TRAFFIC SIGN CLASSIFIER

2025

by Fiz Garrido Escudero

UNIVERSIDADE DA CORUÑA
Data Mining and Neural Networks Course

General objective

Enhance road safety and driver assistance by building a model that can reliably recognize traffic signs in real time, reducing human errors and supporting ADAS/autonomous vehicles.

Specific objectives

- Explore a labeled image dataset
- Learn to design and train CNN and KNN models
- Apply data-mining, cleaning, and preprocessing techniques
- Evaluate and interpret results rigorously

Project scope

Problem

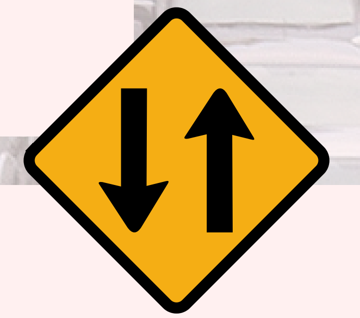
Drivers' weak sign knowledge

+

visual overload

=

Wrong interpretation of signs



Deliverables

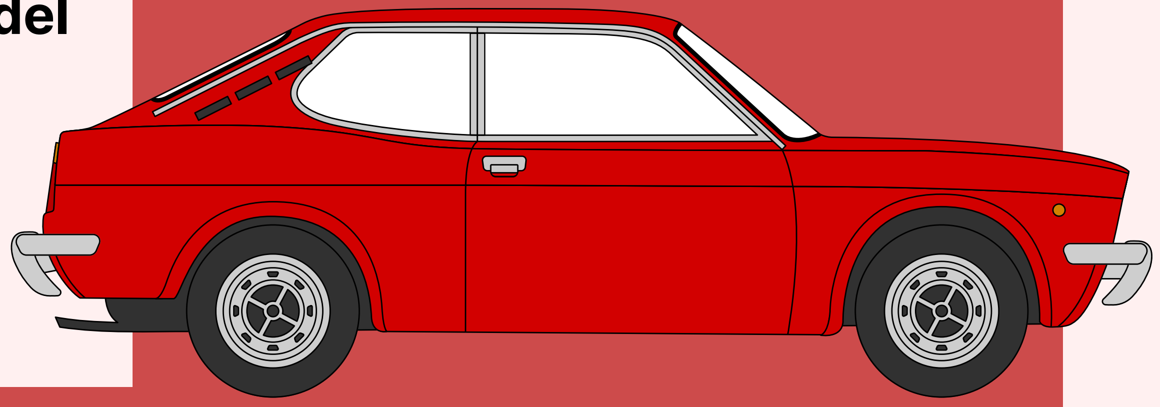
- Trained models and evaluation report.
- Executable Jupyter notebook

In Scope

- Use a made of images dataset (43 cropped traffic-sign classes).
- Data pipeline: loading, cleaning, resizing, normalizing, etc
- Develop and train a CNN and a KNN model
- Evaluation & interpretation
- Documentation & reproducibility: runnable notebook + presentation.

Out of Scope

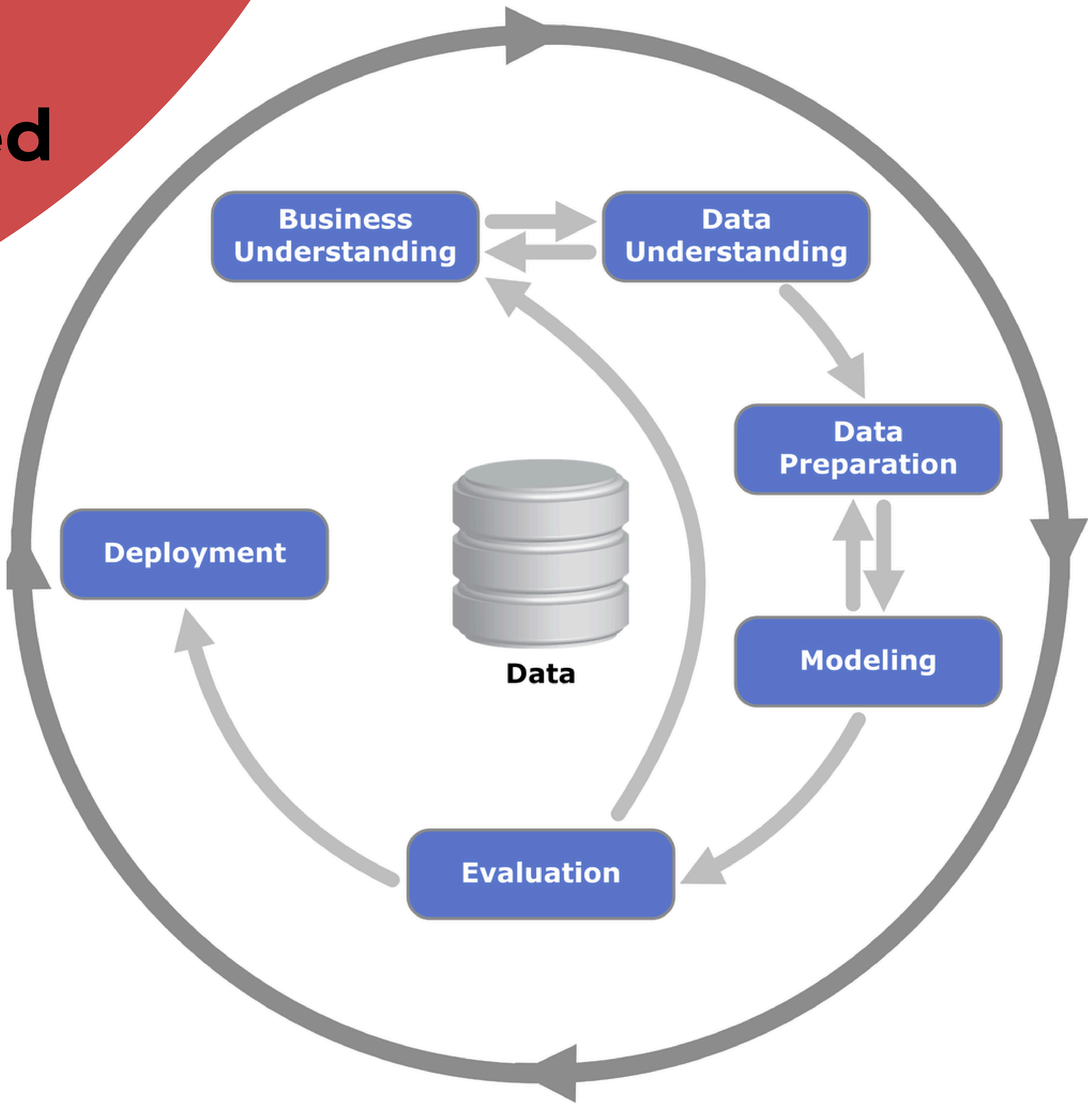
- Detecting signs in full road scenes
- Real-time deployment
- Multi-country sign coverage.



Assumptions

- Correct labels
- Clean data
- Colab's type GPU
- Limited time

Methodology used



Algorithms applied

CNN

Pretrained MobileNetv2

+

classification head

Optimization & Training Tricks

- Adam optimizer
- Dropout
- BatchNormalization
- EarlyStopping

KNN

- It classifies based on the k nearest neighbors of the sample
- Only for checking CNN performance

Key results: CNN

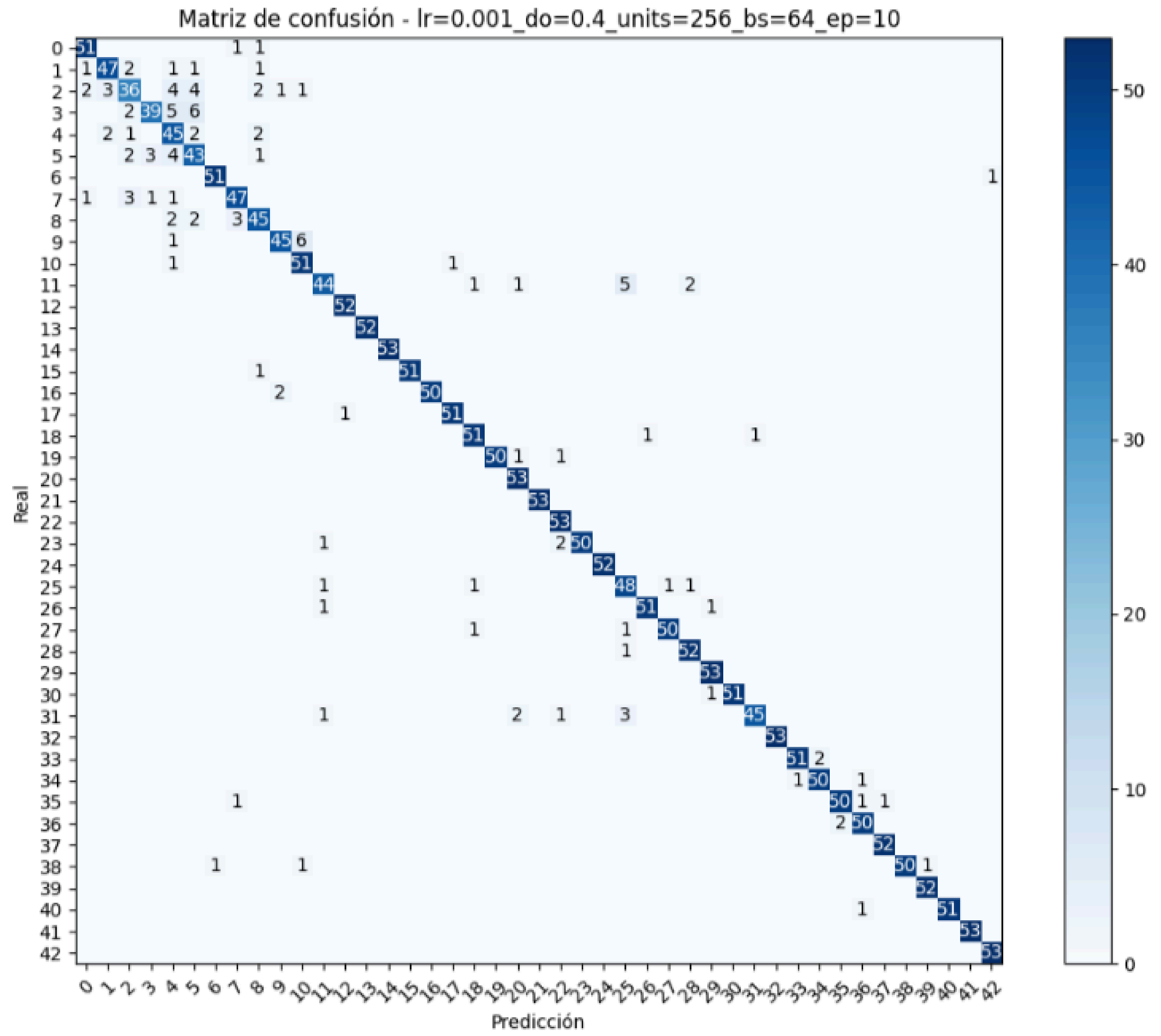
Resultados de Configuraciones del Modelo

config	learning_rate	dropout	dense_units	batch_size	epochs	val_accuracy	test_accuracy	test_loss	f1_macro	f1_weighted	precision_macro	recall_macro
lr=0.001_do=0.4_units=256_bs=64_ep=30	0.0010	0.4000	256	64	30	0.9393	0.9389	0.1924	0.9389	0.9389	0.9404	0.9389
lr=0.001_do=0.4_units=256_bs=32_ep=30	0.0010	0.4000	256	32	30	0.9376	0.9336	0.2383	0.9338	0.9337	0.9355	0.9337
lr=0.001_do=0.4_units=256_bs=64_ep=10	0.0010	0.4000	256	64	10	0.9327	0.9433	0.2045	0.9434	0.9434	0.9454	0.9433
lr=0.0001_do=0.4_units=256_bs=64_ep=30	0.0001	0.4000	256	64	30	0.9327	0.9283	0.2552	0.9280	0.9279	0.9287	0.9283
lr=0.001_do=0.4_units=256_bs=32_ep=10	0.0010	0.4000	256	32	10	0.9296	0.9221	0.2487	0.9221	0.9220	0.9258	0.9222
lr=0.0001_do=0.4_units=256_bs=32_ep=30	0.0001	0.4000	256	32	30	0.9269	0.9283	0.2586	0.9283	0.9282	0.9293	0.9284
lr=0.0001_do=0.4_units=256_bs=32_ep=10	0.0001	0.4000	256	32	10	0.9114	0.9057	0.3456	0.9054	0.9053	0.9072	0.9058
lr=0.0001_do=0.4_units=256_bs=64_ep=10	0.0001	0.4000	256	64	10	0.8933	0.8835	0.4267	0.8831	0.8830	0.8842	0.8836

Test accuracy (best model): 94.33%

Key results: CNN

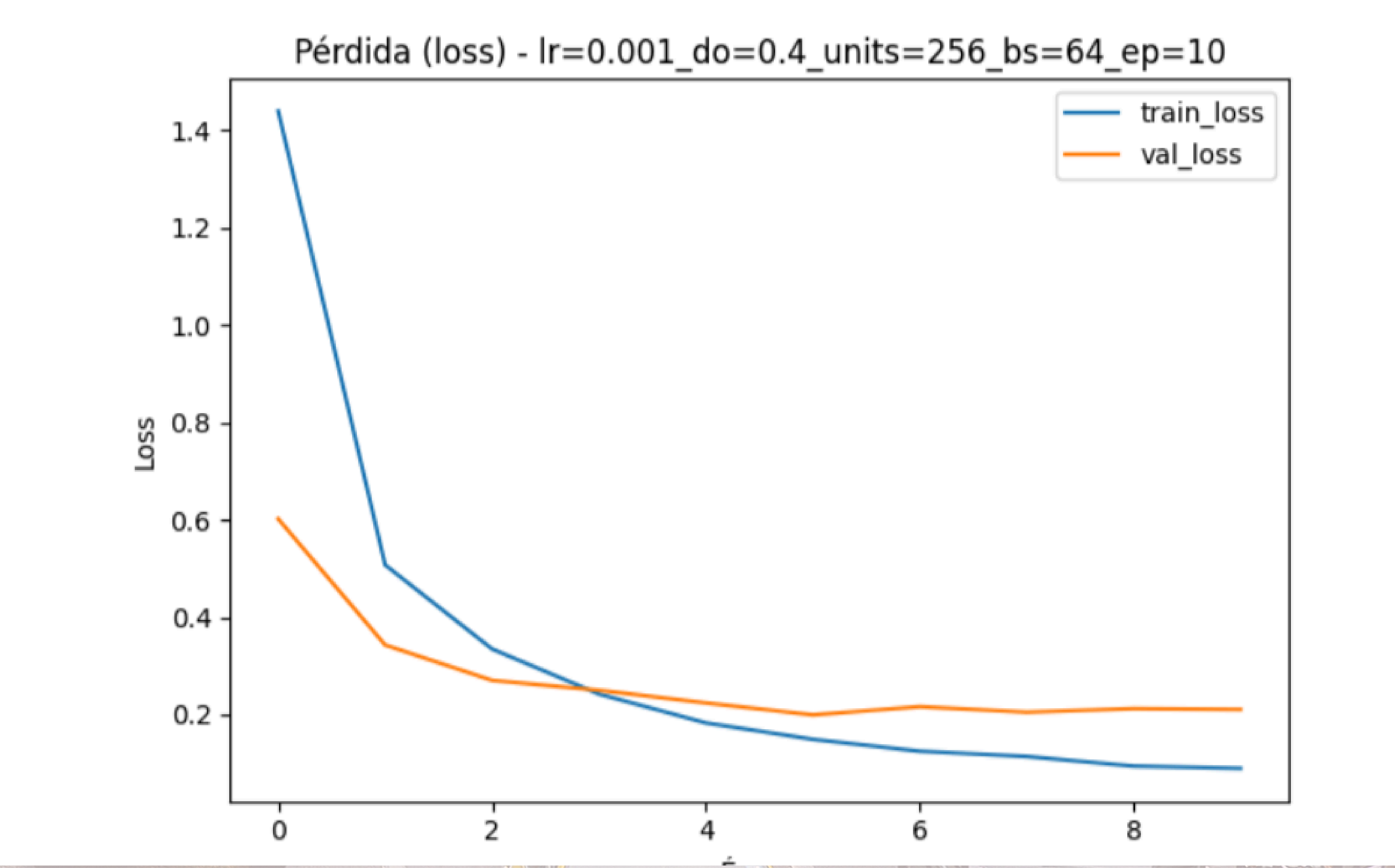
¡¡Robust model !!



Key results: CNN

validation loss
vs
train loss

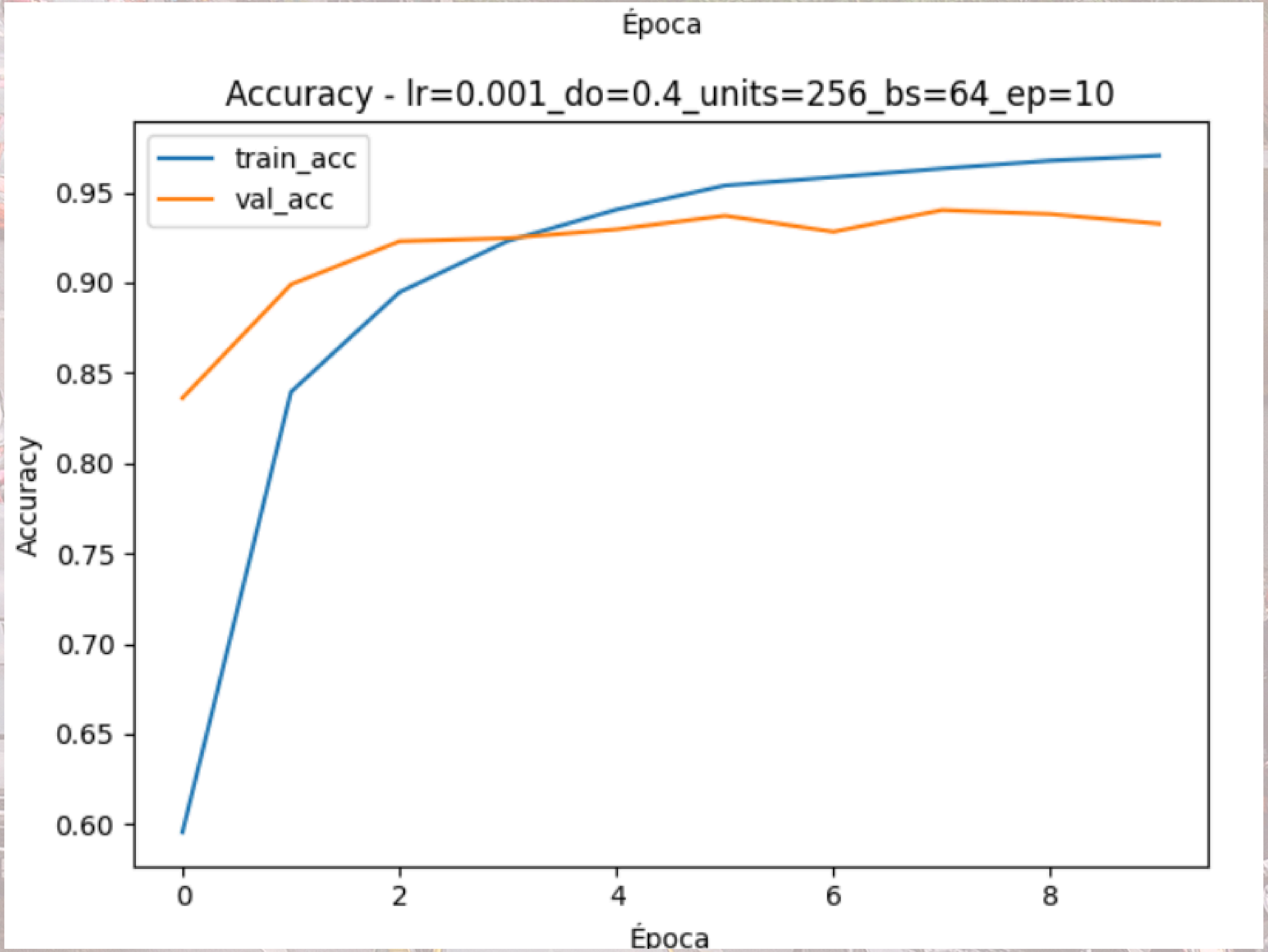
Good learning WITHOUT OVERFITTING



Key results: CNN

validation accuracy
vs
train accuracy

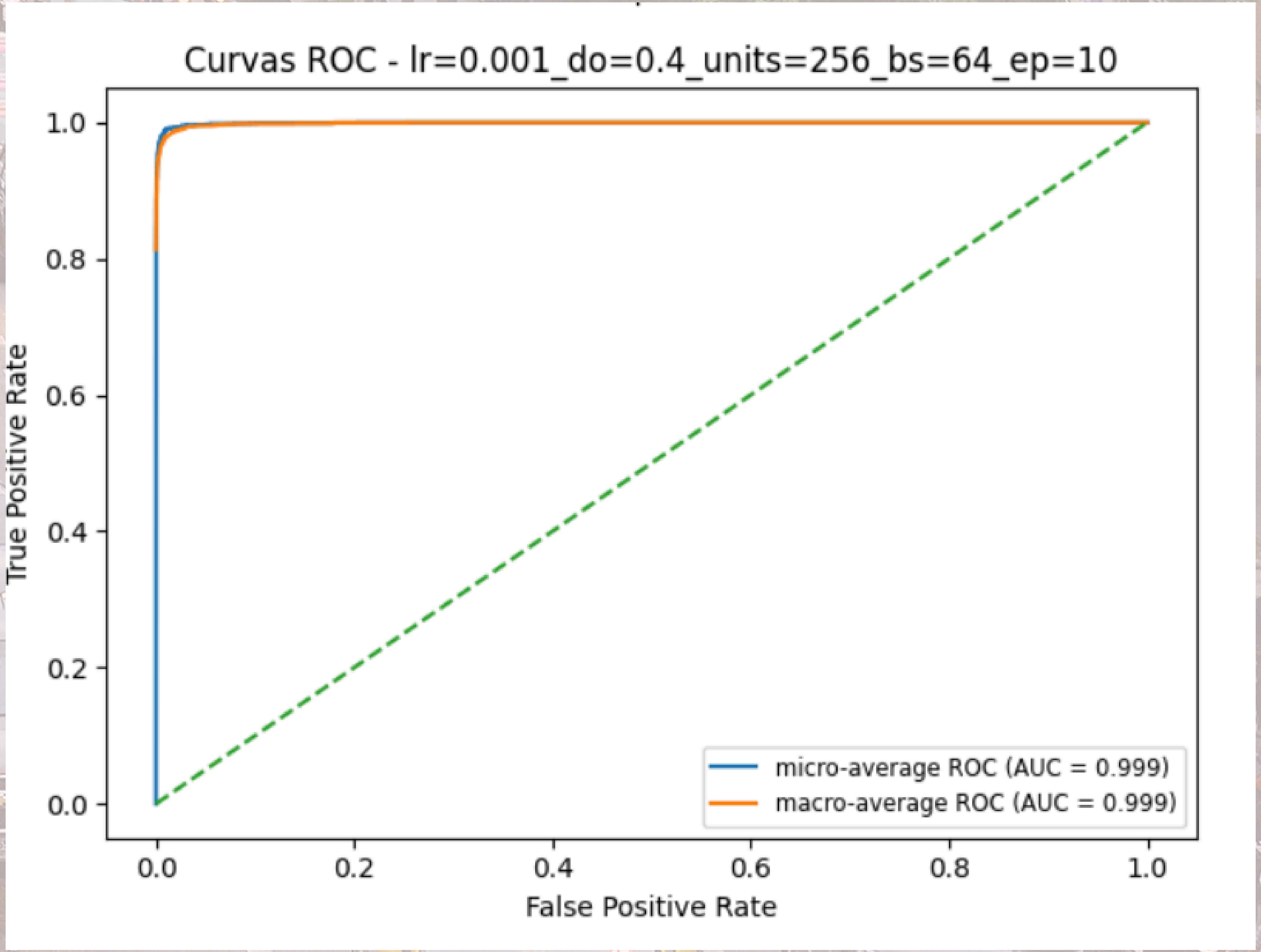
Good learning WITHOUT OVERFITTING



Key results: CNN

ROC Curve

Near-perfect separation between classes



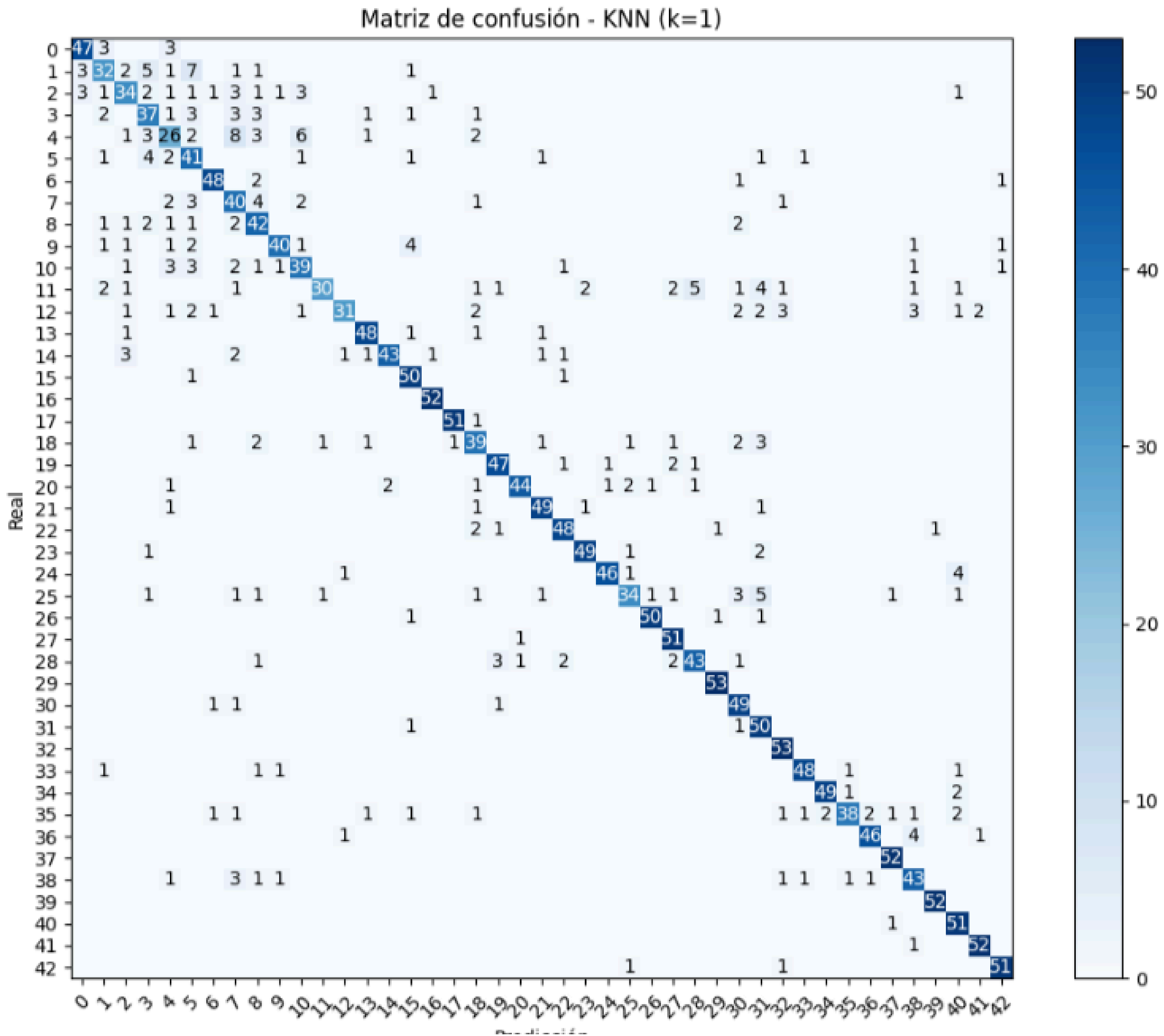
Key results: KNN

Resultados de Configuraciones del Modelo KNN

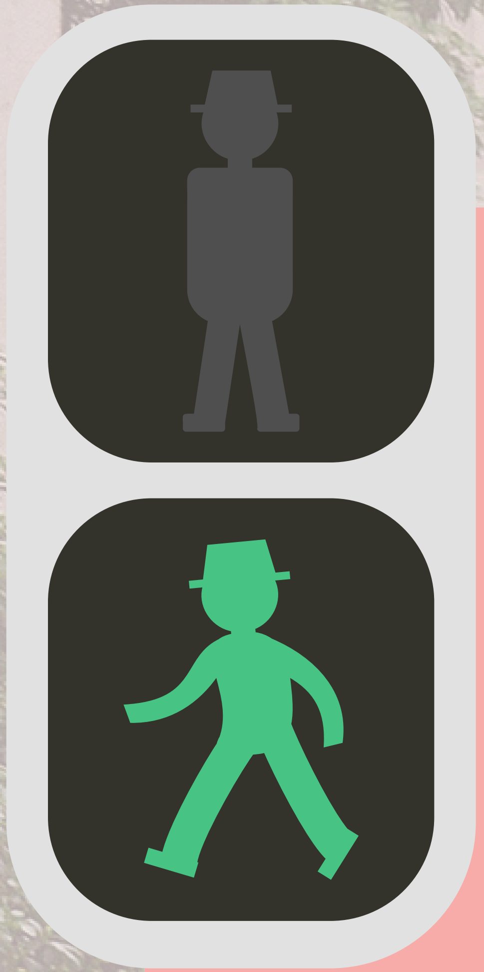
	config	k	test_accuracy	precision	recall	f1_score
0	KNN_k=1	1	0.8494	0.8563	0.8496	0.8480
1	KNN_k=3	3	0.7697	0.7917	0.7700	0.7720
2	KNN_k=5	5	0.7414	0.7577	0.7416	0.7397
3	KNN_k=7	7	0.7095	0.7242	0.7097	0.7055
4	KNN_k=13	13	0.6581	0.6734	0.6584	0.6476

Test accuracy (best model, k = 1): 84.94%

Key results: KNN



Findings



- **CNN clearly outperforms kNN**
- **CNN is expensive to train but cheap to predict; kNN is the opposite.**
- **Training/validation curves stay close**

  → no strong overfitting; early stopping was effective

- **Remaining errors appear between visually similar signs.**

Conclusions

- CNN vs ML classic models
- Necessarity of cleaning and preprocessing
- Difficulties of implementing a real life problem

Recommendations

- Using more RAM and a powerfull GPU
- Collect the largest dataset posible

FOR HYPOTHETICAL FINAL USERS:

¡¡¡ Practice mindful driving and pay attention to road signs !!!