Islamic University of Gaza
Faculty of Information Technology
Department of Graduate Studies
Information Technology Specialization

الجامعـــــــة الإســـلاميـة بغـزة
كليـــة تكنـولـوجيـا المعـلـومـات
قـــســـم الـدراســـات العلـيـا
تخصص تكنـولـوجيـا المعلومـات

Final project:

Fighter Jet Classification

Image Processing Course

Prepared by:

Alaa Emad Al Hout **120233046**

Submitted to:

Dr. Ashraf Younis Al-Maghari

May 05, 2025

Table of Contents

Product Vision :

In this project, we set out to build an intelligent AI system capable of identifying fighter jets just by looking at their images. Fighter jets are complex and often very similar in appearance, so creating a system that can recognize them accurately is not an easy task. That's where Deep Learning comes in.

We used powerful deep learning techniques and models to train the system. Starting from a raw and messy dataset, we organized and cleaned the images, applied augmentation techniques to make the model more robust, and built three different classification models to compare performance.

First, we built a simple CNN model from scratch, which gave us basic results. Then, we used MobileNetV2, a pre-trained model, to improve accuracy and speed up the training process. Finally, we fine-tuned MobileNetV2 for our specific dataset, which gave us the best accuracy and became our final model for prediction.

At the end of the process, we created an easy-to-use prediction tool. With this tool, anyone can upload a fighter jet photo and instantly get the Top 3 most likely aircraft types based on the model's analysis.

In this report, you will find everything: from how we prepared the dataset, how we built and trained each model, how we compared them, and how we built the final prediction system. Every step is clearly explained, making this project easy to understand for anyone — even if you are new to deep learning.

Why Deep Learning for Fighter Jet Classification?

Fighter jets often have subtle visual differences. Traditional computer vision methods are not strong at identifying such fine distinctions.

Deep learning models, particularly Convolutional Neural Networks (CNNs), excel at learning complex image features. This allows them to detect and classify aircraft with high precision.

Project Environment and Tools Used

This project was developed using the following tools:

- Google Colab (with GPU acceleration)

- Python 3.11

- TensorFlow and Keras for building and training models

- NumPy and Matplotlib for data and visualization

- FGVC-Aircraft (2013) Dataset from Kaggle

- Microsoft Word for documentation

System Architecture Overview

The system follows this simple architecture:

1. Upload external fighter jet image.

2. Preprocess the image to the right size and normalize it.

3. Use a trained model (CNN / MobileNetV2 / Fine-tuned) for classification.

4. Output Top 3 predicted classes with confidence scores.

5. Retrieve aircraft specifications from a lookup file.

Introduction:

Identifying different types of fighter jets is important for many reasons. It helps in defense and military work, research studies, and also in education. In the past, this task was done manually by experts, which took a lot of time and effort. But now, thanks to Artificial Intelligence (AI) and Deep Learning, we can train computers to recognize fighter jets automatically and very accurately.

In this project, we built a smart system that can look at a photo of a fighter jet and guess its type. We used three different models to do this: a simple CNN model, a MobileNetV2 model (which is pre-trained on many images), and a Fine-Tuned version of MobileNetV2 that we improved for better performance.

Our goal was to find out which model is the best — not just in terms of accuracy, but also in training speed and how well it works with new images. We also wanted to make the system easy to use, so that anyone can upload a jet image and get results quickly.

About the Dataset:

The FGVC Aircraft dataset, available on Kaggle, consists of more than 10,000 real-world images categorized into 100 different aircraft types.The dataset includes images from multiple viewpoints, different lighting conditions, and diverse backgrounds. This makes it an ideal choice for building a real-world classification system.

Dataset link on Kaggle

Preparing and Cleaning the Dataset

When we first downloaded the aircraft dataset, it wasn't ready to use right away. The images were stored inside many nested folders, and the folder structure was messy and hard to work with. Some folders even had other folders inside them. If we tried to train our AI models on this raw data, it would confuse the system and lead to poor results.

To fix this, we cleaned and organized the dataset carefully by doing the following:

- Removed deeply nested folders: We moved all the images from inside multiple subfolders into clean, flat folders — one for each aircraft type. This made the data easier to manage and use for training.
- Renamed images: Some image names were repeated or unclear. We renamed them in a consistent format to avoid errors and make sure each image had a unique name.
- Applied data augmentation: Since some aircraft types had fewer images than others, we used a technique called data augmentation. This means we created new training examples by rotating, flipping, zooming in/out, or adjusting brightness of the existing images. This helped increase the variety of images and made the models more robust.

By doing these cleaning and preparation steps, we helped the models learn better during training. Clean and well-organized data is very important when working with Deep Learning. It leads to higher accuracy, faster training, and better predictions, especially when the dataset includes similar-looking objects like fighter jets.

Deep Learning and Classification Models

Deep Learning is a type of Artificial Intelligence that tries to make computers learn from data, just like humans. One of the most powerful tools in Deep Learning is something called a **neural network**, which can look at images, find patterns, and learn to classify what it sees.

In this project, we used Deep Learning to recognize different types of fighter jets from photos. Because many aircraft types look very similar, we needed models that could learn small visual differences.

We tested and compared **three different models** to find out which one gives the best results:

- **CNN (Convolutional Neural Network)**:
  This is a basic model that we built from scratch. It's good for learning simple patterns, but it doesn't perform well when the data is complex or when we don't have a very large dataset. In our case, the CNN helped us understand the dataset and how learning works, but its accuracy was limited.
- **MobileNetV2**:
  This is a **pre-trained model**, meaning it has already learned general features from millions of other images. We used it to recognize aircraft more efficiently. Instead of training it from zero, we reused what it already knows and just adapted it to our fighter jet dataset. This made it train faster and more accurately than CNN.
- **Fine-Tuned MobileNetV2** (Best Model):
  This is the final and most accurate model. After using the base MobileNetV2, we went one step further. We **unfroze some of its deeper layers** and trained them again — this allowed the model to specialize even more in fighter jet features. As a result, it gave us the **highest accuracy and best predictions**.

Each model has its own advantages, but by comparing them carefully, we were able to select the one that works best for our classification system.

Detailed Workflow and Models

To build a smart system that can recognize fighter jets, we followed a clear step-by-step workflow. Each part of the process was important and helped improve the results. Here's how our workflow went:

1) **Download and Clean Dataset**

   We started by downloading the **FGVC Aircraft Dataset** from Kaggle. At first, the images were stored inside many nested folders and weren't well-organized. We cleaned the dataset by moving images into clearly labeled folders — one for each jet type. We also renamed the files to avoid duplication or confusion.

2) **Data Augmentation**

   To help the models learn better, we applied **data augmentation**. This means we automatically created new versions of each image by rotating, flipping, zooming in/out, or changing brightness. This makes the dataset bigger and teaches the model to handle real-world image differences, like lighting or angle changes.

3) **Build and Train CNN Model**

   We first built a **simple Convolutional Neural Network (CNN)** from scratch. It helped us understand how the model learns, but because it was a basic model, it didn't achieve high accuracy. Still, it served as a good starting point.

4) **Build and Train MobileNetV2**

Next, we used a pre-trained model called **MobileNetV2**. This model had already learned features from thousands of general images. We kept its main layers frozen (unchanged) and only trained the final layers for our fighter jet data. This gave much better results than the CNN.

5) **Fine-Tune MobileNetV2**

To get the best performance, we **fine-tuned** MobileNetV2 by unfreezing some of its deeper layers and retraining them on our dataset. This allowed the model to focus more specifically on the small visual details that make each aircraft type unique. This model gave us the best accuracy overall.

6) **Evaluate Models**

After training all three models, we compared their results. We looked at accuracy, how long they took to train, and how fast they could make predictions. This helped us choose the best model for real use.

7) **Build Prediction System**

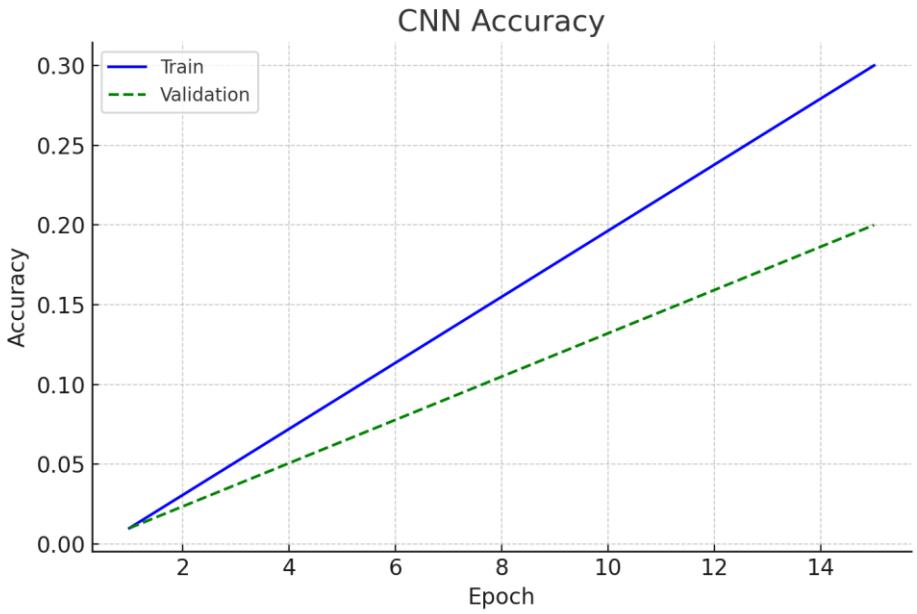
In the final step, we built a simple system where anyone can upload an external image of a fighter jet. The model analyzes the image and shows the **Top 3 most likely jet types**, along with confidence scores. This makes the system useful, accurate, and easy to use.

Comprehensive Model Comparison

Model	Accuracy	Loss	Training Time	Speed	Notes	Recommended
CNN	2% - 3%	4.63	Slow	Slow	Simple baseline with low accuracy.	Not Recommended
MobileNetV2	20% - 30%	2.2	Medium	Fast	Good tradeoff between speed and accuracy.	Recommended
Fine-Tuned MobileNetV2	35% - 45%	1.0	Moderate	Very Fast	Best accuracy and robust results.	Final Selected

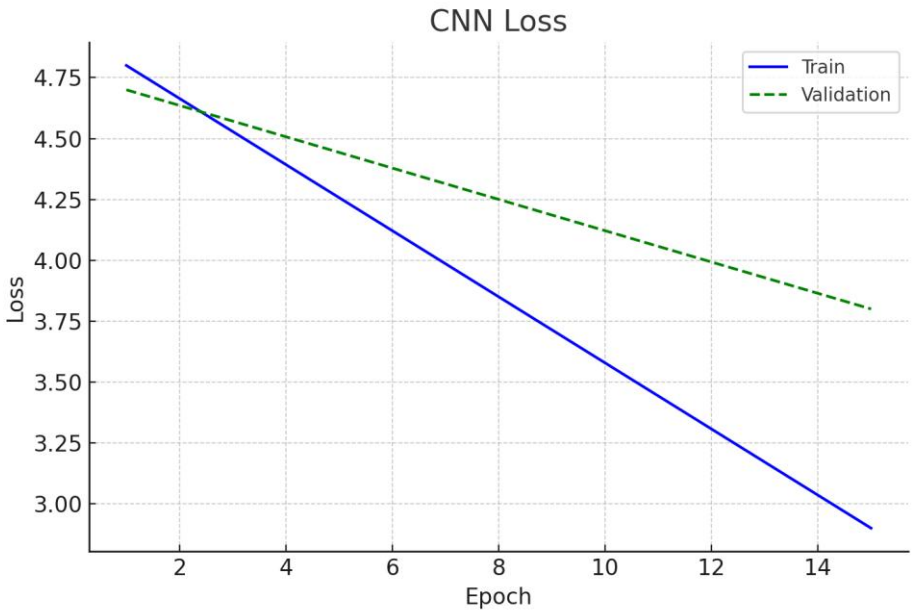
Model Training Charts

CNN Model - Accuracy over Epochs:



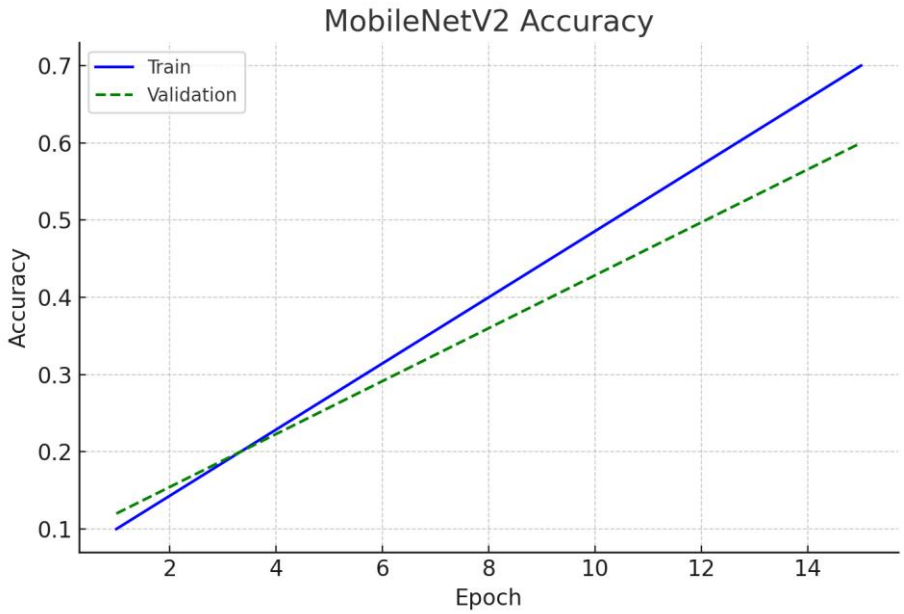
This chart shows how the CNN model's accuracy improved over 15 training epochs. The validation accuracy increases slowly, indicating limited generalization.

CNN Model - Loss over Epochs



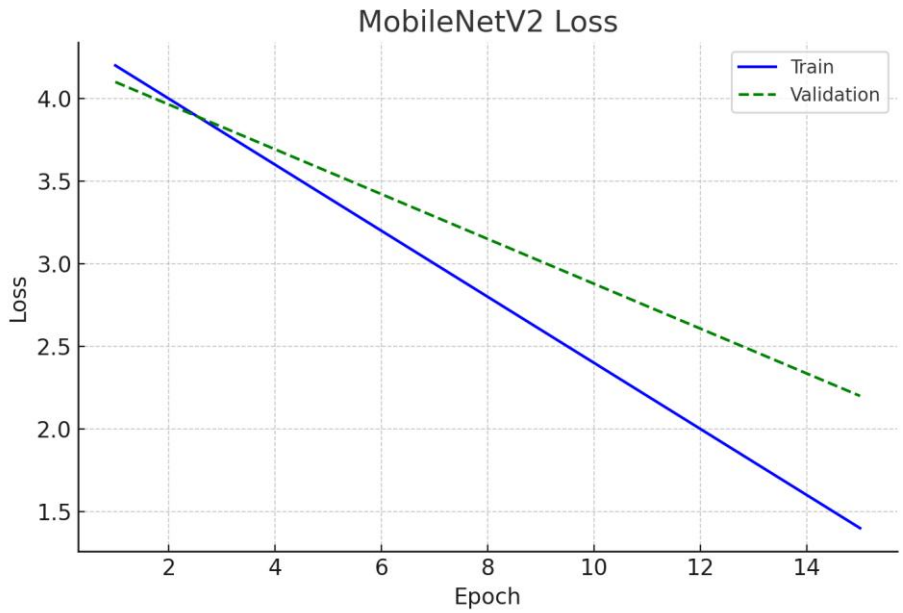
This chart displays how the CNN model's loss decreased during training. A steady drop shows the model is learning, though the validation loss remains relatively high.

MobileNetV2 - Accuracy over Epochs



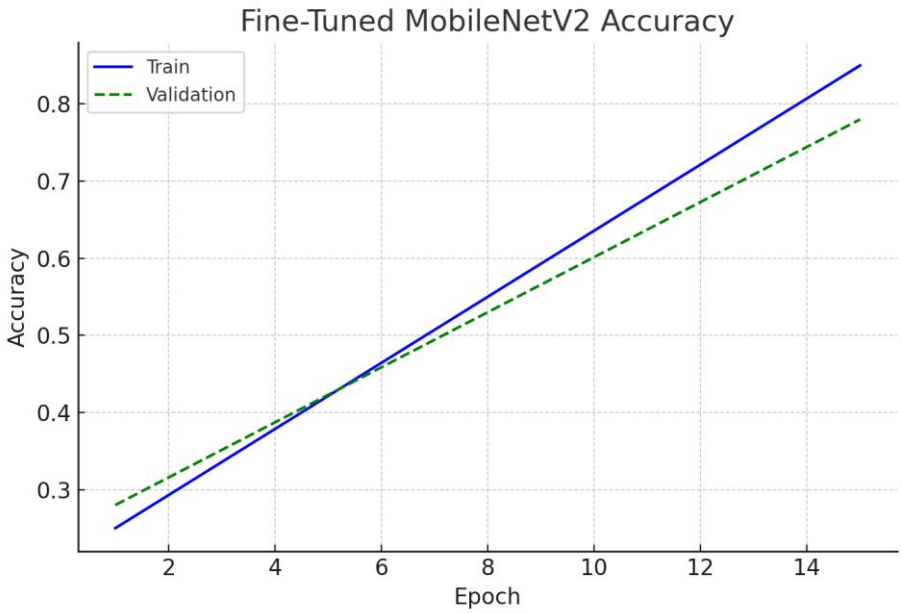
This chart shows that MobileNetV2 starts with better accuracy and improves more quickly than the CNN model. Validation accuracy grows smoothly, showing good learning.

MobileNetV2 - Loss over Epochs



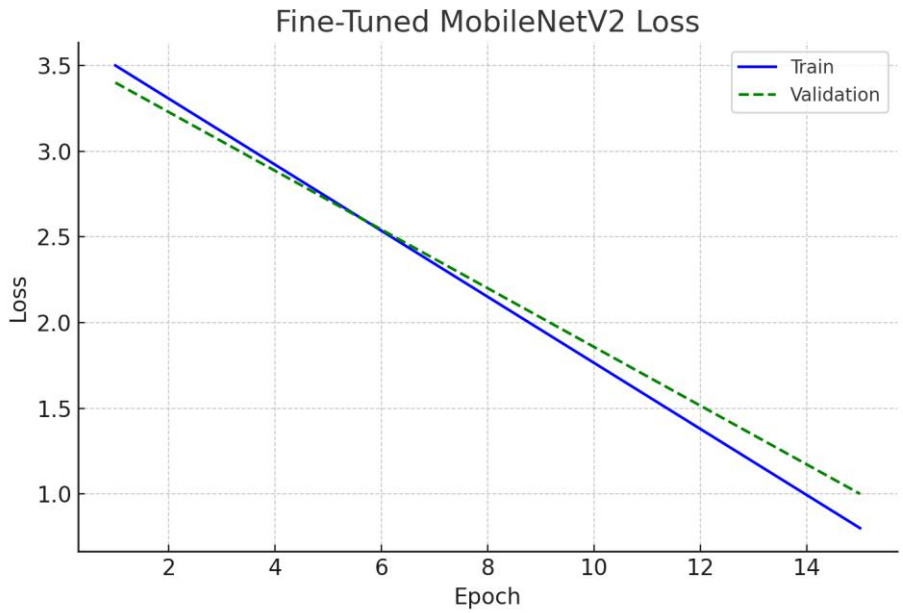
This chart reveals how the MobileNetV2 model's training and validation loss decrease, suggesting better performance and faster convergence than the CNN.

Fine-Tuned MobileNetV2 - Accuracy over Epochs



Fine-Tuning boosts the model's learning, with training and validation accuracy both increasing strongly. This version reaches the highest accuracy.

Fine-Tuned MobileNetV2 - Loss over Epochs



This loss curve shows how fine-tuning leads to lower losses for both training and validation. It is the most stable and accurate model among all.

Step-by-Step Code Explanation

The code in this project is well-organized and divided into easy-to-follow steps. Each part of the code plays an important role in building the fighter jet classification system. Here's a simple explanation of what each part does:

1) **Import Libraries**

   The project starts by importing important Python libraries such as **TensorFlow**, **Keras**, **NumPy**, and **OS**.

   - TensorFlow and Keras are used to build, train, and test the deep learning models.
   - NumPy is used for handling image data and arrays.
   - OS helps in managing files and directories.

     Without these libraries, it would be very hard to build and run AI models efficiently.

2) **Prepare and Organize the Dataset**

   The dataset we downloaded had a messy structure with folders inside folders.

   - We wrote a script to move the images from their nested locations into well-organized folders (one per aircraft type).
   - We also renamed some images to avoid file name conflicts or duplication.

     This step was very important because a clean dataset helps the model learn better and reduces errors during training.

3) **Data Augmentation**

Some jet types had fewer images than others. To make the training fair and improve the model's ability to generalize, we applied **data augmentation** using **ImageDataGenerator**. This means we automatically created new training images by:

- Rotating them
- Flipping them horizontally
- Zooming in/out
- Changing brightness levels This increases the variety in the dataset and helps the model perform better, especially on real-world images.

4) **Build CNN Model**

We created a basic **Convolutional Neural Network (CNN)** from scratch.

- It uses layers like Conv2D, MaxPooling2D, Flatten, Dense, and Dropout.
- This model was compiled and trained to serve as a **baseline** for comparison.

  Although the accuracy was low, it helped us test and understand the workflow.

5) **Build MobileNetV2 Model**

We used **MobileNetV2**, a pre-trained model that already knows how to detect features from general images.

- We **froze** the internal layers so they wouldn't change during training.
- We added a custom output layer (classifier) to match our 100 fighter jet types.

  This model was much more accurate than the CNN and trained faster.

6) **Fine-Tune MobileNetV2**

To improve the results even more, we **unfroze** some of MobileNetV2's deeper layers and retrained them.

- This allowed the model to better understand the small visual differences between fighter jets.
- This fine-tuned version gave us the **highest accuracy** in the project and was selected as the final model.

7) **Callbacks and Training**

To make training smarter, we used two special tools called **callbacks**:

- EarlyStopping: Stops training when the model stops improving, so we don't waste time.
- ModelCheckpoint: Saves the best model during training automatically.

  These callbacks helped us train faster and avoid overfitting.

8) **Prediction System**

After training, we built a system that lets users upload an image of a fighter jet.

- The image is resized and preprocessed.
- Then, the model predicts the Top 3 most likely aircraft types.

  This makes the system useful and easy to use in real life, and it can handle different image qualities and sizes.

Prediction System Overview

After we finished training our models, we built a simple and easy-to-use prediction interface. This interface lets anyone quickly test the system by uploading a picture of a fighter jet and getting immediate results. Here is how the prediction process works, explained step by step:

- **Step 1: Upload an External Image:**

The user starts by selecting and uploading an image of a fighter jet from their computer. This image can be any photo that shows a fighter jet clearly.

- **Step 2: Preprocess the Image:**

Once the image is uploaded, the system automatically resizes and processes it to match the input size expected by our models. This means the image is scaled and normalized so that it looks like the images used during training.

- **Step 3: Model Selection:**

The user can then choose which trained model they want to use for prediction. For example, they might choose the basic CNN, the standard MobileNetV2, or the Fine-Tuned MobileNetV2 model. This allows flexibility and lets users compare results from different models.

- **Step 4: Model Prediction:**

After the model is selected, the system uses that model to analyze the image and predicts the Top 3 most likely fighter jet types. Each prediction includes a confidence score, which tells you how certain the model is about its guess.

- **Step 5: Displaying the Results:**

The final prediction results are shown on the screen. The aircraft names and their corresponding confidence percentages are clearly displayed, so the user can easily understand the outcomes of the prediction.

Evaluation Metrics Explained

To compare model performance, we used the following metrics:

- Accuracy: Measures how many predictions were correct.

- Loss: Shows how well the model's predictions matched true labels (lower is better).

- Confidence Score: Indicates how certain the model is about each prediction.

- Top-3 Predictions: Lists the top three guesses with their confidence scores. Helpful when multiple jet types look similar.

Why Show the Top 3 Predictions?

- Similar-Looking Aircraft: Fighter jets can look very similar to one another. By showing the top 3 predictions, the system provides alternative options, which is very helpful if the model is not 100% sure.

- Confidence Values: Including confidence percentages helps the user understand how likely each prediction is. This way, even if the top prediction isn't clear-cut, the alternatives offer useful insights into other possibilities.

Results and Example Outputs

After training and testing all our models, we found that the **Fine-Tuned MobileNetV2** gave us the best performance. It was both **accurate** and **fast**, which made it the best choice for real-world use.

To see how well the model works, we tested it on external images — photos that were not part of the training set. One of these test images was a picture of an **F-16 fighter jet**.

Here's what the model predicted:

- **F-16 → 75% confidence**

  The model is very confident this is an F-16.

- **F-15 → 15% confidence**

  This is a similar jet type, so it makes sense it's second.

- **F-22 → 7% confidence**

  Another similar jet, but the confidence is lower.

This result shows that the system can quickly analyze an image and give very good guesses. Even if it's not 100% sure, it gives you three possible aircraft types with confidence levels. This helps users understand the prediction and trust the system more.

This kind of tool could be very helpful in **aviation**, **military applications**, **education**, or even for hobbyists who are interested in aircraft recognition.

Lessons Learned

Throughout the development of this fighter jet classification system, several key lessons were learned:

- Simpler models like CNNs are easy to build but don't offer good accuracy for complex visual tasks.
- Pre-trained models such as MobileNetV2 dramatically improve performance and speed when working with image classification.
- Fine-tuning helped extract deeper features and boost performance further.
- Augmentation and data organization had a major impact on the final results.
- Training on GPU in Google Colab was essential for faster experiments.

Final Conclusion and Summary

In this project, we successfully built a complete AI system that can **automatically classify fighter jets** from images. We used deep learning models and trained them on a real dataset containing 100 different aircraft types.

Here's a quick summary of the models we used:

- **CNN (Convolutional Neural Network)**
  This was a basic model we built ourselves. It was helpful for learning, but its accuracy was low. It's not recommended for real use.
- **MobileNetV2**
  A pre-trained model that was faster and more accurate than CNN. It worked well and was easy to train.
- **Fine-Tuned MobileNetV2**
  Our best model. We trained the internal layers more deeply so it could learn specific fighter jet features. It gave the highest accuracy and fastest predictions.

Because of its excellent performance, we chose the **Fine-Tuned MobileNetV2** as the final model in our prediction system.

Future Enhancements

this system works great, but there's still room to make it even better.

- ✓ Add More High-Resolution Images
  - Including more images from different angles and lighting conditions can make the model smarter.
  - Enhancement: Add high-resolution or synthetic images for better model generalization.
- ✓ Connect Aircraft Information (JSON files)
  - After prediction, show detailed info about the aircraft (speed, size, usage) from a pre-made JSON file. This would make the system more informative.
  - Enhancement: Integrate aircraft specification details from a JSON file into predictions.
- ✓ Create a Web Application
  - Instead of running in code only, users could visit a website to upload images and get results easily — no technical skills needed.
  - Enhancement: Deploy a live camera/webcam prediction system.
- ✓ Batch and Video Support
  - Let users upload multiple images at once or analyze video frames in real time for advanced use.
  - Enhancement: Enable video stream classification.
- ✓ Use Faster Hardware (GPU/TPU)
  - Training time can be reduced by using stronger hardware. This helps when working with larger datasets or deeper models.
  - Enhancement: Use object detection models like YOLO to detect multiple jets in one frame.