IMAGE CLASSIFICATION

Task 3: Convolutional Networks

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

MCV MASTER IN COMPUTER VISION Barcelona

UAB Universitat Autònoma de Barcelona · Universitat de Barcelona · UOC Universitat Oberta de Catalunya · UPC Universitat Politècnica de Catalunya BarcelonaTech · upf. Universitat Pompeu Fabra Barcelona

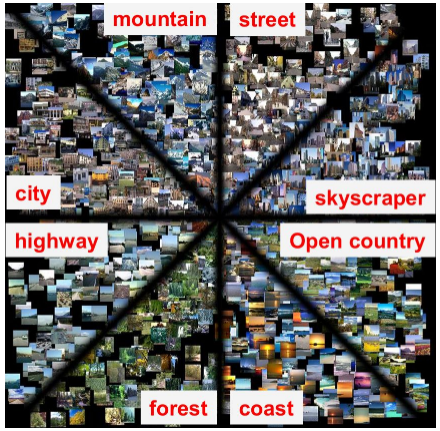
0. New dataset

New dataset

** We use 4-fold cross-validation for quantitative results.*
For our qualitative analysis, we take samples from Fold 1.

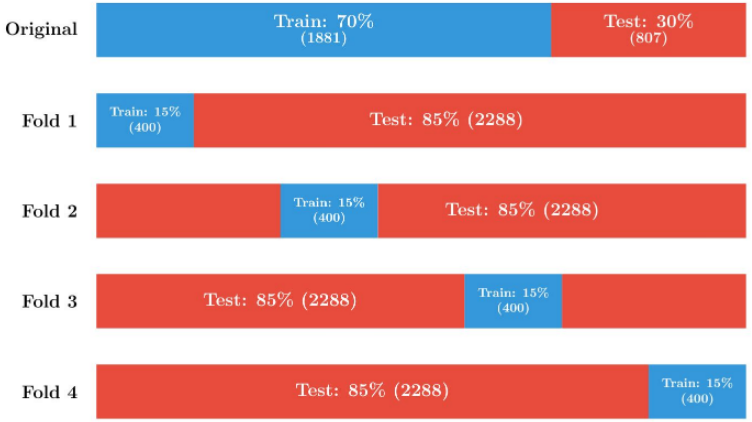
In this assignment we received a new dataset. Its 8 classes are far more specific than the previous 11 ones.

There is a *large* version of the dataset, and 4 *small* ones (corresponding to different folds*).



Overview of new dataset.

Ambiguity in previous dataset. It is *sports and leisure* despite existing the *home or hotel* class.



Data partitioning: Comparison of the *large* split versus the four *small* (few-shot) training folds.

New dataset - Analysis

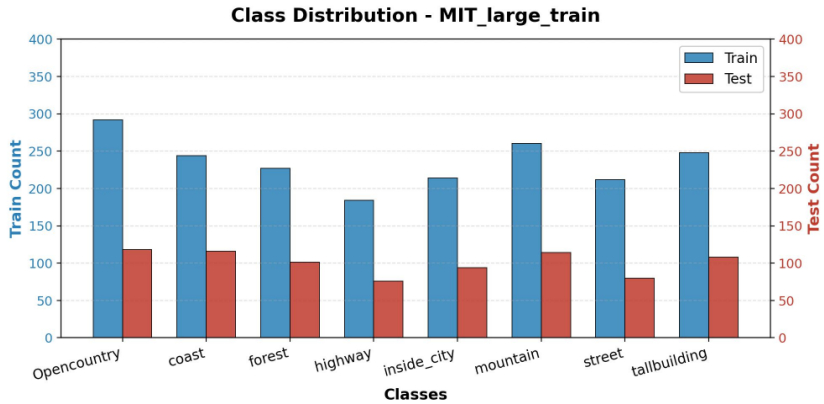
Note: The class distribution is identical across all 4 folds.

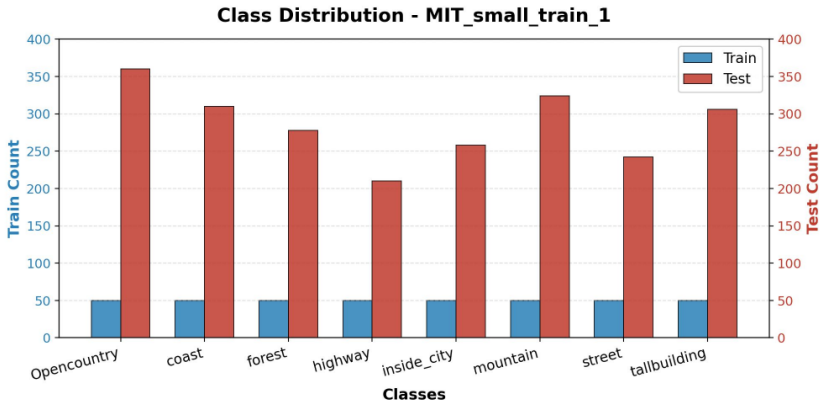
Large split:

- The **class proportions** are **preserved** across the training and test partitions.
- Significant **class imbalance** (*Opencountry* vs. *highway*) risks biasing models towards majority classes.

Small splits:

- The **test set** shares the **original class distribution**, ensuring models are evaluated realistically.
- **Training sets** are strictly **balanced** (50 images per class), removing the potential frequency bias.





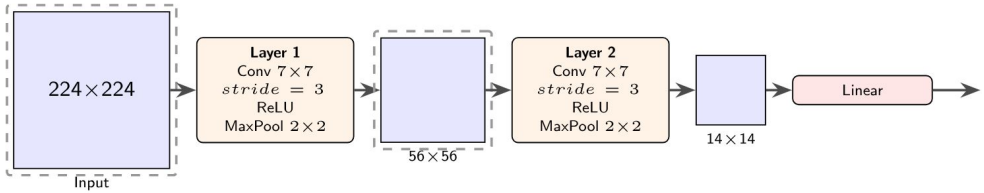
New dataset - Updated results

We evaluate our previous best configurations:

<u>Week 1:</u>

- **Local descriptor**: Dense SIFT (*stride* = *scale* = 8).
- **Fisher vectors** with 25 components.
- **Dimensionality Reduction:** PCA (32 components).
- Histogram normalization.
- **Spatial pyramid** with 0 levels (full images).
- **Classifier**: SVM with C = 1 and RBF kernel.

<u>Week 2:</u> CNN



Metrics on the test set (previous dataset)

	Accuracy	F1	Precision	Recall
Week 1 (%)	**44.5**	**43.9**	**44.0**	**44.5**
Week 2 (%)	36.7	36.4	37.1	36.7

Metrics on the test set (*large* dataset)

	Accuracy	F1	Precision	Recall
Week 1 (%)	**88.7**	**89.1**	**89.2**	**89.0**
Week 2 (%)	69.3	69.7	71.3	69.4

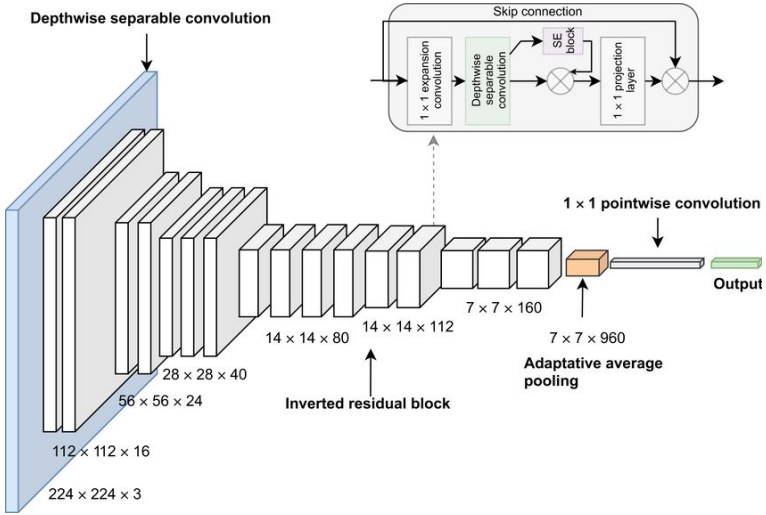
Note: Just transitioning to the new dataset doubles our best results.

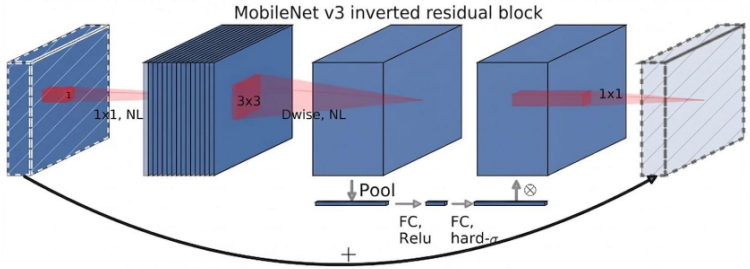
1. Our baseline: MobileNetV3

MobileNetV3 - Architecture

MobileNetV3 is a deep convolutional network that balances lightness (~5.4M parameters) and high accuracy. Its core component is the **Inverted Residual Block**:

1. Input channels are **expanded** with **1x1 convolutions**.
2. **Depthwise convolutions** reduce computational cost by applying a **single filter per channel** separately.
3. The **Squeeze-and-Excitation (SE)** mechanism helps the network **focus** on what matters. It uses global pooling (*squeeze*) and FC layers (*excite*) to produce an **importance weight** (from 0 to 1) to each channel.
4. A final **1x1 convolution compresses** the rich features to a lower channel count for the next block.





Andrew Howard et al., "Searching for MobileNetV3" in ICCV, 2019.

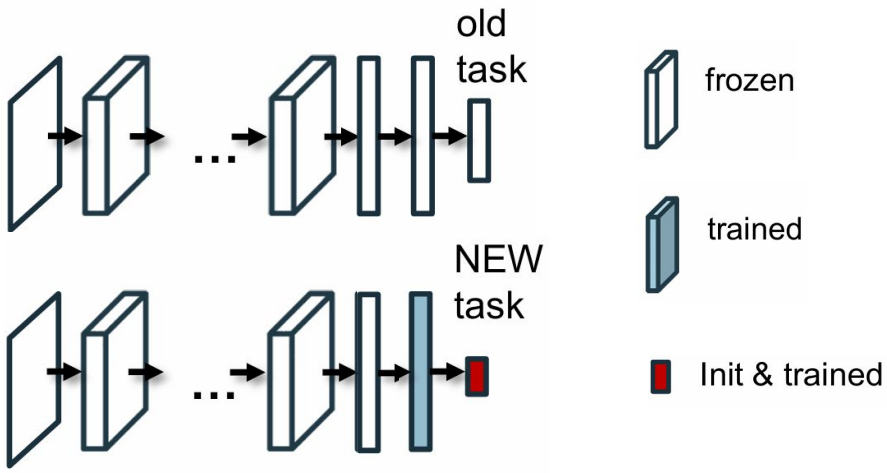
MobileNetV3 - Large dataset

A natural first question arises: *How does this model compare to our previous approaches*? To answer this, we leverage **transfer learning**, initializing with ImageNet weights and **fine-tuning** only the **final layer**.

➢ **Hypothesis**: A CNN capable of strong performance on ImageNet (with 1.28M images) has learned rich semantic information. Adapting it to our data should surpass our previous results.

Experimental setup:

- **Image size**: 224x224 pixels.
- **Training duration**: 20 epochs.
- **Learning rate**: 0.001.
- **Batch size**: 256.
- **Optimizer**: Adam.
- **Loss function:** Cross-entropy.



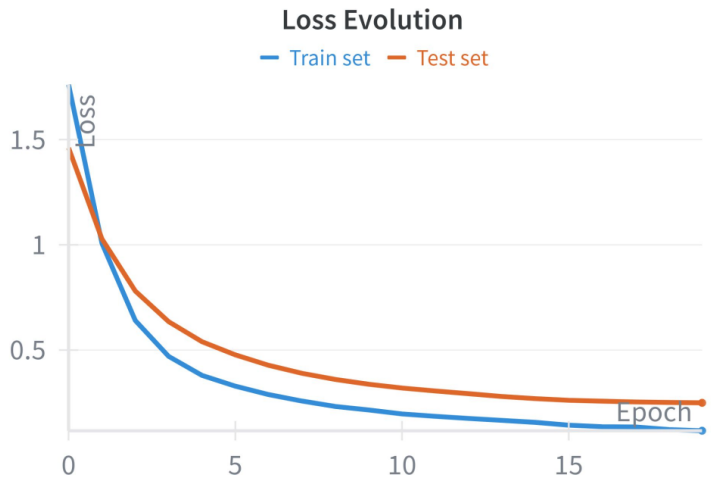
MobileNetV3 - Large dataset

❖ **HYPOTHESIS:** ✔

The model achieves a remarkable 92.4% of accuracy, confirming that semantic features learned from ImageNet transfer effectively to the *large* dataset. Furthermore, the loss curves demonstrate stable convergence with a minimal generalization gap, indicating robust learning without significant overfitting.

Metrics on the test set (*large* dataset)

	Accuracy	F1	Precision	Recall
Week 1 (%)	88.7	89.1	89.2	89.0
Week 2 (%)	69.3	69.7	71.3	69.4
Current (%)	**92.4**	**92.7**	**92.8**	**92.6**



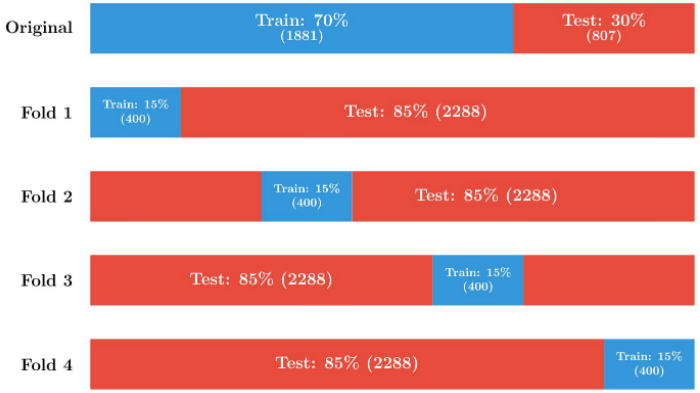
2. Restricting the data

Addressing data scarcity

While MobileNetV3 has proved to be superior to our previous approaches with a simple fine-tuning (reaching >92% accuracy), **real-world scenarios** often **suffer** from **data scarcity**. When we restrict access to a small fraction of the training data, we face the **risk** of the model **failing** to **generalize**.

Therefore, what would happen if we transition to the *small* versions of the dataset? We might wonder:

- Does the model maintain its robustness?
- Can visual diagnostics reveal what happens inside it?
- Furthermore, is there a way to mitigate the data issue?



After that, we propose the following enhancements:

- Modify the architecture to reduce its complexity.
- Improve learning with stabilization techniques.
- Perform a search to optimize the model hyperparameters.

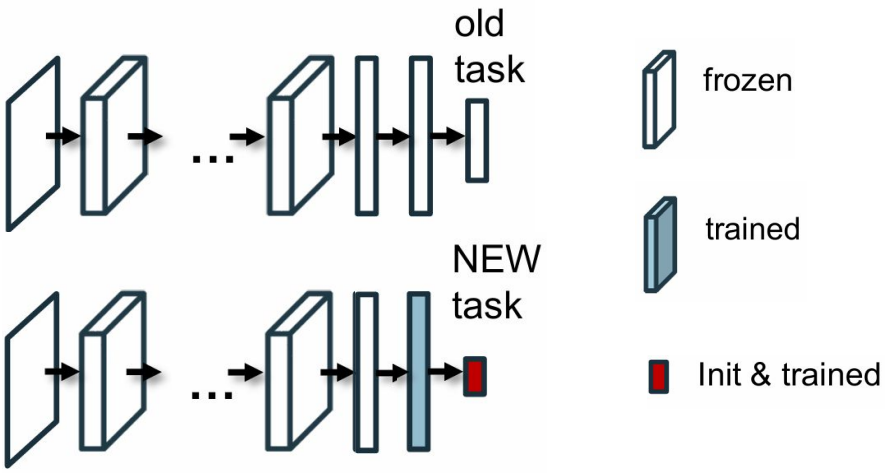
MobileNetV3 - Small dataset

To quantify the performance drop caused by **limited data**, we repeat our previous experiment using the *small* datasets. We maintain the same configuration (pretrained ImageNet weights) to ensure a fair comparison.

➢ **Hypothesis**: Due to the scarcity of samples (400 images), we expect the model to overfit rather than generalize. We think the network is essentially too complex for the amount of data available.

Experimental setup:

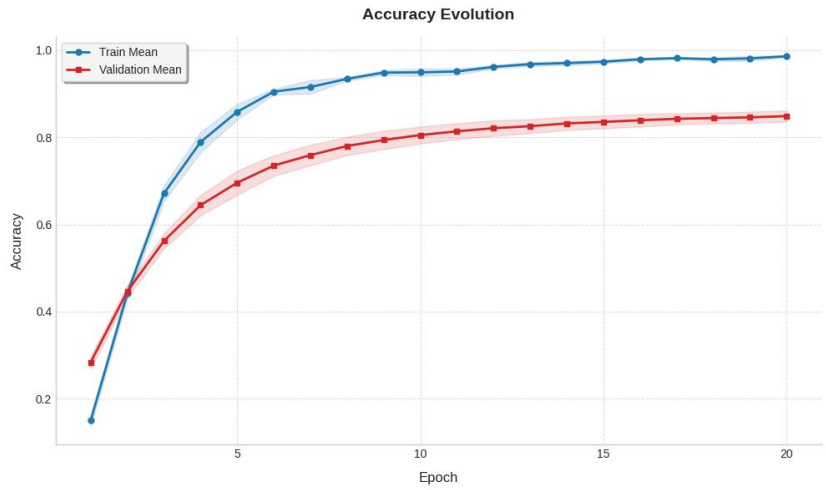
- **Image size**: 224x224 pixels.
- **Training duration**: 20 epochs.
- **Learning rate**: 0.001.
- **Batch size**: 256.
- **Optimizer**: Adam.
- **Loss function:** Cross-entropy.

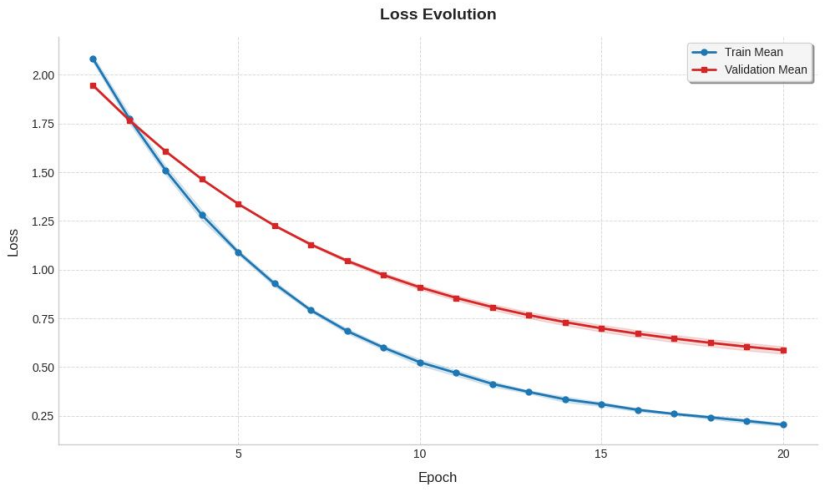


MobileNetV3 - Small dataset

❖ **HYPOTHESIS:** ✔

While the model achieves a respectable 84.9% accuracy, the training accuracy approaches 100%. The diverging loss curves highlight a significant generalization gap. This confirms the model, having too much capacity for the *small* datasets, is memorizing the training examples rather than learning robust features.



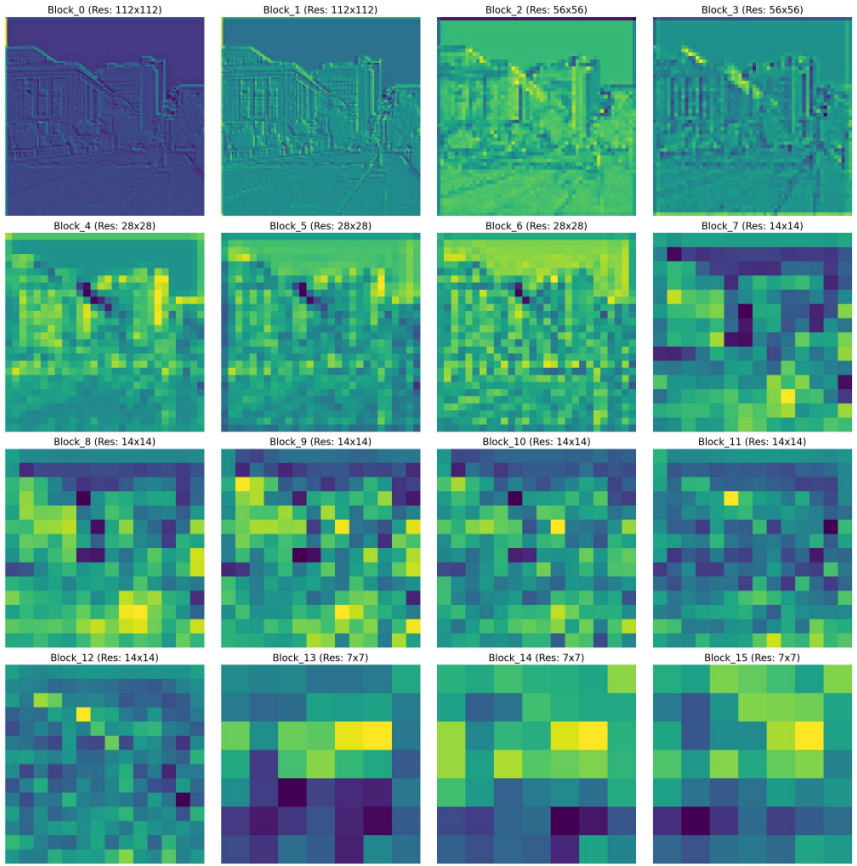


Activation maps

Visualizing the internal **activation maps** across MobileNetV3 layers exposes the hierarchical nature of the network's feature extraction process.

- **Early blocks:** They act as low-level filters, capturing primitive features (e.g., edges).
- **Deep blocks:** The network aggregates these primitives into complex, abstract patterns representing semantic concepts.

While these maps confirm the network is learning, their abstraction makes them **difficult** to **interpret**. To identify the specific pixels driving the decisions, we require a specialized technique: Grad-CAM.



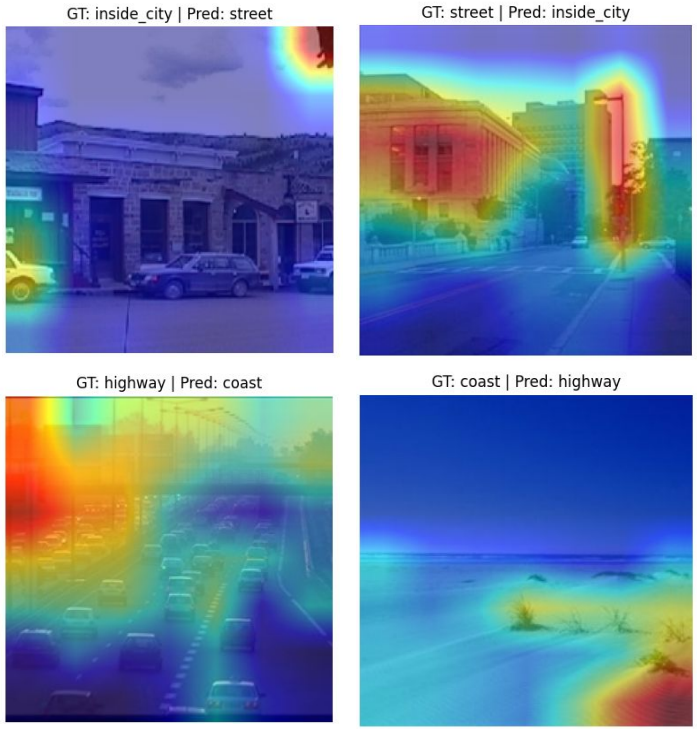
Grad-CAM

We utilize **Grad-CAM** to visualize the decision regions for the **predicted** classes. The heatmaps reveal the model relies on incidental correlations rather than scene semantics.

In the top images, the model ignores the global scene and fixates on specific objects (a car, a building and traffic poles).

In the bottom samples, the model fails to capture the beach or road. It just focuses on local information (image corners).

In general, the model **lacks** sufficient **data** to learn **invariant features**. To force the network to look at the whole picture and reduce this fragility, we introduce data augmentation.



Ramprasaath R. Selvaraju et al., "Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization" in ICCV, 2017.

Data augmentation

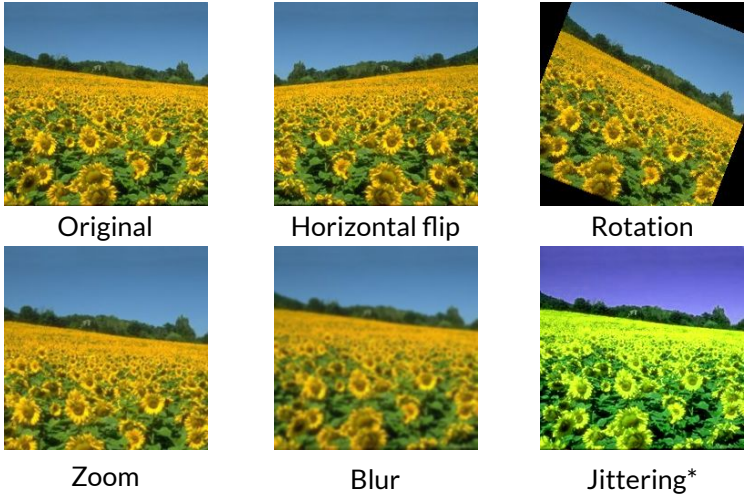
** Jittering refers to random shifts on color properties like brightness, contrast… Here, its effect is exaggerated for illustrative purposes.*

Data augmentation allows us to artificially **expand** the **training set** by generating modified versions of our images (we increase the data by a factor of six). This technique directly combats data scarcity by forcing the model to **learn invariant features** (ensuring it recognizes objects regardless of their orientation, scale…).

➢ **Hypothesis**: We expect all augmentations to outperform the baseline. Furthermore, we anticipate the combination of them yields the highest accuracy, providing strong regularization against overfitting.

Experimental setup:

- **Image size**: 224x224 pixels.
- **Training duration**: 20 epochs.
- **Learning rate**: 0.001.
- **Batch size**: 256.
- **Optimizer**: Adam.
- **Loss function:** Cross-entropy.

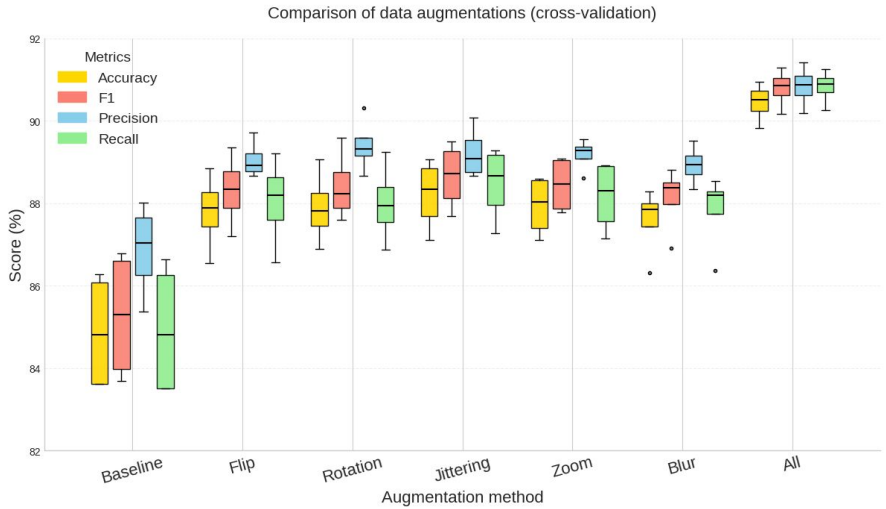


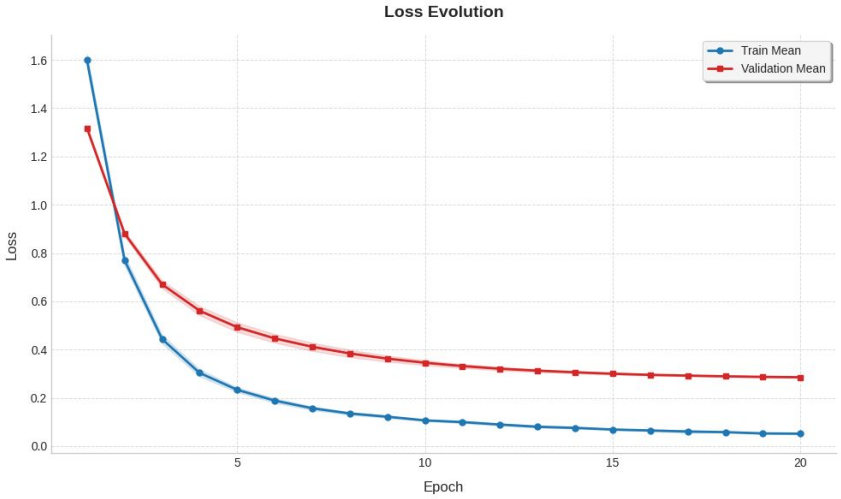
Data augmentation

** The discussion of the loss curves refers to the results of the previous experiment.*

❖ **HYPOTHESIS: ✔**

Individual transformations improve the baseline from ~85% accuracy to ~88%, and their combination boosts it to ~91%. Moreover, the configuration maintains low variance, indicating consistent performance. This is confirmed by the loss plot; the generalization gap is reduced, and the validation loss is approximately halved*.





Removing blocks

** From this experiment onwards, we adopt Early Stopping (patience of 5 epochs) to accelerate development, discarding unpromising configurations to focus our resources only on viable models.*

We create lighter versions of MobileNetV3 by removing feature blocks from the top down. This **reduces** model **capacity**, ensuring it **learns general patterns** rather than overfitting to specific training examples. We fine-tune the full classification head to adapt to the changing input dimensions of the pruned blocks.

➢ **Hypothesis**: We believe that removing blocks improves generalization by aligning the model's capacity to the size of the small dataset. Nevertheless, excessive pruning carries the risk of underfitting.

Experimental setup:

- **Image size**: 224x224 pixels.
- **Training duration**: 20 epochs*.
- **Learning rate**: 0.001.
- **Batch size**: 256.
- **Optimizer**: Adam.
- **Loss function:** Cross entropy.
- **Data augmentation**.

Index	Type	Channels	Resolution
0	Conv2dNormActivation	16	112x112
1	InvertedResidual	16	112x112
2	InvertedResidual	24	56x56
3	InvertedResidual	24	56x56
4	InvertedResidual	40	28x28
5	InvertedResidual	40	28x28
6	InvertedResidual	40	28x28
7	InvertedResidual	80	14x14
8	InvertedResidual	80	14x14
9	InvertedResidual	80	14x14
10	InvertedResidual	80	14x14
11	InvertedResidual	112	14x14
12	InvertedResidual	112	14x14
13	InvertedResidual	160	7x7
14	InvertedResidual	160	7x7
15	InvertedResidual	160	7x7
16	Conv2dNormActivation	960	7x7

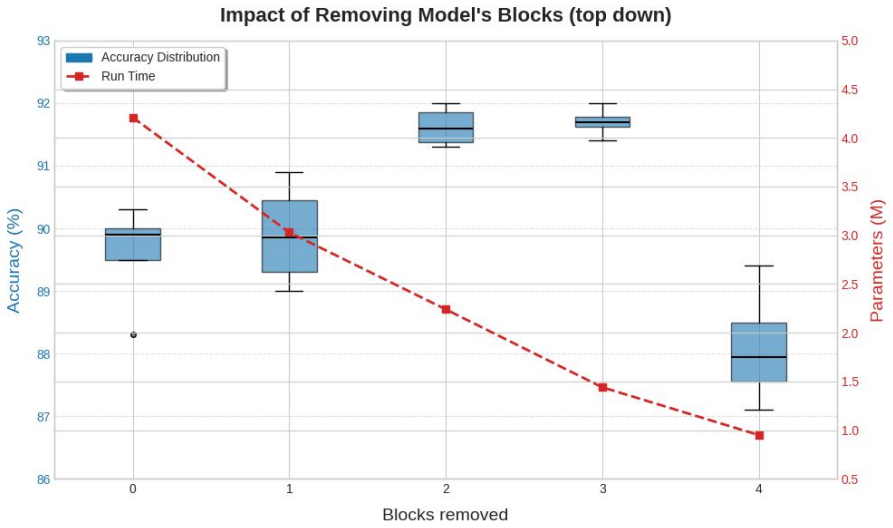
MobileNetV3's Feature Extractor

Removing blocks

- **HYPOTHESIS:** ✔

Removing 2 or 3 blocks provides the highest accuracy, confirming that the full model was over-parameterized for this dataset. Beyond this point (4 blocks), the model loses essential capacity, causing a sharp performance drop.

Decision: We select the configuration in which 3 blocks are removed. It achieves the same accuracy as the 2-block version but with higher efficiency (fewer parameters) and lower variance (it is more robust).

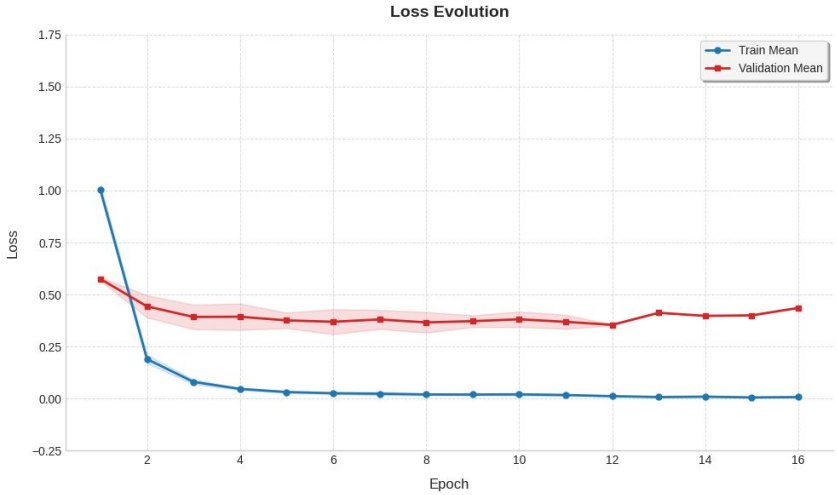


Removing blocks

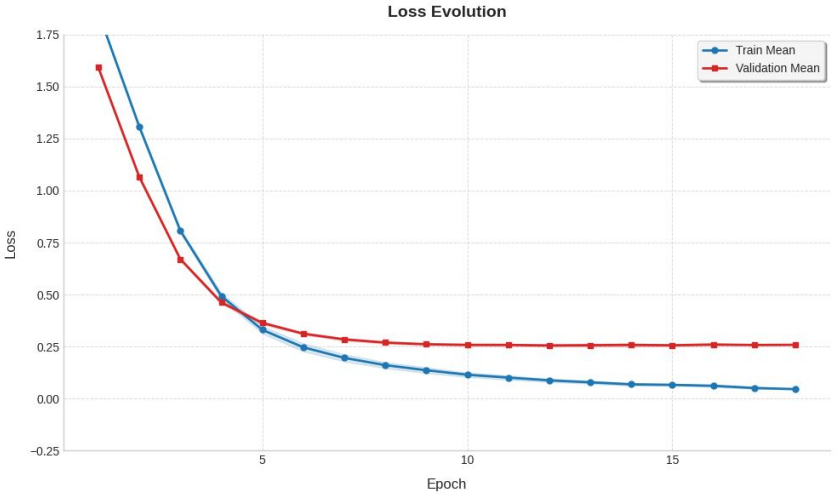
- **HYPOTHESIS:** ✔

The loss curves validate our choice: the model struggles to generalize when no blocks are removed. Removing three blocks provides both a reduction of the generalization gap and smoother convergence.

Blocks removed: 0



Blocks removed: 3



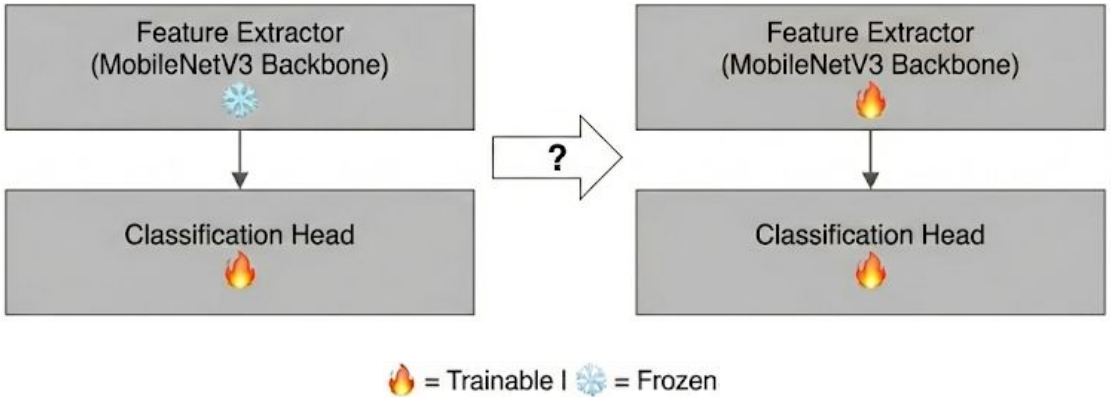
What to train?

Until now, we have **only trained** the **classification head**, keeping the backbone frozen. However, what would happen if we **fine-tuned** the **whole model**? Would the results improve?

➢ **Hypothesis**: Training all the network's parameters leads to overfitting. With the massive increase in trainable parameters, the model is complex enough to memorize the entire training set.

Experimental setup:

- **Image size**: 224x224 pixels.
- **Training duration**: 20 epochs.
- **Learning rate**: 0.001.
- **Batch size**: 256.
- **Optimizer**: Adam.
- **Loss function:** Cross entropy.
- **Data augmentation**.
- **Removed blocks**: 3.



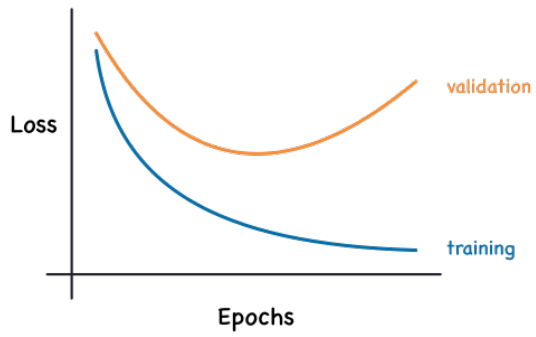
What to train?

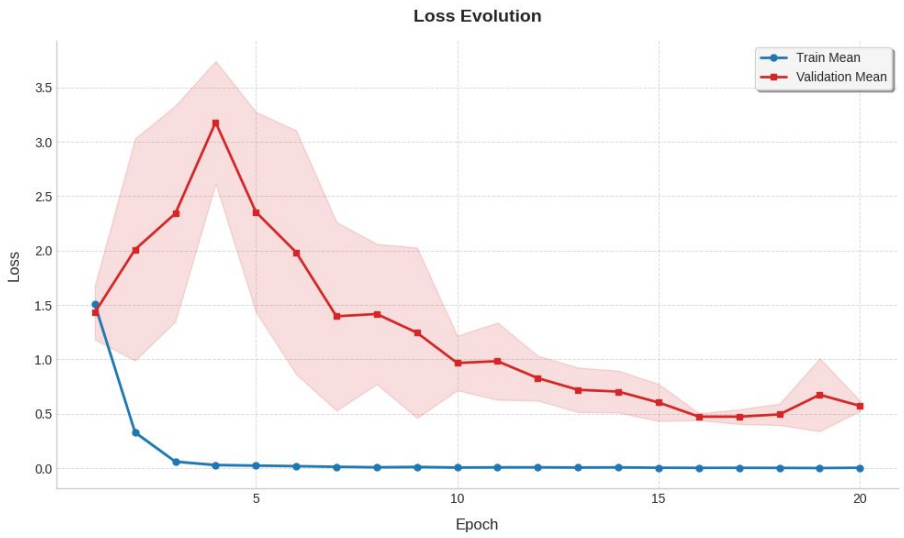
- **HYPOTHESIS:** ✔

Contrary to standard overfitting (top image), we observe severe training instability (spike at Epoch 4). Although the validation loss eventually stabilizes, it does not converge to a competitive value.

Applying large updates (learning rate of 0.001) to the *pre-trained* backbone destroyed ImageNet weights before they could be adapted to the new dataset.

Decision: Due to this instability and the resulting poor generalization compared to the frozen backbone approach, we discard full fine-tuning.





Hyperparameter search

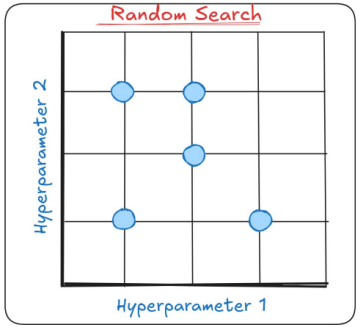
We now fix the architecture and optimize the training configuration to maximize performance.

Stage 1: Exploration (Random Search):

- **Goal:** Identify promising regions in the hyperparameter space.
- **Duration:** 20 runs.
- The Top 5 performing configurations are selected to initialize the next stage.

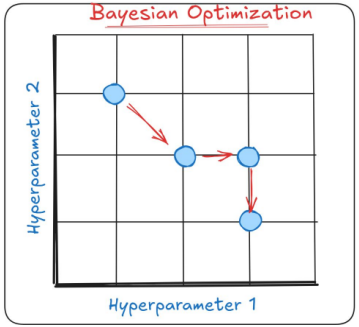
Stage 2: Exploitation (Bayesian Optimization):

- **Goal:** Fine-tune the best candidates to find the optimal configuration.
- **Duration:** 50 runs.
- It uses a probabilistic model to learn from past results. (Conceptual Explanation).



Hyperparameter Space:

- **Optimization:** Optimizer, Learning Rate, Momentum.
- **Training Loop:** Batch Size, Epochs.
- **Regularization:** Dropout, Weight Decay.



Hyperparameter search

Top 5 configurations obtained in Random Search:

Full Random Search Space:

- Batch Size: [32, 64, 128, 256, 512]
- Dropout: [0.0, 0.2, 0.5]
- Learning Rate: [1e-4, 1e-3, 1e-2, 0.1, 0.2, 0.3]
- Momentum: [0.0, 0.2, 0.4, 0.6, 0.8, 0.9]
- Epochs: [10, 50, 100]
- Optimizer: [SGD, RMSprop, Adagrad, Adadelta, Adam, Adamax, Nadam]
- Weight Decay: [0.0, 1e-6, 1e-5, 1e-4, 1e-3]

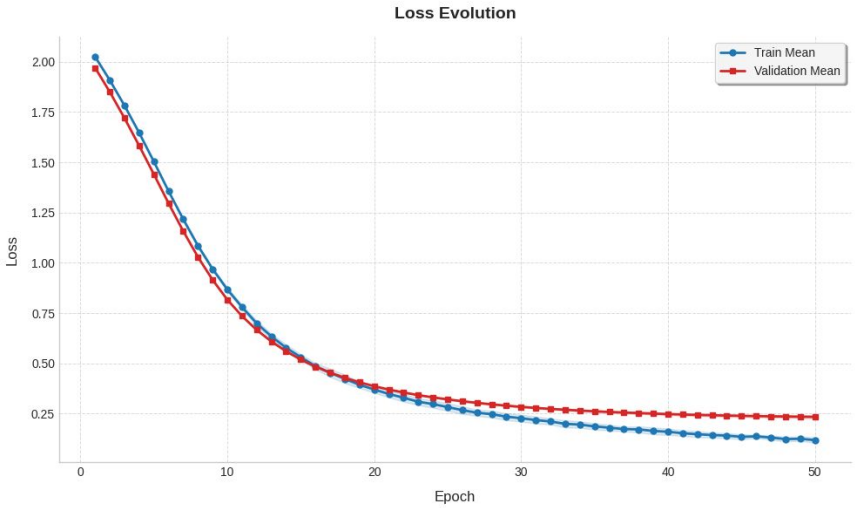
	Batch Size	Dropout	Learning Rate	Momentum	Epochs	Optimizer	Weight Decay	Accuracy (%)
Top 1	64	0.2	1e-3	0.0	50	Adagrad	1e-4	92.1
Top 2	64	0.0	1e-3	0.8	10	Adamax	1e-6	92.1
Top 3	64	0.0	1e-4	0.8	10	RMSprop	1e-3	91.9
Top 4	128	0.2	1e-3	0.8	10	Nadam	1e-4	91.8
Top 5	64	0.2	0.3	0.4	50	Adagrad	1e-6	91.2

Hyperparameter search

Top configuration obtained in Bayesian Optimization:

Batch Size	Dropout	Learning Rate	Momentum	Epochs	Optimizer	Weight Decay
64	0.110	7e-5	0.152	50	Nadam	2.74e-5

The generalization gap from earlier experiments is reduced, confirming the configuration's robustness.



The optimized model achieves 92.4% of accuracy, matching the performance obtained in the first experiment with the *large* dataset. We have successfully compensated for the data scarcity, delivering a model both lightweight and precise.

Metrics on the test set (*small* dataset)

Accuracy	F1	Precision	Recall
92.4 %	92.7 %	92.7 %	92.8 %

3. Conclusions

Experiments summary

Our experiments yielded progressive improvements:

- The most significant gain (+5.6% accuracy) was achieved by introducing data augmentation .
- Optimizing the architecture (removing 3 blocks) contributed a further +1.2% increase.
- Unfreezing the backbone did not improve the results, so we discarded the approach.
- With our hyperparameter search we provide an upgrade of 0.7%, reaching 92.4% of accuracy.

Metrics on the test set (*small* dataset)

		Accuracy	F1	Precision	Recall
Experiment Score (%)	1	84.9	85.3	86.9	84.5
	2	90.5	90.8	90.8	90.8
	3	91.7	92.1	92.2	92.1
	4	91.7	92.1	92.2	92.1
	5	**92.4**	**92.7**	**92.7**	**92.8**

*Top 3 Blocks Removed

Final model

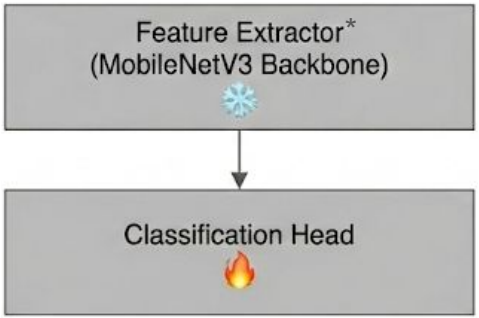
- The optimized MobileNetV3 balances capacity and generalization on the *small* dataset.
- We successfully overcame data scarcity, matching the performance on the *large* dataset.
- The high F1 score (92.7%) indicates the model is not just accurate, but precise and robust.

Metrics of baseline model (*large* dataset)

	Accuracy	F1	Precision	Recall
Score (%)	92.4	92.7	92.8	92.6

Metrics of final model (*small* dataset)

	Accuracy	F1	Precision	Recall
Score (%)	92.4	92.7	92.6	92.8



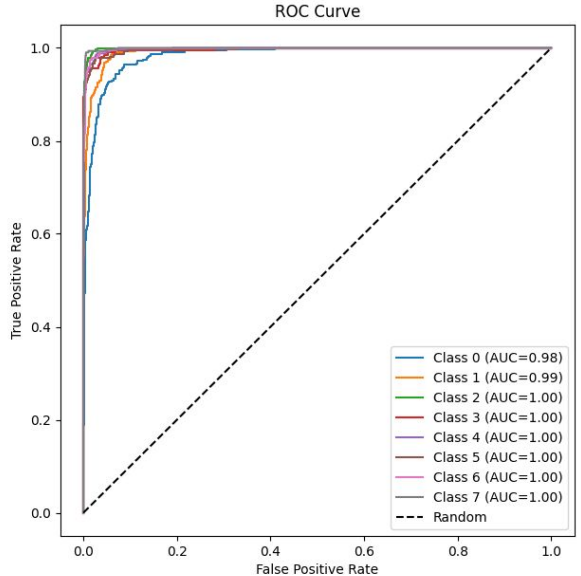
Optimal configuration

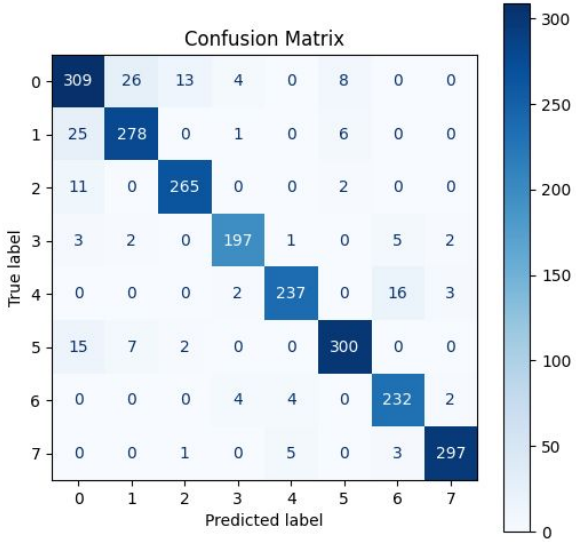
Batch Size	Dropout	Learning Rate	Momentum	Epochs	Optimizer	Weight Decay
64	0.110	7e-5	0.152	50	Nadam	2.74e-5

Error analysis - Small dataset

Our model shows excellent classification performance, as **most categories achieve perfect AUC**.

The major confusion is between *Open Country* and *Coast*. The model misclassifies them around 25 times in each case. Additionally, we observe slight difficulties in distinguishing other similar classes; *Inside City* is treated as *Street* 16 times, and *Mountain* is seen as *Open Country* 15 times.





Class 0	Class 1	Class 2	Class 3
Open Country	Coast	Forest	Highway
Class 4	**Class 5**	**Class 6**	**Class 7**
Inside City	Mountain	Street	Tall Building

4. Extra: ViT

ViT - Swin Transformer

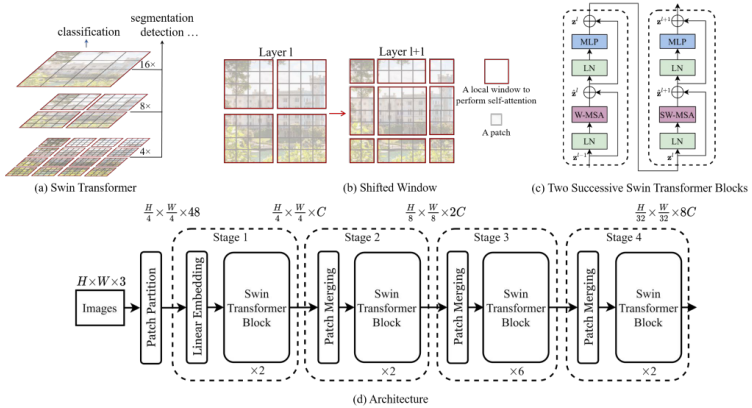
Ze Liu et al., "Swin Transformer: Hierarchical Vision Transformer using Shifted Windows" in ICCV, 2021.

To benchmark against state-of-the-art methodology, we introduce Swin Transformer. Unlike CNNs like MobileNetV3, **Vision Transformers** (ViTs) use self-attention to **capture global dependencies** across the image. Swin's hierarchical structure allows it to be locally efficient (like a CNN) while keeping global context.

➢ **Hypothesis**: The results obtained by Swin transformer will exceed MobileNetV3 metrics.

We evaluate Swin Transformer in two scenarios:

1. *Large* dataset: Swin vs. MobileNetV3 (baseline).
2. *Small* dataset: Swin vs. MobileNetV3 (final model).



Experimental setup

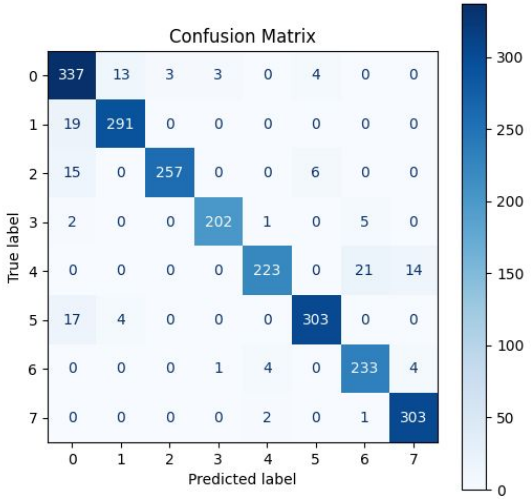
	Large dataset	*Small* dataset
Image size	224x224 pixels	224x224 pixels
Training duration	20 epochs	50 epochs
Learning rate	0.001	7e-5
Batch size	256	64
Optimizer	Adam	Nadam
Loss function	Cross entropy	Cross entropy
Data augmentation	No	Yes

ViT - Swin Transformer

**On the small dataset, when compared with the confusion matrix of our final model.*

❖ **HYPOTHESIS: ✔**

Swin Transformer outperforms our final results in both scenarios, proving that global self-attention enhances generalization even without architecture-specific tuning. Interestingly, the confusion matrix reveals similar error patterns*, suggesting remaining misclassifications are due to dataset ambiguity. We leave a dedicated hyperparameter search to maximize performance as future work.

Metrics on the test set (%)

	Accuracy	F1	Precision	Recall
***Large* dataset**	95.2	95.2	95.3	95.3
***Small* dataset**	94.0	94.0	94.4	93.9