IMAGE CLASSIFICATION

Task 2: Deep Learning / BoW

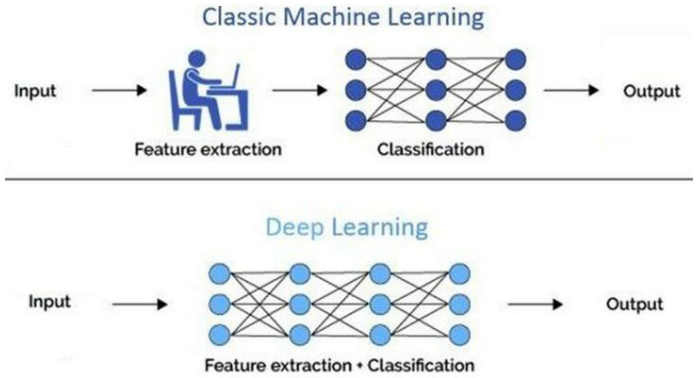
Team 5: Marc Artero Pons, Diego Hernández Antón, Bernat Medina Pérez and Pau Monserrat Llabrés

Overview

Last week, we classified images using **handcrafted features**. Now, however, we switch to **learnt representations**; features are automatically extracted by neural networks (NNs, the baseline is an MLP).

This paradigm shift raises several critical questions. We tackle the following:

- Do NNs alone improve last week's results?
- How is the learning affected by the size of the images?
- Similarly, what about network depth (number of layers)?
- Can last week's classifiers enhance NNs performance?
- Is an approach like spatial pyramids applicable here?
- Can we combine learnt features with Week 1 pipeline?
- What alternatives exist to MLPs?



Our experimental strategy is twofold. For hyperparameter tuning, we use 5-fold cross-validation to ensure robustness. However, for comparative analysis (where we just want to assess the impact of an approach rather than perform a selection) we train on the entire training set and evaluate directly on the test set.

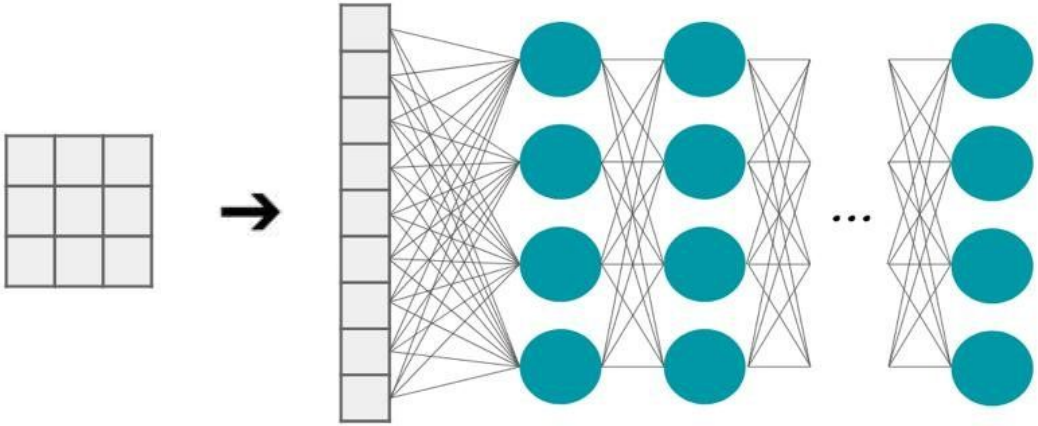
1. The baseline

The baseline

We want to know if the performance in the classification improves just with this paradigm shift. For that, we compare the baseline model (the Multi-Layer Perceptron from the base code) with our Week 1 pipeline.

➢ **Hypothesis**: The baseline MLP model does not improve last week's performance.

We believe this because MLPs process images as flattened vectors. These networks, per se, have no local spatial information: adjacent pixels are treated the same way as those on opposite sides of the images.



The baseline

We use last week's best configuration:

- **Local descriptor**: Dense SIFT (*stride* = *scale* = 8).
- **Codebook size**: 256.
- **Fisher vectors** with 25 components (N).
- Histogram normalization.
- **Spatial pyramid** with 0 levels (full images).
- **Classifier**: SVM with C = 1 and RBF kernel.

Metrics on the test set

	Accuracy	F1	Precision	Recall
Week 1	44.5	44.0	44.2	44.6
Week 2	22.8	22.2	28.3	22.8

The baseline for this week is the following:

- **Network:** MLP with 2 hidden layers.
- **Number of units in hidden layers:** 300.
- **Image size**: 224x224 pixels.
- **Training duration:** 20 epochs.

Note: Other aspects (e.g., optimizer) are held constant in the experiments: they fall outside the scope of our analysis.

❖ **HYPOTHESIS:** ✔

As expected, the baseline by itself does not improve the best performance got last week.

2. Image size

Image size

Image size determines the number of input features of the network. The higher, the richer information the network receives. However, this also increases the likelihood of the model **overfitting** to the training data.

➢ **Hypothesis**: The performance of the baseline MLP benefits from increasing the size of the input images. After an optimal value, though, the model starts to overfit, yielding lower accuracy.

The rest of the model is fixed:

- **Network:** MLP with 2 hidden layers.
- **Number of units in hidden layers:** 300.
- **Training duration:** 20 epochs.

Image size

- **HYPOTHESIS:** ✔

Despite expecting the optimal image size to be higher, we get a trend that satisfies our hypothesis. The performance increases after when switching from 2x2 to 4x4 images.

A plausible explanation for this result is that at 4x4 resolution, the image is reduced to 16 average colors, allowing the MLP to learn robust global patterns (e.g., sky vs. ground). Since MLPs treat pixels independently, at higher resolutions we introduce complex information that detriments classification.

Decision: The best image size is 4x4 pixels.

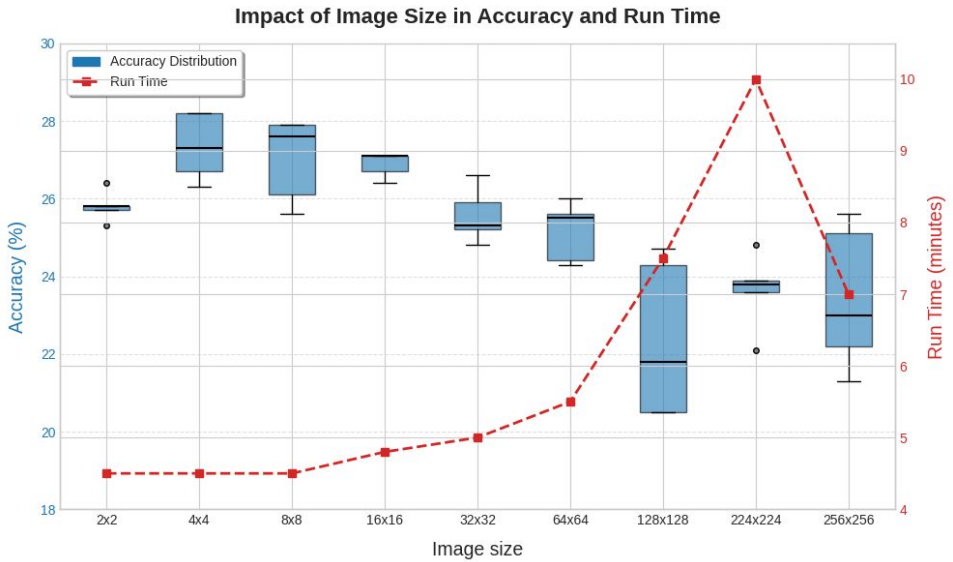


Image size

- **HYPOTHESIS:** ✔

The model struggles to generalize at higher resolutions, as exemplified by the diverging losses in the 256x256 case. On the other hand, for 4x4 images, the validation loss closely tracks the training loss.

Image size: 4x4

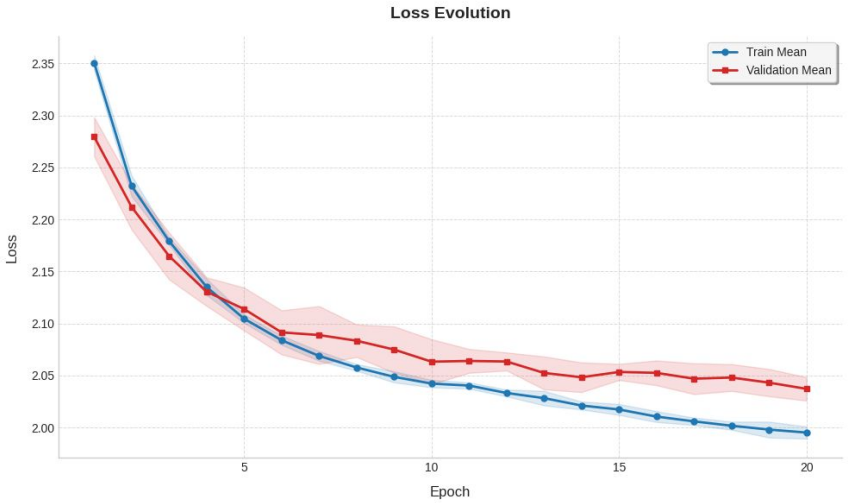
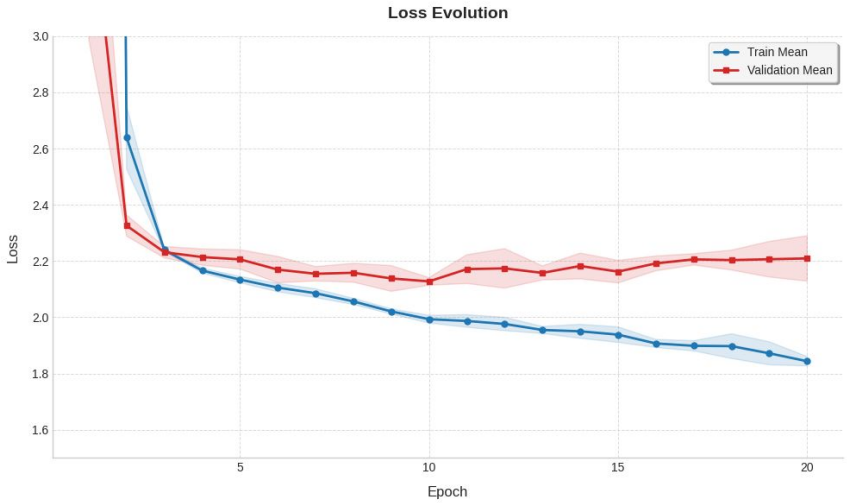


Image size: 256x256



3. Hidden layers

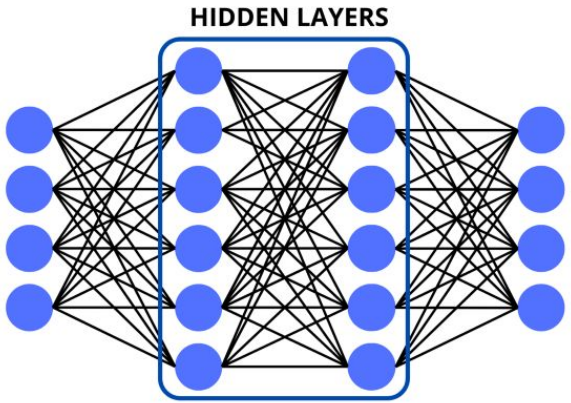
Hidden layers

The number of hidden layers is related to the model's complexity. As in the previous experiment, the greater it is, the better its learning capabilities but making it more prone to overfitting.

➢ **Hypothesis**: Again, there is a tradeoff between generalization and performance. Increasing the amount of hidden layers improves the results obtained by the model, but only until an optimal value.

Experimental setup:

- **Network:** MLP with *x* hidden layers (*x* is to be determined).
- **Number of units in hidden layers:** 300.
- **Image size**: 4x4 pixels.
- **Training duration:** 20 epochs.



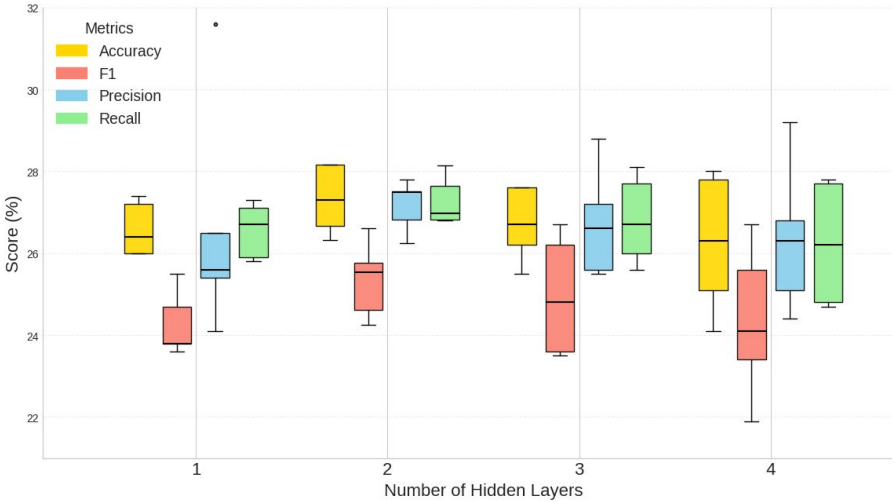
Note: The MLP's complexity depends on both h. We fix the number of units to only tune complexity with one (hyper)parameter.

Hidden layers

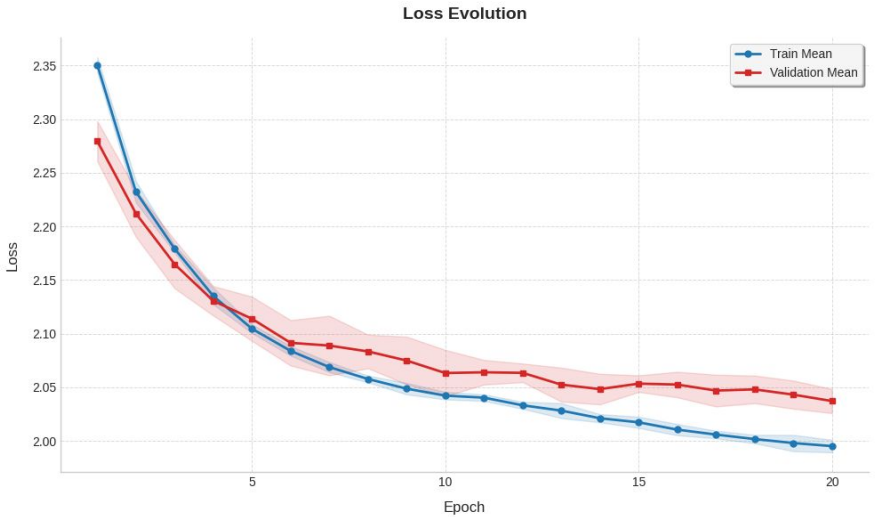
Decision: We choose to use 2 hidden layers. The loss curves for this configuration indicate robust generalization.

- **HYPOTHESIS:** ✔

The results peak at 2 hidden layers. The models with 3 or 4 hidden layers not only underperform on average but also exhibit larger variance, suggesting that the increased complexity is leading to overfitting.



Number of hidden layers: 2



4. Learning + SVM

Learning + SVM

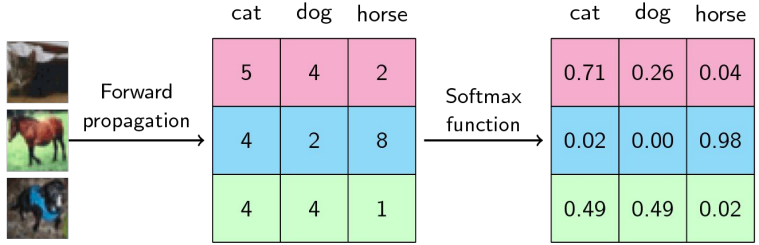
We train our networks using the Cross-Entropy loss: the output is implicitly combined with the Softmax function to produce a vector of predicted probabilities for each category. Hence, the **classification** process is **simple**: given an image, the model predicts the class associated to the highest entry in the output vector.

Last week we observed the SVM classifier achieved the best results. We want to compare both approaches.

➢ **Hypothesis**: Feeding the learnt features of the model's penultimate layer to a SVM classifier improves the performance compared to just computing the *argmax* of the model's output.

Experimental setup:

- **Network:** MLP with 2 hidden layers.
- **Number of units in hidden layers:** 300.
- **Image size**: 4x4 pixels.
- **Training duration:** 20 epochs.
- **Classifier:** SVM with C = 1 and RBF kernel (our last week's best).

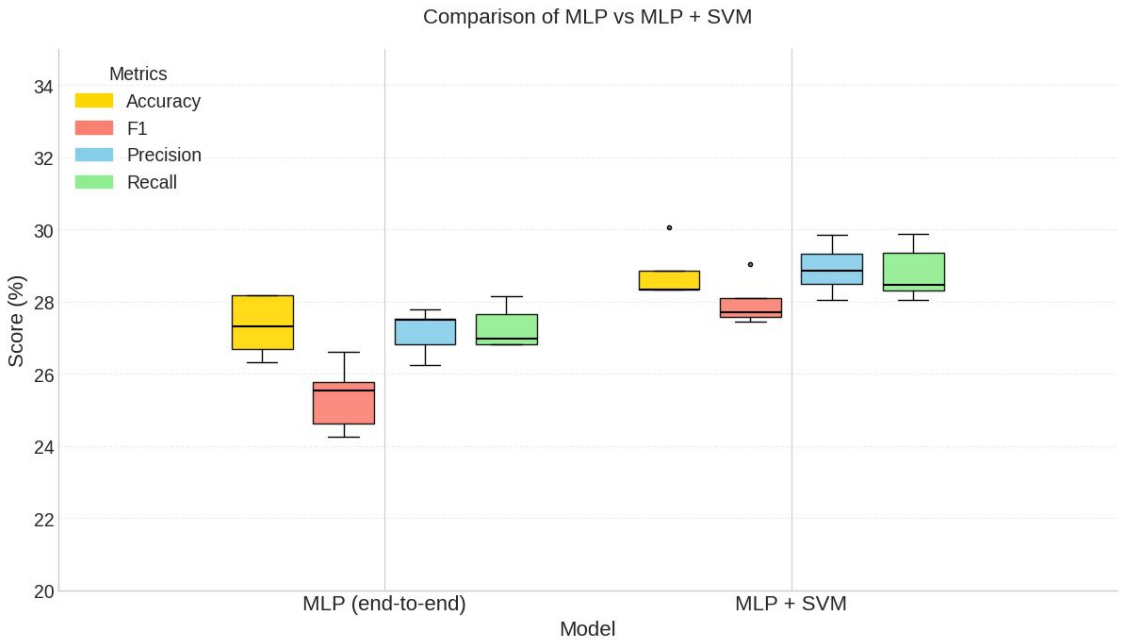


Learning + SVM

- **HYPOTHESIS:** ✔

We observe a consistent improvement across all metric when using the SVM classifier. In particular, we increase accuracy ~1.5%, reaching ~29%.

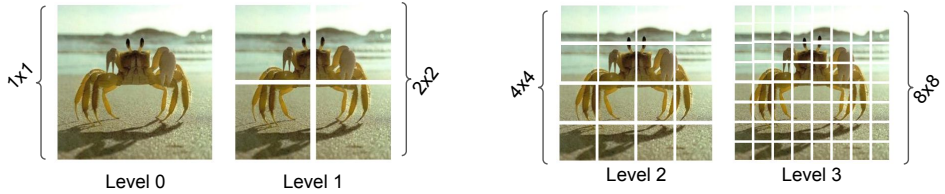
Furthermore, the model with the SVM classifier presents lower variance (in mean), suggesting greater stability.



Decision: Integrate SVM in the model.

5. Adding patches

Adding patches



We partition images into fixed grid patches (like we did with spatial pyramids last week). This **forces** the **MLP** to **process local regions** explicitly, keeping, hence, a degree of spatial information in the learnt features. We consider 3 levels of partition, the same as last week. In the *x* level, images are divided into 2^xx2^x patches.

➢ **Hypothesis**: Performance increases with the number of patches. At some point, though, global context is lost (too detailed patches). Hence, the best metrics are achieved at an intermediate level.

Experimental setup:

- **Network:** Same MLP used in the previous experiment (except image size).
- **Image size:** For level x, (224/2^x)x(224/2^x) pixels (224 is the size in the baseline).
- **Classifier**: SVM with C = 1 and RBF kernel.

Methodology: We classify full images by summing the predicted probabilities of their individual patches (soft voting) to determine the final label. Crucially, we enforce strict separation between training and testing data: images are split before being divided into patches, ensuring information from training images does not *leak* into the test set.

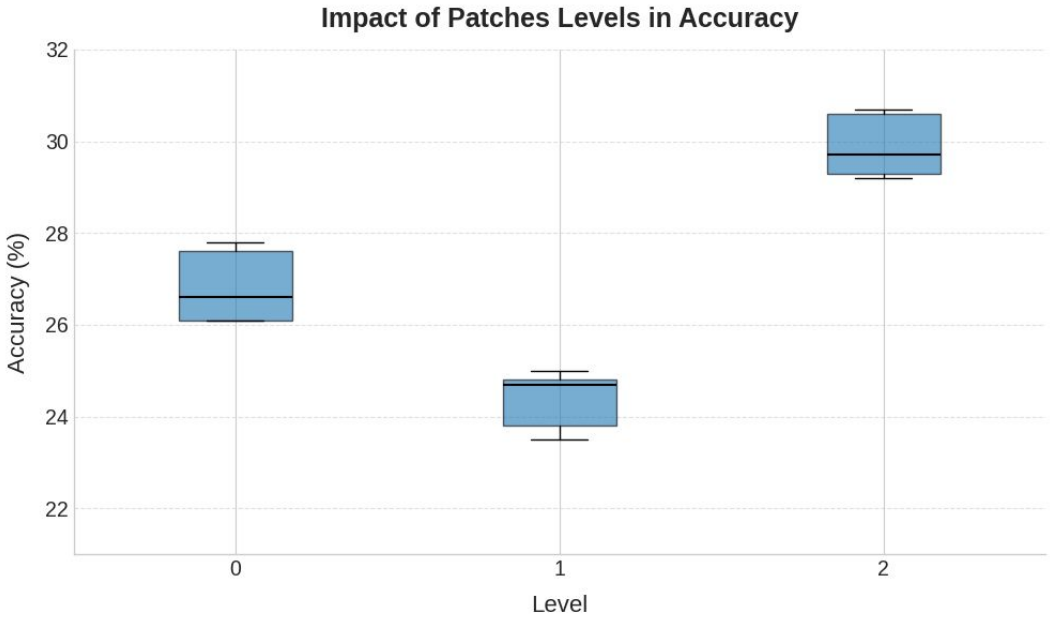
Adding patches

Note: Reducing image resolution could also improve the model in the pointed direction. However, we wanted to strictly isolate the effect of partitioning images in patches: we have seen it is beneficial on its own.

❖ **HYPOTHESIS: ✗**

Although the accuracy peak is reached at level 2, the performance does not increase for level 1. It loses global context provided by level 0, without gaining rich spatial information.

The results imply that analyzing smaller patches could improve the features extracted. We limit the scope to level 2, though, due to the computational cost associated. However, the patching strategy proves to be useful, with a net accuracy improvement of ~3%.



Decision: Use patches from level 2 partition.

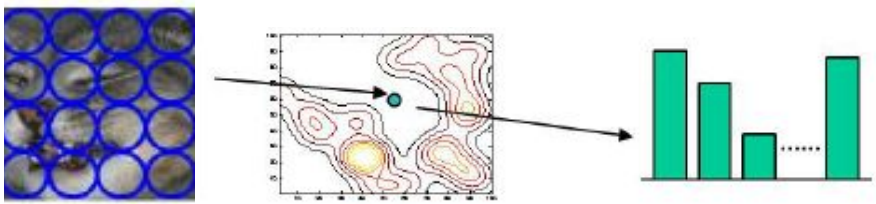
6. Patches + MLP + Fisher Vectors + SVM

Patches + MLP + Fisher Vectors + SVM

We combine this week's approach with last one's by extracting the features of the penultimate layer of the MLP and using them as the dense descriptor (replacing Dense-SIFT). The rest of the pipeline is the same as last week's best configuration. To treat our MLP model as a descriptor, we use the patches approach.

➢ **Hypothesis**: By capturing the statistical distribution of the learnt descriptors (from patches), we improve the results obtained with the previous patch-based voting classification strategy.

Experimental setup:



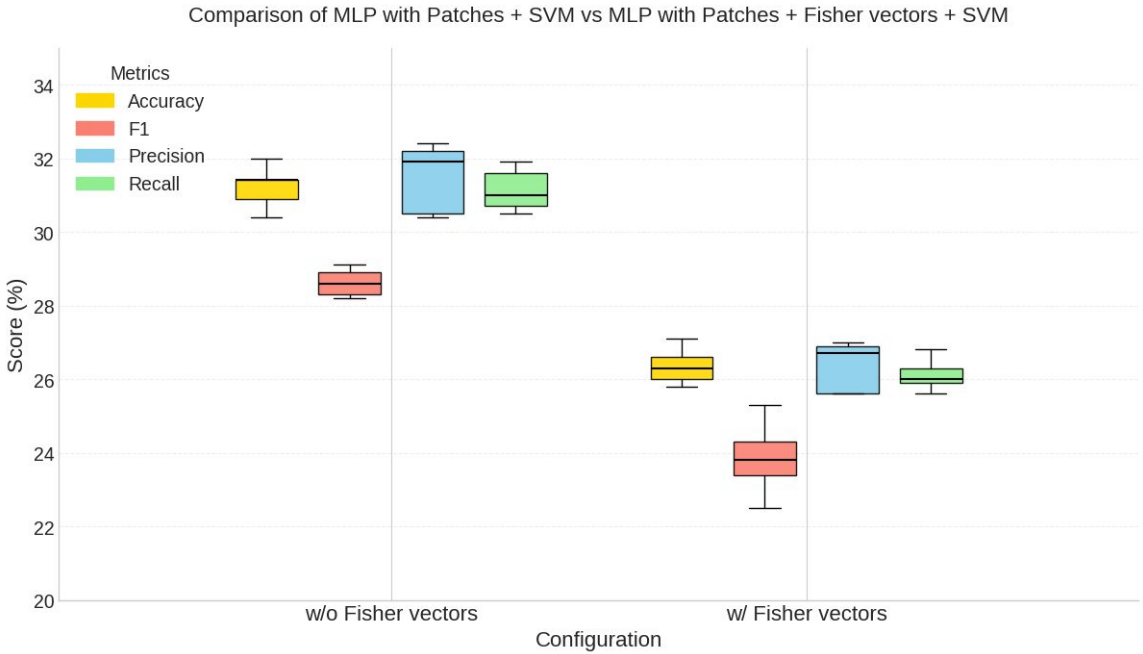
- **Network:** MLP with 2 hidden layers.
- **Number of units in hidden layers:** 300.
- **Image size:** $(224/2^2)$x$(224/2^2)$ pixels (level 2).
- **Training duration:** 20 epochs.
- **Dimensionality Reduction:** PCA (32 components).
- **Encoder:** Fisher Vectors (GMM with 25 components).
- **Classifier:** SVM with C = 1 and RBF kernel.

Patches + MLP + Fisher Vectors + SVM

❖ **HYPOTHESIS: ✗**

Contrary to our beliefs, encoding the learnt features with Fisher Vectors degrades the performance. All metrics decrease (accuracy drops by ~12%).

The features given by the MLP are already discriminative by themselves. Re-encoding them discards the semantic structure learnt by the network and the spatial consensus in the patch-level classification.



Decision: Discard this approach.

Note: Fisher Vectors are intended for dense local descriptors, and in our previous experiment we saw it would be interesting to use finer patches. Investigating this approach with improved partitions remains a promising direction for future work.

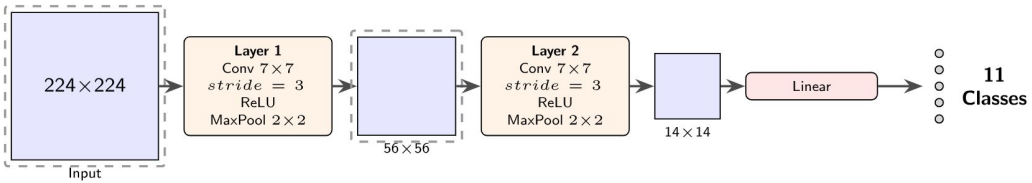
7. CNNs

CNNs

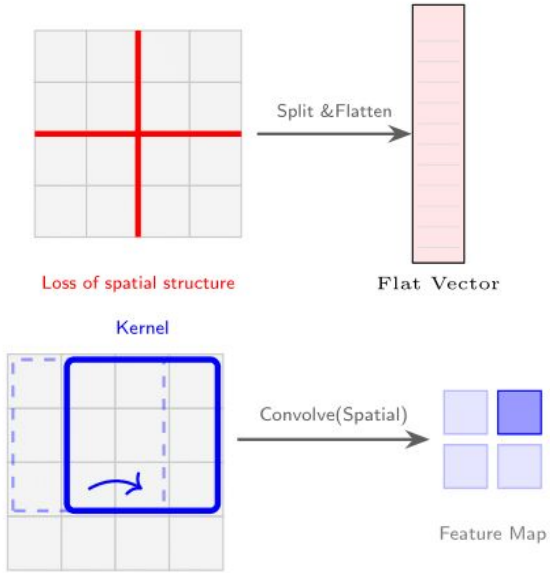
In the last two experiments, we divided images into patches as an attempt to capture spatial information. Nonetheless, MLPs still treat pixels independently within each patch. To fully address this issue, we use a Convolutional Neural Network (CNN). This type of network uses learnable **space-invariant filters** (kernels) to extract features from small, overlapping neighborhoods, preserving the spatial integrity of the image.

➢ **Hypothesis**: CNNs outperform MLPs (in our specific context).

Experimental setup *(CNN's architecture)*:



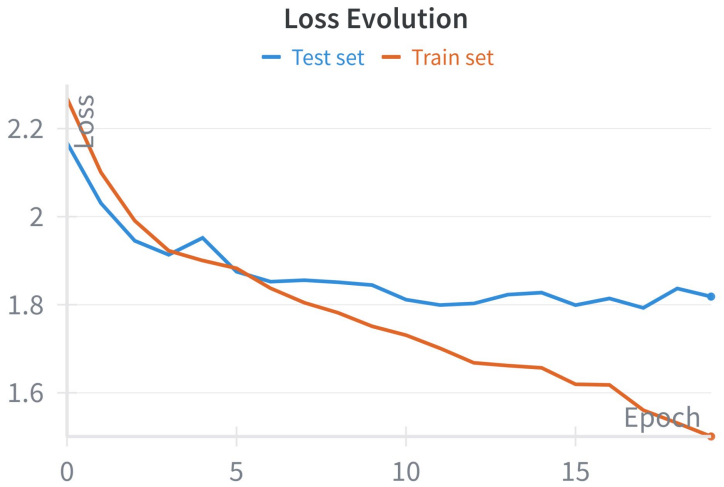
Note: We just replace the linear layers of the baseline model with convolutional blocks and add a fully connected layer at the end to combine all the spatial information learnt. Optimizing the architecture of the CNN is outside the scope of this week's assignment.

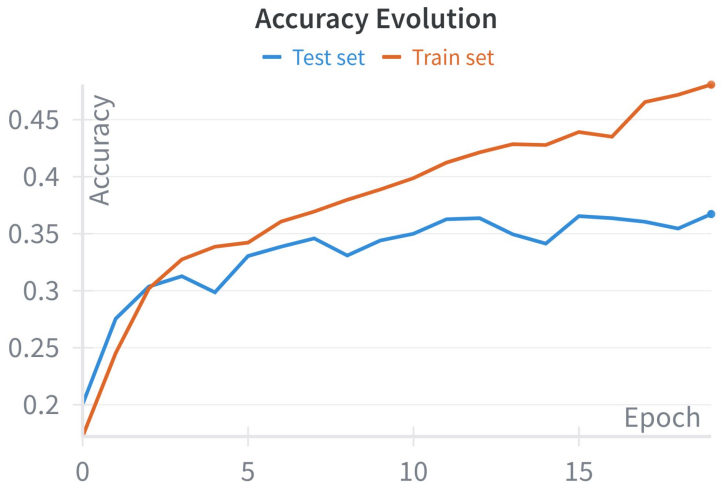


CNNs

- **HYPOTHESIS: ✔**

The loss curves reveal a generalization gap. While the training loss decreases monotonically, the test one plateaus. Despite that, though, the model reaches ~36.7% of accuracy, a massive improvement over this week's baseline (22.8%). This confirms that even an unoptimized CNN can perform better than an MLP.





8. Conclusions

Experiments Summary

- The baseline MLP gives far lower accuracy than last week's best model (22.8% vs 44.5%).
- We improve the baseline to 27.3% by reducing the input image size to 4x4 pixels.
- Using SVMs provides a slight boost to 27.6%.
- Classifying images by patches provides another significant upgrade to 31.7% of accuracy.
- For us, Fisher Vectors impact negatively the results, downgrading the accuracy to 29.5%.
- The CNN architecture achieves the best results, giving a final accuracy of 36.7%.

Metrics on the test set

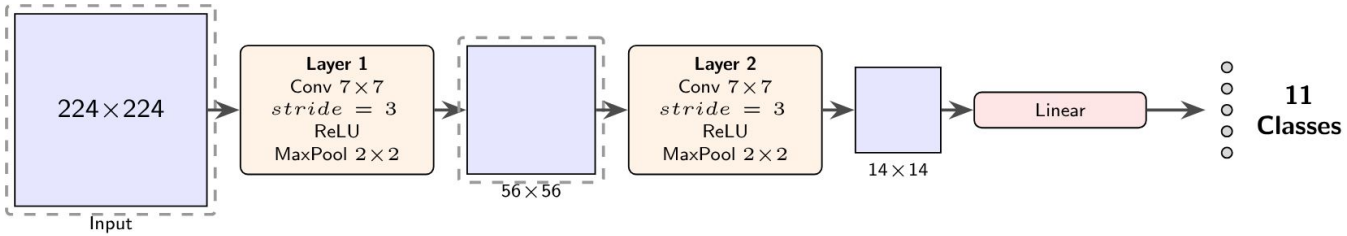
		Accuracy	F1	Precision	Recall
Experiment	1	22.8	22.2	28.3	22.8
	2	27.3	24.9	27.3	27.3
	3	27.3	24.9	27.3	27.3
	4	27.6	26.9	27.5	27.6
	5	31.7	29.9	31.6	31.7
	6	29.5	27.9	29.2	29.5
	7	**36.7**	**36.4**	**37.1**	**36.7**

Best Model: CNN

The CNN emerges as the superior architecture, achieving a 36.7% accuracy. This represents a **massive improvement** over the baseline MLP (22.8%) and surpasses all the other experimental results. This confirms that exploiting spatial locality with learnable kernels is hugely effective. The balanced Precision and Recall scores also demonstrate that the model learns robust features.

Metrics on the test set

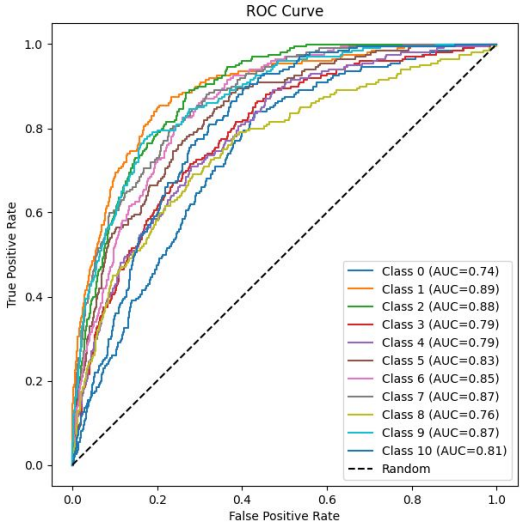
	Accuracy	F1	Precision	Recall
Score (%)	36.7	36.4	37.1	36.7

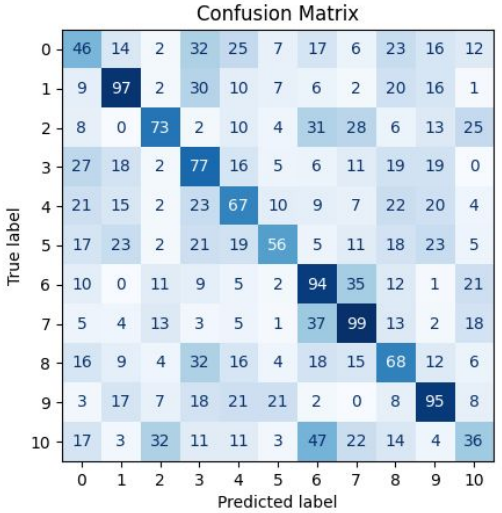


Best Model: CNN

The model shows acceptable classification performance, as all the curves are above the diagonal (random guessing) in the ROC Curve.

Class 1 is the best performing category (AUC = 0.89), closely followed by Class 2 (0.88). Class 0 yields the lowest performance (AUC = 0.74). An anomaly occurs for Class 10, predicting Class 6 more than itself.





Project Summary

Best model's metrics on the test set

	Accuracy	F1	Precision	Recall
Week 1	**44.5**	**43.9**	**44.0**	**44.5**
Week 2	36.7	36.4	37.1	36.7
Week 3	?			
Week 4				