Dextrocardia Detection Using CNN

1. Introduction

During my research on AI-based medical imaging projects, I came across several pneumonia detection projects. While exploring them, an idea struck me: why not focus on **dextrocardia**, a rare condition where the heart is located on the right side of the chest instead of the left? Interestingly, I have this condition myself, which made the idea even more personal and motivating.

Dextrocardia occurs in roughly **1 in 10,000 people**, making it a rare congenital anomaly. Detecting it automatically using AI could help medical professionals in diagnosis, especially in situations where subtle differences in chest X-rays may be overlooked. With this goal in mind, I decided to create a **CNN-based model** to classify X-ray images as either normal or showing dextrocardia.

---

2. Objective

The main objective of this project was straightforward: **to build an AI model capable of detecting dextrocardia from chest X-rays**. Along the way, I wanted to learn and demonstrate the full AI workflow—from dataset preparation and model building to evaluation and deployment—while making the project meaningful and personally relevant.

---

3. Model Selection

I chose a **Convolutional Neural Network (CNN)** for several reasons. CNNs are designed for images, and chest X-rays contain complex spatial patterns like shapes, edges, and textures. Unlike traditional models like SVMs, Decision Trees, or fully connected networks, CNNs **automatically learn these features** without manual extraction.

I also considered pretrained models like VGG or ResNet. However, our dataset is small, and using these large networks could have easily caused **overfitting**. Pretrained models are also unnecessarily complex for a simple binary task like “normal vs dextrocardia.” Building a **custom CNN from scratch** allowed me to control the architecture, depth, and number of parameters, keeping the model lightweight, efficient, and tailored to the dataset.

In short, a custom CNN was **the perfect balance between simplicity, efficiency, and accuracy** for this project.

---

4. Dataset

The dataset I used was actually **a pneumonia X-ray dataset from Kaggle**. It included normal and pneumonia-labeled images. Since there are no publicly available datasets specifically for dextrocardia, I **creatively transformed the data** by flipping the X-ray images horizontally. This created a mirrored version of the chest X-rays, simulating a right-sided heart orientation.

After this transformation, I applied standard preprocessing steps:

- **Resizing** all images to a uniform size.
- **Normalizing** pixel values to scale them between 0 and 1.
- **Data augmentation**, including rotations and shifts, to increase the effective dataset size and help the model generalize better.

This approach allowed me to create a dataset that represents dextrocardia while keeping the project manageable and reproducible.

---

5. Workflow / Pipeline

The project followed a clear, step-by-step workflow:

1. **Data Collection:** Downloaded pneumonia X-ray images from Kaggle.
2. **Data Transformation:** Flipped images horizontally to simulate dextrocardia.
3. **Data Preprocessing:** Resized, normalized, and applied data augmentation.
4. **Model Design:** Built a custom CNN consisting of convolutional, pooling, flatten, dense, and dropout layers.
5. **Training:** Trained the CNN using **backpropagation** and the **Adam optimizer**, with **binary cross-entropy loss** for classification.

6. **Validation:** Monitored validation accuracy and loss to prevent overfitting.
7. **Testing & Evaluation:** Tested the model on unseen images to check performance.
8. **Prediction:** The model outputs whether the X-ray shows normal heart orientation or dextrocardia.

By following this workflow, I ensured that each step—from dataset creation to prediction—was structured and reproducible.

---

6. Model Architecture

The CNN architecture is simple yet effective:

- **Conv2D layers:** Extract low-level to high-level features like edges, shapes, and textures.
- **MaxPooling layers:** Reduce the spatial dimensions while keeping important information.
- **Flatten layer:** Converts 2D feature maps into 1D vectors.
- **Dense layers:** Integrate features and perform classification.
- **Dropout layers:** Prevent overfitting by randomly deactivating neurons during training.
- **Sigmoid activation:** Produces probabilities for binary classification.

This combination of layers allows the model to learn the distinguishing features of normal vs flipped X-rays efficiently.

---

7. How the Model Works

Once the images are fed into the CNN, the **convolutional layers detect patterns** in local regions of the X-ray. MaxPooling layers reduce dimensions and keep computational efficiency high. The **flattening and dense layers** then combine these features for final classification.

During training, the **backpropagation algorithm** calculates gradients of the loss function, and **gradient descent (Adam optimizer)** updates the weights to minimize errors. After sufficient

training, the model can predict whether an X-ray shows normal heart orientation or dextrocardia with high accuracy.

---

8. Knowledge of Algorithms

This project uses a combination of fundamental algorithms and techniques:

1. **Convolution:** Extracts local features from images.
2. **Pooling (MaxPooling / AveragePooling):** Downsamples feature maps to reduce computation and prevent overfitting.
3. **Activation Functions (ReLU, Sigmoid):** Introduce non-linearity; Sigmoid outputs probabilities.
4. **Dropout:** Regularization to prevent overfitting.
5. **Flattening:** Converts feature maps into vectors for dense layers.
6. **Dense Layers:** Perform final classification.
7. **Backpropagation:** Computes gradients for weight updates.
8. **Gradient Descent / Adam Optimizer:** Minimizes the loss function.
9. **Binary Cross-Entropy Loss:** Measures error in binary classification.
10. **Data Augmentation Algorithms:** Flipping, rotation, and shifting to expand the dataset and improve generalization.

Together, these algorithms enable the model to **learn efficiently from images and classify them accurately**.

---

9. Future Scope / Advancement

Although this project was initially created for **GDG purposes**, I envision many ways to advance it:

- Extend the model to detect **situs inversus** and related congenital conditions.
- Build a **full medical imaging platform** with frontend and backend, allowing users to upload X-rays and receive predictions.
- Deploy the platform on **cloud services** like Streamlit Cloud, Heroku, or AWS for real-world accessibility.
- Add advanced AI features like anomaly highlighting, automated reports, and patient management tools.

These improvements could transform the project from a **proof-of-concept** into a **fully functional medical imaging solution**.

---

10. Conclusion

In summary, this project demonstrates a **personal, practical application of AI** to detect a rare medical condition affecting roughly **1 in 10,000 people**. By creatively transforming a pneumonia dataset and building a custom CNN, I developed a workflow that covers data preparation, model training, evaluation, and prediction.

The project highlights how AI can provide innovative solutions in medical imaging, even for rare conditions like dextrocardia, and lays the foundation for future expansion into more comprehensive medical tools.