Mini project how we did it

Step 1: first know what we need to is the d.l
It mimics the human brain

Step 2: what architecture it follows
These are the cnn models
•**ShuffleNet**

•**SqueezeNet**

•**EfficientNet B0**

•**ResNet-50**

MobileNet

Step 3: why are we doing this project?
Normally there are no proper algorithms for finding the fruit quality by just using the images .so we want to fill that gap
“Possible questions”:
Q: What do you mean by “no proper algorithms”?
👉 *Most existing methods are not accurate or image-based.*

Q: Why is fruit quality classification important?
👉 *To reduce waste and ensure better quality for consumers.*
Q: Why use images instead of sensors?
👉 *Images are cheaper, faster, and non-destructive.*

Q: Why deep learning for this task
👉 *It automatically learns features and gives high accuracy.*

Q: What challenges do you expect?
👉 *Small datasets, class imbalance, and visual similarity.*

Q: What are the real-world uses?
👉 *Sorting machines, supermarkets, and mobile apps.*

Q: How does your project fill the gap?
👉 *It provides a practical image-based deep learning solution.*

Q: Which model performed best in your project?
👉 *MobileNet gave the highest classification accuracy*

Step 4: now first discuss about the cnn and later we can go deep into the project

Sure! Here's a **detailed yet easy-to-understand explanation of CNN (Convolutional Neural Network)** — including an example at the end.

🧠 **What is CNN?**

CNN (Convolutional Neural Network) is a type of **deep learning algorithm** that is **specially designed to work with images**.

Just like your brain processes visual patterns (like recognizing faces, objects, letters), CNN helps computers **see and understand images**.

🔍 **Why CNN for images?**

Normal neural networks (like dense layers) don't work well with large images — too many pixels = too many connections = slow & less accurate.

CNN solves this by:

- **Focusing on small image areas (features)** at a time
- **Reusing filters** to find patterns like edges, shapes, and textures

- **Reducing the number of parameters**

🧱 Basic Layers in a CNN

1. 🟨 **Convolutional Layer**
    - Applies **filters (kernels)** to the image
    - Extracts **features** like edges, corners, textures
      Example: Finds where the edges of an apple are
2. 🟦 **ReLU Layer (Activation)**
    - Adds **non-linearity** so the model can learn complex patterns
      Example: Helps distinguish between round (apple) and long (banana)
3. 🟪 **Pooling Layer (usually MaxPooling)**
    - Reduces the image size (downsampling)
    - Keeps important information only
      Example: Shrinks image from 100x100 to 50x50 while keeping key features
4. 🟩 **Fully Connected Layer (Dense)**
    - Connects all neurons to make the final decision
      Example: Predicts if it's a **fresh apple**, **rotten apple**, or **banana**
5. 🟥 **Output Layer**
    - Gives the final **probabilities or classes**
      Example: [0.9 fresh apple, 0.1 rotten apple]

📊 CNN Architecture Example (Simple)

Suppose you are building a fruit quality checker using CNN:

Input Image (e.g., 100x100 image of a mango)
↓
Conv Layer 1 → 16 filters detect edges
↓
ReLU Activation

↓
Max Pooling → reduce size
↓
Conv Layer 2 → more filters detect texture, color
↓
ReLU + Pooling again
↓
Flatten + Fully Connected Layer
↓
Output Layer → Predicts: Fresh Mango, Rotten Mango, Not Mango

---

📷 **Real-Life Example:**

Let's say you input this picture:

👉 

CNN does the following:

- **Detects edges** (outline of the mango)
- **Finds patterns** (dark spots, wrinkles)
- **Summarizes features** (texture, shape)
- **Classifies it** as: **"Rotten Mango"**

✅ **Key Advantages of CNN:**

- Learns features automatically
- Requires fewer parameters than normal neural nets
- Works great for **image classification**, **object detection**, **face recognition**, etc.

🧠 **Summary:**

CNN is a powerful deep learning model that mimics the way humans see. It processes images through layers to detect patterns and classify objects.

Step 5:

We are classifying the fruits apple,banana,oranges

Training samples: 10901
Test samples: 2698

Split sizes:
- Train: 8720 samples
- Validation: 2181 samples
- Test: 2698 samples

`ImageDataGenerator` is used for:

- **Preprocessing + Augmentation** in one step
- Feeding data **in batches** during training
- Automatically **resizing**, **normalizing**, and **labeling** images

---

🧠 Summary:

You're using `ImageDataGenerator` to **create more diverse training images** by randomly altering them — which helps your CNN **learn better and perform well on new data**.