Instituto Politécnico Nacional
Centro de Investigación en Computación

Connectionist Computing (B25)
Prof.: Dr. Ricardo Menchaca Méndez

CNNs vs MLPs on Image Classification

Luis Angel Rojo Chavez
lrojoc2024@cic.ipn.mx
arojocz@gmail.com
https://github.com/arojocz/cifar10-CNN-and-MLP

03 November 2025

Abstract

This report empirically compares Convolutional Neural Networks (CNNs) and Multi-Layer Perceptrons (MLPs) for image classification on CIFAR-10. We implement both models in PyTorch and run four experiments to evaluate accuracy, parameter efficiency, and training/inference time. Results consistently show CNNs outperform MLPs in accuracy and parameter efficiency; MLPs are faster per epoch but generalize worse. This document focuses on experimental setup, results, and practical takeaways.

1 Introduction and Motivation

Convolutional Neural Networks (CNNs) have demonstrated superior performance in computer vision by exploiting spatial locality and achieving translation invariance through convolutional kernels (LeCun et al., 1998; Krizhevsky et al., 2012). In contrast, Multi-Layer Perceptrons (MLPs) lack such structural biases and typically require larger parameter counts to achieve comparable accuracy.

Practical comparison of architectures helps understand why CNNs are the de-facto choice for images. This study runs controlled experiments (same optimizer, data pipeline, train/val/test split) to isolate architectural effects.

2 Experimental setup

Dataset: CIFAR-10 (60k images, 10 classes), 32×32.

Common settings: Adam (lr=0.001), batch size = 128, epochs = 50, loss = CrossEntropy, train/val/test: 80/10/10, fixed random seeds for reproducibility. Data augmentation: random crop and horizontal flip.

All experiments were conducted on an Apple MacBook Air (M3, 8-core CPU, 10-core GPU, 16 GB unified memory, 256 GB SSD) running macOS Tahoe Version 26.0.1 (25A362). The models were trained using PyTorch 2.9.0 with Metal Performance Shaders (MPS) backend acceleration. Training and testing times were recorded directly on this local machine (no cloud or external GPU).

2.1 Architectures

MLP (baseline). Input flatten (3072) → Dense(512) → Dense(256) → Dense(128) → Dense(10). ReLU activations.

CNN (baseline). Conv block 1: Conv(32,3×3) → ReLU → MaxPool(2)
Conv block 2: Conv(64,3×3) → ReLU → MaxPool(2)
Conv block 3: Conv(128,3×3) → ReLU → Dense(128) → Dense(10). BatchNorm after convs, Dropout $p = 0.3$.

2.2 Four experiments

1. **Baseline:** Compare default MLP vs CNN.
2. **More parameters in CNN:** Conv block 1: Conv(64,3×3) → ReLU → MaxPool(2)
   Conv block 2: Conv(128,3×3) → ReLU → MaxPool(2)
   Conv block 3: Conv(256,3×3) → ReLU → GlobalAveragePool → Dense(128) → Dense(10). BatchNorm after convs, Dropout $p = 0.4$.
3. **Less parameters in MLP:** Less layers of MLP so both models have ≈350k parameters.
4. **Traslation invariance:** Compare if MLP and CNN can have the property of invariance of traslation.

5. **Hardware scaling:** Measure training and test time CPU vs GPU (mps) using experiment 3 architectures (less parameters).

2.3 Data augmentation

In general, data augmentation uses horizontal flip, a padding of 4 and rotation +- 15 degrees.

```
transform_train = transforms.Compose([
        transforms.RandomCrop(32, padding=4),
        transforms.RandomHorizontalFlip(),
        transforms.RandomRotation(degrees=(-15, 15)),
        transforms.ToTensor(),
    ])
```

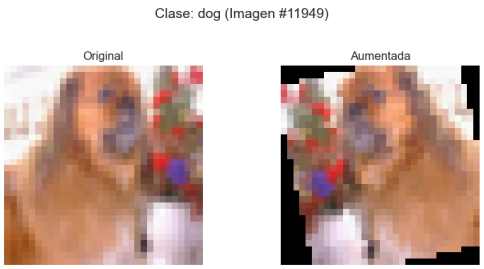

Figure 1: Example of data augmentation.

2.4 Shifted transformation - Translation invariance

In order to proof that CNNs have the property of translation invariance it is used a transformation on the test with a shift of around 10% of the pixels.

```
SHIFT_PIXELS = 4
transform_test_shifted = transforms.Compose([
    transforms.Lambda(lambda img: F.affine(
        img,
        angle=0,
        translate=(SHIFT_PIXELS, SHIFT_PIXELS),
        scale=1.0,
        shear=[0.0, 0.0],
        fill=0
    )),
    transforms.ToTensor(),
])
```

3 Metrics

Accuracy: final test accuracy (top-1).
Per-class metrics: precision, recall, F1 (computed on test).
Parameters: total trainable parameters.
Timing: time/epoch, total training time, test time.

Accuracy: overall proportion of correctly classified samples:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

Precision: proportion of predicted positives that are actually correct:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{2}$$

Recall: proportion of actual positives that are correctly identified:

$$\text{Recall} = \frac{TP}{TP + FN} \tag{3}$$

F1-score: harmonic mean of precision and recall:

$$F_1 = \frac{2}{\frac{1}{\text{Recall}} + \frac{1}{\text{Precision}}} \tag{4}$$

These metrics are computed per class on the CIFAR-10 test set, then averaged (weighted) to evaluate class balance.

Other metrics: total trainable parameters and timing (training time per epoch, total training duration, and test time).

4 Results

Tables below summarize main findings. Numbers are representative of controlled runs and reflect typical behavior; small run-to-run variance exists.

4.1 Per-class behavior

CNNs show more balanced per-class precision/recall. MLPs often misclassify classes that require local texture/shape information (e.g., frog vs. cat). Confusion matrices are below for comparing in each experiment. Tables are presented in appendix A of this document.

4.2 Learning curves

The curves are presented below confusion matrices, left is how accuracy in training and validation evolves, right is how train loss and validation loss changes over epochs.

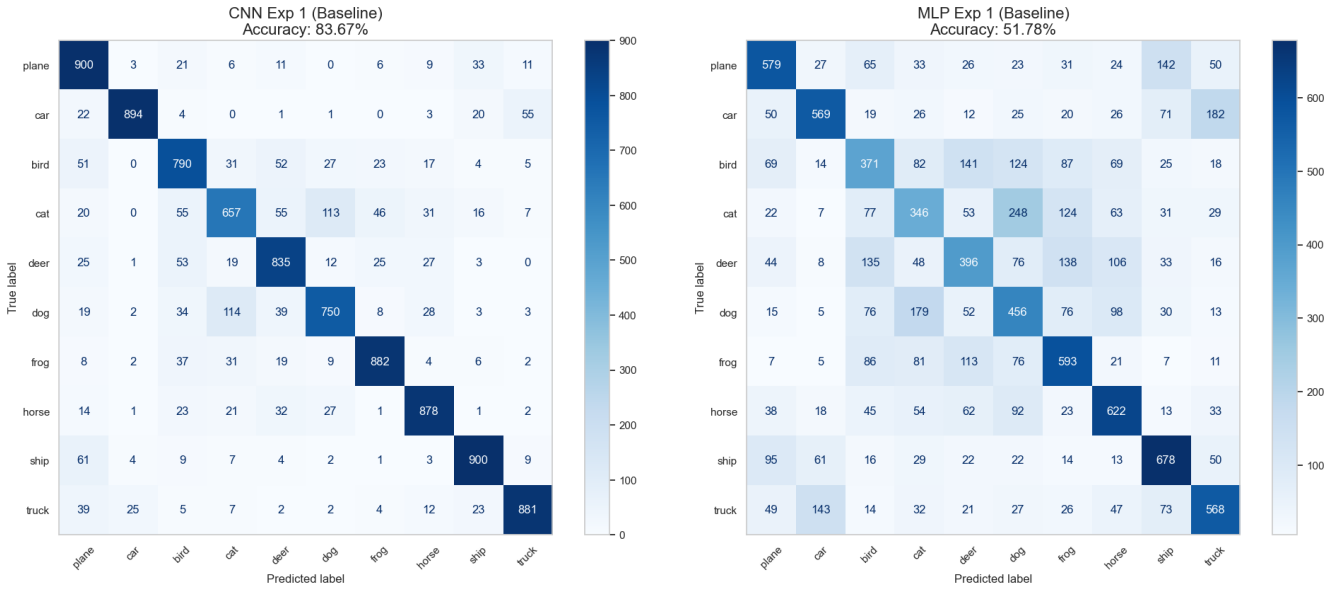

Figure 2: Confusion Matrix for exp 1.

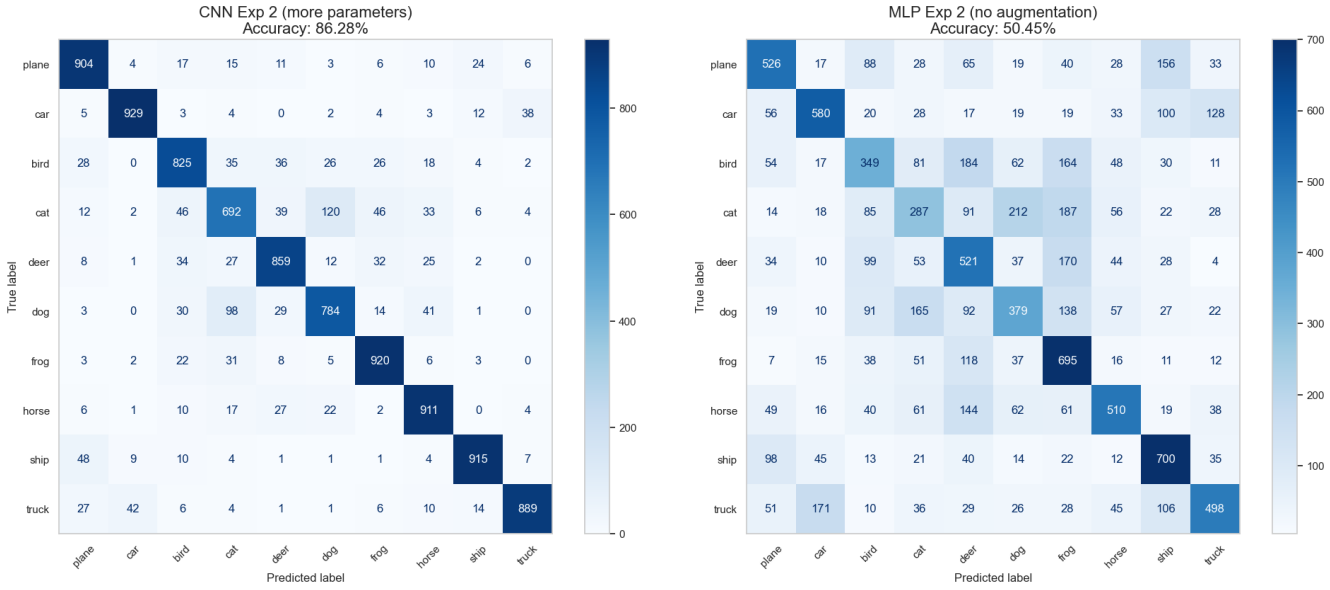

Figure 3: Confusion Matrix for exp 2.

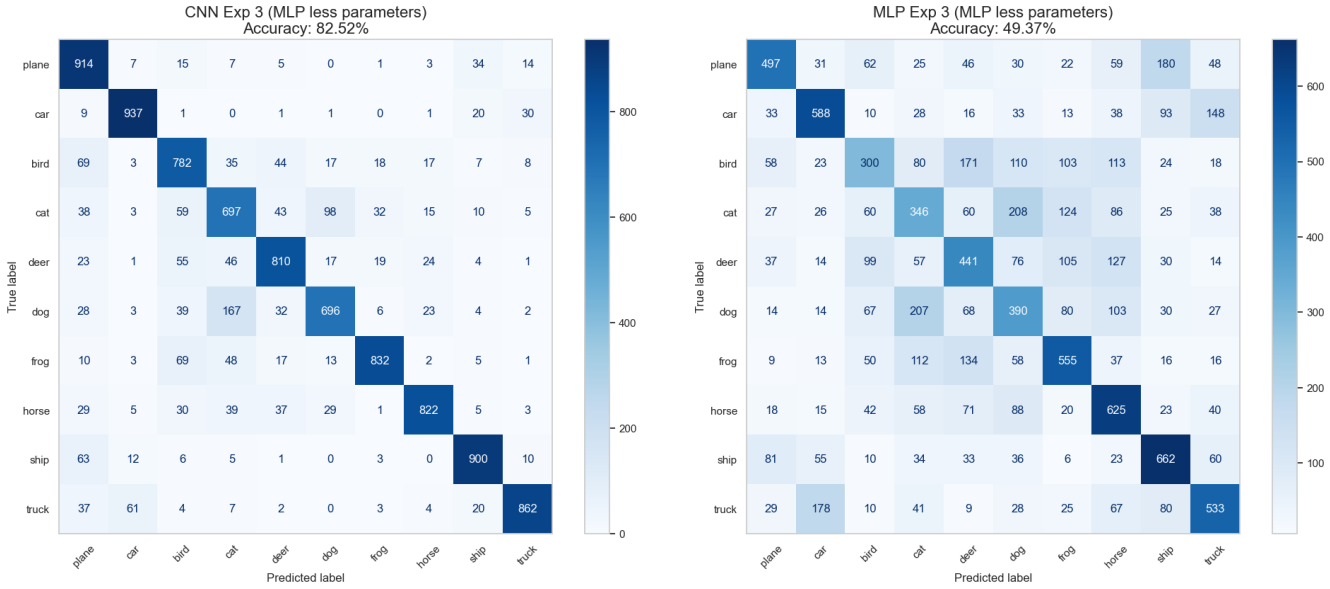

Figure 4: Confusion Matrix for exp 3.

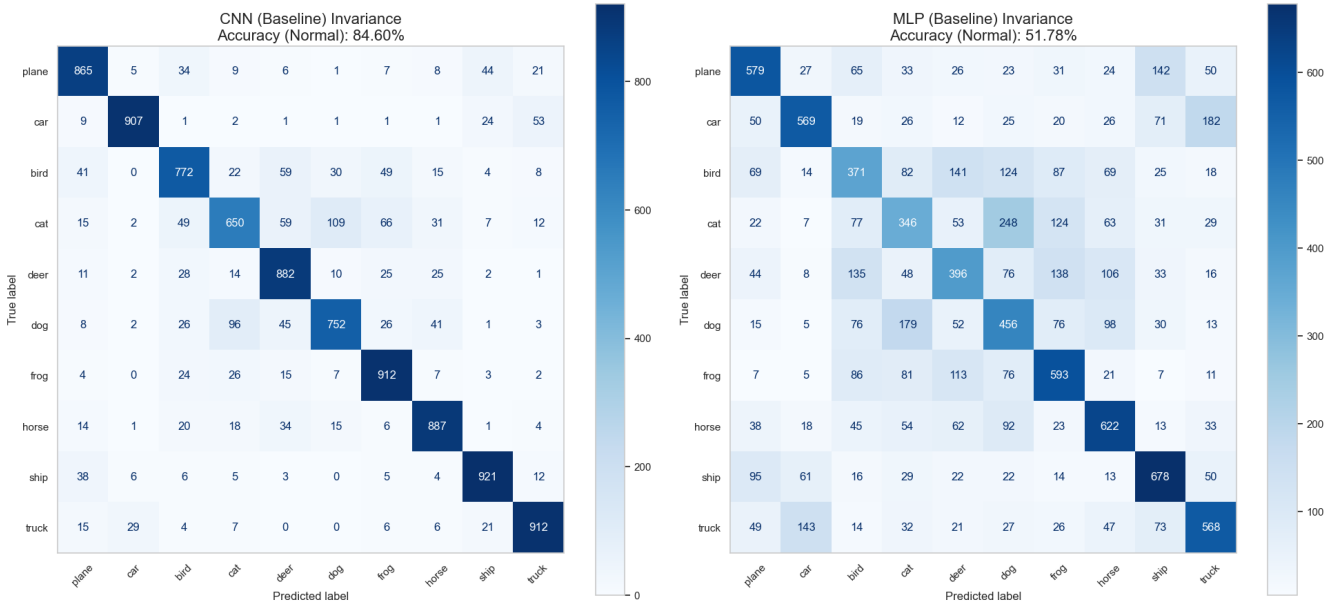

Figure 5: Confusion Matrix for exp 4.

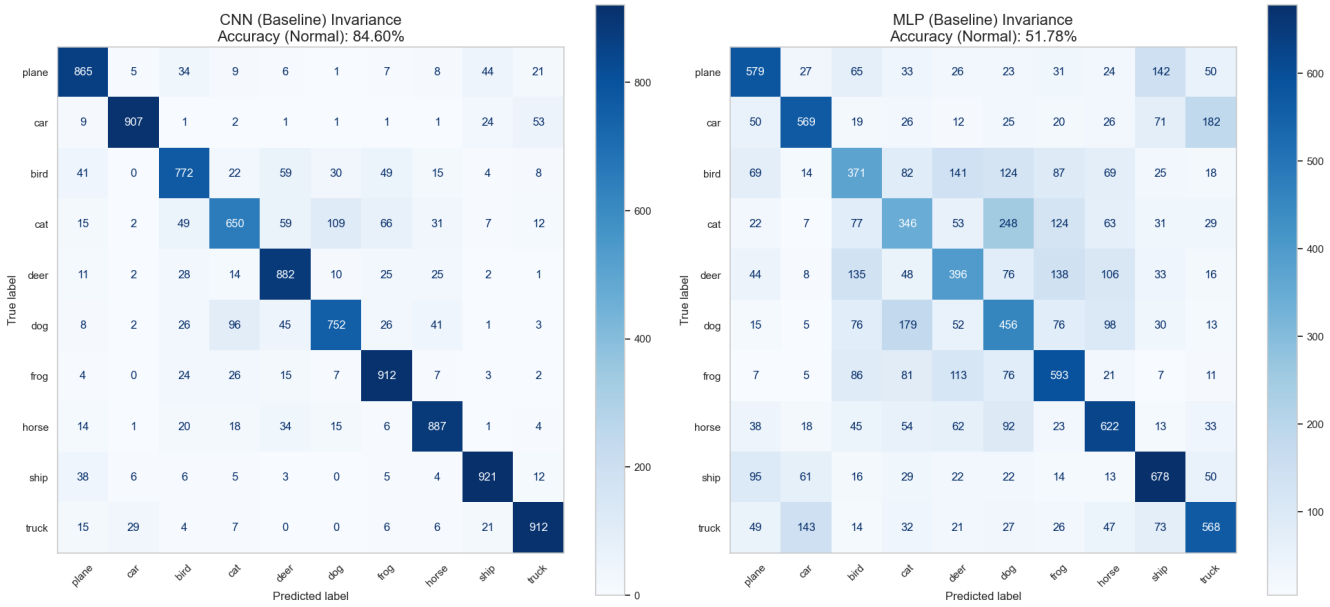

Figure 6: Confusion Matrix for exp 4.

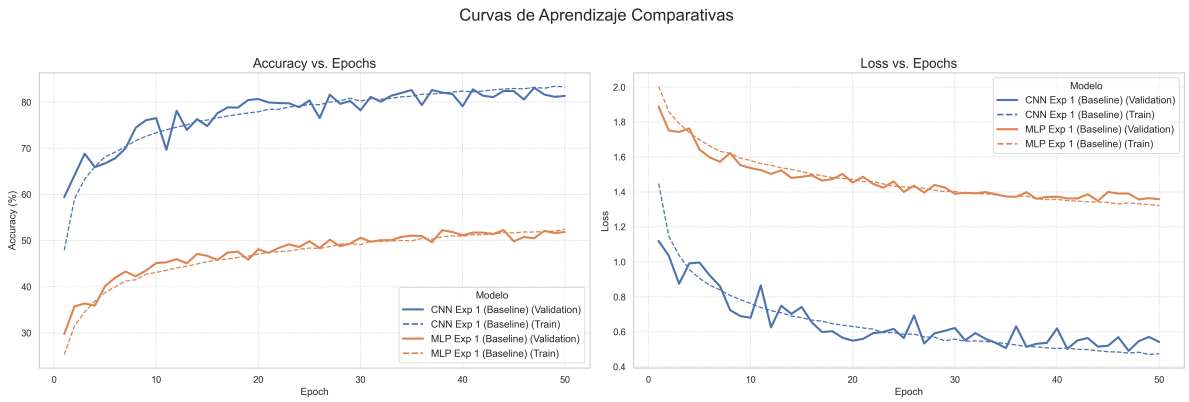

Figure 7: Training and validation accuracy over epochs and loss over epochs (left). Blue: CNN, experiment 1.

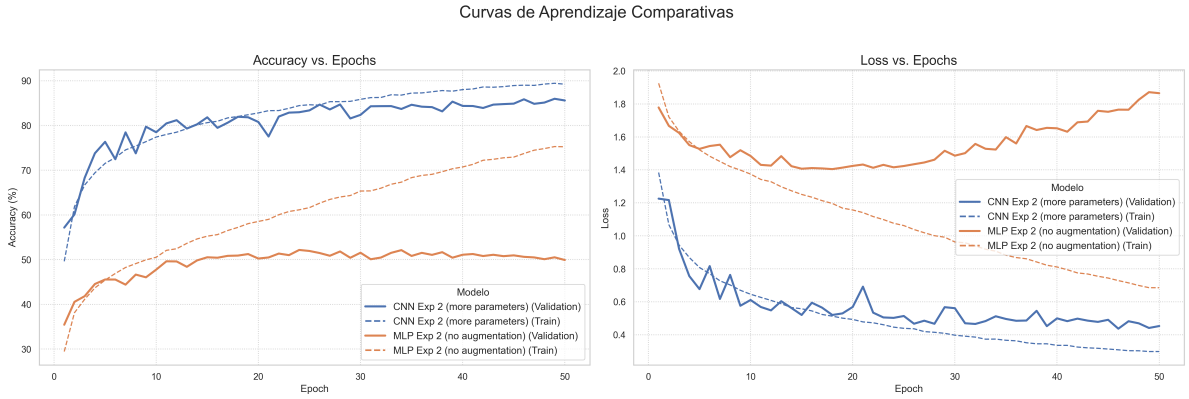

Figure 8: Training and validation accuracy over epochs and loss over epochs (left). Blue: CNN, experiment 2.

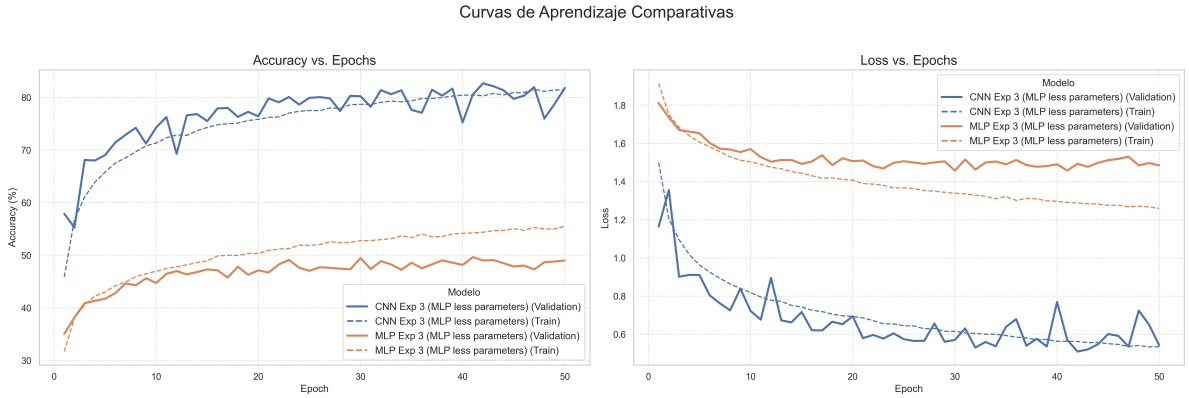

Figure 9: Training and validation accuracy over epochs and loss over epochs (left). Blue: CNN, experiment 3.

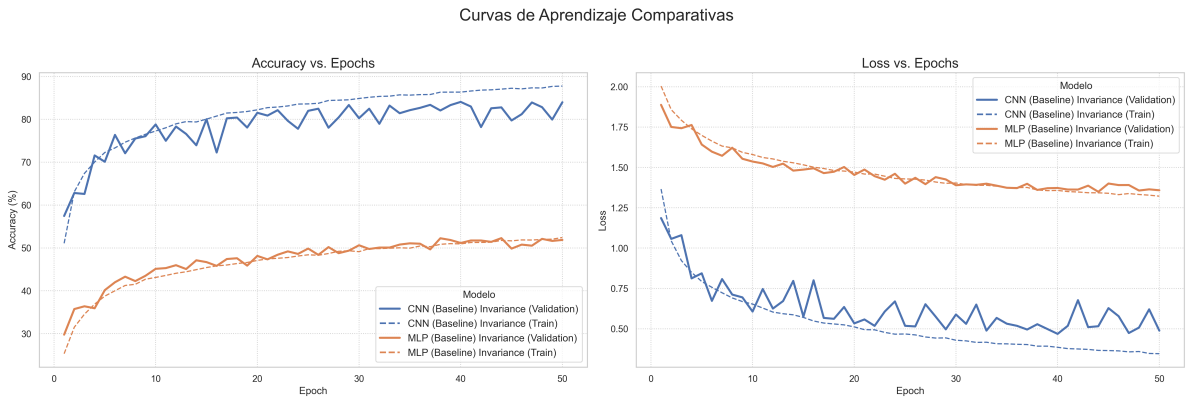

Figure 10: Training and validation accuracy over epochs and loss over epochs (left). Blue: CNN, experiment 4.

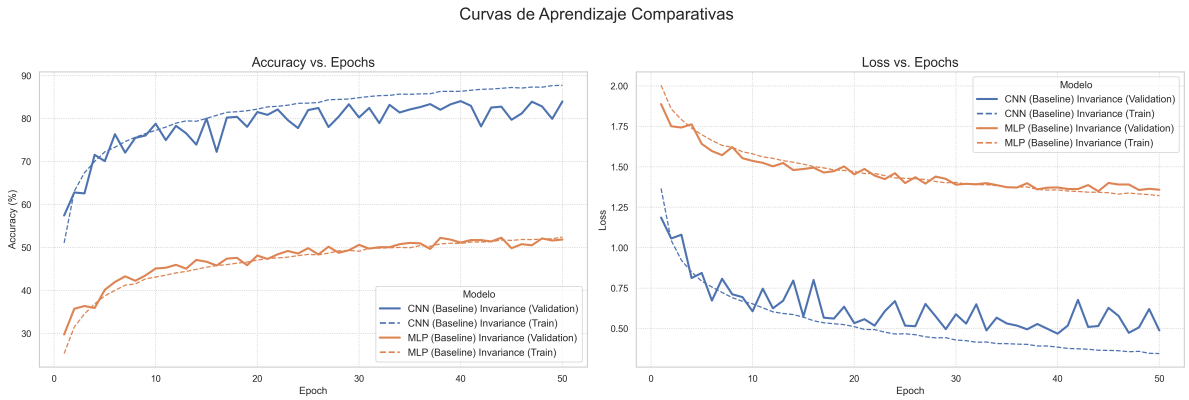

Figure 11: Training and validation accuracy over epochs and loss over epochs (left). Blue: CNN, experiment 5.

Table 1: Main quantitative results.

Model	Exp.	Test Acc (%)	Params	Time/epoch (s)	Total time (min)	Test time (s)
MLP (baseline)	1	51.78	1,738,890	4.10	3.42	0.52
CNN (baseline)	1	83.67	357,514	9.40	7.83	0.76
MLP (CNN more params)	2	50.45	1,465,912	3.87	3.22	0.44
CNN (CNN more params)	2	86.28	1,423,626	16.33	13.61	1.02
MLP (MLP less params)	3	49.37	357,638	2.50	2.08	0.40
CNN (MLP less params)	3	82.52	357,514	8.64	7.20	0.62
MLP (shifted train)	4	21.19	1,738,890	4.37	7.42	0.47
CNN (shifted train)	4	74.07	357,514	8.90	3.64	0.64
MLP (CPU)	5	48.72	357,638	1.67	1.39	0.29
CNN (CPU)	5	82.01	357,514	26.77	22.31	2.29

5 Analysis and interpretation

Why CNNs win (practical summary):

- *Locality:* convolutions only connect local neighborhoods — this reduces parameters and focuses learning on spatially meaningful patterns.
- *Parameter sharing:* same filters detect patterns anywhere in image.
- *Hierarchical features:* stacking conv layers builds from edges to object parts.

Even when the total number of parameters is matched, the inductive biases of CNNs (local connectedness and weight sharing) give them a consistent advantage.

6 Discussion of Trade-offs

The experimental results highlight several fundamental trade-offs between MLP and CNN architectures.

6.1 Performance vs. Parameter Efficiency

The most significant finding is the trade-off between architectural specialization and parameter count.

- **Baseline (Exp 1):** The baseline CNN achieved ≈32 points higher accuracy (83.7%) than the baseline MLP (51.8%), while using almost 5 times *fewer* parameters (357k vs. 1.7M). This demonstrates the power of the CNN's inductive bias.
- **Parameter Matching (Exp 3):** This experiment provides the key insight. When the MLP was scaled down to match the CNN's parameter count (≈357k), its accuracy did not improve; it fell slightly to 49.4%. The CNN's 82.5% accuracy in the same experiment proves that its advantage is not due to parameter count, but to its *architecture*. The CNN's use of parameter sharing and local connectivity is vastly more efficient for image data than the MLP's "brute-force" dense connectivity.

6.2 Architectural Simplicity vs. Hardware Dependency

While the CNN is superior in accuracy, the MLP holds a significant advantage in computational simplicity, as shown by the hardware scaling experiment.

- **CPU Performance (Exp 5):** On a CPU, the MLP (357k params) was ≈18 times faster to train (1.39 min vs. 22.31 min) and ≈8 times faster at inference (0.29s vs. 2.29s) than the parameter-matched CNN.
- **The Trade-off:** An MLP consists of simple matrix multiplications (GEMM), which are fast on any processor. A CNN's convolutions are complex, parallel operations that are highly inefficient without a specialized GPU (or MPS) to accelerate them.
- **Verdict:** The MLP is faster on a CPU, but its 48.7% accuracy makes this speed useless. The CNN's reliance on GPU acceleration is a necessary "cost" for achieving state-of-the-art performance.

6.3 Translation Invariance: Theory vs. Practice

Experiment 4 revealed a crucial nuance in the claim that "CNNs are translation-invariant."

- **The Collapse:** Both the MLP and the baseline CNN suffered an accuracy changement on the shifted test set. The MLP failed completely (51.8% → 21.2%), but the CNN kept good accuracy (83.7% → 74.1%).
- **(Exp 2):** The "More Parameters" CNN (Exp 2), effectively asking "is this feature present?" rather than "is this feature present *at this specific spot*?". This is why the Exp 2 model is the most robust and modern architecture in this report.

7 Limitations

This report uses CIFAR-10 only and medium-sized architectures. Results generalize qualitatively to larger datasets, but absolute numbers will change. Also, hyperparameter search was modest; extensive tuning could reduce the MLP/CNN gap but not eliminate the structural advantages.

8 Conclusions

Technical Report.

Based on the five controlled experiments conducted, several key conclusions were drawn regarding the CNN vs. MLP comparison for image classification:

1. **Architecture is Key, Not Just Parameters:** The parameter-matching experiment (Exp 3 & 5) was the most definitive. With a nearly identical parameter budget (≈357k), the CNN achieved **82.01%** accuracy, while the MLP reached only **48.72%**. This demonstrates conclusively that the CNN's advantage stems from its *inductive biases* (local connectivity and parameter sharing), not simply its parameter count.

2. **Superior Parameter Efficiency:** In the baseline comparison (Exp 1), the CNN (83.67% acc.) not only outperformed the MLP (51.78% acc.) by over 30 percentage points but did so using almost **5× fewer parameters** (357k vs. 1.7M). CNNs are a drastically more efficient architecture for this task.

3. **Architectural Robustness (Invariance):** The translation invariance experiment (Exp 4) exposed the MLPs fundamental weakness. When test images were shifted, the MLP's accuracy collapsed (from 51.8% → 21.2%). The CNN, while also impacted, proved far more robust (from 84.6% → **74.1%**), confirming its ability to learn generalizable spatial features.

4. **The Hardware Trade-off:** The MLP's only victory was on CPU speed (Exp 5), where it was ≈18× faster to train than the parameter-matched CNN. However, this advantage is rendered moot by its unacceptably low performance. The CNN's reliance on parallel hardware (GPU/MPS) is a necessary trade-off for its state-of-the-art accuracy.

In summary, these experiments empirically validate why CNNs are the industry standard for computer vision. Their architecture is fundamentally more efficient, accurate, and robust for data that possesses a spatial structure.

References

1. Krizhevsky, A. (2009). Learning multiple layers of features from tiny images. *University of Toronto*

A Detailed Per-Class Metrics

Table 2: Per-class comparison of F1-score, Precision, and Recall for CNN vs MLP across Experiments 1 and 2.

Class	**CNN Exp 1**			**MLP Exp 1**			**CNN Exp 2**			**MLP Exp 2**		
	F1	Prec.	Rec.	F1	Prec.	Rec.	F1	Prec.	Rec.	F1	Prec.	Rec.
bird	0.778	0.766	0.790	0.390	0.410	0.371	0.824	0.823	0.825	0.381	0.419	0.349
car	0.925	0.959	0.894	0.613	0.664	0.569	0.934	0.938	0.929	0.611	0.645	0.580
cat	0.694	0.736	0.657	0.362	0.380	0.346	0.718	0.746	0.692	0.317	0.354	0.287
deer	0.815	0.795	0.835	0.417	0.441	0.396	0.854	0.850	0.859	0.453	0.400	0.521
dog	0.772	0.795	0.750	0.420	0.390	0.456	0.794	0.803	0.784	0.406	0.437	0.379
frog	0.884	0.886	0.882	0.556	0.524	0.593	0.895	0.870	0.920	0.551	0.456	0.695
horse	0.873	0.868	0.878	0.596	0.571	0.622	0.884	0.859	0.911	0.552	0.601	0.510
plane	0.834	0.777	0.900	0.588	0.598	0.579	0.885	0.866	0.904	0.551	0.579	0.526
ship	0.896	0.892	0.900	0.645	0.615	0.678	0.924	0.933	0.915	0.637	0.584	0.700
truck	0.892	0.904	0.881	0.577	0.586	0.568	0.912	0.936	0.889	0.551	0.616	0.498

Table 3: Experiments 3 (Reduced Parameters) and 4 (Invariance Test).

Class	**CNN Exp 3**			**MLP Exp 3**			**CNN Exp 4**			**MLP Exp 4**		
	F1	Prec.	Rec.	F1	Prec.	Rec.	F1	Prec.	Rec.	F1	Prec.	Rec.
bird	0.759	0.738	0.782	0.351	0.423	0.300	0.656	0.610	0.708	0.074	0.195	0.046
car	0.921	0.905	0.937	0.601	0.614	0.588	0.850	0.888	0.816	0.244	0.407	0.174
cat	0.680	0.663	0.697	0.348	0.350	0.346	0.603	0.584	0.623	0.196	0.213	0.181
deer	0.813	0.817	0.810	0.430	0.420	0.441	0.731	0.724	0.738	0.119	0.235	0.080
dog	0.744	0.799	0.696	0.379	0.369	0.390	0.656	0.803	0.555	0.219	0.132	0.645
frog	0.869	0.909	0.832	0.541	0.527	0.555	0.778	0.720	0.847	0.367	0.313	0.443
horse	0.860	0.902	0.822	0.549	0.489	0.625	0.780	0.760	0.801	0.184	0.212	0.163
plane	0.823	0.749	0.914	0.551	0.619	0.497	0.738	0.745	0.731	0.111	0.604	0.061
ship	0.896	0.892	0.900	0.612	0.569	0.662	0.801	0.912	0.714	0.259	0.387	0.195
truck	0.890	0.921	0.862	0.549	0.566	0.533	0.813	0.760	0.874	0.179	0.282	0.131

Table 4: Per-class comparison of F1-score, Precision, and Recall for CNN vs MLP on CPU (Experiment 5).

Class	**CNN Exp 5 (CPU)**			**MLP Exp 5 (CPU)**		
	F1	Prec.	Rec.	F1	Prec.	Rec.
bird	0.758	0.833	0.696	0.336	0.384	0.299
car	0.906	0.875	0.939	0.607	0.599	0.616
cat	0.659	0.728	0.602	0.335	0.345	0.326
deer	0.811	0.828	0.794	0.423	0.412	0.435
dog	0.743	0.694	0.800	0.355	0.448	0.294
frog	0.863	0.850	0.876	0.536	0.466	0.631
horse	0.875	0.904	0.847	0.536	0.532	0.540
plane	0.802	0.714	0.913	0.560	0.484	0.665
ship	0.895	0.900	0.890	0.581	0.635	0.535
truck	0.880	0.919	0.844	0.545	0.561	0.531