Deep Learning Image Classification for Cars and Planes

Felix Groer
Data Science Tech Institute
felix.groer@edu.dsti.institute
fearlix/drl

February 9, 2025

Abstract

This paper presents a deep learning approach to image classification, specifically focusing on binary classification of planes and cars. A key aspect of this research involved creating a quality dataset, ensuring a selection of images for training and evaluation. Different Convolutional Neural Network architectures were analyzed on the basis of their accuracy and efficiency to find the best performing model for the image classification. Additionally, we discuss key findings regarding the trade-offs between model complexity and generalization. Finally, we outline potential improvements for future projects.

1 Introduction

In 1956, John McCarthy introduced the term Artificial Intelligence (AI) during the Dartmouth Summer Research Project on AI. AI aims to replicate human cognitive abilities like learning, reasoning, and problem-solving, differing from traditional algorithms that follow strict, predefined steps. [7] A major milestone in AI history was the Turing Test, proposed by Alan Turing. This test suggested that machines are "intelligent" if a human cannot distinguish it from another human in conversation. [8] Over time, AI has evolved through different stages, each improving its ability to process information and make decisions. AI development has also progressed from rule-based systems to Machine Learning in the 1980s, allowing models to adjust based on input data. [6] Around 2010, Deep Learning became dominant, using artificial neural networks inspired by the human brain to detect complex patterns in large datasets. Unlike earlier models, Deep Learning requires minimal human intervention, enabling AI to improve autonomously. [2] Breakthroughs like AlexNet on ImageNet accelerated research into advanced architectures such as VGG, ResNet, and Transformers, further enhancing AI's ability to process and understand vast amounts of data [4].

2 Project Approach

2.1 Data Collection and Preparation

In this project, we created a custom dataset by gathering images from multiple sources to ensure sufficient variety and balance between the two classes planes and cars. First, we coordinated with several plane spotters, who shared their photographs on JetPhotos. After securing permission from these contributors, we implemented a data pipeline that accessed JetPhotos via the REST API and downloaded 10,000 plane images. These images are made available in the *fearlixg/planes_splitted* [1] folder, which is already divided into training and testing sets.

Next, we searched for car images to complement our plane data and found the Stanford Cars dataset on *Kaggle*.[2] From this resource, we extracted a total of 8,150 car images, ensuring coverage of various brands and models. We organized these images into the *fearlixg/cars_splitted* [3] folder, which is already divided in a train-test split. The dataset sample size can be found in Table 1.

Class	Training Samples	Test Samples
Planes	8,000	2,000
Cars	6,520	1,630

Table 1: Train and test dataset distribution before augmentation

To ensure that the project is reproducible and can be used by other researchers, the data can be reused or extended for further image classification experiments without having to perform the data preparation twice.

2.2 Data Augmentation and Transformations

Data augmentation is crucial for improving model generalization and weakening overfitting [5]. In our project, we designed an augmentation pipeline that had the goal to balance the two classes planes and cars when one class has fewer samples than the other. The primary steps of the transformation are:

- **RandomResizedCrop and RandomHorizontalFlip:** These operations allow the model to see objects at slightly different scales and orientations, reducing sensitivity to exact positioning.
- **RandomRotation:** Slight rotations increase the model's robustness to angle variations in both planes and cars.
- **ColorJitter and RandomAffine:** Adjusting brightness, contrast, and hue, as well as applying translations, helps the model learn color-invariant and position-invariant features.

[1] `https://huggingface.co/datasets/fearlixg/planes_splitted`

[2] `https://www.kaggle.com/datasets/jessicali9530/stanford-cars-dataset?resource=download`

[3] `https://huggingface.co/datasets/fearlixg/cars_splitted`

- **GaussianBlur and RandomErasing:** These forms of noise and partial occlusion train the model to focus on relevant features rather than memorizing fine details.

The function `merge_and_balance_datasets` has the goal to merge separate plane and car datasets and check the size of each class. If a disparity is detected, the function oversamples the minority class by selecting additional samples and applying the aforementioned augmentation pipeline. This process prevents the network from memorizing the duplicated images and implements better feature learning. As a result, the training set becomes balanced, which is essential for avoiding bias toward the majority class.

After applying all necessary transformations and oversampling, the final distribution of training samples is summarized in Table 2. The table reflects the balanced nature of the resulting dataset. By ensuring class equality, the model gains a more robust understanding of both feature classes.

Class	Before Augmentation	After Augmentation	Increase (%)
Planes	10,000	10,000	0
Cars	8,150	10,000	22.7

Table 2: Train dataset distribution before and after augmentation and oversampling.

By using these augmentation and balancing techniques, the final dataset offers a more diverse selection of samples. This helps to improve model performance and generalization, as the Convolutional Neural Network (CNN) can better handle variations in object appearance and environmental conditions. This adjustment helped to minimize the negative effects of oversampling and made the classification model more stable and reliable.

2.3 Model Selection

Choosing the right model is important for balancing performance and generalization in deep learning. We first tested **ResNet-50**, a well-known model for large-scale image classification [3]. However, it showed overfitting problems in our binary classification task. Even with regularization techniques like data augmentation, dropout, and weight decay, ResNet-50 memorized the training data too well, which led to 60-70% training accuracy but an 100% validation accuracy.

To improve this, we tested **MobileNetV3**, a smaller model optimized smart phone CPUs [1]. While it reduced overfitting, it also had challenges. MobileNetV3 was trained on 1,000 ImageNet classes, and fine-tuning it for our two-class problem did not always give the best results.

These tests showed the trade-off between complexity and generalization deep models like ResNet-50 tend to overfit on small datasets, while lightweight models like MobileNetV3 may not capture enough details.

After this we then decided to focus on a **custom CNN** designed specifically for our task. It includes:

- **Smaller and Simpler Architecture**: Fewer layers and parameters prevent excessive memorization while still extracting useful features.

- **Adaptive Pooling**: Standardizes feature maps to improve consistency across different input sizes.

- **Stronger Regularization**: Dropout (0.55), weight decay, and label smoothing to prevent overconfidence.
- **Dynamic Learning Rate**: A scheduler adjusts the learning rate based on validation performance for stable training.

With these adjustments, our custom CNN achieved better balance between training and test accuracy. Unlike ResNet-50, which overfitted, and MobileNetV3, which sometimes lacked expressiveness, our model performed consistently across multiple experiments. This confirms that bigger is not always better, and choosing the right model depends on the dataset and problem complexity.

2.4 Training Process

The training process is important for making sure the model learns efficiently and performs well on new data. Our approach updates the model step by step while showing live progress and saving the best model versions after each epoch of training. The training process includes several techniques to improve learning and avoid overfitting:

- **Tracking Progress, Accuracy and Estimated Time:** During each epoch, we display how much of the training is completed and how much time is left. Additionally after every epoch, we record both training and validation accuracy.
- **Validation and Early Stopping:** After each epoch, we test the model on the train validation set. If the accuracy improves, the model is saved. If there is no improvement for several epochs, early stopping is used to prevent unnecessary training loops.
- **Adjusting the Learning Rate:** A scheduler is used to change the learning rate based on validation accuracy. If the model stops improving, the learning rate is lowered to help fine-tune the learning process.
- **Final Testing and Visualization:** Once training is finished, we calculate the total training time, show accuracy trends over epochs, and create a confusion matrix. These visual tools help us understand model performance and make improvements if needed.

This training process helped the model learn well while reducing overfitting. By using early stopping, adjusting the learning rate, and saving model checkpoints, we ensure the training is efficient and reliable. The recorded metrics and graphs also helped us in the development process to monitor progress, detect problems, and make better decisions for improvements.

2.5 Model Validation

After the training is finished, we run a validation step to check how well the model works on new, unseen data. Each image from the test set is given to the model, and the outputs are analyzed to get both the predicted class argmax and the confidence level of the prediction softmax. We then calculate different performance measures to understand how good the model is:

- **Accuracy** shows the percentage of correct predictions.
- **Confusion matrix** helps us see where the model makes mistakes by showing how many images were classified correctly and incorrectly for each class.
- **Precision, recall, and F1-score** give more details about how well the model performs for each class, especially useful when one class is harder to identify.
- **ROC-AUC score** measures how well the model separates the two classes by looking at how confident it is in its predictions.

By using these different measurements, we can better understand the strengths and weaknesses of our trained model. Based on that we did more improvements and retrained the model until it reached the final state.

2.6 Model Checkpointing

To make collaboration and model sharing easier, we use the Hugging Face platform in our workflow. The pipeline automatically creates a repository and set up a naming convention for the model files. The best-performing model from the training process is saved and uploaded to Hugging Face. To keep track of different versions, we add a timestamp to each model filename. These steps create a structured process for saving, tracking, and sharing trained models. By using Hugging Face, we make it easier for others to access, test, and improve the models without having to repeat the entire training process. The models that were created in this project can be found in *Hugging Face.*[4]

[4] `https://huggingface.co/fearlixg/cars_vs_planes_model`

3 Results, Discussion and Limitations

The final training process was carried out over 20 epochs, with the model improving step by step. Table 3 shows the key accuracy results for training and testing.

Epoch	**Train Accuracy (%)**	**Test Accuracy (%)**
1	75.05	82.36
5	89.82	92.39
10	92.37	95.54
15	93.76	96.20
20	94.43	96.56

Table 3: Training and test accuracy over time

At the beginning, the model improved fast. The training accuracy increased quickly, but the test accuracy changed slightly as the model learned new patterns. After 10 epochs, the model reached 92.37% training accuracy and 95.54% test accuracy, which showed good learning progress. In the final epochs, the model reached 94.43% training accuracy and 96.56% test accuracy, meaning that it was performing well on unseen data. For the final model, we calculated important classification metrics that can be found in Table 4.

Metric	**Score (%)**
Test Accuracy	96.56
Precision	97
Recall	97
F1-Score	97
ROC-AUC Score	100

Table 4: Final model evaluation metrics

Table 5 gives details about the model's performance for each class.

Class	**Precision (%)**	**Recall (%)**	**F1-score (%)**
Plane (Class 0)	99	93	96
Car (Class 1)	95	99	97
Overall Accuracy		96.56	

Table 5: Final classification report

These results show that the model was very good at telling the difference between planes and cars. The recall for cars was 99%, meaning that the model correctly identified almost all car images. However, the recall for planes was slightly lower at 93%, showing that some planes were misclassified. This could be because some plane pictures have also cars visible in the training dataset due to cars being also on ground at airports.

Even though the model performed well, it still had some limitations:

- **Overfitting Issues:** The test accuracy was stable, but the difference between training and test accuracy became larger in later epochs. This suggests that some overfitting might have happened. Adding stronger regularization, having a larger dataset or stopping training earlier could reduce this issue.
- **Dataset Bias:** The dataset was created using images from JetPhotos and Stanford Cars, which means that some specific types of images might be overrepresented. Also on some plane images cars were visible which could also mislead the model. If the model is tested on new, real-world images, it might not perform as well.
- **Class Imbalance:** Even though we used oversampling and data augmentation, the recall score for planes was still lower than for cars. This shows that the model was slightly better at recognizing cars than planes. A better balance between the two classes and more data for each of the two classes could help improve this.

4 Conclusion and Future Work

This research studied deep learning for image classification, focusing on distinguishing between planes and cars using a custom CNN. We showed that careful model selection, data augmentation, and regularization techniques were important for achieving high accuracy.

Our best model achieved a test accuracy of 96.56%, with a precision, recall, and F1-score of 97%. We compared different architectures and found that ResNet-50 was too complex for our dataset and suffered from overfitting. Meanwhile, MobileNetV3 was lightweight but had trouble learning enough features for accurate classification. The custom CNN provided a good balance between performance and generalization.

For future improvements, we suggest:

- **Increasing the Dataset:** Using more diverse images could help the model generalize better.
- **Using Semi-Supervised Learning:** Pretraining the model on a larger dataset before fine-tuning on our dataset could improve performance.
- **Explaining Model Decisions:** Methods like Grad-CAM or SHAP could be used to visualize which features the model is focusing on when making predictions.

In conclusion, this project shows that deep learning can be successfully used for binary image classification. However, it also highlights the importance of choosing the right model, preprocessing data properly, and reducing overfitting. These findings can be applied to future projects.

References

[1] Adam, H., Chen, B., Chen, L.-C., Chu, G., Howard, A., Le, Q. V., Pang, R., Sandler, M., Tan, M., Vasudevan, V., Wang, W., Zhu, Y., *Searching for MobileNetV3*, Google, 2019.

[2] Cole, T., *Erfolgsfaktor Künstliche Intelligenz*, Hanser, 2020, pp. 41–45.

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

[4] Hinton, G., Krizhevsky, A., Sutskever, I., "ImageNet classification with deep CNNs," *Advances in Neural Information Processing Systems*, 2012, pp. 1097–1105.

[5] Khoshgoftaar, T., Shorten, C., "A survey on image data augmentation for deep learning," *Journal of Big Data*, vol. 6, 2019, pp. 1–48.

[6] Matsuo, Y., *Is Artificial Intelligence Behind Human: What's Beyond Deep Learning*, Kadokawa, 2019, pp. 39–60.

[7] McCarthy, J., Minsky, M. L., Rochester, N., Shannon, C. E., *A Proposal for the Dartmouth Summer Research Project on Artificial Intelligence, August 31, 1955*, *AI Magazine*, vol. 27, 2006, pp. 12–14.

[8] Turing, A. M., *Computing Machinery and Intelligence*, *Mind*, vol. 59, 1950, pp. 433–460.