Title of the Final Project

First Author
firstauthor@i1.org

Second Author
secondauthor@i2.org

Third Author
thirdauthor@i2.org

Fourth Author
fourthauthor@i2.org

Abstract

Brief summary of the work developed as well as the main results and contributions.

1. Introduction

Here, the problem to be solved is described (what do we want to do?), the motivation (why is it relevant to do it?), and the goals (what specific objectives are we going to address in order to solve the problem?).

We use the LaTeX format of the CVPR conference. This document can be written either in English or Spanish.

What follows is a tentative approximation of the sections the document should have. If students consider that they need others, as well as subdividing the sections into different subsections, they can do it without problem.

Students can take inspiration from the cs231n final projects website: http://cs231n.stanford.edu/project.html, where different projects are presented and developed.

This whole final report can have from 6 to 8 pages (no more and no less).

2. Background

This section presents the fundamental concepts necessary to understand the work.

3. Related Works

It presents what has been done in the field previously, and what the best methods are currently. It is very important, in general, not only in this section, to adequately document the relevant literature. To do this, you must use the *.bib* file, in the way I show here: [1–3]

4. Methods

4.1. Classification Models

Detailed description of the methods used and/or proposed, and clear justification of why these methods are used and not others.

5. Experiments

The data used, the experimental validation protocol, the metrics used, the experiments carried out, the results obtained, and their discussion are presented here.

5.1. Dataset

6. Conclusions

Section that presents, briefly and as a summary, the main conclusions of the work carried out. It also usually includes future possible works. That is, what are the most promising lines to continue with this work, as well as possible proposals for improvement. IMPORTANT: these are the scientific conclusions reached in the project; not your personal conclusions about the work you have done!

References

[1] Stéphane Lathuilière, Pablo Mesejo, Xavier Alameda-Pineda, and Radu Horaud. A comprehensive analysis of deep regression. *IEEE transactions on pattern analysis and machine intelligence*, 42(9):2065–2081, 2019. 1

[2] Pablo Mesejo, Daniel Pizarro, Armand Abergel, Olivier Rouquette, Sylvain Beorchia, Laurent Poincloux, and Adrien Bartoli. Computer-aided classification of gastrointestinal lesions in regular colonoscopy. *IEEE transactions on medical imaging*, 35(9):2051–2063, 2016. 1

[3] Víctor A Vargas-Pérez, Pablo Mesejo, Manuel Chica, and Oscar Cordón. Deep reinforcement learning in agent-based simulations for optimal media planning. *Information Fusion*, 91:644–664, 2023. 1

Feature	ResNet	EfficientNet (V1)	EfficientNetV2
First Published	2015 (He et al.)	2019 (Tan & Le, Google)	2021 (Tan & Le, Google)
Core Idea	Residual (skip) connections to enable training of very deep networks	Compound scaling: jointly scale depth, width, and resolution using a fixed coefficient	Improved compound scaling + faster training: smarter architecture + adaptive regularization
Basic Block	• BasicBlock (2× 3×3 convs) for ResNet18/34 • Bottleneck (1×1 → 3×3 → 1×1 convs) for ResNet50+	MBConv (Mobile Inverted Bottleneck): 1×1 exp → 3×3 depthwise → SE → 1×1 proj	Fused-MBConv (early stages): regular 3×3 conv instead of 1×1+depthwise MBConv (later stages) SE only in later blocks
Skip Connections	✓ Yes (additive residual)	✓ Yes (MBConv uses residual when strides=1)	✓ Yes
Attention	× No	✓ Squeeze-and-Excitation (SE) in every block	✓ SE, but only in later stages (to reduce overhead)
Scaling Method	Manual depth increase (18 → 34 → 50 → 101 → 152)	Uniform compound scaling: Depth ×α, Width ×β, Res ×γ	Non-uniform compound scaling + progressive learning (small → large images during training)
Input Resolution	Fixed (usually 224×224), but flexible	Scales with model size: B0: 224 → B7: 600	Smaller than V1 for same accuracy: S: 384, M: 480, L: 480–512
Typical Sizes	• ResNet18 (11M params) • ResNet34 (21M) • ResNet50 (25M) • ResNet101 (44M)	• B0 (5.3M) • B3 (12M) • B5 (30M) • B7 (66M)	• S (21M) • M (54M) • L (120M)
Accuracy (ImageNet)	• ResNet50: ∼76% • ResNet101: ∼78%	• B0: ∼77% • B3: ∼82% • B7: ∼84.4%	• S: ∼83.9% • M: ∼85.2% • L: ∼85.7%
Training Speed	Moderate	Slow (due to high-res inputs & depth)	Much faster (up to 11× vs V1 at similar accuracy)
Inference Efficiency	Good for small models (18/34); heavier for 101+	Very parameter-efficient, but high-res hurts latency	Better FLOPs/accuracy trade-off; optimized for TPU/GPU
Best For	• Baseline models • Transfer learning with limited data • Fast prototyping	• Resource-constrained deployment • When you need high accuracy with fewer params	• Production systems needing speed + accuracy • Training from scratch or fine-tuning
fastai / PyTorch	✓ Native in torchvision & fastai	✓ Via `timm` (e.g., `efficientnet_b0`)	✓ Via `timm` (e.g., `efficientnetv2_s`)
Key Weakness	Suboptimal parameter efficiency; outdated vs modern models	Slow training; SE adds compute overhead; high-res = memory heavy	Slightly less mature ecosystem; not in torchvision

Table 1. Comparison of ResNet, EfficientNetV1, and EfficientNetV2 Architectures