W A R S A W
S C H O O L   O F   C O M P U T E R   S C I E N C E

A R I S A P i l o t P r o j e c t
(M L E n g i n e e r)

Fusion of deep learning models for aircraft classification in aerial imagery

Wojciech Karwowski

W A R S AW 2025

1. Introduction

In recent years, the space sector has been developing at a remarkable pace, as evidenced by the growing number of observation satellites delivering optical images with ever-increasing spatial resolution [1]. This data is opening up new possibilities for monitoring the environment, infrastructure, and human activity, while also supporting the development of advanced analytical systems based on artificial intelligence (AI).

The classification of aircraft in aerial and satellite imagery is a key issue in the fields of image recognition, security, and airspace monitoring. Automatic classification systems are used in both military and civilian contexts – ranging from supporting intelligence operations and analyzing aviation activity to air traffic control and the monitoring of critical infrastructure. In the military sphere, these systems enable rapid identification of aircraft types, which is crucial for assessing potential threats and planning operational activities.

In recent years, object detection methods have become particularly popular, allowing for the localization of aircraft within images. However, classification itself – that is, distinguishing between different types of aircraft – remains a fundamentally important task, especially in the context of evaluating operational capabilities and conducting technical reconnaissance. For this reason, the present project focuses exclusively on the classification problem, treating it as a foundational step toward building more complex systems.

Although this problem may seem fundamental, it is accompanied by a range of challenges, such as the great diversity of aircraft designs, subtle visual differences between specific types, and variability in imaging conditions (observation angle, lighting, atmospheric interference). Classical image analysis methods have proven insufficient under such circumstances, which is why, in recent years, deep neural networks – capable of automatically extracting discriminative features from visual data – have come to play a dominant role.

Despite the increasing availability of high-resolution aerial and satellite imagery, the reliable and accurate classification of aircraft types remains challenging due to the limited quality and size of publicly available datasets, strong intra-class variability, and the sensitivity of deep models to changing imaging conditions. These limitations hinder the deployment of robust AI-based solutions in real operational environments, where both accuracy and resilience to perturbations are crucial. In this study, I therefore investigate not only individual deep learning-based classifiers but also their ensemble combinations, aiming to leverage model complementarity to improve predictive performance and robustness. I consider two ensemble strategies: weighted soft-voting on logits without an additional learning stage, which aggregates the confidence scores of base models, and stacking with logistic regression, which learns an additional meta-classifier on the validation-set outputs of the base models. The objectives of this work are to benchmark several state-of-the-art convolutional neural network (CNN) classifiers and their ensembles on a curated aircraft dataset, analyze their accuracy and identify the most effective strategies for future integration into operational recognition and monitoring systems.

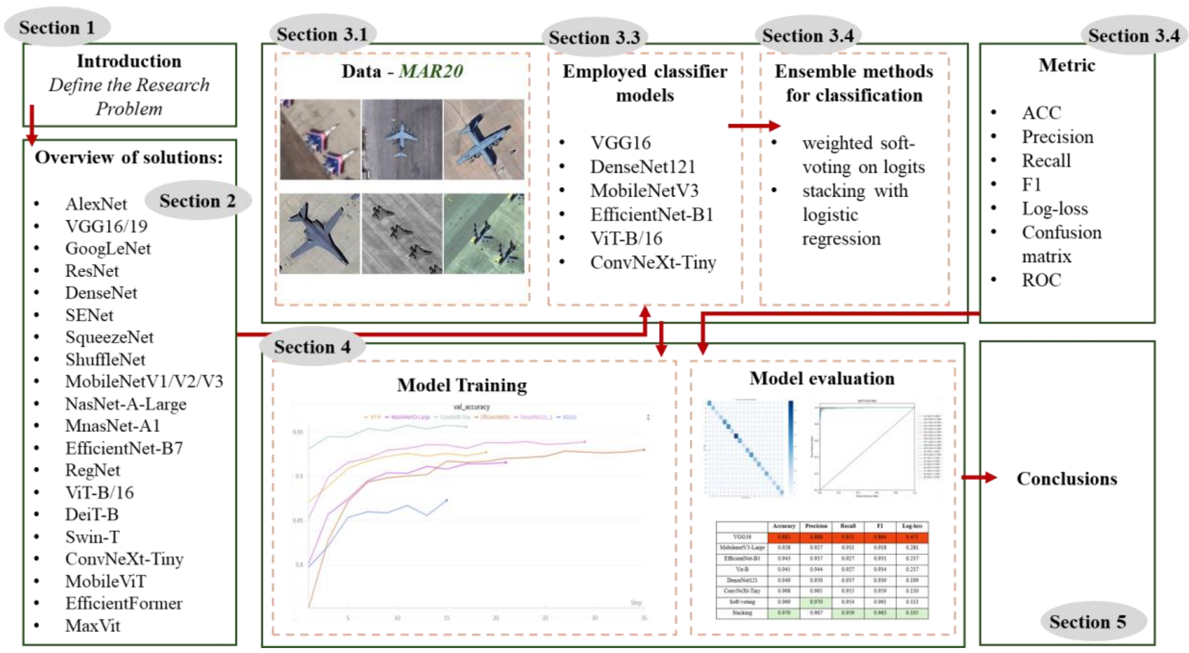


Fig. 1. Workflow of the study.

Fig. 1 presents the workflow of the study, reflecting the logical structure of the paper. Section 1 introduces the research problem and discusses the motivation for undertaking the study. Section 2 provides a review of existing classification methods. Section 3 covers the theoretical part – the characteristics of the dataset used, a description of the applied tools and classification models, and the presentation of two ensemble-fusion strategies. Section 4 is devoted to the practical part and includes the experimental design, the training process of the selected models and classifier ensembles, as well as the analysis of the obtained results with corresponding conclusions. The final Section 5 provides a concise summary of the conducted research and the main concluding remarks.

2. Overview of solutions

The development of image classification methods has been highly dynamic in recent years. The starting point was simple yet groundbreaking convolutional neural networks (CNNs), such as AlexNet [2], which revolutionized the ImageNet competition in 2012 thanks to its deeper architecture, use of ReLU nonlinearity, and GPU-based training. The next step was the introduction of models like VGG16 and VGG19 (2014) [3], characterized by an orderly, sequential structure of 3×3 convolutions and fully connected layers. Although these networks had an enormous number of parameters and were computationally expensive, they became the foundation for further research.

The following generation brought architectures that enabled the training of deeper networks while maintaining gradient stability. ResNet (2015) [4] introduced the concept of residual connections, allowing the construction of models with hundreds of layers, while DenseNet (2016) [5] went a step further by introducing dense connections between all layers, improving information flow and reducing the number of parameters. In parallel, networks such as GoogLeNet and Inception (2016) [6] were developed, employing multi-scale modules and global average pooling, which increased computational efficiency.

As the need grew for models to be deployed on devices with limited computational power, a new line of architectures optimized for efficiency emerged. This group includes SqueezeNet (2016) [7], ShuffleNet (2018) [8], and MobileNet (v1-v3) (2018) [9] [10] [11]. The latter introduced depthwise separable convolutions and the Squeeze-and-Excitation mechanism in version V3, significantly reducing complexity while maintaining high prediction quality. Simultaneously, approaches that automate the network design process, such as NASNet (2017) [12] and MnasNet (2018) [13], based on Neural Architecture Search methods, were developed.

The next stage featured models that introduced the idea of systematically scaling architecture. EfficientNet (2019) [14] automatically balances depth, width, and resolution, offering an excellent trade-off between quality and computational complexity. RegNet (2020) [15], on the other hand, proposed a simple, regular approach to scaling architectures, making it easier to transfer models across different computational budgets.

In recent years, we have witnessed the modernization of classic CNNs, inspired by Transformers. ConvNeXt (2022) exemplifies this approach: it simplifies the structure of convolutional blocks, increases kernel sizes, and replaces traditional normalization with newer solutions, achieving performance comparable to Transformers while retaining the advantages of CNNs. Variants of ResNet (2015) [4] and DenseNet (2016) [5] enhanced with channel attention mechanisms, such as SENet (2017) [16], have also gained popularity.

However, the most significant change in recent years has been the introduction of Transformers into computer vision. The Vision Transformer (ViT) (2020) [17] treats an image as a sequence of patches and applies the self-attention mechanism known from natural language processing (NLP). Although it requires large training datasets and considerable computational power, it achieves excellent results. An improved version, DeiT (2020) [18], uses distillation to enable training Transformers on smaller datasets. Swin Transformer (2021) [19] introduces hierarchical sliding windows, combining global context with local relationships, making it competitive with CNNs in classification and detection tasks. The latest trends focus on combining convolutions and attention mechanisms in hybrid models such as MobileViT (2021) [20], EfficientFormer (2022) [21], and MaxViT (2022) [22], as well as on developing very large models (foundation models) capable of adapting to multiple tasks with minimal fine-tuning.

In summary, the evolution of image classifiers has progressed from simple, sequential architectures (AlexNet, VGG), through networks with mechanisms supporting gradient propagation (ResNet, DenseNet), lightweight and optimized mobile models (MobileNet, ShuffleNet), scalable architectures (EfficientNet, RegNet), to modern CNNs inspired by Transformers (ConvNeXt) and Transformers themselves (ViT, Swin). Current research is focused on further improving accuracy while simultaneously enhancing computational efficiency, which enables the use of advanced classifiers both in scientific research and in applications requiring real-time operation, such as the analysis of aerial or satellite images. Appendix A (Table A1) presents the distinguishing features of the most popular classification models along with their accuracy on the ImageNet dataset.

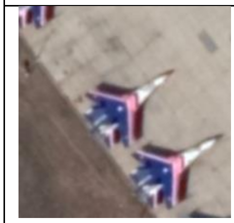
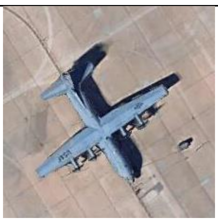
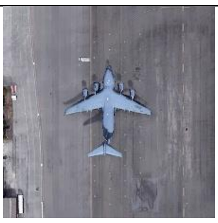
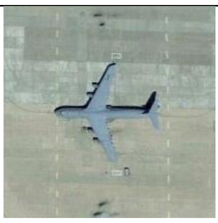
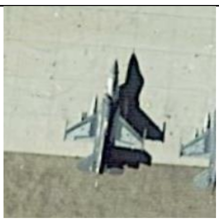
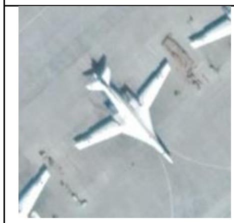
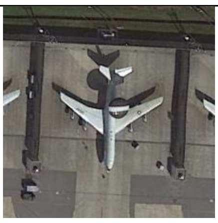
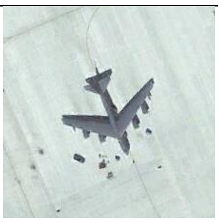
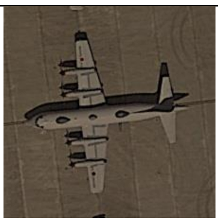
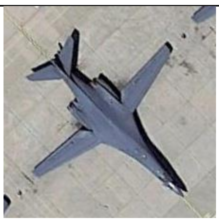
Six architectures were selected for further research: VGG16, DenseNet121, MobileNetV3-Large, EfficientNet-B1, ViT-B/16, and ConvNeXt-Tiny. This set was chosen to represent different stages in the development of image classification methods and to showcase diverse architectural approaches. VGG16 serves as a classic benchmark, illustrating the capabilities of early, sequential convolutional networks. DenseNet121, by introducing dense connections between layers, enables more efficient use of information and parameters, making it a significant point of comparison. MobileNetV3-Large represents the family of lightweight models designed for mobile applications and systems with limited resources, which is especially important in the context of images obtained from UAVs. EfficientNet-B1 exemplifies a modern approach to automatic architecture scaling, providing an optimal compromise between quality and computational complexity. ViT-B/16 brings the Transformer paradigm into computer vision, allowing for the analysis of global dependencies in an image and achieving top performance in many tasks. Meanwhile, ConvNeXt-Tiny represents contemporary efforts to modernize classic CNNs by integrating solutions inspired by Transformers, offering high effectiveness while maintaining a convolutional character.

This historical overview directly guided the selection of models evaluated in my study. To address the aircraft-classification task under varying operational constraints, I selected a representative subset of architectures spanning the key milestones of this evolution: high-capacity residual networks (e.g., ResNet, DenseNet), lightweight mobile-optimized CNNs (e.g., ShuffleNet, MobileNet, EfficientNet), and more recent Transformer-inspired designs (e.g., ConvNeXt). This diversity allowed me to examine how architectural depth, efficiency, and attention mechanisms affect classification accuracy and robustness in aerial imagery. The selected methods are discussed in greater detail in Subsection 3.3.

3. Data and methodology

3.1. Data

To prepare the classifier, the public MAR20 (Military Aircraft Recognition) [23], database was used, containing aerial images with annotations for 20 types of military aircraft (Fig. 2). For the classification stage, objects (aircraft) were cropped from the source images according to the annotations and scaled to a fixed resolution of 224×244 px, which simplifies batch processing and aligns with the common practice of using input sizes of approximately 224 px in ImageNet-based models.

A1 (Su-35)	A2 (C-130)	A3 (C-17)	A4 (C-5)	A5 (F-16)
				
A6 (Tu-160)	A7 (E-3 AWACS)	A8 (B-52)	A9 (P-3C)	A10 (B-1B)
				

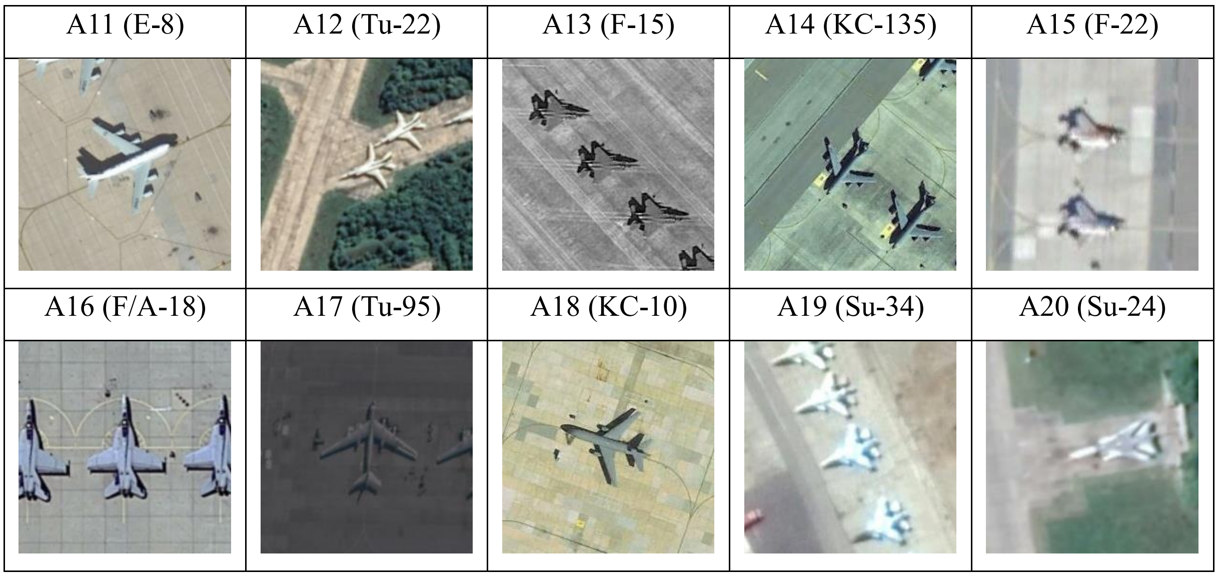


Fig. 2. Examples of aircraft classes prepared from the MAR20 dataset.

The data prepared in this way was divided into:

- Training set (70%) – used to estimate model parameters;
- Validation set (15%) – used to monitor the training process. Early stopping was applied if there was no improvement on this set;
- Test set (15%) – used exclusively for the final evaluation of model performance on data not seen during training.

The number of images in each set is provided in Table 1.

Table 1. Number of images in the training, validation, and test datasets.

Class	**Train**	**Validation**	**Test**
A1	1152	247	248
A2	1175	253	252
A3	816	176	176
A4	456	98	96
A5	868	186	187
A6	331	71	72
A7	459	99	99
A8	664	143	143
A9	726	156	156
A10	652	140	140
A11	368	80	80

A12	505	109	109
A13	1147	246	246
A14	1250	269	269
A15	461	99	99
A16	1826	392	392
A17	967	208	208
A18	228	49	49
A19	846	182	182
A20	639	137	137
sum	**14 897**	**3 203**	**3 203**

Before being fed into the network, each image crop was converted to the RGB color space, normalized (according to ImageNet statistics), and standardized in size to ensure consistency among samples and compatibility with the requirements of the architectures used in the experiments.

3.2. Tools

A computational environment for the project was prepared using the Anaconda platform. This environment was specifically configured for training deep learning models with the PyTorch library. For model evaluation (calculation of assessment metrics), functions from the scikit-learn library were used. PyTorch (together with the torchvision and torchaudio modules) enables the implementation and training of neural network architectures with full GPU support. The model training process was monitored using the Weights & Biases platform, which allowed for tracking metrics, visualizing the learning process, and comparing results across different experimental configurations. The code used is available: https://github.com/wojciech-karwowski/ARISA-Project.

3.3. Employed classifier models

This subsection focuses on the specific architectures selected for the experiments in this study. These models were chosen to represent different families of image classifiers – from classical convolutional networks to modern Transformer-inspired designs – in order to investigate how architectural diversity influences aircraft-classification performance in aerial imagery.

One of the first convolutional architectures to gain widespread recognition in image classification was VGG16 (2014) [3]. It is characterized by a simple, sequential structure based on cascading convolutional layers with small filter sizes (3×3), interleaved with pooling layers. Its advantages included simplicity and ease of transfer to new tasks, but at the cost of a very large number of parameters and significant computational load.

Next came DenseNet121 (2017) [5], which expanded on the idea of inter-layer connections by densely linking each layer to all previous ones. This type of construction increased feature reuse, improved gradient propagation, and reduced the number of parameters compared to earlier architectures of similar depth.

The next generation of lightweight networks included MobileNetV3 (2019) [11] and EfficientNet B1 (2019) [14]. MobileNetV3 was designed with mobile devices in mind, optimizing the trade-off between accuracy and speed by employing inverted residual blocks, depthwise separable layers, and architecture optimization via neural architecture search (NAS). EfficientNet B1, on the other hand, is part of a larger family of networks based on the joint scaling of depth, width, and resolution. Thanks to so-called compound scaling, it achieved high accuracy while maintaining computational efficiency.

In parallel, transformer-inspired architectures were also being developed. The Vision Transformer (ViT-B) (2020) [17] marked a breakthrough departure from classic convolutions, replacing them with a self-attention mechanism. Here, images are divided into non-overlapping patches, which are then treated as sequences in a language modeling task. ViT-B stands out for its high accuracy on large datasets but requires significant computational resources and proper scaling.

The most recent architecture described is ConvNeXt-Tiny (2022) [24], which modernizes classic convolutional neural networks (CNNs) in the spirit of Transformers. This architecture increases filter sizes, simplifies block structures, and applies modern normalization techniques, allowing it to achieve competitive results compared to self-attention-based models, while retaining the efficiency and stability advantages of CNNs. The key features of the models studied are presented in Table 2.

Table 2. Overview of the characteristic features of the models employed in the classification task.

Architecture	**Year**	**Features**	**Params (M)**	**FLOPs (GFLOPs, @224×224)**	**Input**	**Top-1 (ImageNet)**
VGG16 [3]	2014	Simple sequential structure, 3×3 convolutions, large number of parameters	138	~15.5	224×224	~71%
DenseNet121 [5]	2017	Dense inter-layer connections, improved gradient flow	8.0	~2.9	224×224	~75%
MobileNetV3-Large [11]	2019	Lightweight for mobile, inverted residual block, NAS	5.4	~0.22	224×224	~75.2%
EfficientNet-B1 [14]	2019	Compound scaling (depth, width, resolution)	7.8	~0.7	240×240	~79.1%
ViT-B/16 [17]	2020	Self-attention mechanism, image as patch sequence (16×16), no convolutions	86	~17.5	224×224	~77.9%
ConvNeXt-Tiny [24]	2022	Modernized CNN in Transformer style, large filters (7×7), simplified blocks	29	~4.5	224×224	~82.1%

3.4. Ensemble methods for classification

To improve generalization and prediction stability, classifier ensembles were employed, in which the decisions of multiple independently trained models are combined into a single result. From the perspective of classical error decomposition, such combination reduces the variance component (by averaging independent fluctuations), can partially compensate for the systematic error of individual architectures (as models make mistakes on different examples), and in practice often improves probability distribution calibration (e.g., lower log-loss, Brier score, ECE). Two complementary schemes were used: weighted soft-voting on logits (without an additional learning stage) [25] and stacking with logistic regression trained on the validation set [26].

In soft-voting with logits, each base model $m = 1, \ldots, M$ produces a logits vector $\ell^{(m)}(x) \in \mathbb{R}^C$ for a given sample x. These logits are aggregated using a weighted sum (1), and the resulting vector is transformed into a probability distribution via (2).

$$\tilde{\ell}(x) = \sum_{m=1}^{M} w_m \, \ell^{(m)}(x) \tag{1}$$

$$\hat{p}(x) = softmax\left(\tilde{\ell}(x)\right) \tag{2}$$

$$w_m = \frac{\left(F1_{macro}^{(m)}\right)^{\alpha}}{\sum_{k=1}^{M}\left(F1_{macro}^{(k)}\right)^{\alpha}}, \qquad \alpha = 2 \tag{3}$$

The weights $\{w_m\}$ are determined automatically on the validation set based on the quality of the individual models (3), which amplifies the contribution of better classifiers (in terms of macro-F1), while not zeroing out the contribution of weaker (often complementary) models. When validation is unavailable, weights fall back on equal $w_m = \frac{1}{M}$. Aggregating logits, rather than probabilities directly, avoids excessive "flattening" of the distribution (the mean of probabilities can be overly conservative) and in practice leads to sharper and often better-calibrated predictions.

In stacking, by contrast, an additional lightweight meta-classifier is trained to learn how to combine signals from the base models. For each sample, we construct a feature vector (4), i.e., the concatenation of logits from all models. Then, using pairs $\{z(x_i), y_i\}$ from the validation set (disjoint from the training data for the base models), we train a multinomial logistic regression (multinomial LogisticRegression) by minimizing negative log-likelihood (cross-entropy).

$$z(x) = \left[\ell^{(1)}(x)||\ell^{(2)}(x)|| \ \dots || \ \ell^{(M)}(x)\right] \in \mathbb{R}^{C \cdot M} \tag{4}$$

$$\hat{p}_{meta}(x) = softmax(W_z(x) + b) \tag{5}$$

The meta-model returns a distribution (5). The choice of logistic regression is justified: (i) it minimizes a proper scoring rule, (ii) it introduces moderate regularization and interpretability (a linear combination of logits), with negligible computational cost at inference time. It is critical to avoid information leakage: the meta-classifier is trained exclusively on the validation set (or - in an even “cleaner” version - on out-of-fold (OOF) predictions from a K-fold split).

Both strategies are complementary: soft-voting provides a simple and stable baseline, requiring no additional training, while stacking can further improve metrics (e.g., macro-F1, log-loss), especially when the base models are heterogeneous (different architectures, augmentations, training criteria). In both cases, class mapping consistency between models was ensured, and weights and meta-parameters were determined exclusively on the validation set to reliably assess generalization to the test set.

3.5. Metric

To evaluate the classification models, the most popular metrics from computer vision and machine learning were used. The quality assessment indicators employed are presented in Table 3. Additionally, Receiver Operating Characteristic (ROC) measures and the confusion matrix, which illustrates the distribution of errors between classes, were used in the analysis.

Table 3. Summary of the classification model evaluation metrics used.

Metric	Description	Goal
Accuracy (ACC)	$ACC = \frac{1}{N}\sum_{i=1}^{N} 1[\hat{y}_i = y_i]$	1
Precision	$Prec = \frac{1}{C}\sum_{i=1}^{C} \frac{TP_c}{TP_c + FP_c}$	1
Recall	$recall = \frac{1}{C}\sum_{i=1}^{C} \frac{TP_c}{TP_c + FN_c}$	1
F1	$F1 = \frac{1}{C}\sum_{i=1}^{C} \frac{2PrecRecall}{Prec + Recall}$	1
Log-loss	$CE = -\frac{1}{N}\sum_{i=1}^{N} \log p_{i,y_i}$	0

Where: N- number of samples, C – number of class, y_i – true label of sample, $\hat{y}_i$ – predicted label, $p_{i,c}$ – predicted probability for class *c* in sample *i*, $1[\cdot]$ - indicator function, TP_c, FP_c, FN_c – respectively, true positives/errors for class *c*.

4. Results

4.1. Model Training Procedure

All analyzed deep learning models – VGG16, MobileNetV3-Large, EfficientNet-B1, Vision Transformer (ViT-B), DenseNet121, and ConvNeXt-Tiny – were trained in the PyTorch environment. The data preparation process included standardizing the image resolution to 224 × 224 pixels, converting images to tensor format, and applying random horizontal flipping (RandomHorizontalFlip) as an augmentation technique to increase sample diversity.

Additionally, images were normalized to the values used in models pre-trained on the ImageNet dataset (mean: [0.485, 0.456, 0.406], standard deviation: [0.229, 0.224, 0.225]).

Training was conducted with a learning rate of 1×10^{-4} and a batch size of 32. The chosen learning rate ensured stable and gradual convergence, without the risk of the model settling into local minima too abruptly.

The learning rate of 1×10^{-4} and batch size of 32 were selected following commonly used settings for fine-tuning ImageNet-pretrained CNNs and were confirmed in preliminary trials on the validation set. These trials involved testing learning rates in the range $[1 \times 10^{-3}, 5 \times 10^{-5}]$ to ensure stable convergence without oscillations or divergence, while maintaining a batch size that balanced GPU memory constraints and gradient-estimate stability.

To mitigate the risk of overfitting, early stopping was employed with parameters: patience = 4, delta = 0.001, mode = "min". This allowed the training process to stop if no significant improvement in validation performance was observed for four consecutive epochs, while also guarding against premature stopping due to natural fluctuations in the loss curve.

Given the limited number of available images relative to the very large number of model parameters, transfer learning was applied. This technique involves leveraging knowledge (weights) acquired during training on large, general datasets – in this case, ImageNet – and transferring it to a new task. Weights trained on ImageNet were used for extracting low- and mid-level features, such as edges, textures, and spatial patterns, while the final layers responsible for image classification were retrained on the analyzed dataset to adapt the model to the specifics of the new problem.

4.2. Model training and evaluation

The benefit of using transfer learning is a significant reduction in model training time, since most parameters are already pre-tuned. Moreover, it improves generalization ability and reduces the risk of overfitting, as the networks rely on robust representations learned from millions of ImageNet images, rather than solely on the small set of available samples. The Weights & Biases (W&B) platform was used to monitor the training process, providing detailed tracking of metrics, visualization of progress, and comparison of results across different experiments. The following figures (Fig. 3, Fig. 4) present the loss and accuracy curves for the validation dataset.

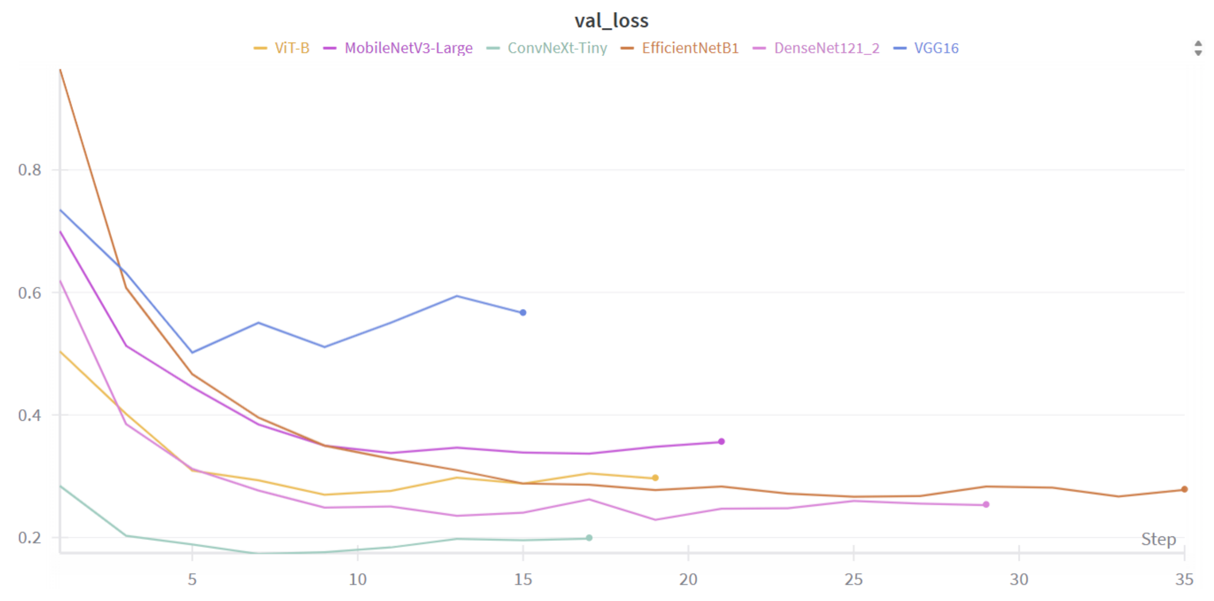


Fig. 3. Loss curve for the validation dataset.

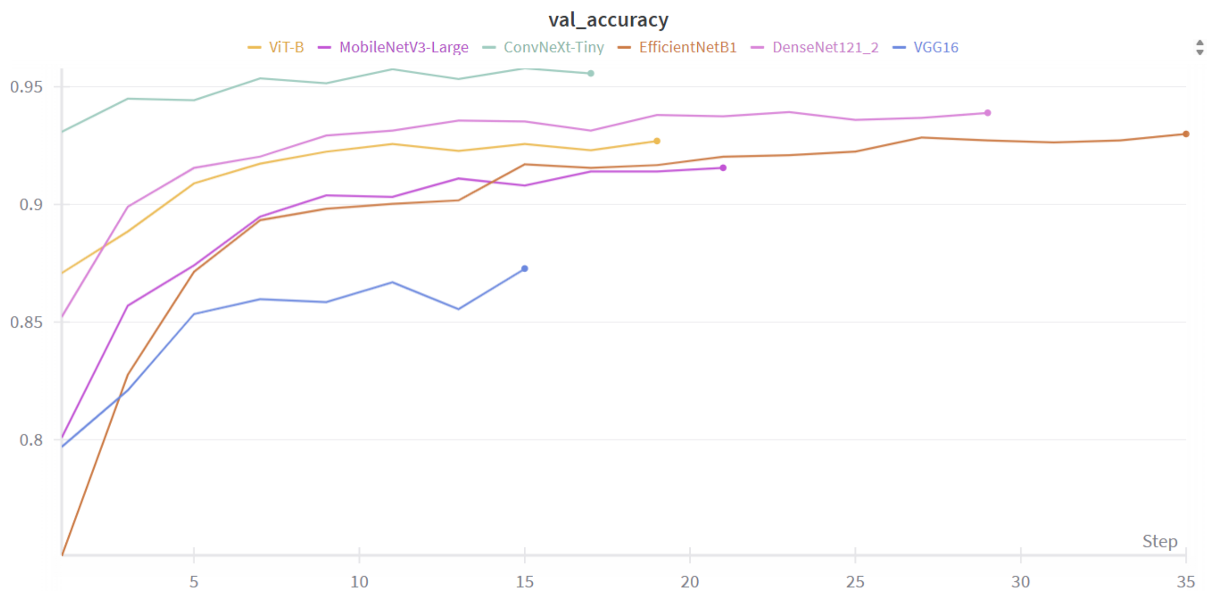


Fig. 4. Accuracy curve for the validation dataset.

Analysis of these graphs reveals a clear decrease in loss values during the initial epochs, accompanied by a simultaneous increase in accuracy, indicating an effective learning process. The ConvNeXt-Tiny model achieved both the lowest loss and the highest accuracy, while VGG16 demonstrated slower and less stable convergence.

To provide a more comprehensive assessment of performance, the models were further evaluated using the metrics presented in Table 3. The tests were conducted on a separate test dataset that was not involved in training. The evaluation results are shown in Table 4.

Table 4. Evaluation metric values on the test dataset.

	Accuracy	Precision	Recall	F1	Log-loss
VGG16	0.881	0.888	0.851	0.864	0.451
MobilenetV3-Large	0.928	0.927	0.911	0.918	0.281
EfficientNet-B1	0.943	0.937	0.927	0.931	0.237
Vit-B	0.945	0.944	0.927	0.934	0.237
DenseNet121	0.949	0.950	0.937	0.939	0.199
ConvNeXt-Tiny	0.968	0.965	0.955	0.959	0.130

The best results were achieved by ConvNeXt-Tiny (Accuracy 0.968, Precision 0.965, Recall 0.955, F1 0.959), which also recorded the lowest log-loss value (0.130). This indicates not only the highest decision accuracy (argmax), but also the most reliable probability distributions – the model more frequently assigns high probabilities to the correct class and is less likely to be "overconfident" in incorrect predictions. The second-best performance came from DenseNet121 (Accuracy 0.949, F1 0.939) with a log-loss of 0.199. The next two models – EfficientNet-B1 (Accuracy 0.943, F1 0.931, log-loss 0.237) and ViT-B (Accuracy 0.945, F1 0.934, log-loss 0.237) – achieved nearly identical quality, despite being based on different architectures (convolutional vs. transformer). MobilenetV3-Large (Accuracy 0.928, F1 0.918, log-loss 0.281) performed somewhat worse, but its model contains only about 5.4 million parameters. Moreover, it requires just 0.22 GFLOPS, making it a suitable solution for scenarios with limited computational resources. Among the algorithms tested, the oldest model – VGG16 (Accuracy 0.881, F1 0.864, log-loss 0.451) – performed the weakest.

An interesting characteristic of the evaluated models is that their precision values exceed recall (for example, ConvNeXt-Tiny 0.965 vs. 0.955, DenseNet121 0.950 vs. 0.931, VGG16 0.888 vs. 0.851). From a probabilistic quality perspective, differences in log-loss are pronounced: moving from VGG16 (0.451) to MobilenetV3-Large (0.281), then to EfficientNet-B1/ViT-B (0.237), and DenseNet121 (0.199). ConvNeXt-Tiny (0.130) illustrates a systematic improvement in calibration and the "sharpness" of probability distributions. This has practical significance for threshold tuning and for model combination methods (such as soft-voting and stacking), which directly benefit from higher-quality probability estimates.

Appendix B (Table B1) presents examples of misclassifications made by the trained models. Analysis shows that some errors stem from incorrect class labels. Additionally, many models tend to overgeneralize image features, leading to the omission of distinctive aircraft components (for example, confusing E-3 and E-8 types).

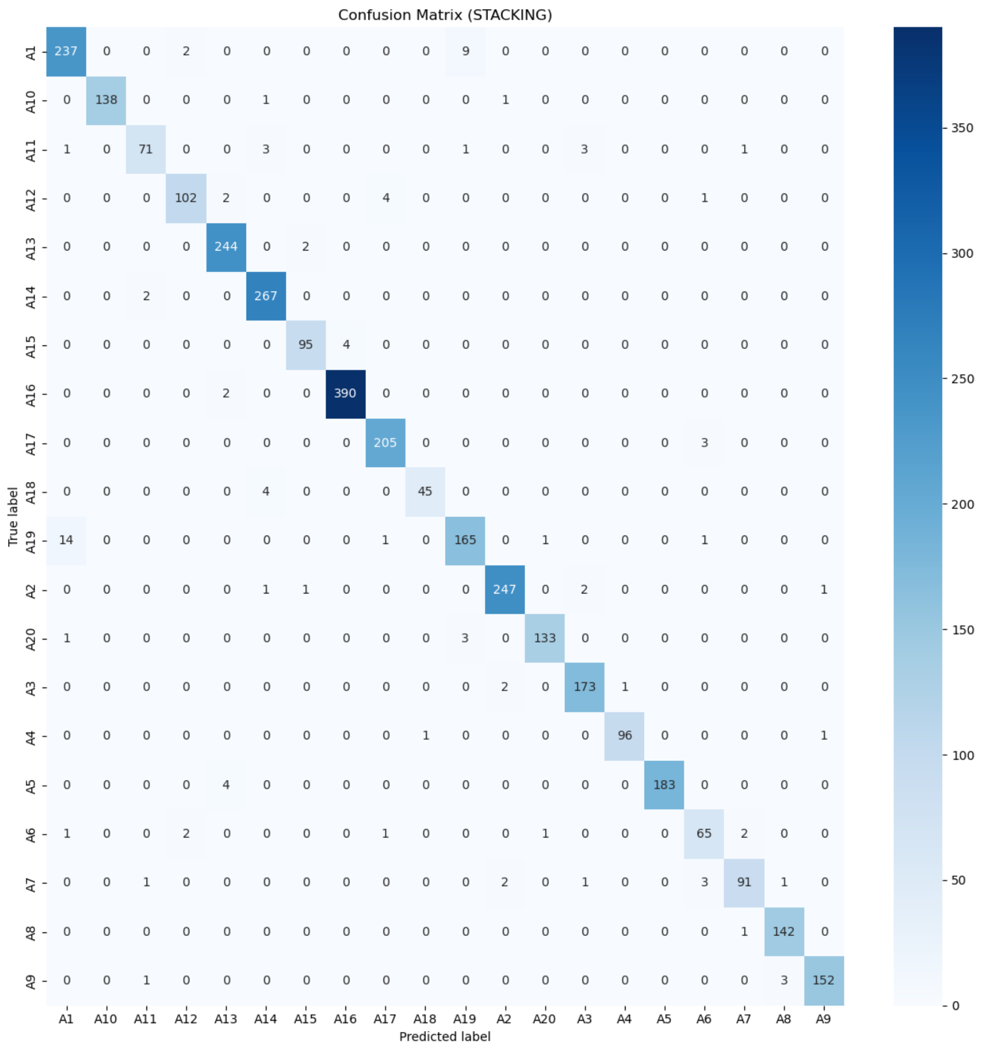
To improve classification quality, ensemble classifiers described in Section 3.4 were used. All trained classifier models were included in the ensemble preparation. The first ensemble approach applied was soft-voting. For each model, logits were computed and then combined into a weighted average. Weights were assigned in proportion to the models' performance on the validation set, with stronger models having greater influence (the F1 score was squared to determine the weight). The resulting sum of logits was normalized using the softmax function to obtain final class probabilities. The resulting weights were: VGG16 – 0.148, DenseNet121 – 0.173, EfficientNet-B1 – 0.168, ConvNeXt-Tiny – 0.180, MobileNetV3-Large – 0.163, and ViT-B – 0.169. In this way, models with higher validation performance played a larger role in the final prediction.

The ensemble based on weighted soft-voting achieved the highest results among all approaches (Table 5) – slightly surpassing the best single model, ConvNeXt-Tiny, in terms of Accuracy (0.969 vs. 0.968) and F1 (0.961 vs. 0.959), while significantly improving log-loss (0.113 vs. 0.130), indicating better calibration and greater reliability of predictions.

Table 5. Evaluation metrics for models and ensemble classifiers on the test dataset. The best results are highlighted in green, and the worst in red.

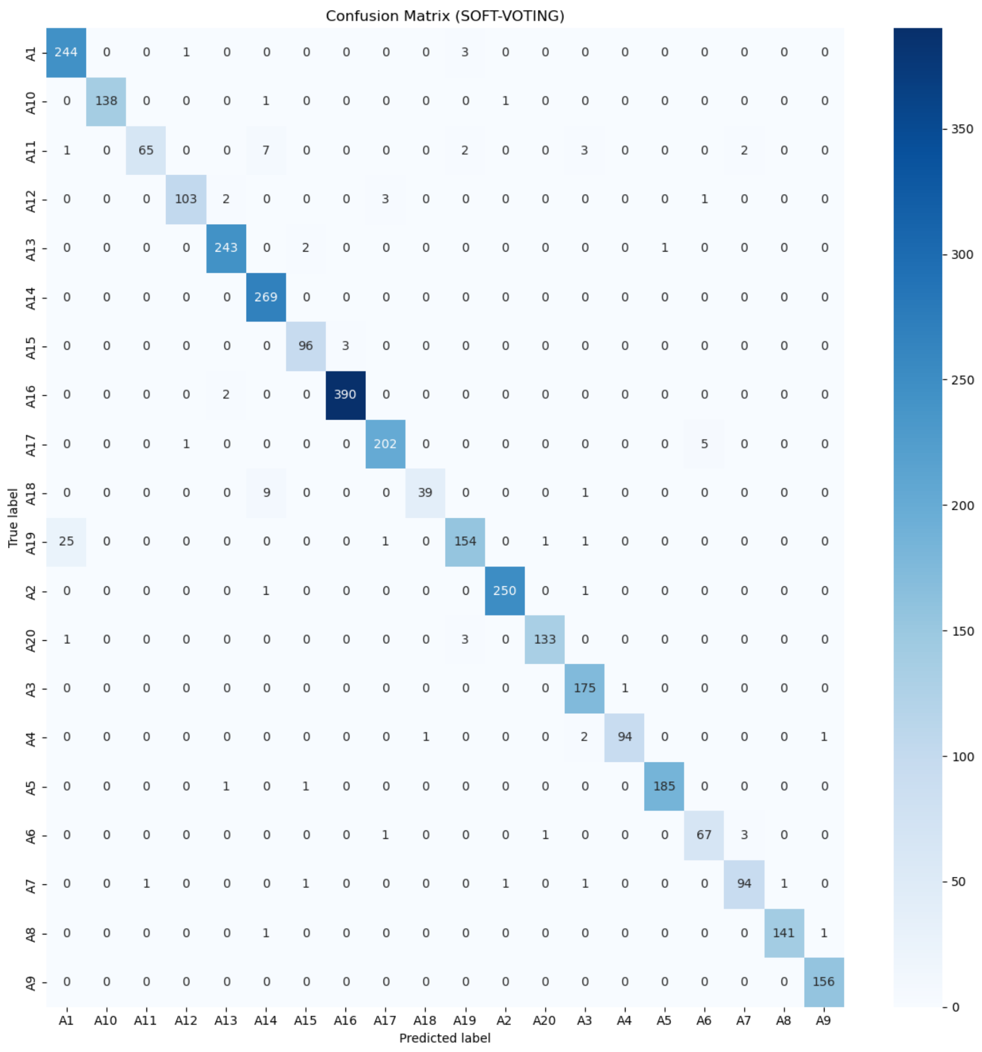
	Accuracy	**Precision**	**Recall**	**F1**	**Log-loss**
VGG16	0.881	0.888	0.851	0.864	0.451
MobilenetV3-Large	0.928	0.927	0.911	0.918	0.281
EfficientNet-B1	0.943	0.937	0.927	0.931	0.237
Vit-B	0.945	0.944	0.927	0.934	0.237
DenseNet121	0.949	0.950	0.937	0.939	0.199
ConvNeXt-Tiny	0.968	0.965	0.955	0.959	0.130
Soft-voting	0.969	0.970	0.954	0.961	0.113
Stacking	0.970	0.967	0.959	0.963	0.105

The second ensemble method tested was stacking with logistic regression. This approach achieved the best evaluation metrics on the test set. Compared to the best single model (ConvNeXt-Tiny) as well as the classic soft-voting method, it enabled a further increase in prediction quality – reaching the highest values for Accuracy (0.970) and F1 (0.963). Notably, its advantage is particularly evident in terms of log-loss (0.105), indicating the best calibration and the highest reliability of the generated probabilities. These results confirm that using the stacking method, which learns to optimally combine the decisions of multiple base classifiers, provides better generalization than both single models and simpler ensemble schemes.



True label \ Predicted label	A1	A10	A11	A12	A13	A14	A15	A16	A17	A18	A19	A2	A20	A3	A4	A5	A6	A7	A8	A9
A1	237	0	0	2	0	0	0	0	0	0	9	0	0	0	0	0	0	0	0	0
A10	0	138	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0
A11	1	0	71	0	0	3	0	0	0	0	1	0	0	3	0	0	0	1	0	0
A12	0	0	0	102	2	0	0	0	4	0	0	0	0	0	0	0	1	0	0	0
A13	0	0	0	0	244	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0
A14	0	0	2	0	0	267	0	0	0	0	0	0	0	0	0	0	0	0	0	0
A15	0	0	0	0	0	0	95	4	0	0	0	0	0	0	0	0	0	0	0	0
A16	0	0	0	0	2	0	0	390	0	0	0	0	0	0	0	0	0	0	0	0
A17	0	0	0	0	0	0	0	0	205	0	0	0	0	0	0	0	3	0	0	0
A18	0	0	0	0	0	4	0	0	0	45	0	0	0	0	0	0	0	0	0	0
A19	14	0	0	0	0	0	0	0	1	0	165	0	1	0	0	0	1	0	0	0
A2	0	0	0	0	0	1	1	0	0	0	0	247	0	2	0	0	0	0	0	1
A20	1	0	0	0	0	0	0	0	0	0	3	0	133	0	0	0	0	0	0	0
A3	0	0	0	0	0	0	0	0	0	0	0	2	0	173	1	0	0	0	0	0
A4	0	0	0	0	0	0	0	0	0	1	0	0	0	0	96	0	0	0	0	1
A5	0	0	0	0	4	0	0	0	0	0	0	0	0	0	0	183	0	0	0	0
A6	1	0	0	2	0	0	0	0	1	0	0	0	1	0	0	0	65	2	0	0
A7	0	0	1	0	0	0	0	0	0	0	0	2	0	1	0	0	3	91	1	0
A8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	142	0
A9	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	152

Fig. 5. Confusion matrix for stacking.

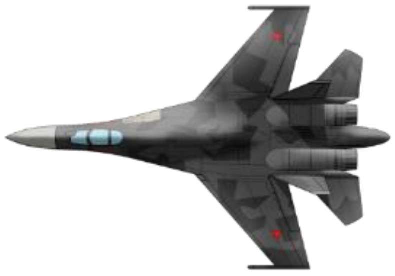
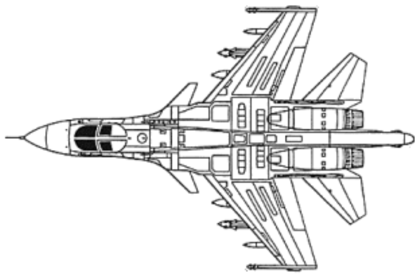
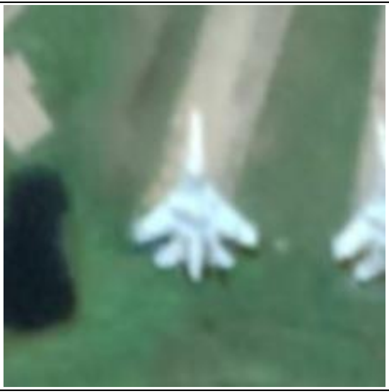
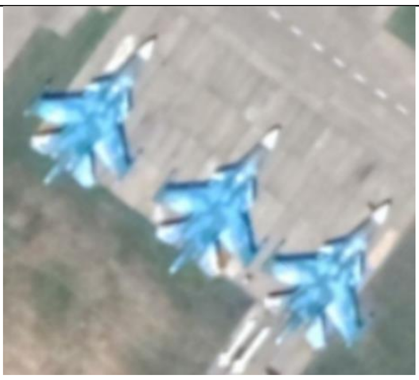


	A1	A10	A11	A12	A13	A14	A15	A16	A17	A18	A19	A2	A20	A3	A4	A5	A6	A7	A8	A9
A1	244	0	0	1	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0
A10	0	138	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0
A11	1	0	65	0	0	7	0	0	0	0	2	0	0	3	0	0	0	2	0	0
A12	0	0	0	103	2	0	0	0	3	0	0	0	0	0	0	0	1	0	0	0
A13	0	0	0	0	243	0	2	0	0	0	0	0	0	0	0	1	0	0	0	0
A14	0	0	0	0	0	269	0	0	0	0	0	0	0	0	0	0	0	0	0	0
A15	0	0	0	0	0	0	96	3	0	0	0	0	0	0	0	0	0	0	0	0
A16	0	0	0	0	2	0	0	390	0	0	0	0	0	0	0	0	0	0	0	0
A17	0	0	0	1	0	0	0	0	202	0	0	0	0	0	0	0	5	0	0	0
A18	0	0	0	0	0	9	0	0	0	39	0	0	0	1	0	0	0	0	0	0
A19	25	0	0	0	0	0	0	0	1	0	154	0	1	1	0	0	0	0	0	0
A2	0	0	0	0	0	1	0	0	0	0	0	250	0	1	0	0	0	0	0	0
A20	1	0	0	0	0	0	0	0	0	0	3	0	133	0	0	0	0	0	0	0
A3	0	0	0	0	0	0	0	0	0	0	0	0	0	175	1	0	0	0	0	0
A4	0	0	0	0	0	0	0	0	0	1	0	0	0	2	94	0	0	0	0	1
A5	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	185	0	0	0	0
A6	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	67	3	0	0
A7	0	0	1	0	0	0	1	0	0	0	0	1	0	1	0	0	0	94	1	0
A8	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	141	1
A9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	156

Fig. 6. Confusion matrix for soft-voting,

Fig. 5 and Fig. 6 present the confusion matrices for both ensemble methods employed: stacking (Logistic Regression on logits) and soft-voting. In both cases, the vast majority of samples were correctly assigned to the appropriate class, confirming the high level of accuracy achieved in quantitative evaluation. Only a few classification errors are visible – these most often concern classes with similar aircraft silhouettes, which may indicate difficulties in distinguishing between types with similar geometry. The largest number of misclassifications is observed between classes A1 (Su-35) and A19 (Su-34). These are aircraft with similar fuselage designs, and the differences visible in satellite imagery are very subtle. The Su-35 is a multirole fighter with a twin vertical stabilizer aerodynamic layout and a relatively slender fuselage, while the Su-34 is a tactical bomber whose most distinctive feature is its widened cockpit with side-by-side seating for two pilots. However, from orbital altitude, with limited resolution, this difference in the front of the fuselage is difficult to discern, and both aircraft have similar wingspans, empennage, and top-down silhouettes (Table 6). As a result, the classifier – despite its high overall performance – makes the most errors when distinguishing between these two types of aircraft.

Table 6. Differences between the Su-35 and Su-34.

	Su-35	**Su-34**
Model		
Sample		
DIMENSIONS		
Length	21.90	23.34
Width	15.30	14.70
Height	6.32	6.09

A comparison of both approaches shows that stacking is characterized by slightly higher stability, as evidenced by a lower number of misclassifications in some classes. Soft-voting also achieves very high results but occasionally generates additional errors in classes with fewer samples.

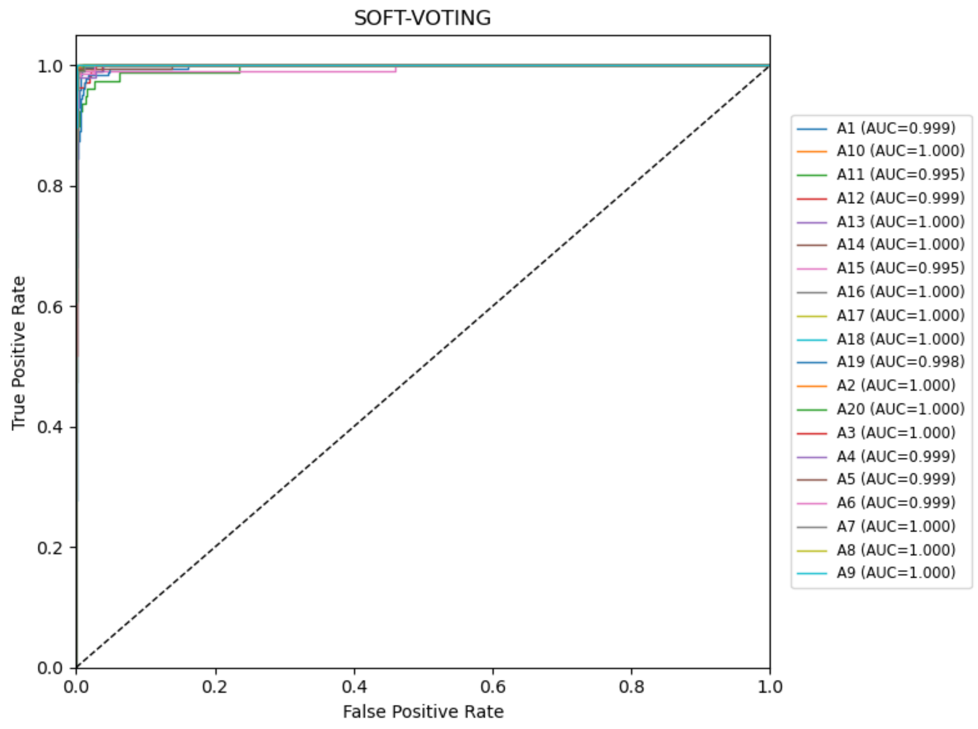


Fig. 7. ROC for soft-voting.

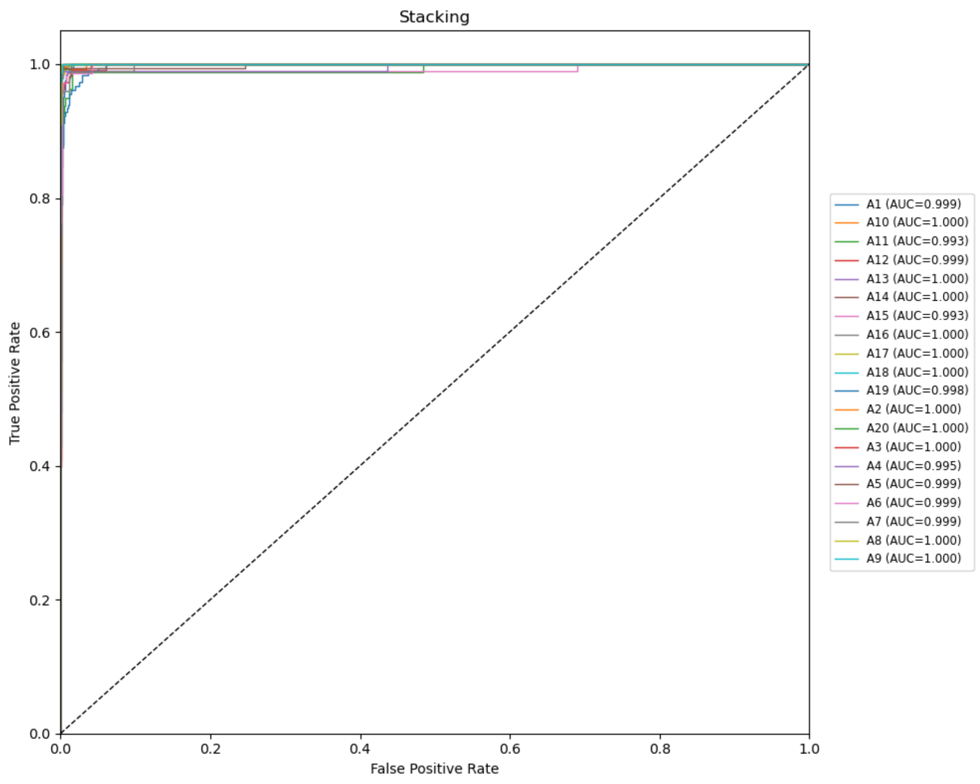


Fig. 8. ROC for stacking.

Fig. 7 and Fig. 8 present the ROC (Receiver Operating Characteristic) curves for all classes. The AUC (Area Under the Curve) for each class ranges from 0.995 to 1.000, confirming that both stacking and soft-voting offer excellent class separation capability. Such high AUC values indicate strong model confidence in label assignment and very good probability calibration.

5. Conclusions

The conducted research confirms that deep neural networks are an effective tool for classifying aircraft in aerial and satellite images. Among the single models tested, the ConvNeXt-Tiny architecture achieved the best results, reaching the highest accuracy and the lowest log-loss value, indicating very good probabilistic calibration. At the same time, comparing different approaches showed a systematic improvement in quality as the field moves from older, classical networks (e.g., VGG16) toward modern convolutional and transformer-based architectures.

Within the study, the application of ensemble methods yielded even clearer benefits. Both soft-voting and stacking allowed for further improvements in prediction quality compared to the best base models. Soft-voting provided a small but consistent increase in all metrics, as well as improved calibration of probability distributions. Stacking, on the other hand, thanks to its meta-classifier learning, achieved the best overall results, surpassing other approaches in terms of accuracy, F1-score, and log-loss. These results confirm that combining different architectures within ensembles increases the stability and reliability of classification. An analysis of the confusion matrix revealed that errors are concentrated among pairs of aircraft with similar silhouettes and dimensions, such as the Su-35 and Su-34, or the E-3 and E-8. This indicates that image resolution and subtle structural differences remain limiting factors. Nevertheless, AUC values ranging from 0.995 to 1.000 demonstrate that the classification systems exhibit a high capacity for class separation and strong confidence in their predictions.

In summary, the research has shown that deploying modern deep learning architectures in combination with ensemble methods enables very high effectiveness in classification tasks. Importantly, the results obtained can also be applied in practice – for example, as a guideline for selecting and modifying the backbone in YOLO-family object detectors, where an appropriately chosen feature extractor architecture can significantly improve the detection and classification of objects in satellite imagery.

While the obtained metrics – in particular AUC values close to 1.0 and the consistent gains from ensemble methods – demonstrate near-operational performance under controlled experimental conditions, the deployment of such systems in real-life scenarios would require further validation on imagery acquired under varying atmospheric conditions, sensor geometries, and noise levels. Additional domain-adaptation steps and continuous model monitoring would be essential to ensure robustness in operational settings.

References

[1] ‘Satellite Database | Union of Concerned Scientists’. Accessed: Mar. 17, 2025. [Online]. Available: https://www.ucs.org/resources/satellite-database

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, ‘ImageNet Classification with Deep Convolutional Neural Networks’, in *Advances in Neural Information Processing Systems*, Curran Associates, Inc., 2012. Accessed: Apr. 03, 2024. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2012/hash/c399862d3b9d6b76c8436e924a68c45b-Abstract.html

[3] K. Simonyan and A. Zisserman, ‘Very Deep Convolutional Networks for Large-Scale Image Recognition’, Apr. 10, 2015, *arXiv*: arXiv:1409.1556. doi: 10.48550/arXiv.1409.1556.

[4] 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*, 2016, pp. 770–778.

[5] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, ‘Densely Connected Convolutional Networks’, Jan. 28, 2018, *arXiv*: arXiv:1608.06993. doi: 10.48550/arXiv.1608.06993.

[6] C. Szegedy *et al.*, ‘Going Deeper with Convolutions’, Sep. 17, 2014, *arXiv*: arXiv:1409.4842. doi: 10.48550/arXiv.1409.4842.

[7] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, ‘SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size’, Nov. 04, 2016, *arXiv*: arXiv:1602.07360. doi: 10.48550/arXiv.1602.07360.

[8] X. Zhang, X. Zhou, M. Lin, and J. Sun, ‘ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices’, Dec. 07, 2017, *arXiv*: arXiv:1707.01083. doi: 10.48550/arXiv.1707.01083.

[9] A. G. Howard *et al.*, ‘MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications’, Apr. 17, 2017, *arXiv*: arXiv:1704.04861. doi: 10.48550/arXiv.1704.04861.

[10] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, ‘MobileNetV2: Inverted Residuals and Linear Bottlenecks’, Mar. 21, 2019, *arXiv*: arXiv:1801.04381. doi: 10.48550/arXiv.1801.04381.

[11] A. Howard *et al.*, ‘Searching for MobileNetV3’, Nov. 20, 2019, *arXiv*: arXiv:1905.02244. doi: 10.48550/arXiv.1905.02244.

[12] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, ‘Learning Transferable Architectures for Scalable Image Recognition’, Apr. 11, 2018, *arXiv*: arXiv:1707.07012. doi: 10.48550/arXiv.1707.07012.

[13] M. Tan *et al.*, ‘MnasNet: Platform-Aware Neural Architecture Search for Mobile’, May 29, 2019, *arXiv*: arXiv:1807.11626. doi: 10.48550/arXiv.1807.11626.

[14] M. Tan and Q. V. Le, ‘EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks’, Sep. 11, 2020, *arXiv*: arXiv:1905.11946. doi: 10.48550/arXiv.1905.11946.

[15] I. Radosavovic, R. P. Kosaraju, R. Girshick, K. He, and P. Dollár, ‘Designing Network Design Spaces’, Mar. 30, 2020, *arXiv*: arXiv:2003.13678. doi: 10.48550/arXiv.2003.13678.

[16] J. Hu, L. Shen, S. Albanie, G. Sun, and E. Wu, ‘Squeeze-and-Excitation Networks’, May 16, 2019, *arXiv*: arXiv:1709.01507. doi: 10.48550/arXiv.1709.01507.

[17] A. Dosovitskiy *et al.*, ‘An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale’, Jun. 03, 2021, *arXiv*: arXiv:2010.11929. doi: 10.48550/arXiv.2010.11929.

[18] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jégou, ‘Training data-efficient image transformers & distillation through attention’, Jan. 15, 2021, *arXiv*: arXiv:2012.12877. doi: 10.48550/arXiv.2012.12877.

[19] Z. Liu *et al.*, ‘Swin Transformer: Hierarchical Vision Transformer using Shifted Windows’, Aug. 17, 2021, *arXiv*: arXiv:2103.14030. doi: 10.48550/arXiv.2103.14030.

[20] S. Mehta and M. Rastegari, ‘MobileViT: Light-weight, General-purpose, and Mobile-friendly Vision Transformer’, Mar. 04, 2022, *arXiv*: arXiv:2110.02178. doi: 10.48550/arXiv.2110.02178.

[21] Y. Li *et al.*, ‘EfficientFormer: Vision Transformers at MobileNet Speed’, Oct. 11, 2022, *arXiv*: arXiv:2206.01191. doi: 10.48550/arXiv.2206.01191.

[22] Z. Tu *et al.*, ‘MaxViT: Multi-Axis Vision Transformer’, Sep. 09, 2022, *arXiv*: arXiv:2204.01697. doi: 10.48550/arXiv.2204.01697.

[23] W. Yu *et al.*, ‘MAR20：遥感图像军用飞机目标识别数据集’, *Yaogan Xuebao/Journal of Remote Sensing*, vol. 27, no. 12, pp. 2688–2696, 2023, doi: 10.11834/jrs.20222139.

[24] Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell, and S. Xie, ‘A ConvNet for the 2020s’, Mar. 02, 2022, *arXiv*: arXiv:2201.03545. doi: 10.48550/arXiv.2201.03545.

[25] T. G. Dietterich, ‘Ensemble Methods in Machine Learning’, in *Multiple Classifier Systems*, Berlin, Heidelberg: Springer, 2000, pp. 1–15. doi: 10.1007/3-540-45014-9_1.

[26] D. H. Wolpert, ‘Stacked generalization’, *Neural Networks*, vol. 5, no. 2, pp. 241–259, Jan. 1992, doi: 10.1016/S0893-6080(05)80023-1.

Appendix A

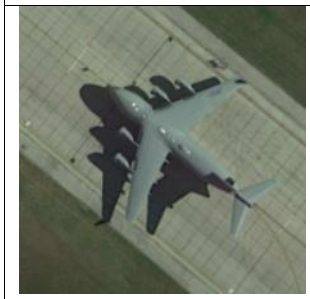
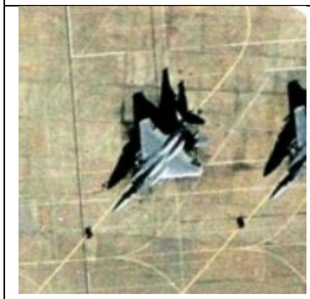
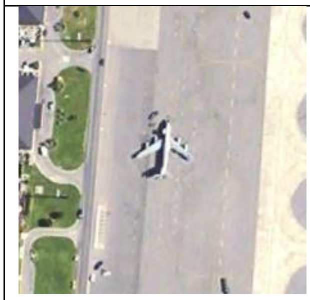
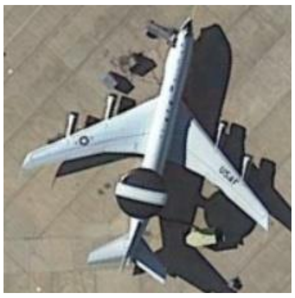
Table A. Feature of the models.

Model	**Year**	**Feature**	**Parameters (M, ~)**	**Acc. (%) on ImageNet**
AlexNet [2]	2012	First deep CNN, ReLU, dropout, GPU	60	57.0
VGG16/VGG19 [3]	2014	Simple sequence of 3×3 convolutions	138	71.5
GoogLeNet (Inception-v1) [6]	2014	Multi-scale modules, Global Avg Pooling	6.8	69.8
ResNet [4]	2015	Residual connections (skip connections)	25–60	76–77
DenseNet-121[5]	2016	Dense connections between layers (dense blocks)	8	74.9
SENet [16]	2017	Channel attention (Squeeze-and-Excitation)	115 (ResNet-152+SE)	82.7
SqueezeNet [7]	2016	Very lightweight network, model <5 MB (fire modules)	1.2	57.5
ShuffleNet [8]	2018	Group convolutions and channel shuffle operation	3.5	69.4
MobileNetV1 [9]	2017	Depthwise separable convolutions	4.2	70.6
MobileNetV2 [10]	2018	Bottleneck + residual, improved efficiency	3.4	71.8
MobileNetV3 [11]	2019	SE blocks + NAS optimization	5.4	75.2
NASNet-A-Large [12]	2017	Architecture discovered by NAS	88	82.7
MnasNet-A1 [13]	2018	NAS + latency optimization (mobile)	4.4	75.2

EfficientNet-B7 [14]	2019	Automated scaling of width/depth/resolution	66	84.3
RegNet [15]	2020	Regular scaling scheme for architectures	83	82.9
Vision Transformer (ViT-B/16) [17]	2020	Patch embedding + self-attention	86	81.8
DeiT-B [18]	2020	ViT with distillation, efficiently trained	86	81.8
Swin Transformer (Swin-T/L) [19]	2021	Hierarchical shifting windows	29–197	81–87
ConvNeXt-Tiny [24]	2022	CNN modernized for Transformer compatibility	29	82.1
MobileViT [20]	2021	CNN + lightweight Transformer blocks hybrid	5–10	~78
EfficientFormer [21]	2022	Lightweight Transformer with MobileNet speed	12–55	79–82
MaxViT [22]	2022	Multi-axis attention (local + global)	120	84–86

Appendix B

Table B1. Examples of model errors.

Image	VGG16	MobilenetV3-Large	EfficientNet-B1	Vit-B	DenseNet121	ConvNeXt-Tiny	True	Comment
	A3	A3	A3	A3	A3	A3	A11	True label error: the models correctly classified the aircraft
	A13	A13	A13	A13	A13	A13	A12	True label error: the models correctly classified the aircraft
	A13	A3	A13	A3	A14	A13	A11	Incorrect classification
	A7	A5	A11	A11	A11	A7	A7	E-3 and E-8 aircraft based on the Boeing 707 fuselage. The E-3 is distinguished by its antenna.