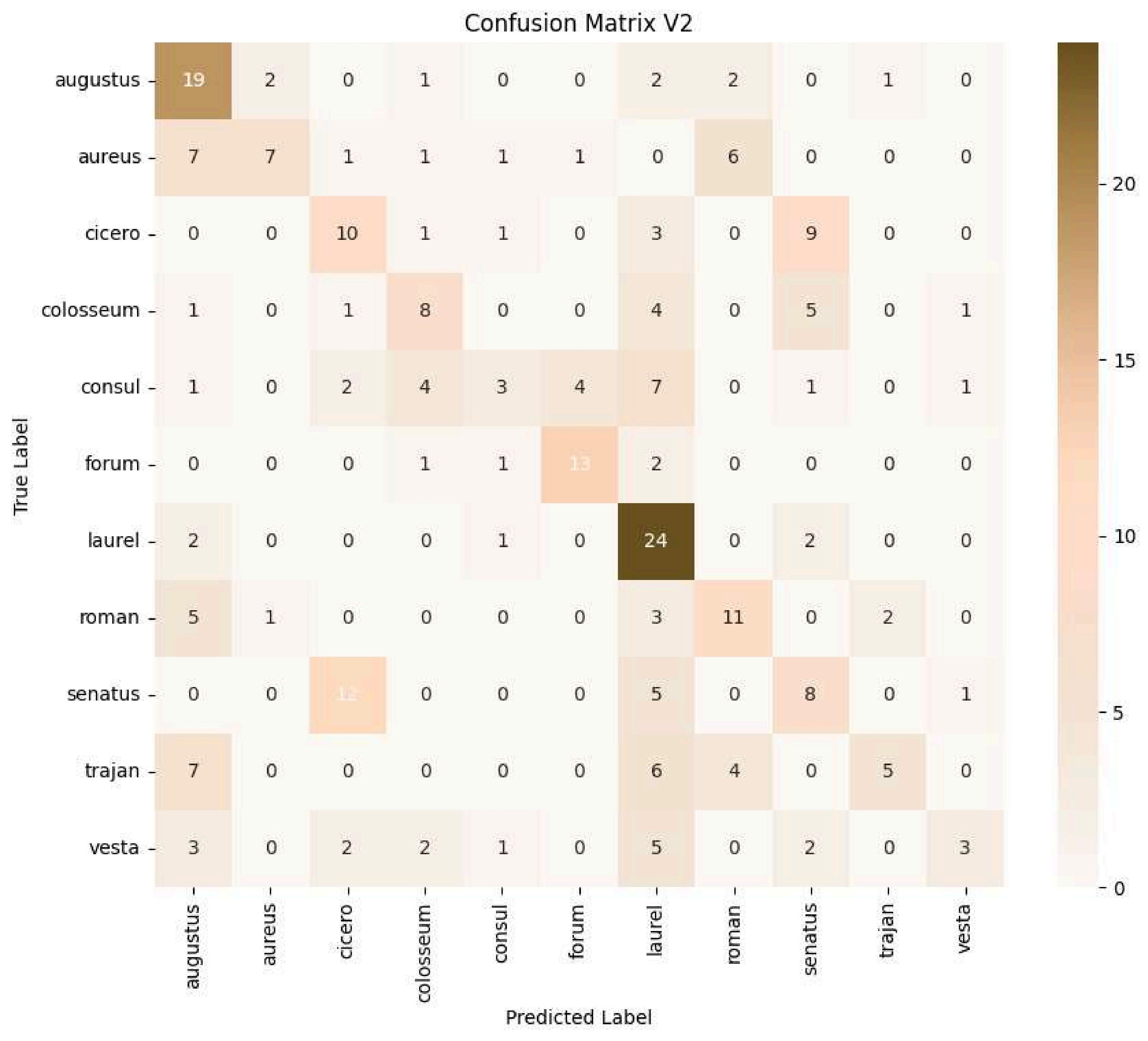
- Class balancing using weighted cross-entropy loss, based on inverse class frequencies
- Learning rate schedulers (CosineAnnealing, ReduceLROnPlateau)
- Early stopping to prevent overfitting
- Mixed precision training with autocast and GradScaler to speed up training and reduce memory usage

EnhancedFontCNN (Version 1 vs Version 2)



Confusion Matrix V2

	augustus	aureus	cicero	colosseum	consul	forum	laurel	roman	senatus	trajan	vesta
augustus	14	0	0	0	0	0	2	7	0	4	0
aureus	4	3	0	0	0	3	3	9	1	1	0
cicero	1	0	14	0	1	0	3	1	2	2	0
colosseum	0	0	0	10	4	0	3	1	2	0	0
consul	4	0	3	1	4	2	7	0	0	2	0
forum	0	0	0	0	2	13	0	2	0	0	0
laurel	0	0	0	1	3	2	21	2	0	0	0
roman	1	0	0	0	0	1	0	18	0	2	0
senatus	0	0	8	0	3	0	5	3	6	0	1
trajan	4	2	0	0	0	0	1	6	1	8	0
vesta	0	0	2	2	1	1	6	2	2	0	2



Confusion Matrix V2

	augustus	aureus	cicero	colosseum	consul	forum	laurel	roman	senatus	trajan	vesta
augustus	19	2	0	1	0	0	2	2	0	1	0
aureus	7	7	1	1	1	1	0	6	0	0	0
cicero	0	0	10	1	1	0	3	0	9	0	0
colosseum	1	0	1	8	0	0	4	0	5	0	1
consul	1	0	2	4	3	4	7	0	1	0	1
forum	0	0	0	1	1	13	2	0	0	0	0
laurel	2	0	0	0	1	0	24	0	2	0	0
roman	5	1	0	0	0	0	3	11	0	2	0
senatus	0	0	12	0	0	0	5	0	8	0	1
trajan	7	0	0	0	0	0	6	4	0	5	0
vesta	3	0	2	2	1	0	5	0	2	0	3

PRE-TRAINED MODELS

Our custom CNNs stalled below ~50% val acc and converged slowly, so we then evaluated several ImageNet-pretrained backbones:

ResNet-18 (frozen backbone)
Training aborted early due to very long runtime, did not complete full 5-epoch run

MobileNetV2 (CPU friendly)

- Batch size = 16, no AMP, 5 epochs
- Validation accuracy by epoch: 42.06%, 46.03%, 51.19%, 56.35%, 58.33%

MobileNetV2 (partial fine-tuning)

- Unfroze last two feature blocks, added dropout(0.4)+linear head
- Validation accuracy by epoch: 49.60%, 60.32%, 65.87%, 68.65%, 70.63%, 69.44%, 72.22%, 73.41%, 70.63%, 70.63%, 69.44%, 73.41%, 72.62%, 70.24%.

MobileNetV2 (no augmentation)

Same setup but no data augmentation, 10 epoch.

Validation accuracy by epoch:

40.08%, 55.16%, 53.57%, 64.68%, 64.68%, 65.08%, 67.46%, 71.43%, 69.05%, 70.24%

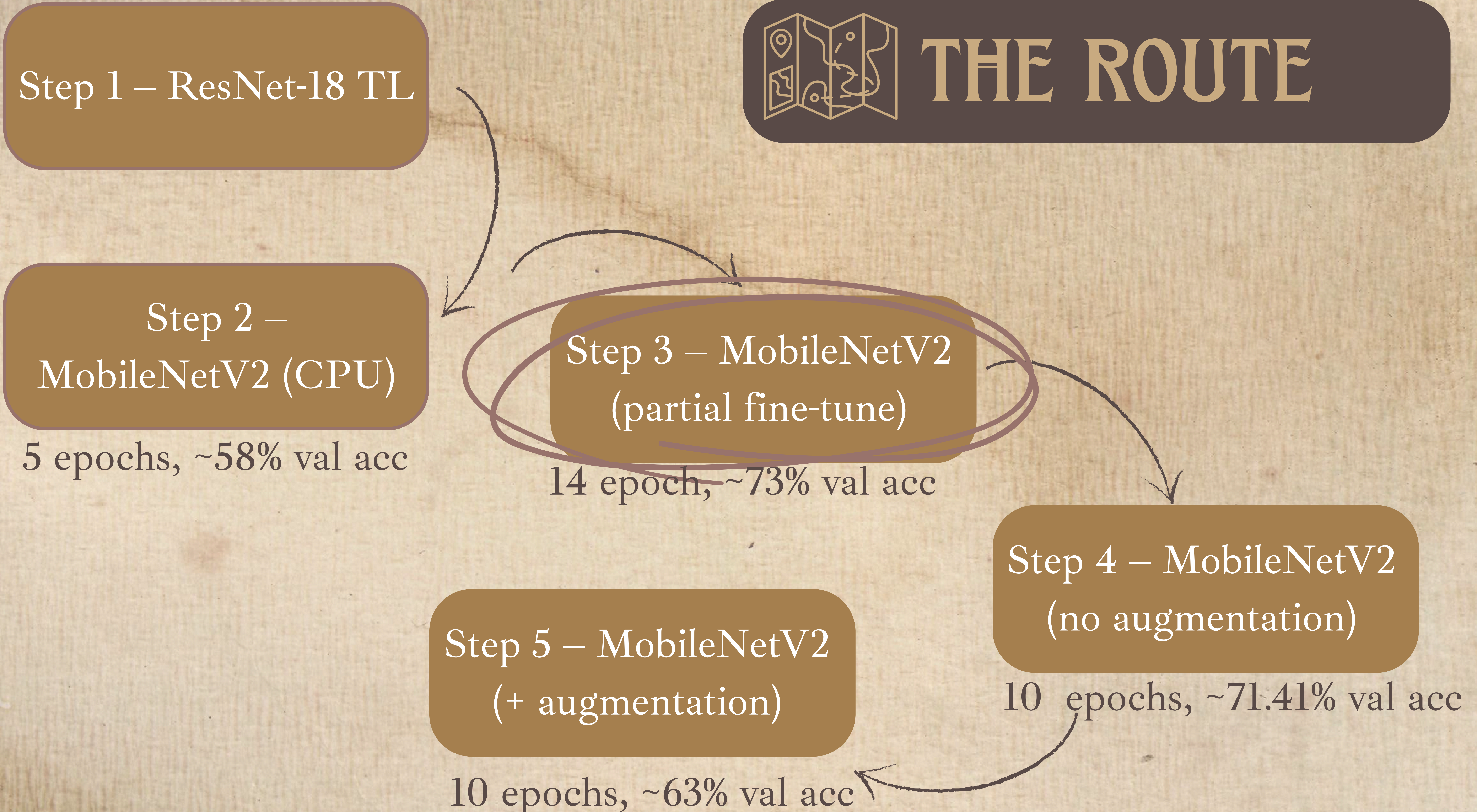
Best val acc: 71.43%

MobileNetV2 (+ aggressive augmentation)

Added flips, rotations, color jitter, grayscale, 7 epochs so far

Validation accuracy by epoch: 38.10%, 50.79%, 50.40%, 50.79%, 56.35%, 59.92%, 61.90%

THE ROUTE



Step 1 – ResNet-18 TL

Step 2 – MobileNetV2 (CPU)

5 epochs, ~58% val acc

Step 3 – MobileNetV2 (partial fine-tune)

14 epoch, ~73% val acc

Step 4 – MobileNetV2 (no augmentation)

10 epochs, ~71.41% val acc

Step 5 – MobileNetV2 (+ augmentation)

10 epochs, ~63% val acc

Model	Base Network	Augmentation	Fine-Tuning	Optimizer	Training Epochs	Device
ResNet18 (TL)	ResNet18	✅	❌	AdamW	5	GPU
MobileNetV2 (CPU)	MobileNetV2	✅	❌	AdamW	5	CPU
Optimized	MobileNetV2	✅	✅ (selected layers)	AdamW	30 (Early Stopping)	GPU
No Augmentation	MobileNetV2	❌	❌	Adam	10	MPS (Apple Silicon)
Augmentation	MobileNetV2	✅	❌	Adam	10	MPS (Apple Silicon)

The model we have chosen:

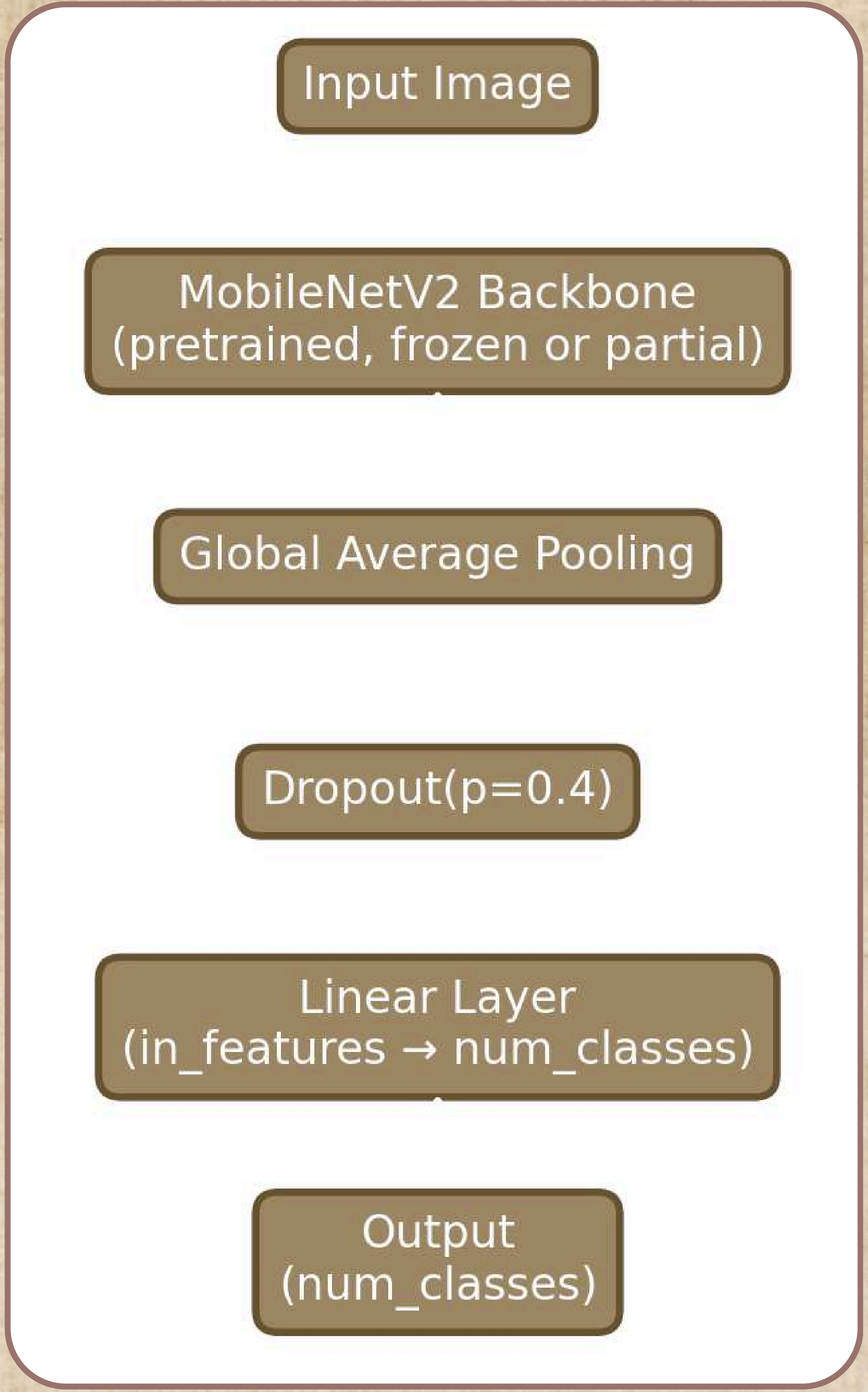
The model leverages MobileNetV2, a lightweight and efficient CNN network, pre-trained on ImageNet and based on inverted residual blocks with linear bottlenecks

A final part that collects the features in a vector via global average pooling and a final classification via a two-layer classifier

Dropout (0.4): Reduces overfitting by randomly disabling 40% of output units.

Linear Layer: that maps the feature vector (of in_f dimension) in the number num_classes of target classes (fonts).

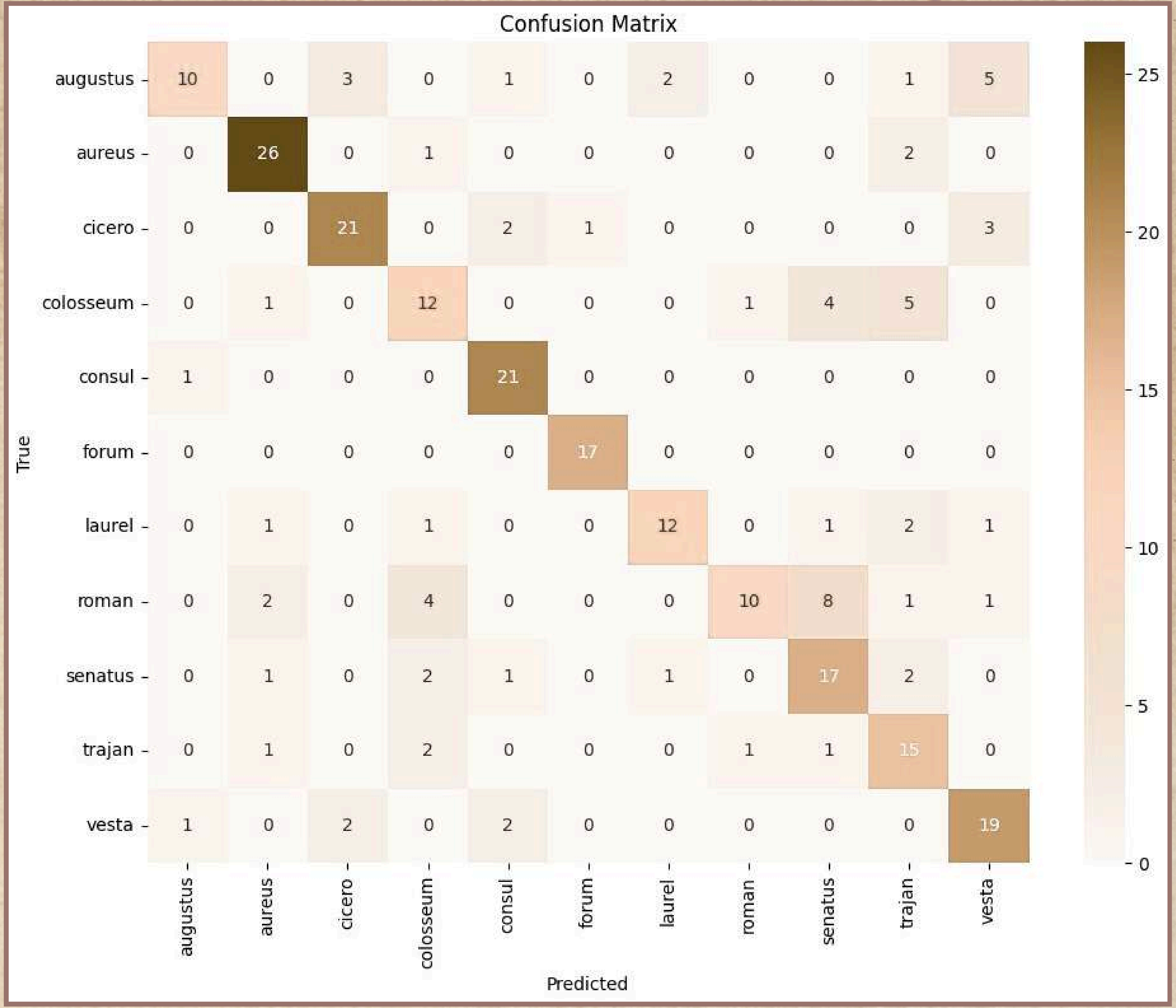
Output: each vector element represents the score associated with each class/font.


Input Image
MobileNetV2 Backbone (pretrained, frozen or partial)
Global Average Pooling
Dropout(p=0.4)
Linear Layer (in_features → num_classes)
Output (num_classes)

THE RESULTS

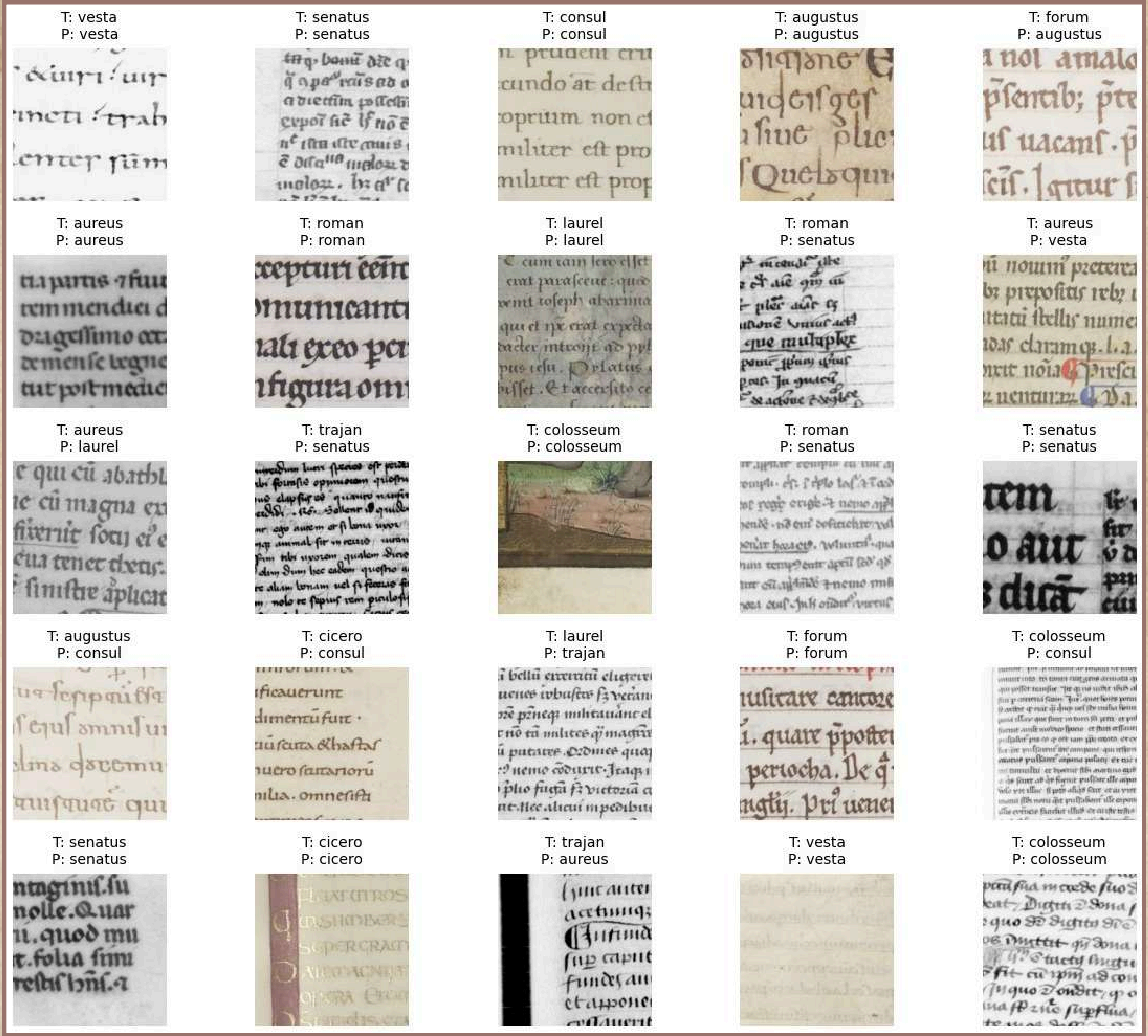
<u>Key insights:</u>

- Strong Diagonal: Most classes (e.g., aureus, consul, forum, vesta) show high true-positive counts.
- Notable Confusions:
  - augustus ↔ cicero (3 misclassified as cicero)
  - colosseum ↔ traian/ roman (5 as trajan, 4 as roman)
  - roman ↔ senatus (8 as senatus)
- Rare Misclassifications: Some fonts like senatus and aureus have very few off-diagonal errors, indicating robust discrimination.

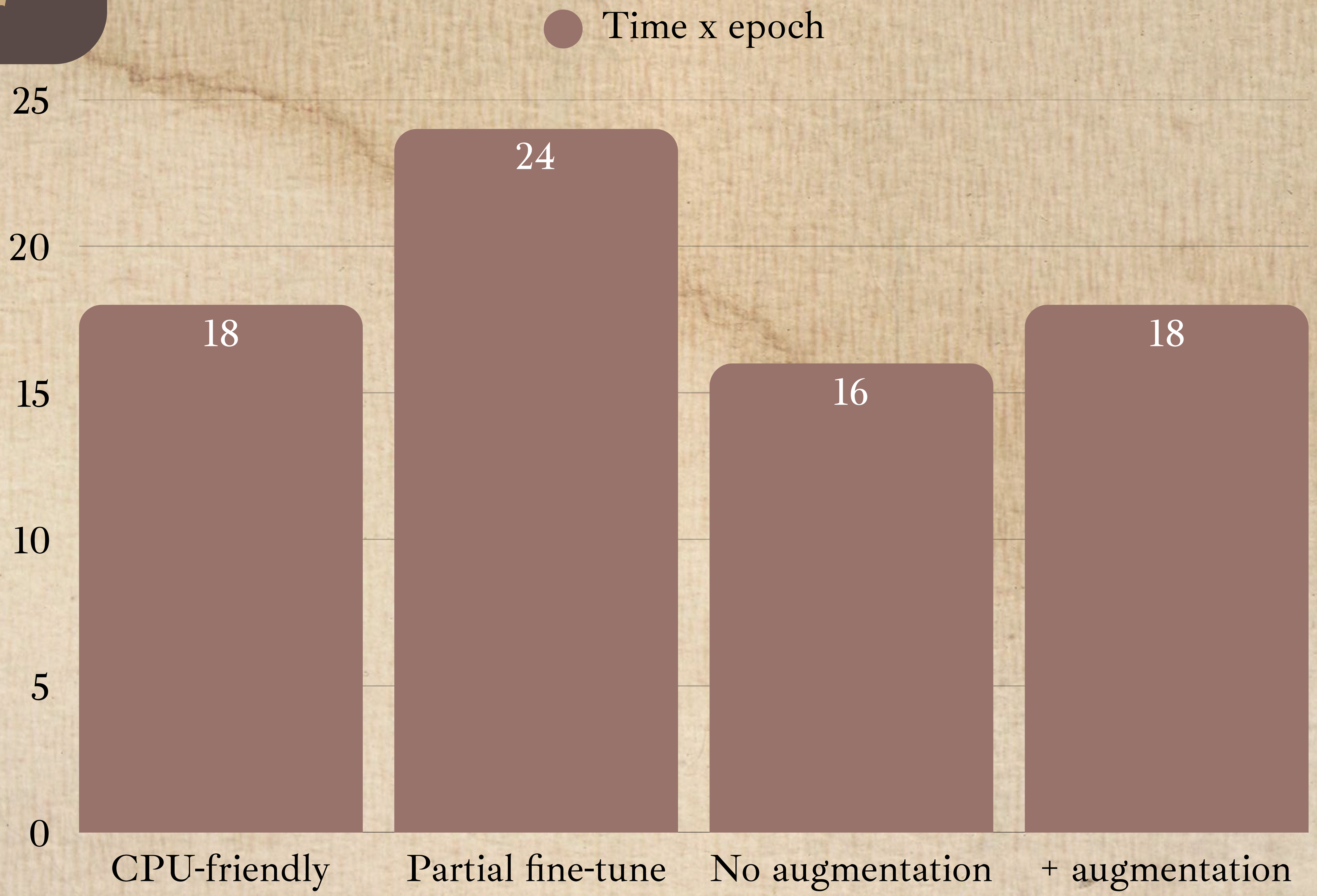


	augustus	aureus	cicero	colosseum	consul	forum	laurel	roman	senatus	trajan	vesta
augustus	10	0	3	0	1	0	2	0	0	1	5
aureus	0	26	0	1	0	0	0	0	0	2	0
cicero	0	0	21	0	2	1	0	0	0	0	3
colosseum	0	1	0	12	0	0	0	1	4	5	0
consul	1	0	0	0	21	0	0	0	0	0	0
forum	0	0	0	0	0	17	0	0	0	0	0
laurel	0	1	0	1	0	0	12	0	1	2	1
roman	0	2	0	4	0	0	0	10	8	1	1
senatus	0	1	0	2	1	0	1	0	17	2	0
trajan	0	1	0	2	0	0	0	1	1	15	0
vesta	1	0	2	0	2	0	0	0	0	0	19

SAMPLE CASE


T: vesta
P: vesta
T: senatus
P: senatus
T: consul
P: consul
T: augustus
P: augustus
T: forum
P: augustus
T: aureus
P: aureus
T: roman
P: roman
T: laurel
P: laurel
T: roman
P: senatus
T: aureus
P: vesta
T: aureus
P: laurel
T: trajan
P: senatus
T: colosseum
P: colosseum
T: roman
P: senatus
T: senatus
P: senatus
T: augustus
P: consul
T: cicero
P: consul
T: laurel
P: trajan
T: forum
P: forum
T: colosseum
P: consul
T: senatus
P: senatus
T: cicero
P: cicero
T: trajan
P: aureus
T: vesta
P: vesta
T: colosseum
P: colosseum

Runtime



● Time x epoch
25
20
15
10
5
0
18
24
16
18
CPU-friendly
Partial fine-tune
No augmentation
+ augmentation

Evaluation Metrics

$$\text{Accuracy} = \frac{\sum_i (TP_i + TN_i)}{\sum_i (TP_i + TN_i + FP_i + FN_i)}$$

$$\text{Precision}_i = \frac{TP_i}{TP_i + FP_i}$$

$$\text{Recall}_i = \frac{TP_i}{TP_i + FN_i}$$

$$\text{F1}_i = 2 \times \frac{\text{Precision}_i \times \text{Recall}_i}{\text{Precision}_i + \text{Recall}_i}$$

$$\text{Train Loss} = \frac{1}{N_{\text{train}}} \sum_j \ell_j$$

$$\text{Val Loss} = \frac{1}{N_{\text{val}}} \sum_j \ell_j$$

$$\text{Time/Epoch} = t_{\text{end}} - t_{\text{start}}$$

Limitations & Future Work

Limitations

- Dataset variability: noisy scans, inconsistent centering, and mixed single/double pages still challenge model robustness.
- Class imbalance: some fonts underrepresented, leading to skewed learning despite weighted loss.
- Compute constraints: aggressive augmentation and fine-tuning pipelines require long runtimes on limited hardware.
- Architecture ceiling: custom CNNs plateaued ~50% val accuracy; transfer learning helped but may not fully capture font nuances

Limitations & Future Work

Future work

- Advanced backbone exploration: Experiment with Transformers;
- Augmentation & preprocessing: Implement adaptive denoising (e.g., Denoising Autoencoders) and page-layout normalization;
- Data expansion: Augment dataset with synthetic font renderings or acquire more historical scans to balance rare classes;
- Hyperparameter optimization: Automate tuning via Bayesian search.

Conclusions

- Problem Addressed: Successfully tackled the challenge of classifying historical fonts from noisy, inconsistent scanned pages.
- Approach & Findings:
  - Custom CNNs provided a solid baseline (~50% val acc) but plateaued quickly.
  - Transfer learning with MobileNetV2 achieved the best performance (up to 71.4% val acc) and faster convergence.
- Key Takeaways
  - Pretrained backbones can dramatically boost accuracy and reduce training time.
  - Data quality and diversity remain critical—robust preprocessing & augmentation pay off.
  - Compute-resource constraints influenced model choice and training strategies

Thank You

ПОЧТОВАЯ КАРТОЧКА
Pocztów
КАРТОЧКА
50190
50190
50190
WEZ
Moreau, édit., Versailles