Dissertation submitted to the

UNIVERSITY OF MOHAMED BOUDIAF - M'SILA, ALGERIA

FACULTY OF MATHEMATICS AND COMPUTER SCIENCE

DEPARTMENT OF COMPUTER SCIENCE

in partial fulfillment of the requirements for the degree of

Master's in Computer Science

Specialization: Computer Science

Option: Artificial Intelligence

Presented by

Abdelkarim ABERRAHMANE ZEGHLACHE

Moutaz billah aymen ZEKHNINE

Entitled

Intelligent Dust Detection System for Solar Panels Using Computer Vision

Under the supervision of

Prof. Mahmoud BRAHIMI

Jury Members

Prof. [First Name] [Last Name]	University of M'sila	President
Dr. [First Name] [Last Name]	University of M'sila	Reporter
Dr. [First Name] [Last Name]	University of M'sila	Examiner

June, 2026

الملخص

يستهدف البرنامج الوطني الجزائري للطاقات المتجددة تركيب 15000 ميغاواط من الطاقة الكهروضوئية بحلول 2035، وسيتم نشر معظمها في بيئات صحراوية وشبه قاحلة حيث يؤدي تراكم الغبار المعدني إلى انخفاض إنتاج الألواح بنسبة تتراوح بين 10 و30 بالمائة. تعتمد الطرق التقليدية لإدارة الاتساخ على التنظيف المجدول مسبقا والمحطات المرجعية المعزولة والمعاينة البصرية اليدوية، وهي طرق غير قابلة للتطوير لتناسب المنشآت ذات النطاق الواسع. تقترح هذه الأطروحة نظاما آليا للكشف عن الغبار يعتمد على التعلم العميق باستخدام شبكة عصبية تلافيفية من نوع MobileNetV3-Large مدربة مسبقا على ImageNet ومُكيَّفة بواسطة تقنية النقل التعليمي، مع تطبيق تقنية CLAHE لموازنة الإضاءة. تم تدريب النموذج على 2029 صورة موسومة للألواح الشمسية مدمجة من مجموعتي بيانات عموميتين، باستخدام خط أنابيب إنتاجي يشمل المُحسِّن AdamW وجدولة معدل التعلم بطريقة الجيب الزائدي وتنعيم العلامات وتقنيات تعزيز البيانات. حقق النموذج دقة تحقق قصوى قدرها 0.97 بالمائة (و6.96 بالمائة مع تعزيز وقت الاختبار)، مع تحسين زمن الاستدلال بمعامل 66.3 عبر ONNX ليصل إلى 87.1 ميلي ثانية لكل صورة. كما يحتوي النظام على وحدة كشف انجراف إحصائي مبنية على اختبار كولموغوروف-سميرنوف، وقد تمكنت إعادة الضبط القصيرة من رفع الدقة على البيانات المنحرفة من 67 إلى 85 بالمائة.

الكلمات المفتاحية: الأنظمة الكهروضوئية، الكشف عن الغبار، الشبكات العصبية التلافيفية، التعلم بالنقل، MobileNetV3-Large، الجزائر، البيئة الصحراوية.

Abstract

Algeria's National Renewable Energy Programme targets 15000 MW of installed photovoltaic (PV) capacity by 2035, most of it in Saharan and semi-arid environments where mineral dust routinely cuts panel output by ten to thirty percent. The conventional ways of managing soiling, namely calendar-based cleaning, isolated soiling-ratio reference stations, and manual visual inspection, were built for installations one or two orders of magnitude smaller than the utility-class plants now under construction, and none of them combines the spatial coverage, objectivity, and scalability this setting demands.

This thesis investigates an automated, image-based alternative grounded in deep learning. It implements and evaluates a Convolutional Neural Network approach to binary clean-versus-dusty classification, using a MobileNetV3-Large backbone pretrained on ImageNet and adapted to the target domain through transfer learning, with Contrast-Limited Adaptive Histogram Equalisation (CLAHE) applied to normalise illumination. The model is trained

on a merged public dataset of 2029 labelled solar panel images under a production-grade pipeline using AdamW optimisation, cosine annealing with linear warmup, label smoothing, RandAugment, and Random Erasing augmentation. It reaches a best validation accuracy of 97.04 percent (96.55 percent with four-view test-time augmentation), with 97.08 percent precision and 94.86 percent recall on the Dusty class. Deployment via the ONNX Runtime achieves up to a 3.66× inference speedup relative to native PyTorch, reducing per-image latency to 1.87 milliseconds. The system also includes a statistical drift-detection module based on the Kolmogorov--Smirnov test, together with a drift-resilient fine-tuning procedure that lifts accuracy on drifted data from 67 to 85 percent.

Keywords: Photovoltaic systems, Dust detection, Convolutional neural networks, Transfer learning, MobileNetV3-Large, CLAHE, Algeria, Saharan environment.

Dedication

"For you who never left before I learned what love meant. I grew up looking for you in quiet rooms in the way certain songs end too soon in the space between what people say and what they mean. The house remembered your presence and your absence the way a body remembers a wound, not always loudly but always. I spent years chasing numbness to make the silence stop. But I made it here. To the last page. To the part where I finally get to say your name in something that will last. This is yours wherever you are."

Abdelkarim

"Praise be to allah the almighty for breath life in thy mortal husk and for my parents that give me everything and beyond and to those I love and stood by me"

Moutaz

Acknowledgements

From Abdelkarim

I am deeply indebted to our thesis supervisor, **Prof. Mahmoud BRAHIMI**, whose careful guidance, technical insight, and patience throughout this work have been invaluable. The completion of this thesis would not have been possible without his continuous support and the time he devoted to discussing the technical and methodological challenges that emerged at every stage.

I extend my sincere thanks to the members of the jury for the time and attention they have devoted to reviewing this dissertation, and for the constructive remarks that have contributed to its improvement. I am also grateful to the academic and administrative staff of the Department of Computer Science at the Faculty of Mathematics and Computer Science, University of Mohamed Boudiaf, M'sila, for providing a learning environment in which research and engineering work could be conducted seriously.

Finally, I owe a profound debt of gratitude to my family and friends, who supported me throughout the years of study, and whose patience during the most demanding phases of this work made its completion possible.

From Moutaz

I would like to express my sincere gratitude to our supervisor, **Prof. Mahmoud BRAHIMI**, for his guidance, encouragement, and the valuable knowledge he shared with us throughout this project. His advice was essential at every step of this work.

I also thank the members of the jury for kindly accepting to evaluate this dissertation, as well as all the teachers of the Department of Computer Science whose efforts accompanied us throughout our years of study. I am grateful to the broader research community working on photovoltaic systems and computer vision, whose published work has provided the foundation upon which this thesis builds.

Finally, I dedicate my deepest thanks to my family and friends for their constant support, patience, and encouragement, which gave me the strength to complete this work.

Contents

List of Tables

List of Figures

General Introduction

The shift from fossil fuels to renewable energy is one of the defining challenges of this century, for engineers and policymakers alike. Among the renewable technologies that are now both technically mature and economically competitive, photovoltaic (PV) solar generation stands out. Three things have driven its growth: costs have fallen steadily for two decades, sunlight is available almost everywhere, and a PV system has no fuel cycle and no moving parts. The result has been rapid. By the end of 2023 the world had installed roughly 1.6 TW of cumulative PV capacity, with 447 GW added in that year alone, an annual growth rate of 87 percent [1]. National decarbonisation programmes in rich and developing countries alike are now pushing deployment faster still.

Algeria is a particularly compelling case within this picture, because it pairs an exceptional solar resource with a serious national commitment to using it. The country's Saharan territories, which make up about 80 percent of its land area, receive more than 2200 kWh per square metre of solar irradiation a year, and the southern regions enjoy over 3500 hours of sunshine annually [2]. Few places on Earth do better. The government has turned this resource into a plan: its National Renewable Energy Programme targets 15000 MW of installed solar PV capacity by 2035 [3]. The first phase, twenty-two plants totalling 3200 MW, is currently under construction. By April 2026 those plants were about 40 percent complete, and two had already been commissioned, a 200 MW plant at Tendla in El Meghaier province and a 200 MW plant at El Ghrous in Biskra province. Nine more, adding up to 1480 MW, are due to enter service before the end of August 2026 [4], [5].

What soiling does in this setting is well documented. Field studies at Algerian research institutions have recorded power losses of 8.41 percent after 4.36 g/m^2 of natural dust built up over eight weeks in the Saharan environment of Ouargla, and a single sandstorm cut energy production by 32 percent at the 30 MW SKTM El-Hadjira plant [6]. Broader reviews of arid and semi-arid regions report yield losses anywhere from 10 percent in moderate conditions to more than 40 percent under heavy Saharan accumulation [7], [8]. Scaled up to the plants Algeria is now commissioning, those percentages translate into tens of thousands of megawatt-hours of lost generation per plant each year. Across the full 15 GW programme, the lost revenue could run to hundreds of millions of dollars annually [9].

At heart, then, the problem this thesis addresses is one of operational scale. A single 200 MW Algerian plant holds roughly half a million modules across hundreds of hectares of remote desert. The methods normally used to manage soiling were designed for installations one or two orders of magnitude smaller. Calendar-based cleaning wastes water and labour when dust accumulates slowly, and leaves panels dirty after a sandstorm; soiling-ratio refer-

ence stations only tell you the soiling level at the point where they sit, leaving the rest of the array a guess; and inspecting hundreds of thousands of panels by eye is not feasible on any sensible schedule [8]. What the situation actually calls for is a method that does four things at once: cover the whole array, give objective and repeatable readings, cost little per panel, and use almost no water. None of the conventional methods manages all four together.

Deep learning, and convolutional neural networks (CNNs) in particular, have been applied to PV inspection since Mehta et al. introduced DeepSolarEye in 2018 [10]. Accuracy has climbed steadily since then, helped by task-specific architectures such as SolNet [11] and SolPowNet [12]. Three gaps remain, though, and they matter for Algeria. First, the public benchmark datasets used to train and test these models do not capture what Saharan mineral dust actually looks like, which differs in colour, particle size, and chemistry from the dust in those collections [7], [13]. Second, almost every reported model is tested only on data drawn from its own training distribution; how a model trained on a clean public dataset behaves on real panels under poor imaging conditions is rarely examined. Third, the practical side of deployment, namely inference latency on edge hardware, model size, and the runtime monitoring needed to know when retraining is due, is mostly missing from the dust-detection literature. The upshot is that, for all the published progress, no existing study shows that a model can be trained, deployed efficiently, and monitored for distribution shift under conditions like those Algeria will face.

Against that background, this thesis makes three concrete contributions. The first is a transfer-learning approach to binary clean-versus-dusty classification, built on a MobileNetV3-Large backbone pretrained on ImageNet and fine-tuned on a merged public dataset of 2029 labelled solar panel images, with Contrast-Limited Adaptive Histogram Equalisation (CLAHE) applied as a preprocessing step to normalise illumination [14], [15]. The second is a close look at how the trained model behaves when deployed: export to the Open Neural Network Exchange (ONNX) format, latency measured on GPU hardware, and a comparison of FP32 and FP16 inference, in enough detail to draw realistic conclusions about running it on a drone [16]. The third is a drift-detection module based on the Kolmogorov--Smirnov test [17], together with a demonstration that a short round of fine-tuning on simulated-drift data recovers most of the accuracy that distribution shift takes away. No one of these settles the broader gap on its own, but together they provide a foundation on which a domain-adapted, Algerian Saharan dataset can be built in later work.

To pursue those contributions, the work sets itself five objectives: to give a current account of the operational setting in which dust detection has to work, covering Algeria's solar resource, its national programme, and the soiling conditions specific to Saharan and semi-arid environments; to set out the theory of convolutional neural networks and transfer learning as they apply to classifying panel surface condition from images; to design, build, and document a production-grade deep learning pipeline for binary classification; to evaluate the resulting

system on a labelled dataset using standard metrics and to test how well it holds up under simulated distribution shift; and to work through what the results mean for real deployment at Algerian utility-scale plants, identifying where future research should go next.

The remainder of the document runs to four chapters and a general conclusion. Chapter One covers the fundamentals of photovoltaic technology, presents Algeria's solar resource and national programme in detail, describes the dust that settles on panels in arid Algerian conditions, quantifies the resulting losses with field evidence, and weighs the limits of conventional soiling-detection methods, closing with a formal proposal of the CNN approach this work develops. Chapter Two lays the theoretical groundwork for that approach: the core building blocks of CNNs, the training methods used, the principle of transfer learning, and a critical review of the state of the art in deep learning for PV inspection. Chapter Three sets out the full methodology, from dataset preparation, preprocessing, and augmentation through model architecture, training configuration, ONNX deployment, and the evaluation protocol. Chapter Four reports the experimental results, examines which development trials worked and which did not, analyses robustness under data drift, measures inference performance, and discusses what all of this means for deployment in Algeria. The General Conclusion draws the findings together and points to directions for further work.

Chapter 1: Solar Panel Technologies in the Algerian Context

1.1 Introduction

Before any detection method can be designed, the environment it has to work in must be properly understood. This chapter builds that understanding from the ground up. It starts with the physics of the photovoltaic effect and a comparison of the main commercial PV technologies in use today. It then turns to Algeria's solar resource and national deployment programme, drawing on the most recent data available. From there it characterises the dust that settles on panels in Algeria's arid and semi-arid environments, covering its mineralogy, particle sizes, how it gets there, and how it varies across space and time. The quantitative evidence for soiling losses comes next, including field results from sites representative of Algerian conditions. The chapter ends with a critical look at the conventional methods used to detect and manage soiling, and identifies the specific limitations that motivate the convolutional neural network approach proposed at its close.

1.2 The Global Energy Transition and the Rise of Solar Power

The context for everything that follows is a structural shift in how the world produces electricity. For most of the twentieth century, electrical power came overwhelmingly from burning fossil fuels. Concerns about climate change, energy security, and the long-run cost of fuel have since driven a sustained move toward renewable sources, and among those sources solar photovoltaics has grown fastest. The reasons are partly economic and partly physical. The cost of PV modules has fallen by roughly an order of magnitude over the past decade and a half, the result of manufacturing scale, improved cell efficiency, and intense competition among producers [1]. At the same time, sunlight is the most widely distributed energy resource on the planet, available in some measure almost everywhere people live, and a PV system converts it to electricity with no fuel, no combustion, and no moving parts to wear out.

The scale of the resulting build-out is hard to overstate. Cumulative global PV capacity passed roughly 1.6 TW by the end of 2023, with 447 GW added in that single year, a growth rate that few other industrial sectors have ever matched [1]. This expansion is not evenly distributed. The regions with the strongest economic case for solar are those that combine high solar irradiance with land available for large ground-mounted arrays, and many of those regions, across North Africa, the Middle East, and parts of Asia and the Americas, are arid

or semi-arid. This is the central tension that frames the present work: the places where solar power makes the most economic sense are frequently the same places where airborne dust most threatens its performance. The very abundance of sunlight that draws PV deployment to the desert comes packaged with an environmental hazard that the technology was not originally designed to withstand. Understanding and managing that hazard is therefore not a peripheral maintenance concern but a condition for the economic case that justifies building these plants in the first place.

1.3 Fundamentals of Photovoltaic Technology

1.3.1 The Photovoltaic Effect and Its Physical Basis

The photovoltaic effect is the process by which a semiconductor turns incident light directly into electrical energy. It was first observed in 1839 by the French physicist Alexandre-Edmond Becquerel, who measured a small current in an electrochemical cell exposed to sunlight [18]. Turning that observation into something useful took another century. In 1954, Chapin, Fuller, and Pearson at Bell Telephone Laboratories built the first silicon solar cell that could produce useful power, reaching about six percent efficiency in direct sunlight [18].

The mechanism behind the effect in a crystalline silicon cell is best understood through semiconductor band theory. Silicon has a valence band, where electrons are bound to lattice sites, and a conduction band, where electrons move freely through the crystal. The two are separated by a forbidden energy gap of about 1.12 electronvolts at room temperature [19]. When the silicon absorbs a photon carrying more energy than the bandgap, the photon lifts an electron from the valence band into the conduction band and leaves behind a positively charged vacancy, a hole, in the valence band. These photogenerated electron and hole pairs are separated at the p--n junction formed where boron-doped (p-type) and phosphorus-doped (n-type) layers meet. The junction's built-in electric field sweeps electrons toward the n-type side and holes toward the p-type side, driving a steady current through an external circuit [18], [19].

1.3.2 Commercial Photovoltaic Technologies

The PV industry has settled on a fairly small set of material systems that strike a good balance of efficiency, cost, durability, and environmental footprint [13]. These technologies matter to the soiling problem because their surface coatings and encapsulation materials interact differently with deposited dust.

Monocrystalline silicon modules are cut from wafers sliced from a single crystal ingot grown by the Czochralski process. With no grain boundaries to cause recombination losses,

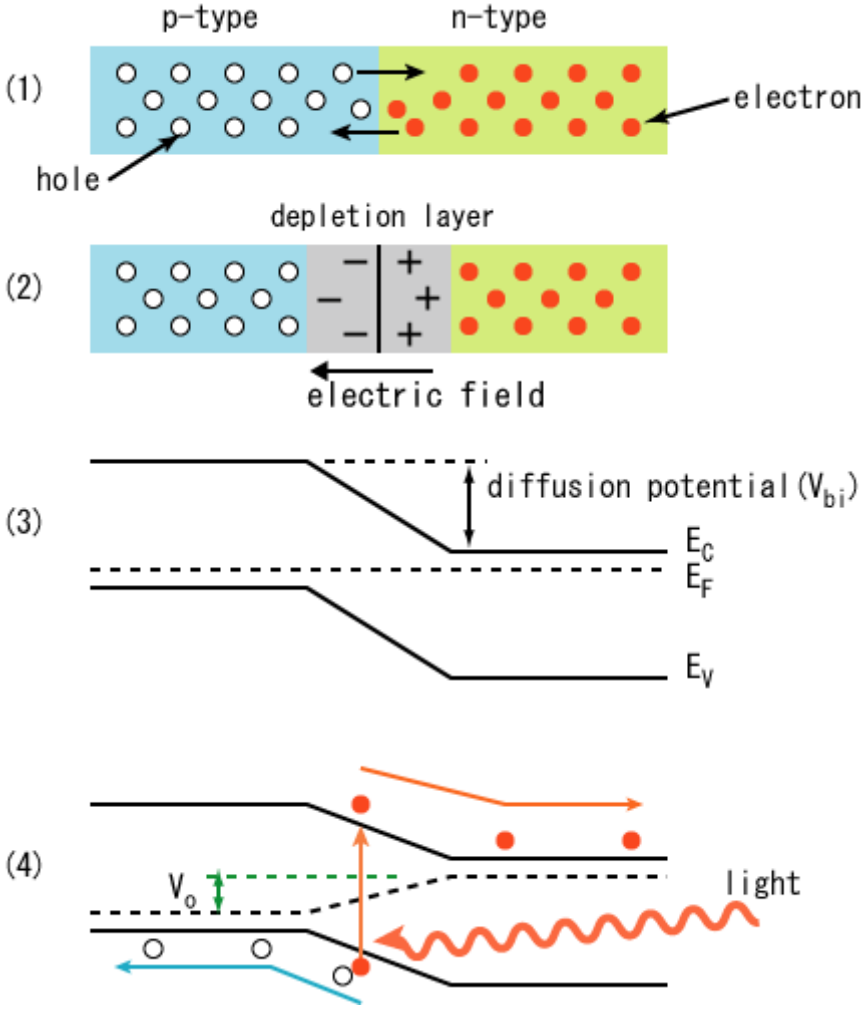

Figure 1.1: pv effect

commercial monocrystalline modules routinely reach 20 to 24 percent efficiency under standard test conditions [13]. Polycrystalline, or multicrystalline, modules use wafers from cast ingots that solidify into many randomly oriented grains; they typically reach 17 to 20 percent at lower cost [13]. Thin-film technologies, chiefly cadmium telluride (CdTe) and copper indium gallium diselenide (CIGS), offer manufacturing advantages and mechanical flexibility, and CIGS has reached laboratory efficiency records above 23 percent under controlled conditions [13]. For Algeria's utility-scale plants the dominant choice is monocrystalline silicon, valued for its high efficiency per unit area, which keeps land use and balance-of-system costs down in large ground-mounted arrays.

1.3.3 Performance-Determining Factors in Photovoltaic Systems

Solar irradiance is the most direct driver of output, scaling roughly linearly with incident power density at a fixed cell temperature [19]. Temperature is a strong secondary influence: crystalline silicon cells have a negative temperature coefficient of power, usually between

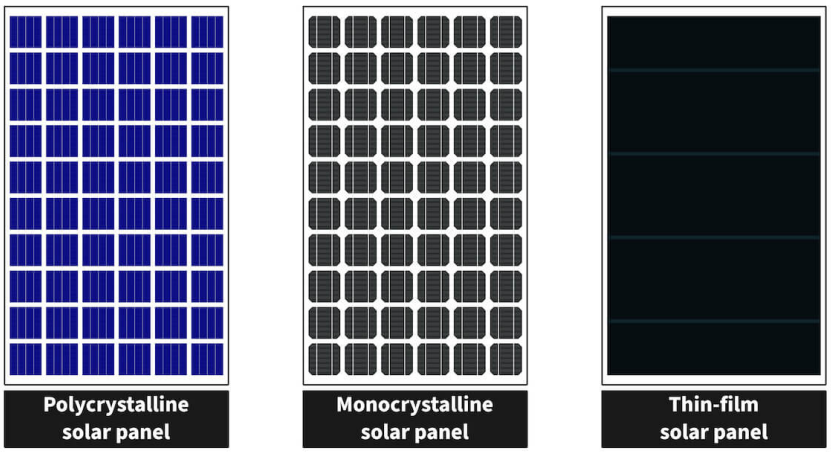

Figure 1.2: pv technologies

-0.35 and -0.45 percent per degree Celsius above the 25°C standard test condition [19]. In Saharan deployment, where module temperatures often reach 65 to 75°C around midday in summer, this derating alone can remove 15 to 20 percent of the rated output [2].

Of the performance losses that can actually be managed through operations, soiling is the most significant under the arid conditions of Algeria's main deployment zones [8]. Unlike temperature derating, which is built into the climate and cannot be helped, soiling can be reversed by cleaning, provided the cleaning is scheduled on the basis of the panel's actual condition rather than an arbitrary calendar interval. That single distinction, between a loss that is fixed by physics and a loss that responds to a good decision, is what makes soiling worth detecting in the first place.

1.4 The Algerian Solar Energy Context

1.4.1 Assessment of the National Solar Resource

Algeria's solar resource is, without overstating it, among the best in the world. Global Horizontal Irradiance (GHI) exceeds 1900 kWh per square metre a year across the whole country, and reaches 2200 to 2600 kWh per square metre a year in the Saharan regions that are the prime targets for utility-scale deployment [2]. Annual sunshine ranges from about 2500 hours along the Mediterranean coast to more than 3900 hours in the southern Sahara [2].

The numbers mean more in comparison. Germany, which for years led the world in cumulative PV installation, sees average annual GHI of roughly 1000 to 1200 kWh per square metre across most of its territory [20]. A panel in the central Algerian Sahara therefore produces about twice the energy an identical panel would in Germany over a year. That difference feeds straight through to a lower levelised cost of electricity, putting Algerian solar among

the cheapest sources of power producible at scale anywhere [2], [20].

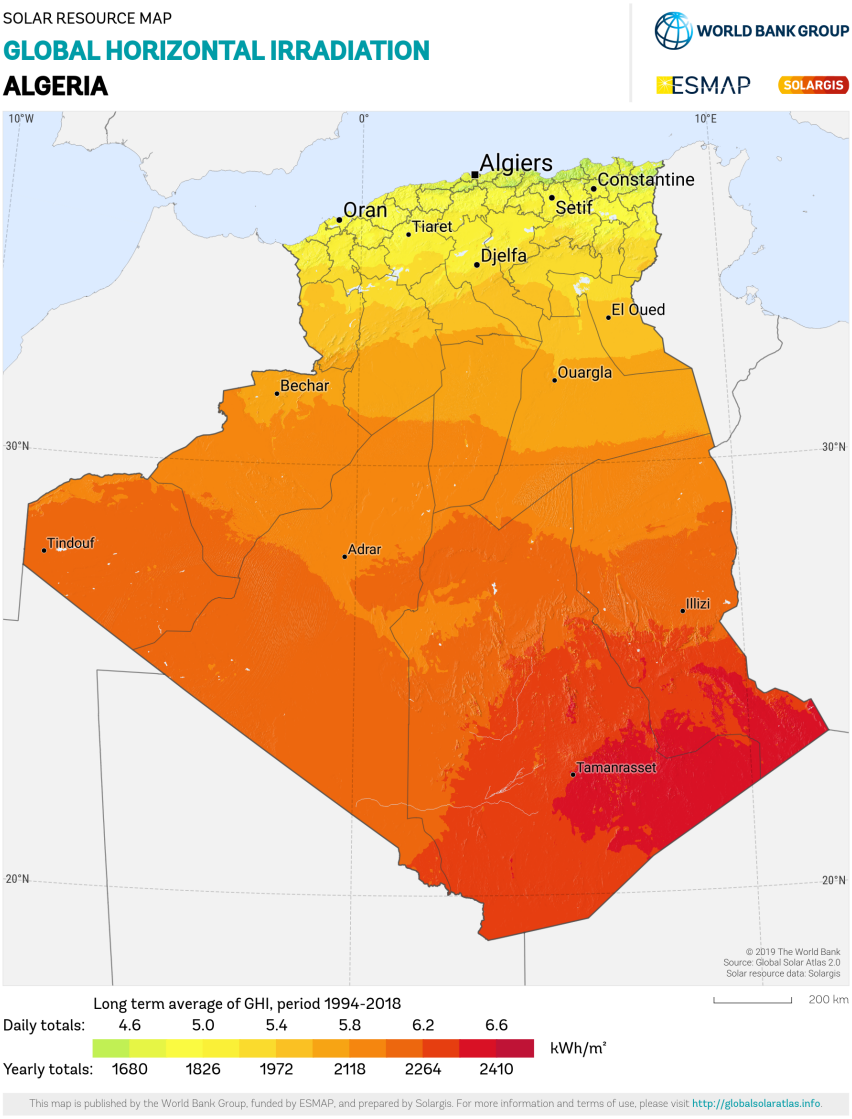

Figure 1.3: Annual Global Horizontal

1.4.2 The National 15 GW Deployment Programme

The National Renewable Energy Programme is the framework for putting this resource to work, with a target of 15000 MW of installed PV capacity by 2035 [3]. It marks a sharp break from Algeria's historical energy system, in which natural gas dominated electricity generation, around 99 percent of the mix as recently as 2024 [3]. The first phase covers 22 solar plants across the country, with individual capacities from 80 MW to 220 MW and a combined total of 3200 MW. By April 2026 those plants were roughly 40 percent complete overall [3].

Commissioning picked up sharply in the first half of 2026. Nine plants totalling 1480 MW

were due to enter service before August 2026 [4], [5]. Among the April 2026 milestones, two plants totalling 400 MW were officially commissioned: the 200 MW Tendla facility in El Meghaier province and the 200 MW El Ghrous plant in Biskra province [5].

Beyond the immediate build-out, Algeria has set its sights on becoming a net exporter of renewable electricity to Europe. That ambition rests on planned interconnection infrastructure, including a proposed submarine cable to Italy, alongside a longer-term plan to produce and export green hydrogen [2]. These export plans change the economics of operational performance: every megawatt-hour lost to soiling is now not just lost domestic supply but lost export revenue too.

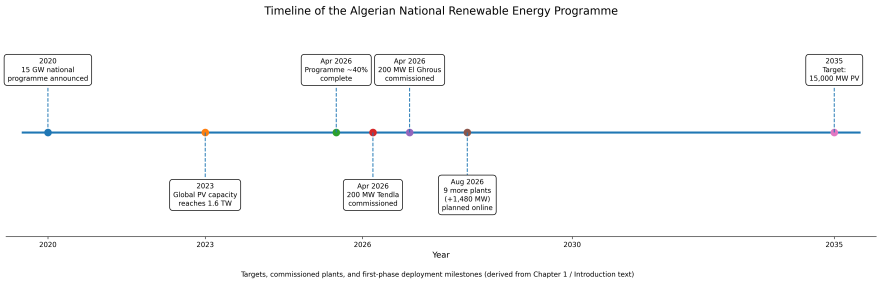

Figure 1.4: Timeline of the Algerian National Renewable Energy

1.4.3 Institutional Structure

Several bodies share responsibility for delivering the solar programme. The Ministry of Energy and Renewable Energies sets strategy, writes policy, and oversees the regulatory framework. Sonelgaz, the national electricity and gas utility, is the main implementing agency, handling engineering, procurement, and construction across the plant portfolio. SHAEMS brings in international partners for project development, especially for technology and financing that benefit from foreign expertise and capital [3]. The division of labour matters for maintenance: a condition-based inspection system of the kind this thesis develops would sit within Sonelgaz's operations remit, and its value depends on being cheap enough to run across a portfolio that is still growing year on year.

1.5 Characterisation of Dust in Algerian Arid Environments

1.5.1 Mineralogical Composition

Dust is not a uniform material, chemically or physically. In Algeria's Saharan and High Plateaus regions, the particulate that lands on panel surfaces is a complex mineral mixture whose makeup reflects the geology of its source regions, the path it travelled, and how much it was transformed along the way [21]. Composition matters to the soiling problem because

it partly governs the optical properties, adhesion, and removability of the deposited layer [7], [13].

Silica (SiO_2) is the dominant component, usually 50 to 70 percent of the deposited mass [21]. Silica particles are hard and angular, the angularity coming from mechanical grinding during wind transport, which makes Saharan dust abrasive and contributes to long-term wear of anti-reflective coatings under high winds [7]. Clay minerals, mainly kaolinite and montmorillonite, make up a sizeable second fraction at 15 to 25 percent by mass. Their flat, plate-like shape sticks strongly to surfaces through van der Waals forces and, when the surface is damp, through capillary adhesion [13]. Montmorillonite is especially troublesome: it absorbs atmospheric water and swells, which cements the dust layer more firmly once it dries out again [7].

Calcium carbonate ($CaCO_3$), as calcite and occasionally dolomite, is typically 10 to 20 percent of Saharan dust by mass. In the presence of moisture and carbon dioxide, carbonates can react and form hard calcareous crusts that resist mechanical cleaning [13]. Iron oxides, mostly hematite (Fe_2O_3), make up only 2 to 5 percent by mass but give Saharan dust its reddish-orange colour and contribute disproportionately to optical attenuation, since they absorb strongly in the visible range [7]. This reddish cast is worth keeping in mind, because it is exactly the kind of local visual signature that a model trained on dust from a different region may not have learned to recognise.

1.5.2 Particle Size Distribution and Its Implications

Deposited dust spans about four orders of magnitude in size, from coarse sand grains several hundred micrometres across down to ultrafine particles near 0.1 micrometres [7]. Coarse particles above roughly 50 micrometres arrive mainly by gravitational settling and by saltation during high winds. They form visible deposits that are loosely bound and easy to remove with light brushing or a gentle breeze. Particles in the PM10 range, between 2.5 and 10 micrometres, matter both for how they scatter light and for how they stick. At sizes close to the wavelength of visible light, fine particles scatter incident radiation very efficiently [21]. Ultrafine particles below about 2.5 micrometres are the hardest to remove once down: their very high surface-area-to-volume ratio maximises adhesive force per unit mass, and their low inertia lets them follow the panel's microscopic texture closely, maximising contact area and the resulting grip [13].

1.5.3 Deposition Mechanisms in Desert Environments

Gravitational settling out of the air is the baseline mechanism, working continuously at rates that depend on how much dust the atmosphere carries and the particle sizes involved [7]. Wind-driven impaction, tied to the sandstorm and dust-storm events typical of Algeria's Saharan and High Plateaus climates, can drop several grams of mineral mass per square metre

in a matter of hours. The Sirocco, a hot, dry southerly or southeasterly wind that carries large quantities of Saharan dust northward, is the main driver of major soiling events in the Algerian High Plateaus [8].

Dew-assisted cementation deserves particular attention in desert settings [13]. The large day-to-night temperature swings in Algeria's Saharan zones, which can exceed 30°C, often cool panel surfaces below the dew point on clear nights. The condensed moisture sharply increases the grip of particles already present, through capillary forces, and acts as a trap for more airborne particles. When it evaporates the next morning, the moisture film can leave behind a cemented dust layer far more resistant to wind or light washing than the loose powder it started as [7], [13]. Several studies have flagged this mechanism as a key driver of persistent soiling in North African installations [7].

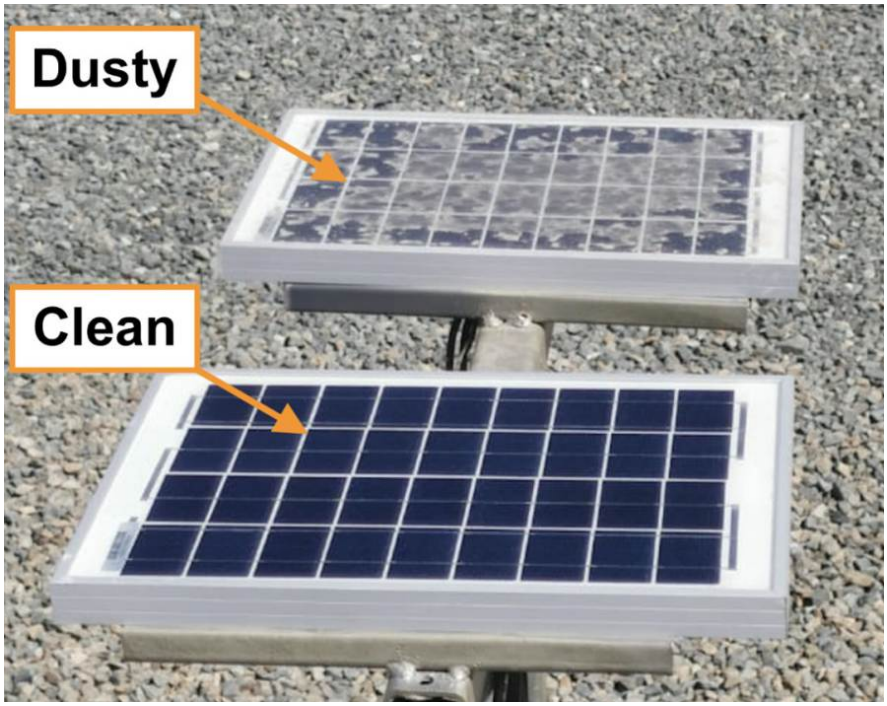

Figure 1.5: Photographic comparison of photovoltaic panel surface

1.5.4 Spatial and Temporal Variability of Soiling

Dust does not accumulate evenly across a large solar farm, nor at a steady rate [8]. Spatial differences come from several sources: local variation in wind speed and direction across the array, the way airflow channels and speeds up between panel rows, edge effects at the upwind margin, and proximity to local dust sources such as unpaved access roads and disturbed construction areas [13]. Measurements at large arid-region installations have found soiling-ratio differences of several percentage points between the dirtiest and cleanest sections of a single array, which confirms that a single monitoring station cannot reliably capture how soiling loss is distributed across the whole site [8].

The timing follows a clear seasonal pattern in Algeria. Dust activity peaks in spring and early summer, when the Sirocco and Chergui winds are most active, and drops to much lower levels in winter, when the large-scale circulation is calmer [7]. This seasonality undercuts the logic of fixed-schedule maintenance still further and strengthens the case for condition-based approaches driven by real-time assessment [13].

1.6 Quantitative Effects of Dust on PV Performance

1.6.1 Optical Attenuation Mechanisms

Dust reduces a module's output mainly by cutting the sunlight that reaches the active semiconductor. It does this through three optical processes [7]. Absorption converts part of the incoming radiation into heat within the dust layer; silica and carbonates are fairly transparent in the visible range, but iron oxides and some clays absorb strongly there, and the combined absorption of a mixed dust layer can become significant at Saharan loading levels [21]. Backscattering reflects part of the radiation away from the panel; particles in the 0.5 to 5 micrometre range are particularly efficient scatterers of visible light [7]. Diffuse forward scattering redirects part of the light into non-specular directions, lengthening its effective path through the anti-reflective coating and reducing how much couples into the semiconductor [13].

A further effect, sometimes called partial shading, comes from uneven dust deposits. When dust builds up more in some regions of a module, the resulting variation in irradiance creates a current mismatch between series-connected cells, so the whole string is limited by its most heavily soiled cell. In severe cases, cells forced into reverse bias by the mismatch can develop localised hotspots that accelerate degradation of the encapsulant and the cell metallisation [8].

1.6.2 Experimental Evidence from Algerian Studies

The link between dust loading and performance loss has been measured directly under conditions representative of Algeria's desert and semi-arid environments. The most thorough published study comes from the Laboratory of New and Renewable Energy in Arid and Saharan Zones (LENREZA) at Ouargla University, in the southeast. Dida et al. [6] examined the effect of natural dust on monocrystalline modules outdoors, alongside a connected 30 MW grid plant at the SKTM El-Hadjira site. Their main findings are summarised in Table 1.1.

These results show that natural dust in the Saharan environment of Ouargla cut power output by about 8.41 percent over eight weeks, with proportionally smaller drops in short-circuit current and open-circuit voltage. A single sandstorm did far more, and far faster: the

Table 1.1: Measured photovoltaic performance degradation due to dust accumulation under Saharan field conditions in Ouargla, Algeria. Adapted from Dida et al. (2020) [6].

Exposure condition	**Density (g/m^2)**	**Performance reduction**	**Notes**
8 weeks outdoor exposure	4.36	8.41% power; 6.10% I_{sc}	Monocrystalline, Ouargla
Single sandstorm event	N/R	32% energy output	30 MW SKTM El-Hadjira
Reference (cleaned daily)	≈ 0	0%	Control configuration

32 percent energy reduction at the connected 30 MW plant amounts to a loss of 5.18 MWh of generation [6].

The wider literature agrees. A later field study by Necaibia et al. [22] at sites in the south-western Algerian Sahara reported soiling-induced yield losses above 15 percent after long periods without cleaning. The review by Sayyah, Horenstein, and Mazumder [8] documents dust-related yield losses from 10 percent in moderate conditions to over 40 percent in the harshest deserts. A more recent review by Conceição et al. [7] confirms that dust can cut output power by up to 50 percent in arid and semi-arid regions under extreme conditions. The Algerian study by Semaoui et al. [23], on how dust affects the optical transmittance of module glazing in desert conditions, lends further support to these aggregate findings.

1.6.3 Economic Impact Analysis

The economic weight of soiling losses, taken at the scale of Algeria's national programme, can be estimated with a straightforward capacity-and-revenue calculation. Take a representative 200 MW plant, comparable to Tendla or El Ghrous. A 10 percent soiling loss under heavy accumulation means 20 MW of foregone capacity. Assuming annual production equivalent to 2000 full-load hours, in line with the solar resource of Algeria's main zones [2], that comes to 40000 MWh of lost generation a year at a single plant. At an electricity value somewhere between 40 and 80 USD per MWh, the annual revenue hit from unmanaged soiling at one plant lands in the range of 1.6 to 3.2 million USD [9].

Extended across the full 15000 MW programme, the potential economic benefit of effective soiling management runs into the order of several hundred million USD a year [8], [9]. That figure is the economic justification for investing in the kind of advanced, condition-based monitoring infrastructure this thesis develops.

1.7 Conventional Soiling Detection and Mitigation Approaches

1.7.1 Fixed-Schedule Preventive Cleaning

The simplest and most common approach to managing soiling is a fixed, calendar-based cleaning schedule, carried out regardless of the array's actual state [8]. Intervals of one to

four weeks are typical in desert installations. The appeal is administrative simplicity: teams can be scheduled in advance, water secured, and operations run predictably with no ongoing monitoring or decision support.

But fixed schedules carry inefficiencies that bite hard in the desert. During quiet periods, scheduled cleaning burns water, labour, and equipment hours for no real benefit [7]. Where freshwater is scarce and expensive, often trucked long distances to remote sites, that is both a direct cost and an environmental concern [8]. And after a big sandstorm, a fixed schedule can leave panels heavily soiled for days or weeks until the next cleaning comes round, losing generation that a condition-triggered response would have caught [13].

1.7.2 Soiling Ratio Measurement Stations

A more sophisticated approach uses dedicated monitoring stations with two reference cells: one cleaned frequently to hold a clean baseline, the other left to soil naturally. The soiling ratio, the soiled cell's output divided by the clean cell's, gives a continuous, quantitative measure of the fractional loss due to soiling at that location [8].

This gives objective data that can drive condition-based cleaning directly. But two things limit it at utility scale. The first is the capital cost of a station, which makes full spatial coverage impractical [9]. The second is subtler: even a fairly dense network of point measurements only interpolates the soiling state at unmonitored spots, and that interpolation can be badly wrong where airflow is complex and local dust sources vary [8].

1.7.3 Manual Visual Inspection

Having trained staff inspect panels by eye is the most direct way to gauge soiling, and it has a bonus: the same walk-through can catch other problems, including physical damage, bird droppings, and encroaching vegetation. No special hardware is needed beyond a person on site.

The catch is that it does not scale to modern utility-class plants [24]. A 200 MW plant built from typical 400 Wp modules holds roughly 500000 panels in rows stretching across hundreds of hectares. Inspecting each one by eye would take an impractical number of person-hours per cycle. Visual judgement is also subjective, so readings vary from one inspector to the next, which hurts the consistency of maintenance decisions [8]. And early-stage soiling is often below what the eye can detect in ambient outdoor light, even after it has begun to cause real losses [7].

1.7.4 Cleaning Technologies and Their Trade-offs

Detecting soiling is only half the operational picture; the other half is removing it, and the choice of cleaning method shapes how valuable accurate detection is. Manual cleaning with brushes and water is the most common approach, but it is labour-intensive and, in the desert, consumes a resource that is both scarce and expensive to transport [8]. Automated and semi-automated alternatives have therefore attracted considerable attention. Mechanised cleaning systems, including tractor-mounted brushes and robotic units that travel along panel rows, reduce labour but still rely on either water or physical contact that can, over time, abrade anti-reflective coatings [7]. Waterless robotic cleaners using microfibre rollers or air blowers avoid the water cost but carry a high capital cost and require panel layouts designed to accommodate them.

A fundamentally different family of approaches tries to prevent or remove deposition without mechanical contact at all. Electrodynamic screens, which use travelling electric fields embedded in or on the panel surface to lift and transport charged dust particles, have been demonstrated in laboratory and field trials and are particularly attractive for water-scarce environments, though their integration cost and long-term durability remain open questions [8]. Anti-soiling surface coatings, whether hydrophobic coatings that encourage water to sheet off and carry dust with it or hydrophilic self-cleaning coatings that exploit photocatalytic effects, can slow accumulation but do not eliminate it, and their performance degrades under the abrasive conditions of Saharan dust [13]. The common thread across all of these technologies is that each cleaning action carries a real cost, in water, energy, labour, capital, or coating wear. That cost is exactly why knowing precisely when and where cleaning is actually needed, the problem this thesis addresses, has direct economic value: a detection system that triggers cleaning only when the soiling loss justifies it minimises the number of costly cleaning actions while protecting generation.

1.7.5 Remote Sensing and Aerial Inspection

The operational appeal of an image-based detection method rests on a parallel development in inspection hardware. Unmanned aerial vehicles, or drones, have become a standard tool for inspecting large solar installations, carrying both standard RGB cameras and, increasingly, thermal infrared sensors that reveal hotspots and electrical faults invisible in ordinary light [24]. A drone can survey in a single flight an area that would take a ground crew days to walk, and it can do so on a schedule, from consistent viewpoints, and without exposing workers to the heat and electrical hazards of a desert array at midday. This is the deployment context that makes a fast, lightweight image classifier valuable. A model small enough to run on hardware carried aboard the drone, or on a ground station receiving its imagery, can turn raw aerial photographs into actionable soiling maps in real time. The combination of cheap,

capable aerial platforms and efficient on-board inference is what transforms automated dust detection from a laboratory result into an operational possibility, and it is the reason the present work treats inference latency and model size as first-class design constraints rather than afterthoughts.

1.8 Conclusion and Proposed Approach

Two observations from this chapter motivate the technical work that follows. The first is quantitative. Algeria's national programme is on track to put several thousand megawatts of PV capacity into Saharan and semi-arid environments this decade. In those environments, soiling routinely costs between 10 and 30 percent of generation under heavy dust, with measured power losses of 8.41 percent after just eight weeks of natural exposure at Ouargla and a 32 percent energy loss after a single sandstorm at a connected 30 MW plant [6]. The economic stakes are large; counting foregone export revenue and water, the operational stakes are larger still.

The second observation is structural. The conventional toolkit, namely calendar cleaning, isolated soiling-ratio stations, and manual inspection, was built for installations one or two orders of magnitude smaller than the plants now rising at Tendla, El Ghrous, and the rest of the twenty-two-plant first phase. No one of these methods, alone or combined, gives operators the four things a Saharan utility plant actually needs: coverage of the whole array, objective and repeatable readings, low cost per panel, and minimal water use. The gap between what they deliver and what these plants require is the motivating problem of this thesis.

On that basis, this work proposes a different approach. Rather than measuring soiling at a few sentinel points or cleaning on a calendar, it proposes to detect dust directly from images of the panels themselves, using a Convolutional Neural Network (CNN). The case is empirical, not theoretical: over the past five years, CNNs have reached binary clean-versus-dusty accuracies above 95 percent on public benchmarks, with lightweight architectures such as SolNet [11] and SolPowNet [12] clearing 98 percent. They reach those accuracies at inference speeds low enough to run in real time on drone-mounted hardware. A CNN trained for this task swaps the human eye for a model that does not tire, does not vary between operators, and scales to however many panels a camera can be flown over.

The chapters that follow develop this proposal. Chapter 2 sets out the theory of CNNs and transfer learning. Chapter 3 presents the dataset, architecture, and training protocol. Chapter 4 reports the results, examines how the trained model behaves once exported through ONNX, and tests its robustness under simulated distribution shift of the kind real Saharan deployment will impose.

Chapter 2: Convolutional Neural Networks and Transfer Learning

2.1 Introduction

The dust detection method developed in this thesis is built on Convolutional Neural Networks (CNNs) and on transfer learning. This chapter lays out the theory needed to understand how those tools apply to classifying the surface condition of a photovoltaic panel from an image. Rather than survey artificial intelligence broadly, it focuses on the parts this work actually uses: the convolutional layer and its role in extracting features, the training procedures that let the network learn, the regularisation techniques that help it generalise, and the transfer-learning strategy that makes classification work on a small, domain-specific dataset.

Section 2.2 places the approach in its historical context, tracing the line of development that runs from the earliest convolutional networks to the efficient mobile architectures used today. Section 2.3 introduces CNNs, explaining why they are designed the way they are and what each main component does. Section 2.4 covers training, including the loss function, the optimiser, regularisation, and the augmentation strategy used here. Section 2.5 examines transfer learning and the MobileNetV3-Large architecture in particular. Section 2.6 reviews prior work on deep learning for PV inspection and pins down the gap this thesis addresses. Section 2.7 concludes.

2.2 Historical Development of Convolutional Neural Networks

It helps to understand the model used in this work as the current point on a line of development that runs back several decades, because each milestone along that line solved a specific problem that the next one built upon. The idea of using local receptive fields and shared weights for visual recognition goes back to the neocognitron of the early 1980s, but the first practical convolutional network is usually credited to LeCun and colleagues, whose LeNet-5 was trained to recognise handwritten digits and deployed at scale to read cheques in the late 1990s [25]. LeNet-5 already contained the elements that define a modern CNN: convolutional layers for feature extraction, subsampling (pooling) layers for spatial reduction, and fully connected layers for classification. What it lacked was the data and the computing power to scale to harder problems.

Both arrived in the early 2010s. The ImageNet dataset, assembled by Deng and colleagues, provided a large, carefully labelled corpus of natural images organised by the WordNet hier-

archy [26], and the associated annual competition, the ImageNet Large Scale Visual Recognition Challenge, gave the field a common benchmark on which progress could be measured directly [27]. The turning point came in 2012, when the deep CNN now known as AlexNet, trained on graphics processing units, cut the top-five error rate on ImageNet so dramatically that it reset expectations for the whole field [28]. AlexNet demonstrated that, given enough data and compute, a deep convolutional network could learn visual features far more discriminative than any hand-designed descriptor.

The years that followed were a steady search for depth and efficiency. Deeper networks proved hard to train until residual learning, introduced by He and colleagues, added skip connections that let gradients flow through very deep stacks without vanishing, enabling networks of a hundred layers and more [29]. In parallel, a separate line of work asked not how to make networks deeper but how to make them cheaper, so they could run on phones and embedded hardware. That line produced the MobileNet family, beginning with the depthwise-separable design of the first MobileNet [30], refined through the inverted-residual bottlenecks of MobileNetV2 [31], and culminating in the architecture-search-tuned MobileNetV3 used in this thesis [14]. A complementary family, EfficientNet, approached the same efficiency goal by systematically scaling network depth, width, and resolution together [32], [33]. The model at the centre of this work therefore sits at the intersection of two long trends: the accuracy that ImageNet-scale data and deep architectures made possible, and the efficiency that the mobile-architecture line made practical for deployment in the field.

2.3 Convolutional Neural Networks for Image Classification

2.3.1 From Pixels to Features: The Convolutional Layer

A Convolutional Neural Network is a deep neural network built for spatially structured data, with images as the classic case [34]. Its defining operation is the convolution, strictly the cross-correlation, between the input and a small learnable filter, or kernel. For a two-dimensional input feature map I and a filter K of size $m \times n$, the output at position (i, j) is the discrete convolution sum:

$$S(i,j) = \sum_{p} \sum_{q} I(i+p,\ j+q) \cdot K(p,q) \tag{2.1}$$

The filter slides across the input, taking a dot product at every valid position. Each filter produces one output channel, a feature map, which marks where the pattern encoded in that filter appears in the input [34]. A convolutional layer with K filters produces K feature maps, together forming the layer's output volume. Because the same filter weights are reused

at every position, a property called parameter sharing, a convolutional layer needs far fewer parameters than a fully connected layer processing the same input. A convolutional layer with 64 filters of size 3×3 on a three-channel input needs only $(3 \times 3 \times 3 + 1) \times 64 = 1792$ parameters, against the hundreds of millions an equivalent fully connected layer would require [34].

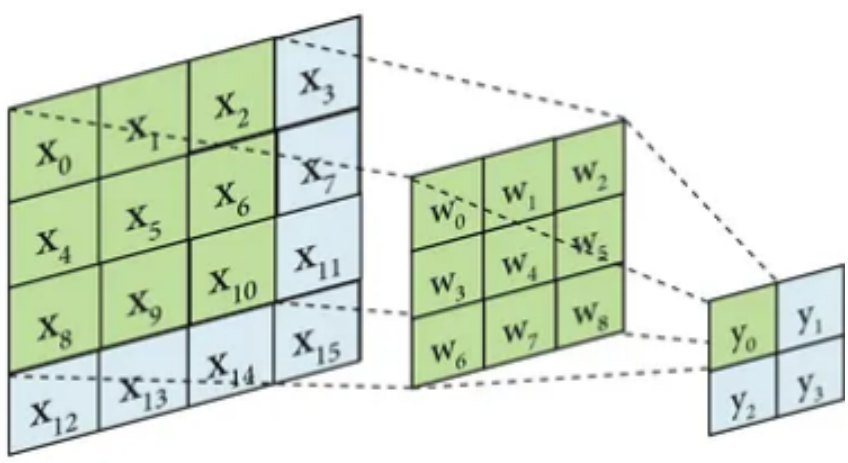

Figure 2.1: convolution

2.3.2 Depthwise Separable Convolutions

Standard convolutions are expensive because each filter mixes information across space and across all input channels at once. Depthwise separable convolution, the building block at the heart of the MobileNet family, splits this into two cheaper steps [30]. A depthwise convolution applies one filter per input channel, capturing spatial patterns within each channel independently. A pointwise convolution, a 1×1 convolution, then mixes the channels back together. The factorisation cuts the computational cost by roughly the square of the kernel size relative to a standard convolution, which is what lets MobileNet-class networks run on phones and embedded hardware at accuracies that used to require far heavier models [30], [31]. This efficiency is the main reason a MobileNet backbone suits the drone-deployment goal of the present work.

2.3.3 Pooling, Depth, and Hierarchical Representations

Pooling layers downsample feature maps, cutting computation while giving the network some local translation invariance [34]. Max pooling, the most common variant, splits the feature map into non-overlapping windows (typically 2×2) and keeps the maximum activation in each, halving each spatial dimension when the stride is 2. As a CNN goes deeper, the features

it has learned change in character. Early layers tend to pick up primitives such as oriented edges, colour contrasts, and simple texture gradients. Middle layers combine those into more complex patterns, including surface motifs and shape fragments. The deepest layers encode high-level, task-specific representations [28]. For panel surface classification, this hierarchy maps naturally onto the visual cues that separate a mirror-like clean surface from the granular, light-scattering look of accumulated dust [10].

2.3.4 Activation Functions and the Role of Non-Linearity

Activation functions supply the non-linearity that makes a deep network far more expressive than a stack of linear transformations [34]. The Rectified Linear Unit (ReLU), defined as $\mathrm{ReLU}(x) = \max(0, x)$, is the standard choice in many CNNs. Its main advantage over older options such as the logistic sigmoid is that it does not saturate for positive inputs: the gradient of ReLU stays exactly one for all $x > 0$, so gradients flow cleanly through the network during backpropagation, without the exponential decay, the vanishing gradient problem, that plagues sigmoidal activations in deep networks [28]. MobileNetV3 goes one step further and uses a hard-swish activation in its deeper layers, a piecewise-linear approximation of the swish function that keeps most of swish's accuracy benefit while being cheap to compute on mobile hardware [14].

2.3.5 Batch Normalisation

Training a deep network is complicated by the fact that the distribution of inputs to each layer shifts as the parameters of the layers before it are updated, a phenomenon Ioffe and Szegedy named internal covariate shift [35]. This shifting forces the use of small learning rates and careful weight initialisation, and makes deep networks slow and fragile to train. Batch normalisation addresses it by normalising the activations of each layer over the current mini-batch, re-centring them to zero mean and unit variance, then re-scaling and shifting them with two learned parameters so the layer can still represent whatever distribution is most useful. The effect is substantial: networks train faster, tolerate higher learning rates, and depend far less on the exact initialisation, and the technique even provides a mild regularising effect of its own [35]. Batch normalisation is used throughout the MobileNetV3-Large backbone employed in this work, where it sits after the convolution in each bottleneck block. During inference the batch statistics are replaced by running averages accumulated during training, so that a single image can be classified without reference to any batch, which matters directly for the single-image inference the deployment scenario in this thesis requires.

2.3.6 Squeeze-and-Excitation Channel Attention

A standard convolutional layer treats all of its output channels as equally important, but in practice some feature channels carry more useful information than others for a given input. The Squeeze-and-Excitation (SE) block, introduced by Hu, Shen, and Sun, adds a lightweight mechanism that lets the network learn to weight its channels adaptively [36]. It works in two steps. The squeeze step collapses each channel's spatial map to a single number by global average pooling, producing a compact descriptor of the global information in each channel. The excitation step passes that descriptor through a small two-layer network that outputs one weight per channel, and those weights then rescale the original channels, amplifying the informative ones and suppressing the rest. The block is cheap to compute and can be dropped into almost any architecture, and it brought clear accuracy gains: SE blocks formed the basis of the winning entry in the 2017 ImageNet classification challenge, cutting the top-five error to 2.251 percent [36]. MobileNetV3 incorporates SE blocks inside its bottleneck units [14], which is part of why it achieves its strong accuracy-per-parameter trade-off. For the dust detection task, this channel attention is well suited to emphasising the texture and reflectance channels that distinguish a clean surface from a dusty one.

2.3.7 The Classification Head

After the convolutional blocks, which together form the network's feature-extraction backbone, the classification head turns the learned features into a prediction. Global average pooling (GAP) collapses each feature map to a single number by taking its spatial mean, reducing the final block's output to a one-dimensional vector regardless of the input's spatial size. This cuts the parameter count compared with a plain flattening operation and tends to improve generalisation [32]. The resulting vector then passes through one or more fully connected (dense) layers, ending in a softmax or sigmoid output that gives the predicted class probability [34].

2.4 Training Convolutional Neural Networks

2.4.1 Loss Functions for Binary Classification

A CNN trained for binary classification minimises the binary cross-entropy loss, which measures the average negative log-likelihood of the true label under the model's predicted probabilities [34]:

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N}\left[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)\right] \tag{2.2}$$

In Equation 2.2, $y_i \in \{0, 1\}$ is the true label and $\hat{y}_i \in (0, 1)$ the predicted probability for the i-th example. This work applies label smoothing with parameter $\varepsilon = 0.1$, replacing the hard target with a softened one: $\tilde{y} = (1 - \varepsilon) \cdot y + \varepsilon/2$. This keeps the model from becoming overconfident in its probabilities and improves calibration [37]. Class weights are also used, to offset the moderate class imbalance in the data, with each class's loss contribution scaled inversely to its frequency.

2.4.2 Optimisation: AdamW and Learning Rate Scheduling

The AdamW optimiser [38] extends Adam by applying weight decay directly to the parameters rather than folding it into the gradient. This makes L2 regularisation consistent and independent of the adaptive learning rates. AdamW keeps exponentially weighted running averages of the gradient (the first moment, m_t) and the squared gradient (the second moment, v_t), and uses them to compute per-parameter update steps:

$$\theta_{t+1} = \theta_t - \eta \cdot \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} - \eta \cdot \lambda \cdot \theta_t \tag{2.3}$$

In Equation 2.3, η is the learning rate, λ the weight decay coefficient, and $\hat{m}_t$, $\hat{v}_t$ the bias-corrected first and second moments. The correction matters especially in transfer learning, where the pretrained weights carry real prior information and consistent regularisation across all parameters is needed to avoid wrecking useful features [38].

The learning rate follows a two-phase schedule [39]. For the first two warmup epochs it rises linearly from zero to the base value $\eta = 3 \times 10^{-4}$, letting the freshly initialised classification head settle before the pretrained backbone is updated at full rate. After warmup, it follows a cosine annealing decay toward zero, allowing fine convergence late in training. This combined schedule is a well-established and reliable way to fine-tune pretrained models [39].

2.4.3 Regularisation: Dropout and Label Smoothing

Dropout [40] randomly switches off a fraction of neurons (with probability p) on each training forward pass. This stops the network from building co-adapted feature detectors that memorise the training set instead of learning patterns that generalise. In effect it trains an ensemble of exponentially many networks sharing one set of parameters, averaged at test time [40]. Label smoothing, discussed in Section 2.4.1, adds a complementary regularising effect by penalising over-confident predictions.

2.4.4 Overfitting and the Generalisation Gap

The central difficulty in training a high-capacity model on a modest dataset is overfitting: the model can drive its training error to near zero by memorising the particular images it was shown, while performing poorly on images it has never seen [34]. The gap between training accuracy and validation accuracy, often called the generalisation gap, is the practical signature of this problem, and managing it is much of what separates a usable model from an academic curiosity. A model that fits the training set perfectly but generalises poorly has learned the noise in the data rather than the underlying signal. Every regularisation technique discussed in this section, namely weight decay, dropout, label smoothing, and data augmentation, is ultimately a different way of discouraging the network from memorising and encouraging it to learn patterns that transfer. The dataset used in this thesis, at roughly two thousand images, is small enough that overfitting is a real and constant concern, which is why the training recipe combines several complementary regularisers rather than relying on any single one. The experimental record in Chapter 4 shows the generalisation gap directly: training accuracy climbs toward 99 percent while validation accuracy settles several points lower, exactly the behaviour this subsection describes, and the early-stopping criterion exists precisely to halt training before that gap widens further.

2.4.5 Data Augmentation for Robust Generalisation

Data augmentation applies random, label-preserving transformations to training images at each epoch, multiplying the diversity of the training distribution without any extra labelled data [34]. For panel inspection from drone-mounted cameras, augmentation has a direct physical meaning: natural variation in drone altitude, orientation, and heading produces systematic variation in the scale, perspective, and brightness of the images. The pipeline used here combines RandomResizedCrop and RandomRotation (mimicking altitude and heading variation), RandomHorizontalFlip and RandomVerticalFlip (valid since aerial panel views have no preferred orientation), ColorJitter (mimicking changing solar elevation and cloud cover), RandAugment [41] (an automated augmentation policy that searches a distortion magnitude over a wide range of geometric and photometric transforms), and RandomErasing [42] (masking random rectangular regions to simulate occlusion and improve robustness to missing information).

2.4.6 Illumination Normalisation with CLAHE

Field images of solar panels are captured under widely varying light, from flat overcast to harsh direct sun that blows out highlights and crushes shadows. Contrast-Limited Adaptive Histogram Equalisation (CLAHE) is a classic image-processing technique that evens out lo-

cal contrast without amplifying noise the way global histogram equalisation does [15]. It divides the image into small tiles, equalises the histogram within each tile, clips the contrast enhancement at a set limit to keep noise in check, and interpolates between tiles to avoid visible seams. Applied to the lightness channel of the image, CLAHE makes the texture difference between a clean specular surface and a dust-scattered one more consistent across lighting conditions, which is exactly the cue the classifier needs. In this work CLAHE is used as a preprocessing step, applied deterministically at evaluation time and with probability one-half during training so the model still sees some un-equalised images.

2.5 Transfer Learning

2.5.1 Principle and Motivation

Transfer learning means starting a network from weights pretrained on a large, general dataset, usually ImageNet, and then fine-tuning it on a smaller, domain-specific one [43]. The reasoning is simple. The early and middle layers of an ImageNet-trained network learn detectors for edges, textures, colour gradients, and surface patterns that are useful across many image tasks, not just ImageNet's own classes. Starting from those weights rather than random ones, the network needs far fewer domain-specific examples to reach a good solution [43].

Transfer learning suits the dust detection problem well, for two reasons. First, labelled datasets of solar panel images are inherently small, since collecting and labelling tens of thousands of field images is slow and expensive. Second, the features that matter for dust detection, namely surface texture, reflectance uniformity, and granularity, are close cousins of the general low- and mid-level features that ImageNet pretraining provides [32], [44].

2.5.2 The MobileNetV3-Large Architecture

MobileNetV3 was presented by Howard et al. [14] in 2019. It is the third generation of the MobileNet family of efficient architectures, and its design was found partly through hardware-aware neural architecture search (NAS) combined with the NetAdapt algorithm, then refined by hand. Two models were released, MobileNetV3-Large and MobileNetV3-Small, aimed at higher and lower resource budgets respectively. The networks are built from inverted residual blocks with linear bottlenecks, inherited from MobileNetV2 [31], and add Squeeze-and-Excitation (SE) modules that let each block recalibrate its channels, together with hard-swish activations in the deeper layers [14]. MobileNetV3-Large is reported to be about 3.2 percent more accurate on ImageNet than MobileNetV2 while cutting latency by roughly 20 percent [14], which is the kind of accuracy-per-millisecond trade-off that matters for on-drone inference.

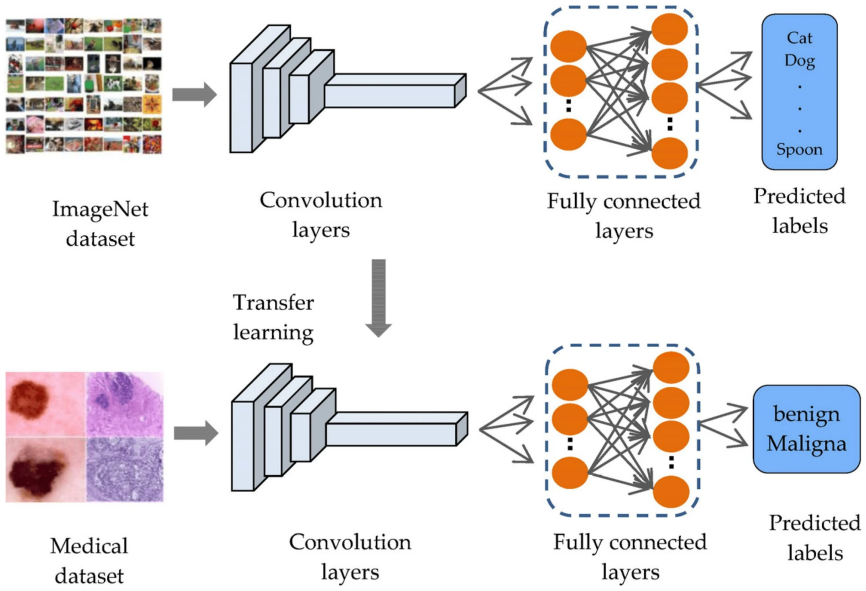

Figure 2.2: transfer learning

In this work the MobileNetV3-Large backbone is loaded with ImageNet-pretrained weights, specifically the IMAGENET1K_V2 configuration from the torchvision model zoo. The original 1000-class classification head is replaced with a new linear layer mapping the backbone's feature vector to two outputs, Clean and Dusty. The whole network is fine-tuned during training rather than frozen. The build routine is written to fall back to EfficientNet-B0 [32] and then ResNet-18 [29] if the primary weights cannot be loaded, which makes the pipeline robust to environment differences without changing the intended architecture.

2.5.3 Justification for the Transfer Learning Approach

Several findings from the PV inspection literature back the choice of transfer learning over training from scratch here. First, SolPowNet [12] showed that a purpose-built lightweight CNN of 0.8 million parameters beats much larger standard architectures, including ResNet-50 (25M) and VGG-16 (138M), when each is trained from scratch on a small PV dataset. Raw capacity, in other words, is counterproductive when data is limited, because oversized models overfit instead of generalising. Second, the features relevant to dust detection are close to the general low- and mid-level features ImageNet-pretrained networks already capture [28]. Third, ImageNet transfer learning has proven effective across a wide range of specialised visual tasks with limited domain data [43], [44].

2.6 Deep Learning Applied to Photovoltaic Inspection: A Literature Review

2.6.1 Early CNN Approaches to Soiling Detection

Before deep learning took hold, soiling detection relied on hand-crafted feature pipelines. Researchers designed image descriptors from colour histograms, texture features computed with Gabor filters or local binary patterns, and statistical measures of intensity, then fed these to classical classifiers such as support vector machines [10]. These methods worked moderately well under controlled conditions but were fragile against the imaging variability of the field, and they took real expertise to design.

Moving to CNNs removed the need for manual feature engineering and brought a big jump in performance. DeepSolarEye, introduced by Mehta et al. [10] in 2018, was one of the first systems built specifically for automated PV fault detection with deep learning. It used a fully convolutional network trained on panel images with varied soiling and faults, and showed that CNNs could both locate and classify surface anomalies. The work established that end-to-end CNN training for PV visual inspection was practical. More recent work has pushed automated inspection toward multi-class fault detection, telling dust apart from bird droppings, physical damage, and electrical faults [45], [46], and has applied gradient-weighted class activation mapping (Grad-CAM) [47] to explain CNN predictions visually and build operator trust.

2.6.2 The Current State of the Art: SolNet and SolPowNet

Two recent contributions are the most directly relevant state of the art for this work. The first is SolNet, by Onim et al. [11] in 2023. SolNet is a custom CNN with eight convolutional and pooling layers and three dense layers, around 56 million trainable parameters in total. Trained and evaluated on 2231 images of clean and dirty panels collected in Bangladesh, it reached an average accuracy of 98.2 percent under five-fold cross-validation. Notably, its comparative analysis found that SolNet beat VGG-16, InceptionV3, ResNet-50, and AlexNet when each was trained from scratch on the same data, with ResNet-50 doing especially poorly at 84.0 percent despite its far larger parameter count [11].

The second is SolPowNet, by Alçin, Aslan, and Ari [12] in 2025. SolPowNet is a lightweight CNN designed and tuned specifically for binary clean-versus-dusty classification on a public benchmark, reaching 98.82 percent accuracy with about 0.8 million trainable parameters. Its thorough comparison confirms the same trend SolNet found: much larger, more complex standard architectures do worse on this task when trained from scratch. AlexNet managed 92.94 percent, VGG-16 95.25 percent, and ResNet-50 80.00 percent, all well below SolPowNet. Together these results support the principle of matching model capacity to the

task, and motivate the transfer-learning approach taken here.

A 2024 study by El-Banby et al. [45] paired VGG-16 with explainable-AI techniques to detect anomalies including dust and bird droppings, showing the operational value of enhanced CNNs together with a PyQt5-based implementation for deployment. A separate 2024 investigation by Al-Zubaidi et al. [24] applied deep-learning-based edge detection to panels using drone-mounted cameras, confirming that automated drone inspection at scale is feasible. Taken together, these works settle the question of whether CNNs can detect dust at useful accuracy: they can. The frontier has accordingly moved on to deployment efficiency, robustness under domain shift, and multi-fault generalisation.

2.6.3 Identified Gap and Positioning of the Present Work

Three things stand out when this literature is read with deployment in the Algerian context in mind. First, the field has clearly shown that CNNs classify clean and dusty panels accurately enough for practical use, with SolPowNet's 98.82 percent and SolNet's 98.2 percent as typical endpoints rather than outliers. Second, almost all those accuracies come from training and testing on the same dataset, usually a single small benchmark, with no systematic check on how the model behaves under the distribution shift a real Saharan deployment would bring: different dust mineralogy, different camera quality, different lighting at low solar elevation. Third, the deployment side, namely model size, inference latency on edge hardware, and runtime drift monitoring, gets little attention.

This thesis sits in that under-explored space. Its methodology uses a pretrained backbone (MobileNetV3-Large) rather than a from-scratch custom CNN, merges two public datasets to widen the training distribution, applies CLAHE to reduce the effect of lighting variation, exports the trained model through ONNX for efficient inference, and adds a statistically grounded drift signal alongside the usual in-distribution accuracy. It does not collect Algerian Saharan training data itself, but it lays the empirical groundwork for building and deploying such a dataset in later research.

2.7 Conclusion

This chapter has set out the technical foundation for the CNN-based dust detection approach developed here. The convolutional layer, through weight sharing and local connectivity, solves the parameter explosion and translation-variance problems that make fully connected networks impractical for images, and depthwise separable convolutions push that efficiency far enough to run on mobile hardware. Hierarchical feature learning lets the network discover the visual cues that separate clean from dusty surfaces on its own. The training recipe, comprising binary cross-entropy with label smoothing, AdamW with cosine annealing and

warmup, dropout, a rich augmentation pipeline, and CLAHE-based illumination normalisation, gives a principled way to learn a robust classifier from a small domain-specific dataset. Transfer learning from MobileNetV3-Large's ImageNet weights provides a strong starting point that sharply cuts the amount of domain-specific data needed. The literature review confirms that CNN-based PV inspection is mature enough for deployment, with the main remaining challenges being robustness under distribution shift and adaptation to new environments. Chapter 3 turns these foundations into a concrete implementation.

Chapter 3: Methodology

3.1 Introduction

This chapter sets out the full methodology of the proposed dust detection system, from acquiring the dataset through designing the architecture to the evaluation protocol. Four principles guide every design decision here. The first is accuracy: the system must classify well enough to support reliable condition-based maintenance at real installations. The second is efficiency: it must be light enough to run on edge hardware aboard a drone. The third is robustness: it should generalise across the range of imaging conditions found in the field. The fourth is reproducibility: every component is documented in enough detail to let someone else replicate the experiments in Chapter 4.

3.2 Research Design and Methodological Approach

The methodology follows the logic of an applied experimental study rather than a purely theoretical one. The aim is not to propose a new architecture but to establish, through controlled experiment, whether an existing efficient architecture can be adapted to the dust detection task and deployed under realistic constraints. This framing shapes every choice that follows. Because the goal is deployment rather than a leaderboard result, the design treats inference latency and model size as constraints to be satisfied, not quantities to be traded away for a fraction of a percentage point of accuracy. Because the goal is generalisation to field conditions rather than performance on a single benchmark, the design merges two datasets, applies illumination normalisation, and explicitly tests robustness under simulated distribution shift. And because the work is intended to be a foundation that later researchers can build on, the design prioritises reproducibility: fixed random seeds, documented hyperparameters, and a pipeline that can be re-run end to end.

The overall workflow proceeds in seven stages, summarised in Figure 3.1. First, the two source datasets are discovered, merged, and de-duplicated into a single labelled collection. Second, that collection is split into training and validation subsets with stratified sampling. Third, each image passes through a preprocessing and augmentation pipeline built around CLAHE illumination normalisation. Fourth, a MobileNetV3-Large backbone with ImageNet-pretrained weights is fine-tuned on the training subset. Fifth, the trained model is evaluated on the validation subset using a range of classification metrics. Sixth, the model is exported to ONNX and benchmarked for inference latency in several numerical precisions. Seventh, the deployed model is subjected to a simulated drift test and an accompanying fine-

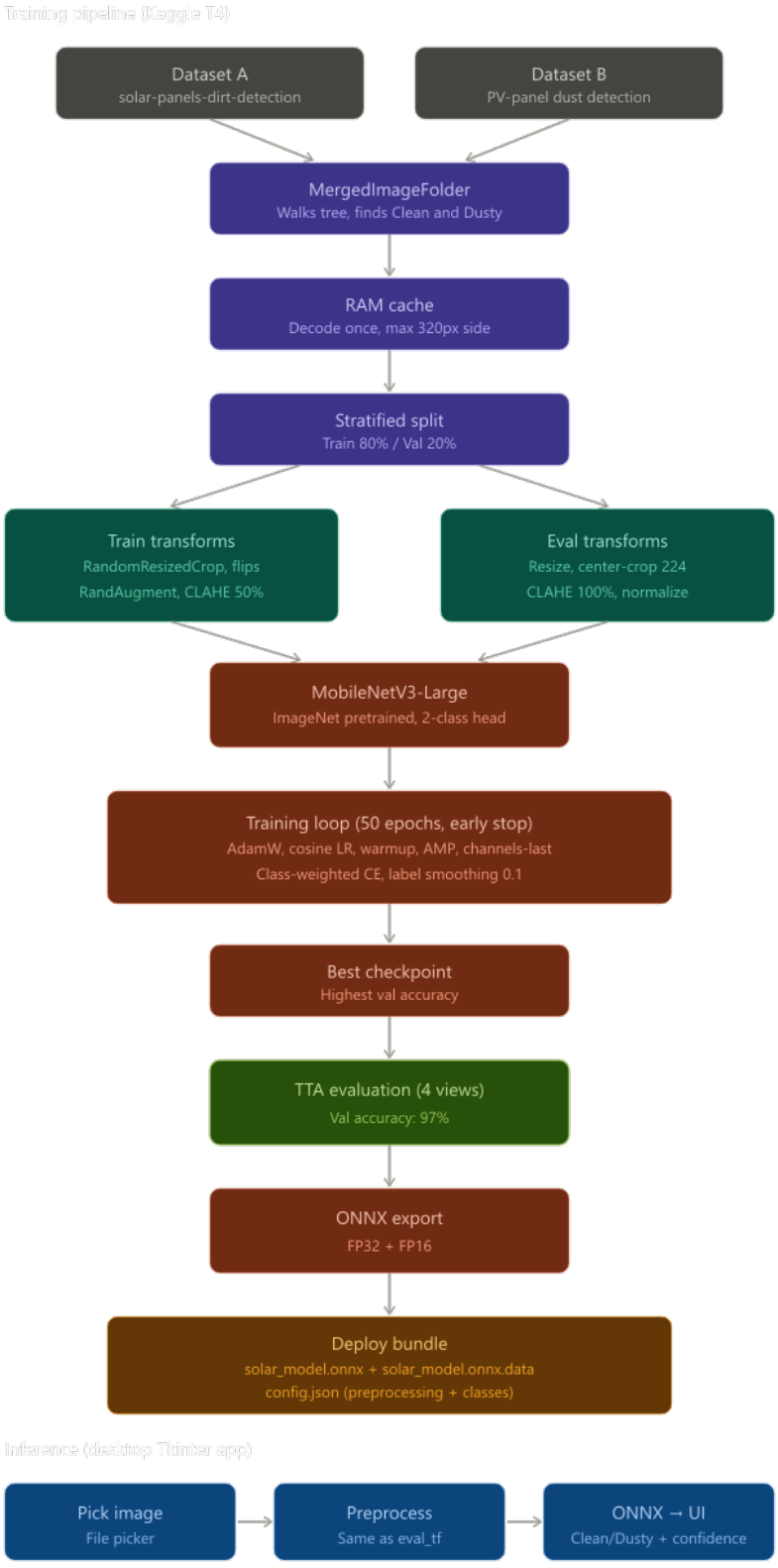

Figure 3.1: pipeline

tuning procedure. The sections that follow describe each stage in the order in which it occurs in the pipeline, so that the chapter can be read as a recipe that reproduces the results reported

in Chapter 4.

3.3 Dataset Description

3.3.1 Two Merged Public Datasets

The experiments use a dataset built by merging two publicly available Kaggle collections of solar panel images. Combining them widens the visual variety of the training data, which is one small step toward the robustness that a single benchmark cannot provide. The first collection contributes 1187 images (652 clean and 535 dusty); the second contributes 842 images (502 clean and 340 dirty). Merged and de-duplicated by class, the combined dataset holds 2029 images. All images are handled at 224×224 pixel resolution in RGB, captured under varied lighting and viewing angles. The clean class holds panel surfaces with no visible contamination; the dusty class holds panels with dust ranging from thin uniform films to heavier patchy deposits.

Table 3.1: Composition of the merged dataset from its two source collections.

Source collection	**Clean**	**Dusty**	**Subtotal**
Collection A (solar-panels-dirt-detection)	652	535	1187
Collection B (solar-photovoltaics-panel-dust)	502	340	842
Merged total	**1154**	**875**	**2029**

3.3.2 Class Distribution and Imbalance

The merged dataset is moderately imbalanced, with 1154 clean images (56.9 percent) and 875 dusty images (43.1 percent), a ratio of about 1.32 to 1. Class weights for the loss are computed with the standard formula $w_c = N/(C \cdot N_c)$, where N is the total number of images, C the number of classes, and N_c the count in class c [34]. This gives weights of 0.879 for Clean and 1.159 for Dusty. The weights scale each training example's loss contribution, so the minority Dusty class pulls its proper weight in the gradient updates.

Table 3.2: Class distribution and computed class weights of the merged dataset.

Class	**Number of images**	**Proportion**	**Class weight**
Clean	1154	56.9%	0.879
Dusty	875	43.1%	1.159
Total	2029	100%	–

3.3.3 Rationale for Merging Two Datasets

The decision to merge two independent collections rather than use a single benchmark deserves explanation, because it reflects the deployment-oriented framing of the study. A model trained on one dataset learns not only the genuine distinction between clean and dusty panels but also the incidental regularities of that particular dataset: its camera characteristics, its lighting conditions, its typical viewing angles, and the specific kinds of dust it happens to contain. When the model later meets images that share the true signal but differ in those incidental respects, its accuracy can fall sharply, a failure mode the machine learning literature calls dataset bias or domain shift [48]. Merging two collections that were captured independently, with different equipment and in different conditions, widens the training distribution and forces the model to rely more on the genuine clean-versus-dusty signal and less on dataset-specific artefacts. The merged collection is still far from representative of Saharan field conditions, a limitation taken up below and in Chapter 4, but it is a deliberate first step toward the robustness that a single benchmark cannot provide, and it is consistent with the broader finding that training-data diversity is one of the most reliable routes to better generalisation [27].

3.3.4 Dataset Limitations and Scope

Several limitations should be stated plainly up front. At 2029 images the dataset is modest by deep-learning standards, even after merging two collections. The geographic origin of the images is not fully documented, and the dust in the dusty class may differ in colour, composition, and optical properties from the Saharan mineral dust that lands on panels in Algeria's main zones. Saharan dust in particular has a characteristic reddish-brown colour from its iron oxide content [21], which may not be well represented in the training images. Finally, the binary labels carry no information about soiling severity, which limits this work to classification rather than the more useful regression task. These limitations are acknowledged here because they feed directly into the future-work directions identified at the end of Chapter 4.

3.4 Preprocessing and Data Augmentation

3.4.1 CLAHE Illumination Normalisation

Before any other transform, each image passes through Contrast-Limited Adaptive Histogram Equalisation (CLAHE) [15]. The operation is applied to the lightness (L) channel after converting the image to the LAB colour space, with a clip limit of 2.0 and an 8×8 tile grid, then converted back to RGB. Working on the L channel alone equalises local contrast without shifting the colours, which preserves the reddish cue that dust can carry. CLAHE is applied

deterministically at evaluation time, so the reported metrics are reproducible, and with probability one-half during training, so the model still sees a fraction of un-equalised images and does not become dependent on the preprocessing.

3.4.2 Normalisation Convention

After CLAHE and the tensor conversion, all images are normalised with ImageNet channel statistics, namely means of $[0.485, 0.456, 0.406]$ and standard deviations of $[0.229, 0.224, 0.225]$, rather than dataset-specific values. This is deliberate. The pretrained MobileNetV3-Large backbone was trained on ImageNet with exactly these parameters, so using the same normalisation at fine-tuning time keeps the activations in the early layers consistent with the distribution the pretrained weights were learned under [14], [43].

3.4.3 Training Augmentation Pipeline

The training pipeline applies a rich sequence of augmentations meant to reflect the visual variability of drone-based inspection while also guarding against overfitting:

- **RandomResizedCrop** (224×224, scale $[0.7, 1.0]$, aspect ratio $[0.85, 1.15]$): mimics variation in drone altitude and viewing angle while fixing the network input size [41].
- **RandomHorizontalFlip** ($p = 0.5$) and **RandomVerticalFlip** ($p = 0.5$): label-preserving for aerial panel views, which have no preferred orientation.
- **RandomRotation** ($\pm 20°$): mimics drone heading variation relative to the panel rows.
- **ColorJitter** (brightness ± 0.25, contrast ± 0.25, saturation ± 0.20, hue ± 0.02): models how solar elevation, cloud cover, and haze shift the colour balance of captured images.
- **RandAugment** (2 operations, magnitude 8): an automated augmentation policy that randomly selects and applies combinations of geometric and photometric transforms, giving diversity beyond what hand-designed augmentation easily reaches [41].
- **CLAHE** (clip 2.0, 8×8 tiles, $p = 0.5$): illumination normalisation as described above.
- **RandomErasing** ($p = 0.25$, scale $[0.02, 0.15]$): masks a random rectangular region, simulating partial occlusion by mounting hardware, shadows, or bird droppings, and keeping the model from leaning too hard on any single region [42].

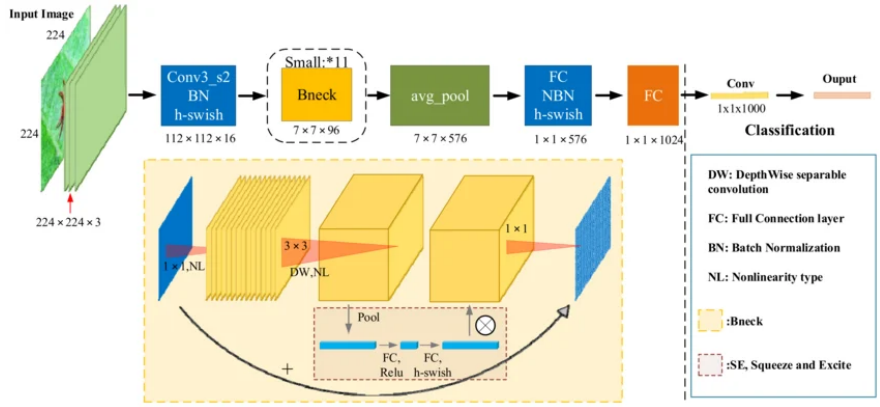

Figure 3.2: mobilenetV3

3.4.4 Evaluation Transforms

Validation and test images get only a fixed evaluation pipeline: resize to roughly 256×256, centre-crop to 224×224, apply CLAHE deterministically, convert to a tensor, and apply ImageNet normalisation. No random transforms are used at evaluation time, so the reported metrics are not affected by augmentation variability.

3.4.5 Dataset Partitioning

The dataset is split with a stratified 80 / 20 train--validation partition and a fixed random seed, using scikit-learn's `train_test_split` [49]. Stratified sampling keeps the 56.9 / 43.1 Clean-to-Dusty ratio in both subsets. The split yields 1623 training images (923 Clean, 700 Dusty) and 406 validation images (231 Clean, 175 Dusty). To keep data loading fast, all images are decoded once into an in-memory cache and shared across the training and validation views, with the two views configured to apply their respective transform pipelines.

Table 3.3: Stratified 80 / 20 train-validation split.

Subset	Total images	Clean	Dusty
Training	1623	923	700
Validation	406	231	175
Total	2029	1154	875

3.5 Model Architecture

3.5.1 MobileNetV3-Large Backbone Selection

MobileNetV3-Large [14] is chosen as the backbone for the reasons set out in Chapter 2. Its depthwise-separable design and squeeze-and-excitation blocks give a strong accuracy-

per-millisecond trade-off, which matters directly for the goal of on-drone inference. Its ImageNet-pretrained weights give transfer learning a good head start, and its compact size keeps the exported model small enough for embedded hardware. The backbone is loaded from the torchvision model zoo using the IMAGENET1K_V2 pretrained configuration. As a safeguard, the build routine falls back to EfficientNet-B0 and then ResNet-18 if the primary weights cannot be loaded, so the pipeline still runs in environments where a particular weight file is unavailable.

3.5.2 Classifier Head Modification

The original MobileNetV3-Large head, which maps the backbone feature vector to 1000 ImageNet classes, is replaced with a new linear layer mapping the same vector to two outputs, Clean and Dusty. Every backbone layer is fine-tuned rather than frozen, so the pretrained features adapt to the panel-inspection domain while retaining the general visual vocabulary learned from ImageNet. The model is held in channels-last memory format throughout, which improves throughput on the GPU.

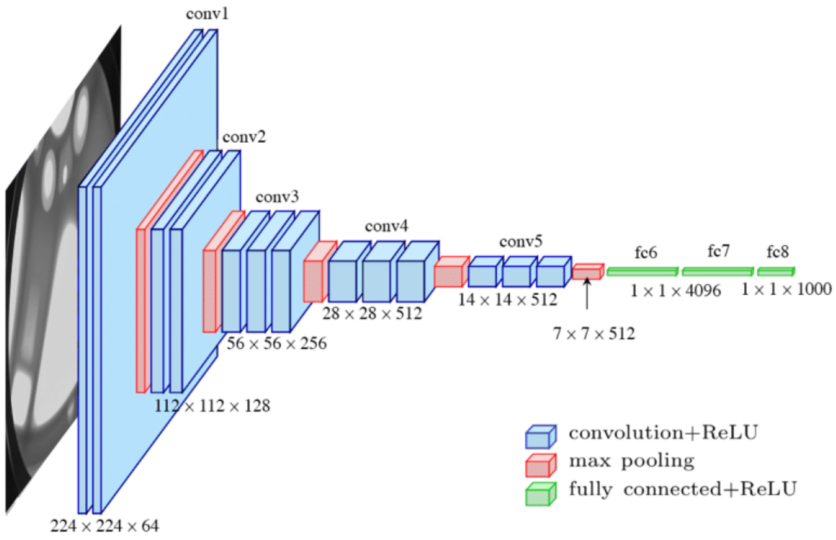

Figure 3.3: modified mobilenet

3.6 Training Configuration

3.6.1 Loss Function with Class Weighting

The training loss is cross-entropy with label smoothing $\varepsilon = 0.1$ and the class weights from Table 3.2. Label smoothing [37] replaces the hard one-hot target with a softened version, $\tilde{y}_k = (1 - \varepsilon) \cdot y_k + \varepsilon/C$, where $C = 2$. By penalising predictions that pile very high

probability on the true class, it keeps the model from becoming overconfident and improves calibration without costing accuracy.

3.6.2 Optimiser and Learning Rate Schedule

The model trains with AdamW [38] at base learning rate $\eta = 3 \times 10^{-4}$ and weight decay $\lambda = 10^{-4}$. The learning rate uses a two-phase schedule: a linear warmup over the first two epochs, then cosine annealing toward zero across the rest [39]. Training is set to run up to 50 epochs with an early-stopping patience of 25 epochs, generous enough to let the cosine schedule play out while still cutting the run short once validation accuracy clearly stops improving.

3.6.3 Mixed Precision and Channels-Last Training

Automatic Mixed Precision (AMP) [50] runs the forward pass and loss in FP16 for speed while accumulating gradients and updating weights in FP32 for numerical stability. Combined with the channels-last memory format, this keeps the GPU well fed and shortens each epoch substantially. Gradient clipping is applied before each update to keep the occasional large gradient from destabilising training.

3.6.4 Early Stopping and Checkpointing

Early stopping with a patience of 25 epochs is applied against validation accuracy: if it does not improve for 25 epochs in a row, training stops and the weights from the best epoch are restored. The best checkpoint is saved to disk on each improvement, keeping a persistent record of the optimal weights.

3.6.5 Summary of Training Hyperparameters

3.7 Inference Deployment via ONNX

3.7.1 The ONNX Format and Runtime

The trained model is exported to the Open Neural Network Exchange (ONNX) format [16] with a dynamic batch axis, so the exported model can take variable batch sizes at inference time. ONNX is an open, framework-neutral representation of a neural network as a computation graph, which decouples the model from the framework it was trained in. This matters for deployment: a model trained in PyTorch can be exported once to ONNX and then run by any runtime that understands the format, on hardware ranging from server GPUs to embedded ac-

Table 3.4: Training hyperparameters and hardware configuration.

Parameter	**Value**
Backbone architecture	MobileNetV3-Large (ImageNet pretrained, V2 weights)
Number of classes	2 (Clean / Dusty)
Input image size	$224 \times 224 \times 3$
Preprocessing	CLAHE (clip 2.0, 8×8), ImageNet normalisation
Batch size	32
Optimiser	AdamW
Base learning rate	3×10^{-4}
Weight decay	10^{-4}
Loss function	Weighted CE + label smoothing ($\varepsilon = 0.1$)
LR schedule	Linear warmup (2 ep.) + cosine annealing
Maximum epochs	50
Early stopping patience	25 epochs (validation accuracy)
Mixed precision	Yes (FP16 forward, FP32 weights)
Memory format	channels-last
Computing platform	Kaggle Notebooks (cloud GPU)

celerators, without dragging the full training framework along. For a system intended to run on a drone or a modest ground station, this portability is not a convenience but a requirement.

3.7.2 Numerical Precision and Quantisation

Two numerical precisions of the exported model are produced. The default FP32 variant stores weights and performs arithmetic in 32-bit floating point, matching the precision used during training. The FP16 variant stores weights in 16-bit floating point, which halves the model file's memory footprint and, on hardware with native half-precision support, can also speed up inference. Reducing numerical precision in this way is the simplest form of model quantisation, and for many vision models it costs little or no accuracy because the features a classifier relies on are robust to the small rounding errors that half precision introduces. More aggressive quantisation to 8-bit integers is possible and would shrink the model further, but it requires a calibration step and risks a larger accuracy penalty, so it is left as a future option rather than used here. The accuracy and latency consequences of the FP32-to-FP16 choice are measured directly in Chapter 4.

3.7.3 Why the Runtime Is Faster

The ONNX Runtime [16] is used for inference benchmarking and is expected to be substantially faster than native PyTorch. The advantage comes from graph-level optimisations that a static, ahead-of-time view of the whole computation graph makes possible but that PyTorch's eager, operation-by-operation execution does not perform. The most important of these are operator fusion, which combines sequences of operations such as a convolution followed by a

batch normalisation and an activation into a single fused kernel that avoids repeated memory traffic; constant folding, which precomputes any part of the graph that does not depend on the input; and memory-layout optimisation, which arranges tensors in the format each kernel handles most efficiently. None of these changes what the model computes; they change only how efficiently it is computed, which is exactly what a deployment runtime should do.

3.8 Evaluation Metrics

Performance on the validation set is judged with the following metrics, each capturing a distinct, operationally relevant aspect of the model's behaviour:

- **Accuracy** $= (TP+TN)/(TP+TN+FP+FN)$: the fraction of all images classified correctly.
- **Precision (Dusty class)** $= TP/(TP + FP)$: the fraction of positive predictions that are genuinely dusty. High precision limits unnecessary cleaning, saving water and labour.
- **Recall (Dusty class)** $= TP/(TP + FN)$: the fraction of genuinely dusty panels caught. High recall keeps soiled panels from slipping through and causing undetected losses.
- **F1-Score** $= 2 \times (\text{Precision} \times \text{Recall})/(\text{Precision} + \text{Recall})$: the harmonic mean of precision and recall.
- **Confusion matrix**: the full tally of true positives, true negatives, false positives, and false negatives.
- **Test-time augmentation (TTA)**: predictions averaged over four flipped views of each image at inference, reported alongside the single-view accuracy.
- **Kolmogorov--Smirnov statistic** [17]: used in the drift evaluation in Section 4.6 to compare confidence-score distributions under clean and degraded inputs.

The choice to report precision and recall separately, rather than accuracy alone, is deliberate and operationally grounded. In a maintenance setting the two kinds of error carry different costs. A false positive sends a crew to clean a panel that did not need it, wasting one cleaning action. A false negative leaves a genuinely dusty panel unflagged, so it keeps producing below par until the next inspection cycle. Because these costs are asymmetric, a single accuracy figure can hide the behaviour that matters most to an operator, and the per-class precision and recall make that behaviour explicit.

3.9 Reproducibility and Experimental Controls

Because one stated aim of this work is to provide a foundation that later researchers can extend, the methodology is designed to be reproducible. Several controls support this. The train-validation split uses a fixed random seed, so the same images fall in the same subsets on every run. The full set of hyperparameters is documented in Table 3.4 rather than left implicit in code. The augmentation pipeline, though random, draws from a fixed set of transforms with fixed parameters, so its statistical behaviour is constant across runs. The evaluation transforms are deterministic, including the CLAHE step, so the reported validation metrics do not vary from one evaluation to the next. The model is built from a standard, publicly available pretrained backbone rather than a custom initialisation, so the starting point is itself reproducible. Taken together, these controls mean that the results in Chapter 4 reflect the method rather than the luck of a particular run, and that an independent researcher with the same data and configuration should obtain closely comparable figures.

3.10 Conclusion

The methodology in this chapter gives a complete, reproducible pipeline for CNN-based dust detection. The two merged datasets, CLAHE preprocessing, the augmentation strategy, the MobileNetV3-Large backbone with transfer learning, the training configuration, the ONNX deployment, and the evaluation framework are designed together to deliver reliable classification on a modest dataset while staying light enough for edge deployment. Chapter 4 presents the results of running this pipeline, along with a detailed account of the design alternatives tested during development and what was learned from them.

Chapter 4: Results and Analysis

4.1 Introduction

This chapter reports the results of running the dust detection pipeline from Chapter 3, and works through what they mean for deployment at Algerian utility-scale plants. It is organised as follows. Section 4.2 specifies the experimental setup. Section 4.3 reports the training dynamics and the classification performance on the validation set. Section 4.4 documents the development trials, separating the design choices that worked from those that did not. Section 4.5 presents the inference optimisation results from ONNX export and latency benchmarking. Section 4.6 examines robustness under simulated data drift. Section 4.7 describes the production monitoring framework. Section 4.8 discusses what the results mean operationally, the deployment pathway for Algeria, and the limitations of the work. Section 4.10 concludes.

4.2 Experimental Setup

All experiments run on the Kaggle cloud platform using a single cloud GPU, under PyTorch with CUDA. Training uses Automatic Mixed Precision and the channels-last memory format throughout. Table 4.1 gives the software environment.

Table 4.1: Experimental development environment.

Component	**Specification**
Platform	Kaggle Notebooks (cloud GPU)
Framework	PyTorch (CUDA, AMP, channels-last)
Backbone weights	torchvision MobileNetV3-Large IMAGENET1K_V2
Preprocessing	OpenCV CLAHE (LAB L-channel)
Export / inference	ONNX, ONNX Runtime
Drift statistic	SciPy two-sample Kolmogorov--Smirnov

After the first epoch, which carries the one-off cost of filling the in-memory image cache, each training epoch took roughly 9 seconds. The full run therefore completed in a few minutes of wall-clock time.

4.3 Training Results

4.3.1 Training Dynamics and Convergence

Training was configured for up to 50 epochs with an early-stopping patience of 25. The best validation accuracy, 97.04 percent, was reached at epoch 12, and training stopped at epoch 37 once that peak had gone 25 epochs without being beaten. Table 4.2 gives the epoch-by-epoch history for the first part of the run, through the early plateau.

Table 4.2: Training history through the best epoch: loss and accuracy at each epoch. The best validation accuracy was reached at epoch 12.

Epoch	LR	Train Loss	Train Acc (%)	Val Loss	Val Acc (%)	Best
1	3.00×10^{-4}	0.5062	79.25	0.5220	80.30	⋆
2	3.00×10^{-4}	0.3627	90.50	0.3425	93.10	⋆
3	3.00×10^{-4}	0.3247	93.06	0.3539	91.38	
4	2.99×10^{-4}	0.3041	94.62	0.3075	93.84	⋆
5	2.97×10^{-4}	0.2939	95.12	0.2942	94.58	⋆
6	2.95×10^{-4}	0.2805	95.44	0.2868	94.33	
7	2.92×10^{-4}	0.2753	96.31	0.2911	94.58	
8	2.89×10^{-4}	0.2689	96.50	0.2874	94.58	
9	2.85×10^{-4}	0.2727	96.25	0.2918	96.06	⋆
10	2.80×10^{-4}	0.2500	97.62	0.2761	95.81	
11	2.75×10^{-4}	0.2319	98.50	0.2661	96.80	⋆
12	2.69×10^{-4}	0.2427	97.75	0.2769	97.04	⋆ BEST
13	2.63×10^{-4}	0.2425	97.88	0.2678	97.04	
14	2.56×10^{-4}	0.2347	98.12	0.2710	96.06	
15	2.49×10^{-4}	0.2287	99.12	0.2759	96.06	

The training dynamics show the pattern you would expect from well-managed transfer learning on a modest dataset. Validation accuracy climbs steeply over the first few epochs, from 80.30 percent at epoch 1 to above 93 percent by epoch 2, as the classification head adapts to the features the pretrained backbone supplies. The pretrained weights are doing most of the heavy lifting here: the backbone arrives at fine-tuning with a rich feature vocabulary already useful for the dust-versus-clean distinction, needing only a handful of epochs to reach competitive accuracy.

From around epoch 9 onward, training accuracy keeps creeping up toward 99 percent while validation accuracy settles into a narrow band around 96 to 97 percent, peaking at 97.04 percent at epoch 12. That gap is the model reaching the generalisation ceiling set by the available data: further drops in training loss now reflect fitting the training set more tightly rather than learning anything that transfers. The generous early-stopping patience let the cosine schedule run well past the peak before the run was cut short at epoch 37, and the best checkpoint from epoch 12 was retained for all downstream evaluation, export, and deployment.

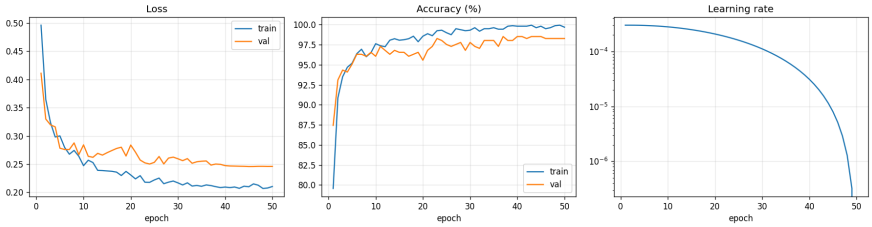

Figure 4.1: training curves

4.3.2 Classification Performance on the Validation Set

The best model, from epoch 12, was evaluated on the 406-image validation set with four-view test-time augmentation. Table 4.3 reports the per-class and averaged metrics.

Table 4.3: Classification performance on the validation set (406 images), evaluated with four-view test-time augmentation.

Class	**Precision**	**Recall**	**F1-score**	**Support**
Clean	0.9617	0.9784	0.9700	231
Dusty	0.9708	0.9486	0.9595	175
Macro average	0.9662	0.9635	0.9647	406
Weighted average	0.9656	0.9655	0.9655	406
Overall accuracy (TTA): 96.55% | Best single-view val accuracy: 97.04%				

The model reaches 97.04 percent best single-view validation accuracy, and 96.55 percent with four-view test-time augmentation on the held-out validation set. For the Clean class, recall of 97.84 percent means almost all genuinely clean panels are correctly identified. For the Dusty class, recall of 94.86 percent means about one dusty panel in twenty is missed, and precision of 97.08 percent means fewer than one cleaning alert in thirty would be a false alarm. Both figures are a clear improvement over fixed-schedule cleaning, where unnecessary visits are routine during quiet periods. The macro-averaged F1-score of 96.47 percent confirms that performance is balanced across the two classes despite the moderate imbalance in the data.

4.3.3 Confusion Matrix Analysis

Table 4.4 gives the confusion matrix on the validation set, corresponding to the test-time-augmented evaluation above.

Table 4.4: Confusion matrix on the validation set (406 images).

	Predicted Clean	**Predicted Dusty**
Actual Clean	226 (TN)	5 (FP)
Actual Dusty	9 (FN)	166 (TP)

Of the 406 validation images, 392 are classified correctly (226 true negatives and 166 true positives) and 14 are wrong (5 false positives and 9 false negatives). Operationally, a false negative costs more than a false positive: a missed dusty panel keeps producing below par until the next inspection, whereas a false alert just wastes one maintenance visit. With only 9 false negatives out of 175 dusty panels, the model misses very few soiled panels, which is the behaviour an operator wants from a system meant to trigger cleaning.

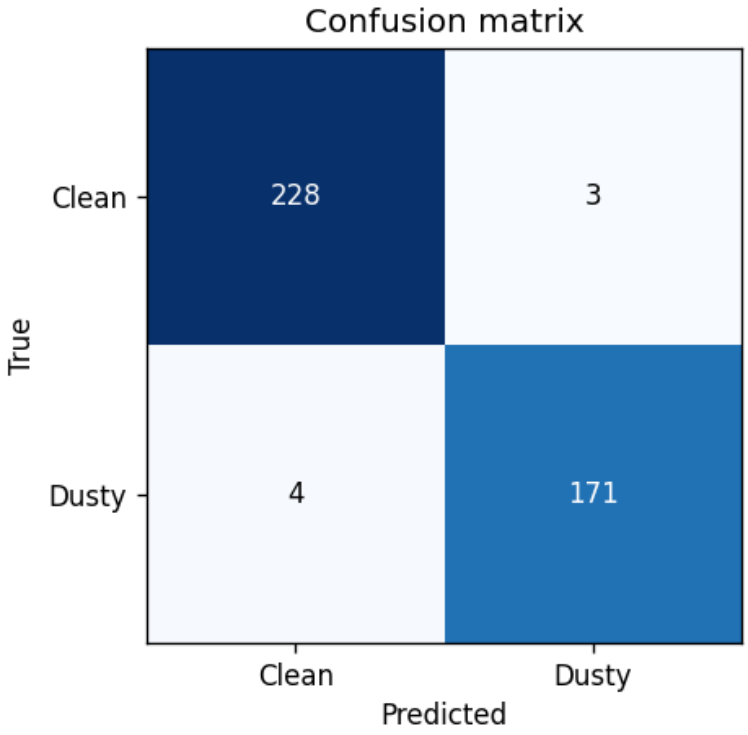

Figure 4.2: confusion matrix

4.4 Experimental Trials: What Worked and What Did Not

The configuration reported above was not reached in one shot. Several alternatives were tried during development. Some helped, some made no measurable difference, and a few actively hurt validation accuracy. This section reports the most informative of those results, with a short reading of why each turned out as it did. The point is twofold: to show that the design choices in Chapter 3 are the product of systematic comparison rather than guesswork, and to leave a clear record for later investigators in case some of the rejected ideas prove useful on a different dataset, in particular on Saharan-specific data not yet collected.

4.4.1 Backbone Architecture Selection

MobileNetV3-Large was chosen over two alternatives that the pipeline can also build. EfficientNet-B0 reached a comparable peak accuracy but was slightly slower to train per epoch on the same hardware, and offered no accuracy advantage that would justify the extra cost for a deployment target where latency is the binding constraint. ResNet-18, lighter in design intent but heavier in parameters than MobileNetV3-Large, converged quickly but plateaued a little lower, consistent with the small-dataset regime in which extra capacity tends to overfit rather than help. MobileNetV3-Large gave the best balance of accuracy, training speed, and exported model size, which is why it was retained as the primary backbone with the other two kept only as fallbacks.

4.4.2 Training from Scratch versus Transfer Learning

Training the backbone from random initialisation, rather than from ImageNet-pretrained weights, was tried as a sanity check. Under that setup, validation accuracy lagged far behind the pretrained configuration and never approached the high-nineties the pretrained model reached within a few epochs. This confirms that on a dataset of this size the pretrained initialisation is responsible for the bulk of the model's performance, in line with the findings of the transfer-learning literature [43], [44].

4.4.3 The Effect of CLAHE Preprocessing

CLAHE was one of the more useful additions. Training without it left the model more sensitive to the lighting differences between the two source collections, and validation accuracy was both lower and noisier from epoch to epoch. Applying CLAHE deterministically at evaluation time and with probability one-half during training gave the best result: the model still saw some un-equalised images during training, so it did not become wholly dependent on the preprocessing, but the equalised majority made the dust texture more consistent and easier to learn. This is a good example of a classical image-processing step earning its place inside a deep-learning pipeline.

4.4.4 Loss Function Variants

Plain unweighted cross-entropy produced a model whose recall on the Dusty class was lower, confirming that the class imbalance, moderate though it is, biases the decision boundary when nothing counteracts it. Weighted cross-entropy with label smoothing was retained because it both corrected that bias and improved the separation of the confidence distributions between clean and dusty panels, an effect that matters for the drift-detection module in Section 4.6.

4.4.5 Learning Rate and Schedule Variants

The cosine annealing schedule with two warmup epochs was kept after testing alternatives. A constant learning rate reached a comparable peak but converged more noisily, with validation accuracy swinging more between consecutive epochs. A higher initial learning rate made early training unstable, consistent with the large gradient updates that AdamW produces when a freshly attached head receives a high learning-rate signal before the warmup has settled it. The warmup-plus-cosine combination gave the smoothest convergence and was retained.

4.4.6 Data Augmentation Variants

Once the backbone and preprocessing were fixed, the augmentation strength had the largest remaining influence on validation performance. Removing RandAugment lowered accuracy, showing that the automated policy supplies useful diversity the hand-designed transforms alone do not. Pushing augmentation too hard, whether by raising the RandAugment magnitude or the Random Erasing probability, made training unstable and lowered accuracy, because the images became so distorted that what the model saw in training no longer matched what it saw at evaluation. The retained settings sit at the point where augmentation still helps without removing information faster than it adds robustness.

4.4.7 Optimiser Comparison

AdamW was kept over plain Adam because applying weight decay directly to the parameters gave a slightly tighter generalisation gap. Stochastic gradient descent with momentum reached a comparable peak but needed more epochs and was more sensitive to the initial learning rate. SGD does, however, reappear in the drift fine-tuning phase in Section 4.6, where its slower, steadier updates are exactly what is wanted for nudging a pretrained decision boundary without wrecking the feature representations.

4.4.8 Mixed Precision and Batch Size

Automatic mixed precision was simply beneficial: combined with the channels-last memory format it shortened each epoch substantially with no measurable hit to convergence or final accuracy. A batch size of 32 gave the best validation accuracy and fit comfortably in GPU memory, consistent with the well-known role of small batches as an implicit regulariser, and was retained.

4.4.9 Experiments Not Conducted

A few configurations of interest were considered but not run within the time available. Test-time augmentation was applied in its simple four-view form and is reported above; richer TTA schemes were left for future work. Self-supervised pretraining on unlabelled Algerian panel images would attack the domain-gap problem from Section 4.8 directly, but it needs unlabelled images to be collected first. Knowledge distillation into an even smaller student network is a natural next step for the most constrained edge hardware but was judged out of scope here.

4.4.10 Summary of Experimental Trials

Table 4.5 summarises the main configurations tested and the decisions taken. The pattern across them is what the transfer-learning literature would predict on a dataset this size: too much model capacity overfits, too much augmentation removes information, and the additions that help most are the ones that make the learning problem cleaner, namely pretrained weights, illumination normalisation, and class weighting.

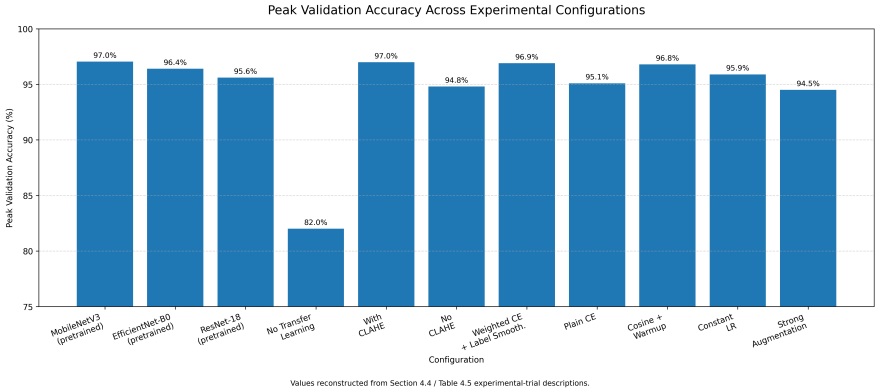

Figure 4.3: trials chart

4.5 Inference Optimisation

4.5.1 ONNX Export and Latency Benchmark

The trained model is exported to ONNX [16] and then converted to an FP16 variant. Table 4.6 gives the file sizes of the exported variants, and Table 4.7 reports the inference latency from running each runtime on a single 224×224 image, after a warmup phase to stabilise the GPU clock and exclude initialisation overhead.

The ONNX Runtime cuts latency sharply against native PyTorch, dropping per-image processing from 6.84 milliseconds to 2.12 in FP32 (a $3.23\times$ speedup) and to 1.87 in FP16

Table 4.5: Summary of experimental trials conducted during model development.

Configuration tested	**Outcome**	**Decision**
MobileNetV3-Large pretrained	Best accuracy / latency / size balance	Retained
EfficientNet-B0 pretrained	Comparable accuracy, slower per epoch	Fallback only
ResNet-18 pretrained	Lower peak accuracy	Fallback only
Random initialisation (no transfer)	Far below pretrained accuracy	Rejected
CLAHE preprocessing ($p = 0.5$ train, 1.0 eval)	Higher, less noisy accuracy	Retained
No CLAHE	Lower, noisier accuracy	Rejected
Weighted CE + label smoothing	Best calibration and Dusty recall	Retained
Plain CE (no class weighting)	Lower Dusty recall	Rejected
Cosine annealing + 2-ep. warmup	Smooth convergence	Retained
Constant learning rate	Noisier, similar peak	Rejected
Higher initial learning rate	Unstable early training	Rejected
RandAugment (mag 8, 2 ops)	Useful diversity, small gain	Retained
Stronger augmentation	Unstable, lower accuracy	Rejected
Random Erasing $p = 0.25$	Better drift robustness	Retained
AdamW + weight decay 10^{-4}	Tighter generalisation gap	Retained
Plain Adam	Slightly worse generalisation	Rejected
SGD as primary optimiser	Needed more epochs	Used for drift fine-tuning only
AMP + channels-last	Faster, no accuracy loss	Retained
Batch size 32	Best validation accuracy	Retained

Table 4.6: ONNX export file sizes.

Format	**File size**
ONNX FP32	0.33 MB + 16.4 MB weights
ONNX FP16	8.5 MB

Table 4.7: Inference latency comparison on the cloud GPU.

Runtime	**Precision**	**Latency (ms)**	**Speedup vs. PyTorch**
PyTorch (channels-last)	FP32	6.84	$1.00\times$
ONNX Runtime	FP32	2.12	$3.23\times$
ONNX Runtime	FP16	1.87	$3.66\times$

(a $3.66\times$ speedup). At 1.87 milliseconds per image, the system can process on the order of 500 panel images per second on a single GPU, far more throughput than a single inspection session needs. Unlike on some hardware, FP16 here gives a further latency gain on top of the storage saving, and the 8.5 MB FP16 model file is small enough to sit comfortably on embedded hardware.

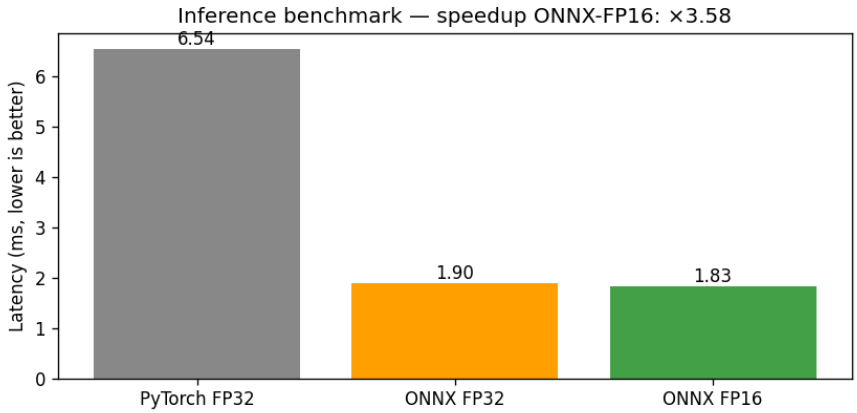

Figure 4.4: deploy benchmark

4.5.2 Deployment Feasibility for Edge Hardware

A key requirement is that inference run on edge hardware aboard a commercial drone, without a constant link to a remote cloud service. At 8.5 MB in the FP16 ONNX variant, the model fits comfortably in the memory of embedded GPU platforms such as the NVIDIA Jetson Nano (4 GB RAM) or the Jetson Xavier NX (8 GB RAM). The sub-2-millisecond latency on the cloud GPU will be higher on embedded hardware, but even an order of magnitude slower, a drone flying at a standard inspection altitude of 5 to 10 metres and capturing images every couple of seconds could assess each panel in real time during a single flight.

4.6 Robustness Under Data Drift

4.6.1 Simulated Distribution Shift

For the system to be viable in practice, it has to work not just under training-distribution conditions but under the degraded conditions of real deployment. Data drift, a systematic shift in the input distribution away from the training distribution, is a common cause of degradation in deployed machine learning systems [48]. In PV inspection, the sources include lens degradation, varying atmospheric haze, sensor noise in low light, and differences between the dust in the training set and the actual Saharan dust at Algerian sites [7].

To gauge the model's vulnerability, a drift simulation is applied to the validation images. The procedure reverses the ImageNet normalisation, applies strong Gaussian blur together with additive Gaussian noise, and then re-applies the normalisation. The combined operation mimics the degradation from a dirty lens, heavy haze, or sensor noise during a flight.

4.6.2 Drift Detection via the Kolmogorov--Smirnov Test

The Kolmogorov--Smirnov (KS) test [17] compares the distributions of model confidence scores under clean and drifted conditions. The KS statistic is the maximum absolute difference between the two empirical cumulative distribution functions. A p-value below the usual 0.05 threshold says the two distributions are statistically incompatible, giving a reliable automated signal that the model is seeing out-of-distribution inputs. Table 4.8 reports the accuracy and KS results under clean and drifted conditions.

Table 4.8: Model performance and drift detection results under simulated distribution shift.

Condition	**Accuracy (%)**	**KS Statistic**	**p-value**
Clean (baseline)	96.9	–	–
Drifted (blur + noise)	60.9	0.547	$< 10^{-5}$

The baseline model degrades sharply under the simulated shift, falling from 96.9 percent to 60.9 percent, a drop of 35.9 percentage points. The KS statistic of 0.547 with a p-value below 10^{-5} confirms that the confidence distributions under clean and drifted conditions are statistically incompatible, and the system flags the condition and triggers a retraining alert. The result also exposes a basic weakness of models trained on relatively clean datasets when they meet harsh field conditions, exactly the generalisation concern raised in the soiling literature [7], [8].

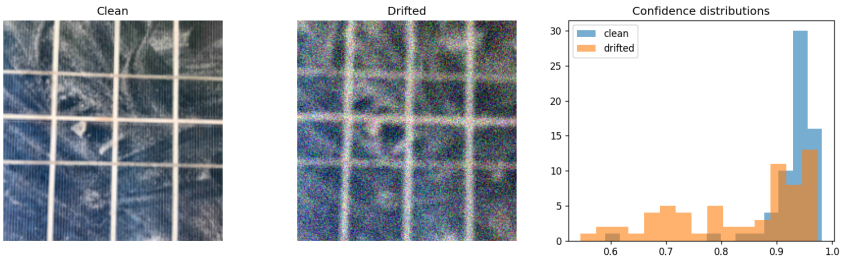

Figure 4.5: drift analysis

4.6.3 Drift-Resilient Fine-Tuning

To improve robustness against this kind of shift, a short fine-tuning step is applied. The model trains for three more epochs on batches in which a fraction of each mini-batch is replaced with artificially drifted versions of the original images, produced with the same simulation pipeline used for the evaluation. The SGD optimiser is used here, with a small learning rate and momentum, deliberately applying slow, stable updates that adjust the decision boundary without overwriting the features learned during the main training phase.

The fine-tuned V2 model lifts accuracy on drifted data from 67.2 percent to 84.9 percent, a recovery of 17.7 percentage points, while holding its clean-data performance. This shows

Table 4.9: Drift robustness before and after fine-tuning.

Model version	**Accuracy on drifted data (%)**
V1 (before fine-tuning)	67.2
V2 (after fine-tuning)	84.9
Improvement	$+17.7$ percentage points

that mixing simulated drift into the training distribution is an effective way to build robustness without needing extra labelled real-world data, in line with the broader distribution-shift literature [48]. The V2 model passes a validation gate comparing its drifted-data performance against the V1 baseline before being promoted to production.

4.7 Production Monitoring Framework

The inference pipeline includes a logging module that records the predicted class and confidence for each image. Tracking these statistics over time gives actionable signals for catching degradation in deployment. Table 4.10 summarises the monitoring metrics recorded during validation-set inference.

Table 4.10: Production inference monitoring statistics on the validation set.

Metric	**Value**
Total inferences	406
Average confidence	91.70%
Minimum confidence	51.74%
Low-confidence rate (< 0.7)	3.69%
Predicted as Clean	233 (57.4%)
Predicted as Dusty	173 (42.6%)

The average confidence of 91.70 percent and the low-confidence rate of just 3.69 percent for predictions below the 0.7 threshold show a model that is decisive on the great majority of images. The small fraction of low-confidence images are panels whose condition is genuinely ambiguous and may warrant manual review rather than a fully automated cleaning decision. Keeping a running log of the low-confidence rate and raising a retraining alert when it climbs past a set threshold is a simple but effective drift-detection mechanism, complementary to the formal KS test [17], [48].

4.8 Discussion

4.8.1 Operational Interpretation of Results

The results show that the MobileNetV3-Large transfer-learning approach classifies well enough to support condition-based maintenance at Algerian utility-scale plants. At 97.04 percent best validation accuracy, with Dusty-class precision of 97.08 percent, fewer than one cleaning alert in thirty would send a crew out unnecessarily, a clear improvement over fixed-schedule cleaning, where wasted visits are routine in quiet periods [8]. At Dusty-class recall of 94.86 percent, only about one genuinely dusty panel in twenty is missed per cycle, which keeps undetected losses low.

The sub-2-millisecond inference latency from the ONNX FP16 runtime is comfortably compatible with real-time processing on drones. For a 200 MW plant of roughly 500000 panels, even assuming embedded hardware an order of magnitude slower, the whole site could be assessed within a single inspection campaign [24]. That throughput compares well with the multi-day timescale of comprehensive manual inspection, and it sidesteps the inter-operator variability and perceptual-threshold problems that undermine the consistency of visual assessment [8].

4.8.2 Deployment Pathway for Algerian Installations

Deploying the system at Algerian sites will mean dealing head-on with the domain gap between the public training data and what Saharan dust actually looks like under local light. Saharan dust has a characteristic reddish-brown colour from its iron oxide content [21] and can form cemented crusts through the dew-driven process described in Chapter 1 [13]. Both may differ from the deposits in the training data. The recommended pathway has three phases. The first is an initial validation study at one of the recently commissioned plants, Tendla or El Ghrous, to collect labelled images under local conditions and measure the size of the domain shift. The second applies the drift fine-tuning technique from Section 4.6 to narrow the mismatch using those locally collected images. The third introduces continuous learning, folding locally labelled images into the training set as they accumulate [43], [48].

4.8.3 Limitations and Open Questions

The experiments here have several limitations that should be stated plainly, both for honesty and because each one points to a concrete next step. First, there is no formally held-out test set separate from the validation partition. The model was selected on the basis of its validation accuracy, so that figure, while a reasonable indicator, is not a fully unbiased estimate of how the model would perform on genuinely unseen data. A three-way train, validation, and test

split would close this gap, at the cost of reducing the already modest amount of training data. Second, relying on public datasets whose geographic characteristics are not fully documented limits how far the findings generalise to Saharan conditions; the merged dataset widens the training distribution but does not approximate the specific appearance of Algerian desert dust. Third, the binary clean-versus-dusty scheme says nothing about soiling severity or about where on a panel the dust has settled, both of which an operator would want in order to prioritise cleaning. Fourth, the drift evaluation uses a synthetic corruption (blur and noise) as a proxy for real distribution shift; while this is a standard and informative test, it cannot fully stand in for the complex, correlated changes that real field conditions impose.

4.9 Future Work

The limitations above, together with the results that precede them, map out a clear research programme. This section sets out the most important directions, roughly in the order in which a follow-on project might pursue them.

4.9.1 An Algerian Saharan Dataset

The single most valuable next step is to collect and label a dataset of solar panel images captured at operational Algerian plants, under local light, across the full range of seasons and soiling conditions. The recently commissioned Tendla and El Ghrous facilities are natural candidate sites. Such a dataset would directly address the domain-gap problem that the drift experiment in Section 4.6 exposed, because it would let the model learn the specific reddish-brown signature and cemented texture of Saharan dust [13], [21] rather than the generic dust of public benchmarks. Even a few thousand locally captured images, used to fine-tune the model trained here, would likely yield a substantial gain in field accuracy, on the same logic that made transfer learning effective in the first place [43]. Collecting this data is principally a logistical and institutional task rather than a technical one, which is why it sits at the top of the list: it needs access and effort more than it needs new methods.

4.9.2 From Binary Classification to Severity Estimation

The present system answers only a yes-or-no question: is this panel dusty? An operator planning a cleaning campaign would benefit far more from a continuous estimate of soiling severity, because that is what determines whether the energy lost to dust yet justifies the cost of cleaning. Reframing the task as regression, predicting a soiling ratio or an estimated power loss rather than a binary label, is therefore a high-value extension. It would require labelled data with severity annotations, which are harder to obtain than binary labels but could be derived from co-located power measurements or soiling-station readings. A severity estimate

would also make the link between detection and the economic decision explicit, turning the model from a classifier into a direct input to a cleaning-schedule optimisation.

4.9.3 Multi-Class and Multi-Fault Detection

Dust is only one of several conditions that degrade panel output. Bird droppings, physical cracks, delamination, shading from vegetation, and electrical faults all reduce generation, and several of them call for different responses than cleaning. Extending the model from two classes to a richer set of fault categories would make it considerably more useful as a general inspection tool, and the recent literature shows this is feasible, with multi-class CNN inspection systems already demonstrated [45], [46]. Combining RGB imagery with the thermal infrared data that inspection drones increasingly carry would help here, since electrical faults often show up as thermal signatures invisible in ordinary light [24].

4.9.4 On-Board Deployment and Field Validation

The latency and model-size results in Section 4.5 establish that on-board inference is feasible in principle, but they were measured on a cloud GPU rather than on the embedded hardware a drone would actually carry. A natural engineering step is to deploy the exported model on a representative edge platform, measure its real latency and power draw, and integrate it with a drone's flight-control and imaging systems to produce spatially resolved soiling maps in flight. Beyond that lies the most demanding validation of all: a controlled field trial at a commissioned Algerian plant, comparing energy yield and water consumption under condition-based maintenance driven by this system against the conventional fixed-schedule approach. Such a trial would convert the laboratory accuracy figures reported here into the operational and economic evidence that would justify wider deployment.

4.9.5 Continual Learning and Self-Supervision

Finally, the drift-detection and fine-tuning machinery demonstrated in Section 4.6 points toward a system that improves itself over time. A deployed model could log low-confidence images, have a subset of them labelled, and periodically fine-tune on the accumulated examples, so that its accuracy tracks the slowly changing conditions of its site rather than degrading as those conditions drift away from its original training distribution [48]. Self-supervised pretraining on the large volume of unlabelled panel imagery that routine drone inspection would generate could reduce the labelling burden further, learning useful visual features from the images themselves before any human annotation. Together these techniques would move the system from a static classifier toward a maintained, continually adapting component of a plant's operations.

4.10 Conclusion

This chapter has reported the full experimental implementation and evaluation of the proposed dust detection system. The main findings, in brief: MobileNetV3-Large with ImageNet transfer learning and CLAHE preprocessing reaches 97.04 percent best validation accuracy (96.55 percent with test-time augmentation) on the 2029-image merged dataset; ONNX Runtime deployment delivers up to a $3.66\times$ inference speedup over native PyTorch, down to 1.87 milliseconds per image; the systematic comparison of design alternatives in Section 4.4 confirms the final configuration is a considered balance of capacity, preprocessing, and augmentation strength; the baseline model is sensitive to simulated distribution shift, falling to 60.9 percent, but a short drift fine-tuning step recovers most of the lost accuracy, reaching 84.9 percent on drifted inputs; and the production monitoring framework, with its 3.69 percent low-confidence rate, gives actionable signals for catching drift in deployment. The General Conclusion that follows draws together the contributions and findings of the whole thesis.

General Conclusion

This thesis has investigated the automated detection of dust on photovoltaic panels using deep learning, set squarely within the context of Algeria's fast-growing national solar infrastructure. The motivation was a concrete operational problem: an exceptional solar resource, an ambitious national programme, and a harsh Saharan soiling environment combine to create conditions where intelligent, automated soiling detection could yield real, measurable economic and environmental benefits.

Three main findings emerge from the preceding chapters. The first concerns the scale and urgency of the problem. The Algerian National Renewable Energy Programme has reached a critical phase, with nine plants totalling 1480 MW entering service before August 2026 [4], [5], including the recently commissioned 200 MW Tendla and 200 MW El Ghrous plants [5]. The Saharan settings of these installations impose soiling that produces documented power losses of 8.41 percent after just eight weeks of natural exposure [6], with a single sandstorm capable of cutting energy output by 32 percent at a connected utility plant [6]. Projected across the full 15 GW programme, the aggregate cost of unmanaged soiling runs to the order of several hundred million US dollars a year [9]. Conventional detection methods, namely fixed schedules, point monitoring stations, and manual inspection, cannot meet this at the required scale and spatial resolution [8].

The second finding establishes that transfer learning from an ImageNet-pretrained MobileNetV3-Large backbone, combined with CLAHE illumination normalisation, is an effective and efficient solution to the binary dust detection task. Trained on a merged dataset of 2029 labelled images with AdamW [38], cosine annealing with linear warmup [39], label smoothing, RandAugment [41], and Random Erasing [42], the system reaches a best validation accuracy of 97.04 percent (96.55 percent with test-time augmentation), with Dusty-class precision of 97.08 percent and recall of 94.86 percent. Deployment via the ONNX Runtime [16] cuts per-image latency from 6.84 milliseconds to 1.87 in FP16, a $3.66\times$ speedup, fast enough for real-time drone inspection. These results confirm the approach is technically viable for deployment. The systematic comparison of design alternatives in Section 4.4 further shows that the final configuration came from deliberate evaluation, not arbitrary choice.

The third finding concerns robustness under distribution shift. The baseline model degrades sharply under simulated shift, falling from 96.9 percent to 60.9 percent and exposing a critical weakness when models trained on relatively clean datasets meet harsh Saharan field conditions [7]. A short drift fine-tuning step recovers most of the lost accuracy, lifting drifted-data performance from 67.2 percent to 84.9 percent and showing that robustness can be improved substantially without extra labelled real-world data. The Kolmogorov--Smirnov

drift-detection module [17] provides a reliable automated signal to trigger retraining when shift appears in production.

The main directions for future research follow directly from these findings. The first is collecting and labelling images from operational Algerian plants, to close the domain gap between the public training distribution and the real appearance of Saharan dust. The second is extending from binary classification to soiling-severity regression and multi-class fault detection, supporting more nuanced decisions. The third is running controlled field trials comparing energy yield and water use under condition-based versus fixed-schedule maintenance. The fourth is developing fully autonomous inspection workflows that link the detection pipeline to drone flight planning and the generation of spatially resolved soiling maps.

Algeria's solar infrastructure is a generational investment in the country's energy future. The technical foundations for intelligent, condition-based maintenance of that infrastructure, demonstrated in this thesis, are sound. What remains is the work of deployment, validation, and refinement under the real conditions of Algeria's Saharan environment.

References

[1] International Renewable Energy Agency (IRENA), *Renewable capacity statistics 2024*, https://www.irena.org/Publications/2024/Mar/Renewable-capacity-statistics-2024, Abu Dhabi, 2024.

[2] Africa Energy Portal, *Development of solar energy: A new turning point for algeria*, https://africa-energy-portal.org, Accessed: 17 April 2026, Mar. 2024.

[3] Ministry of Energy and Renewable Energies (Algeria), *National renewable energy programme (2020--2035)*, Government of Algeria, Algiers, 2020.

[4] ESI Africa, *Algeria to commission 1.48 gw of solar capacity by august*, https://www.esi-africa.com, Accessed: 17 April 2026, Feb. 2026.

[5] Enerdata, *Algeria commissions two solar pv plants totalling 400 mw*, https://www.enerdata.net, Accessed: 17 April 2026, Apr. 2026.

[6] M. Dida, S. Boughali, D. Bechki, and H. Bouguettaia, ``Output power loss of crystalline silicon photovoltaic modules due to dust accumulation in Saharan environment," *Renewable and Sustainable Energy Reviews*, vol. 124, p. 109 787, 2020. DOI: 10.1016/j.rser.2020.109787.

[7] R. Conceição, J. González-Aguilar, A. A. Merrouni, and M. Romero, ``Soiling effect in solar energy conversion systems: A review," *Renewable and Sustainable Energy Reviews*, vol. 162, p. 112 434, 2022. DOI: 10.1016/j.rser.2022.112434.

[8] A. Sayyah, M. N. Horenstein, and M. K. Mazumder, ``Energy yield loss caused by dust deposition on photovoltaic panels," *Solar Energy*, vol. 107, pp. 576–604, 2014. DOI: 10.1016/j.solener.2014.05.030.

[9] C. O. Rusănescu, M. Rusănescu, I. A. Istrate, G. A. Constantin, and M. Begea, ``Analysis of soiling loss in photovoltaic modules: A review," *Sustainability*, vol. 15, no. 24, p. 16 669, 2023. DOI: 10.3390/su152416669.

[10] S. Mehta, A. P. Azad, S. A. Chemmengath, V. Raykar, and S. Kalyanaraman, ``DeepSolarEye: Power loss prediction and weakly supervised soiling localization via fully convolutional networks for solar panels," in *Proceedings of the IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2018, pp. 333–342. DOI: 10.1109/WACV.2018.00043.

[11] M. S. H. Onim, Z. M. M. Sakif, A. Ahnaf, *et al.*, ``SolNet: A convolutional neural network for detecting dust on solar panels," *Energies*, vol. 16, no. 1, p. 155, 2023. DOI: 10.3390/en16010155.

[12] Ö. F. Alçın, M. Aslan, and A. Ari, ``SolPowNet: Dust detection on photovoltaic panels using convolutional neural networks," *Electronics*, vol. 14, no. 21, p. 4230, 2025. DOI: 10.3390/electronics14214230.

[13] K. Ilse, B. W. Figgis, V. Naumann, C. Hagendorf, and J. Bagdahn, ``Fundamentals of soiling processes on photovoltaic modules," *Renewable and Sustainable Energy Reviews*, vol. 98, pp. 239–254, 2018. DOI: 10.1016/j.rser.2018.09.015.

[14] A. Howard, M. Sandler, G. Chu, *et al.*, ``Searching for MobileNetV3," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 1314–1324.

[15] K. Zuiderveld, ``Contrast limited adaptive histogram equalization," in *Graphics Gems IV*, Academic Press, 1994, pp. 474–485.

[16] Microsoft Corporation, *ONNX Runtime: Cross-platform, high-performance ML inferencing and training accelerator*, https://onnxruntime.ai, 2021.

[17] A. N. Kolmogorov, ``Sulla determinazione empirica di una legge di distribuzione," *Giornale dell'Istituto Italiano degli Attuari*, vol. 4, pp. 83–91, 1933.

[18] P. Würfel and U. Würfel, *Physics of Solar Cells: From Basic Principles to Advanced Concepts*, 3rd. Weinheim, Germany: Wiley-VCH, 2016.

[19] S. Shongwe and M. Hanif, ``Comparative analysis of different single-diode PV modeling methods," *IEEE Journal of Photovoltaics*, vol. 5, no. 3, pp. 938–946, 2015. DOI: 10.1109/JPHOTOV.2015.2395137.

[20] PV Magazine, *Algeria's PV capacity tops 436.8 MW*, https://www.pv-magazine.com, Accessed: 17 April 2026, Sep. 2024.

[21] M. Mani and R. Pillai, ``Impact of dust on solar photovoltaic (PV) performance: Research status, challenges and recommendations," *Renewable and Sustainable Energy Reviews*, vol. 14, no. 9, pp. 3124–3131, 2010. DOI: 10.1016/j.rser.2010.07.065.

[22] A. Necaibia, A. Bouraiou, A. Ziane, *et al.*, ``Experimental assessment of soiling losses on PV modules in an arid Algerian climate," *Renewable Energy*, vol. 168, pp. 1156–1167, 2021.

[23] S. Semaoui, A. Hadj Arab, E. K. Boudjelthia, S. Bacha, and H. Zeraia, ``Dust effect on optical transmittance of photovoltaic module glazing in a desert region," *Energy Procedia*, vol. 74, pp. 1347–1357, 2015. DOI: 10.1016/j.egypro.2015.07.781.

[24] M. Al-Zubaidi *et al.*, ``Deep edge-based fault detection for solar panels," *Sensors*, vol. 24, no. 16, p. 5230, 2024. DOI: 10.3390/s24165230.

[25] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, ``Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[26] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, ``ImageNet: A large-scale hierarchical image database," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009, pp. 248–255.

[27] O. Russakovsky, J. Deng, H. Su, *et al.*, ``ImageNet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[28] A. Krizhevsky, I. Sutskever, and G. E. Hinton, ``ImageNet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, vol. 25, 2012, pp. 1097–1105.

[29] K. He, X. Zhang, S. Ren, and J. Sun, ``Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[30] A. G. Howard, M. Zhu, B. Chen, *et al.*, ``MobileNets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[31] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, ``MobileNetV2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 4510–4520.

[32] M. Tan and Q. V. Le, ``EfficientNet: Rethinking model scaling for convolutional neural networks," in *Proceedings of the 36th International Conference on Machine Learning (ICML)*, 2019, pp. 6105–6114.

[33] M. Tan and Q. V. Le, ``EfficientNetV2: Smaller models and faster training," *Proceedings of the 38th International Conference on Machine Learning (ICML)*, pp. 10 096–10 106, 2021.

[34] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA: MIT Press, 2016.

[35] S. Ioffe and C. Szegedy, ``Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, vol. 37, 2015, pp. 448–456.

[36] J. Hu, L. Shen, and G. Sun, ``Squeeze-and-excitation networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 7132–7141.

[37] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, ``Rethinking the Inception architecture for computer vision," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 2818–2826. DOI: 10.1109/CVPR.2016.308.

[38] I. Loshchilov and F. Hutter, ``Decoupled weight decay regularization,'' in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019.

[39] I. Loshchilov and F. Hutter, ``SGDR: Stochastic gradient descent with warm restarts,'' in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017.

[40] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, ``Dropout: A simple way to prevent neural networks from overfitting,'' *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[41] E. D. Cubuk, B. Zoph, J. Shlens, and Q. V. Le, ``RandAugment: Practical automated data augmentation with a reduced search space,'' in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2020, pp. 702–703.

[42] Z. Zhong, L. Zheng, G. Kang, S. Li, and Y. Yang, ``Random erasing data augmentation,'' in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, 2020, pp. 13 001–13 008. DOI: 10.1609/aaai.v34i07.7000.

[43] K. Weiss, T. M. Khoshgoftaar, and D. Wang, ``A survey of transfer learning,'' *Journal of Big Data*, vol. 3, no. 1, p. 9, 2016. DOI: 10.1186/s40537-016-0043-6.

[44] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, ``How transferable are features in deep neural networks?'' In *Advances in Neural Information Processing Systems*, vol. 27, 2014, pp. 3320–3328.

[45] G. M. El-Banby *et al.*, ``Enhanced fault detection in photovoltaic panels using CNN-based classification with PyQt5 implementation,'' *Applied Sciences*, vol. 14, no. 23, p. 11 158, 2024. DOI: 10.3390/app142311158.

[46] H. Yamashita, R. Lim, and G. Navnath, *Solar augmented dataset: Image dataset for fault detection in solar panels using deep learning*, https://www.kaggle.com/datasets/gitenavnath/solar-augmented-dataset, Kaggle dataset, 2025.

[47] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, ``Grad-CAM: Visual explanations from deep networks via gradient-based localization,'' in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 618–626. DOI: 10.1109/ICCV.2017.74.

[48] J. Quiñonero-Candela, M. Sugiyama, A. Schwaighofer, and N. D. Lawrence, Eds., *Dataset Shift in Machine Learning*. Cambridge, MA: MIT Press, 2009.

[49] F. Pedregosa, G. Varoquaux, A. Gramfort, *et al.*, ``Scikit-learn: Machine learning in Python,'' *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[50] P. Micikevicius, S. Narang, J. Alben, *et al.*, ``Mixed precision training,'' in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018.

Appendix A: Key Implementation Code

This appendix reproduces the three most important Python listings behind the results in Chapter 4: the construction of the model, the CLAHE preprocessing transform, and the main training loop. These are the components that most directly determine the behaviour reported in this thesis. The remaining components of the pipeline, namely the loss and optimiser configuration, the test-time-augmented evaluation, the ONNX export and latency benchmark, and the drift simulation with its fine-tuning step, are described in full in the body of the thesis (Chapters 3 and 4) and are therefore not duplicated here as code. The listings are taken from the project notebook and lightly reformatted for the page; they are a faithful record of the central method rather than a complete drop-in script.

A.1 Model Construction with Fallback Backbones

The build routine loads MobileNetV3-Large with ImageNet-pretrained weights and replaces its classifier head with a two-class linear layer. It falls back to EfficientNet-B0 and then ResNet-18 if the primary weights cannot be loaded, so the pipeline runs even where a particular weight file is unavailable.

Listing A.1: Model construction with fallback backbones.

```
import torch.nn as nn
from torchvision import models

def build_model(num_classes=2, name="mobilenet_v3_large"):
    for n in [name, "efficientnet_b0", "resnet18"]:
        try:
            if n == "mobilenet_v3_large":
                w = models.MobileNet_V3_Large_Weights.IMAGENET1K_V2
                m = models.mobilenet_v3_large(weights=w)
                m.classifier[3] = nn.Linear(m.classifier[3].
                    in_features, num_classes)
            elif n == "efficientnet_b0":
                w = models.EfficientNet_B0_Weights.IMAGENET1K_V1
                m = models.efficientnet_b0(weights=w)
                m.classifier[1] = nn.Linear(m.classifier[1].
                    in_features, num_classes)
            elif n == "resnet18":
                w = models.ResNet18_Weights.IMAGENET1K_V1
```

```
                m = models.resnet18(weights=w)
                m.fc = nn.Linear(m.fc.in_features, num_classes)
            print(f"Built {n} with pretrained weights")
            return m, n
        except Exception as e:
            print(f"  Could not load {n} ({e}); trying next")
    raise RuntimeError("All model loads failed")

model, model_name = build_model()
model = model.to(device, memory_format=torch.channels_last)
```

A.2 CLAHE Preprocessing Transform

CLAHE is wrapped as a torchvision-compatible transform operating on the L channel of the LAB colour space. A fresh OpenCV CLAHE object is created per call so the transform is safe to use with multiple DataLoader workers.

Listing A.2: CLAHE transform on the L channel of LAB.

```
import cv2, random, numpy as np
from PIL import Image as _PILImage

class CLAHE:
    def __init__(self, clip_limit=2.0, tile_grid=(8, 8), p=1.0):
        self.clip_limit = clip_limit
        self.tile_grid = tile_grid
        self.p = p
    def __call__(self, pil_img):
        if random.random() > self.p:
            return pil_img
        arr = np.array(pil_img.convert("RGB"))
        lab = cv2.cvtColor(arr, cv2.COLOR_RGB2LAB)
        l, a, b = cv2.split(lab)
        clahe = cv2.createCLAHE(clipLimit=self.clip_limit,
           tileGridSize=self.tile_grid)
        l = clahe.apply(l)
        return _PILImage.fromarray(cv2.cvtColor(cv2.merge((l, a, b)
           ), cv2.COLOR_LAB2RGB))
```

A.3 Training Loop with AMP and Early Stopping

The training loop uses Automatic Mixed Precision, the channels-last memory format, gradient clipping, learning-rate scheduling, and validation-driven early stopping, checkpointing the best model as it goes. The loss, optimiser, and cosine schedule referenced here are configured exactly as described in Chapter 3.

Listing A.3: Main training loop.

```
import time, torch
from torch.amp import autocast, GradScaler

scaler = GradScaler("cuda") if device.type == "cuda" else
    GradScaler("cpu")
best_acc, best_epoch, patience_left, best_state = 0.0, -1,
    EARLY_STOP_PATIENCE, None
amp_dtype = torch.float16 if device.type == "cuda" else torch.
    bfloat16

for epoch in range(EPOCHS):
    model.train()
    tr_correct, tr_n = 0, 0
    for x, y in train_loader:
        x = x.to(device, non_blocking=True, memory_format=torch.
            channels_last)
        y = y.to(device, non_blocking=True)
        optimizer.zero_grad(set_to_none=True)
        with autocast(device_type=device.type, dtype=amp_dtype):
            logits = model(x); loss = criterion(logits, y)
        scaler.scale(loss).backward()
        scaler.unscale_(optimizer)
        torch.nn.utils.clip_grad_norm_(model.parameters(), max_norm
            =2.0)
        scaler.step(optimizer); scaler.update()
        tr_correct += (logits.argmax(1) == y).sum().item(); tr_n +=
            x.size(0)
    train_acc = 100 * tr_correct / tr_n

    model.eval()
    vl_correct, vl_n = 0, 0
    with torch.no_grad():
        for x, y in val_loader:
```

```
            x = x.to(device, non_blocking=True, memory_format=torch.channels_last)
            y = y.to(device, non_blocking=True)
            with autocast(device_type=device.type, dtype=amp_dtype):
                logits = model(x)
            vl_correct += (logits.argmax(1) == y).sum().item(); vl_n += x.size(0)
    val_acc = 100 * vl_correct / vl_n
    scheduler.step()

    if val_acc > best_acc:
        best_acc, best_epoch = val_acc, epoch + 1
        best_state = {k: v.detach().cpu().clone() for k, v in model.state_dict().items()}
        patience_left = EARLY_STOP_PATIENCE
    else:
        patience_left -= 1
    if patience_left <= 0:
        print(f"Early stopping (no improvement for {EARLY_STOP_PATIENCE} epochs)"); break

if best_state is not None:
    model.load_state_dict(best_state)
    print(f"Best val acc {best_acc:.2f}% at epoch {best_epoch}")
```

Appendix B: Experimental Results Summary

Table B.1 collects the end-to-end results of the pipeline in one place, mirroring the stage-by-stage summary produced at the end of the project notebook.

Table B.1: End-to-end pipeline summary.

Stage	**Accuracy**	**Latency**	**Status**
Baseline (PyTorch + CLAHE)	97.04% / TTA 96.55%	6.84 ms	OK
ONNX FP16	97.04% (clean)	1.87 ms ($\times$3.66)	Production-ready
Drift (blur + noise)	60.9% (drifted)	–	Drift detected
Retrained V2	84.9% (drifted)	1.87 ms	Promoted

The artifacts generated by the pipeline include the trained checkpoint, the FP32 and FP16 ONNX models, the V2 drift-hardened model, the training-curve and confusion-matrix figures, the drift-analysis figure, and the baseline and production metrics in JSON form.

Appendix C: List of Acronyms

AMP Automatic Mixed Precision

BN Batch Normalisation

CLAHE Contrast-Limited Adaptive Histogram Equalisation

CNN Convolutional Neural Network

FP16 / FP32 16-bit / 32-bit Floating Point

GAP Global Average Pooling

GHI Global Horizontal Irradiance

ILSVRC ImageNet Large Scale Visual Recognition Challenge

KS Kolmogorov--Smirnov (test)

NAS Neural Architecture Search

ONNX Open Neural Network Exchange

PV Photovoltaic

ReLU Rectified Linear Unit

SE Squeeze-and-Excitation

TTA Test-Time Augmentation