<CSC462 (ML)><CourseProject>

Convolutional Neural Networks for Image Classification

Asma Albahkly

Danah Alomran

Shahad Almutairi

Image classification task and dataset

This project focuses on a binary image classification task: distinguishing between images of cats and dogs. The primary goal is to develop a model capable of correctly identifying whether a given image contains a cat or a dog. This task has practical applications, such as automated pet identification systems and organizing image libraries.

The chosen dataset for this project is the Cats vs. Dogs dataset by Anthony Therrien, sourced from Kaggle. It contains 1,000 images, evenly divided into 500 images of cats and 500 images of dogs.

Methodology

We employed a binary image classification methodology to distinguish between images of cats and dogs using a Convolutional Neural Network (CNN). The workflow consisted of data preprocessing, model design, training, and evaluation to ensure a robust and generalized model.

Data Preprocessing

All images were resized to a uniform dimension of 128×128 pixels and normalized to the range [0, 1] to standardize input for the neural network. The dataset was split using a 5-fold cross-validation strategy to ensure a comprehensive evaluation of the model. This approach divided the dataset into five folds, with each fold serving as a validation set once while the remaining four folds were used for training. This setup helped mitigate overfitting and assessed the model's performance across diverse splits.

Training Strategy

The model training included the use of early stopping with a patience of 5 epochs, which halted training when no significant improvement was observed in the validation loss. This prevented overfitting and ensured efficient use of computational resources. The Adam optimizer with a learning rate of 0.001 was used for optimization, and binary cross-entropy was chosen as the loss function to handle the binary classification task effectively. Model performance was evaluated using metrics such as accuracy, precision, and recall.

Model architectures

The CNN architecture was designed to extract and classify features from images through the following layers:
Input Layer: Accepts input images resized to 128×128×3 (height × width × RGB channels).

Convolutional Layers:
First Layer: 32 filters with 3×3 kernels, ReLU activation, and same padding. Max-pooling with a 2×2 pool size and a dropout layer (0.2) for regularization.
Second Layer: 64 filters same settings.
Third Layer: 128 filters same settings.

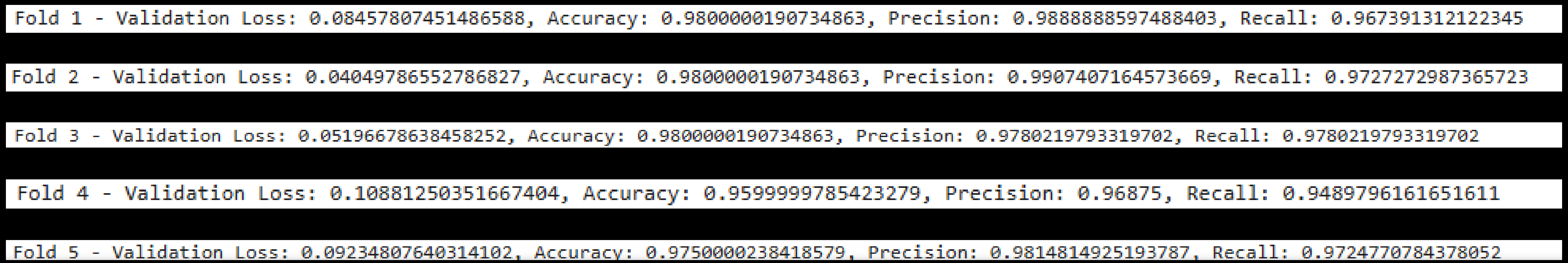
Fully Connected Layers:
Flatten Layer: Converts the feature maps into a one-dimensional vector.
Dense Layer: 512 neurons with ReLU activation, batch normalization for faster convergence and regularization, and dropout (0.5).

Output Layer: A single neuron with sigmoid activation to produce a binary classification output.

This architecture balances complexity and regularization, enabling effective learning from the dataset while preventing overfitting.

<CSC462 (ML)><CourseProject>

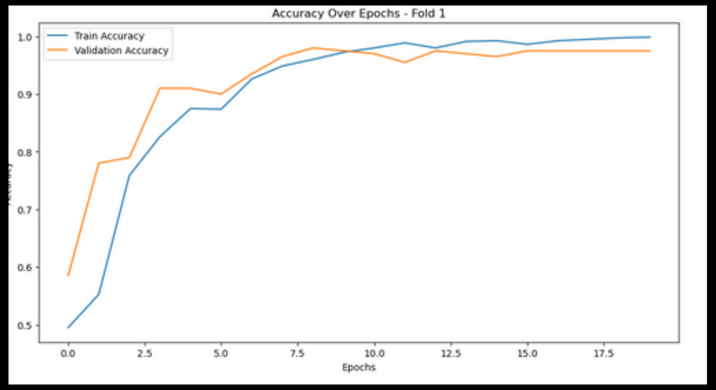
Experimental Results

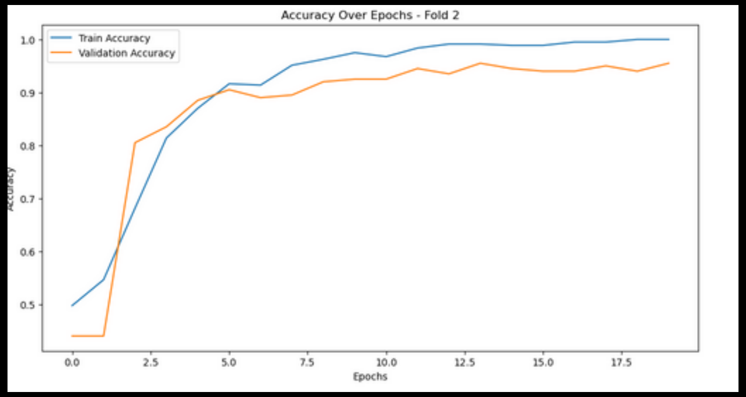


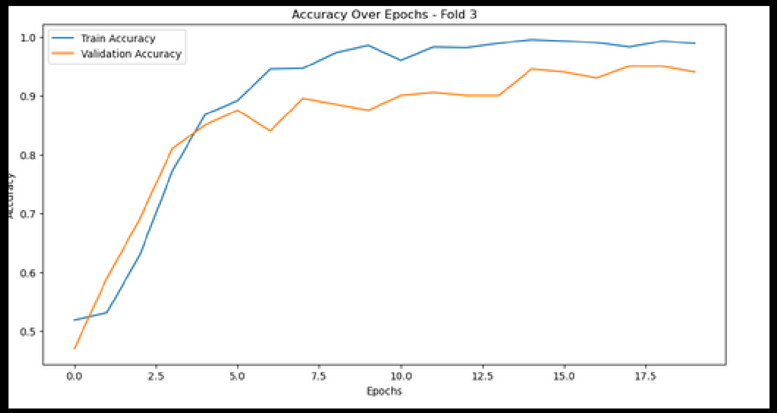
```
Fold 1 - Validation Loss: 0.08457807451486588, Accuracy: 0.9800000190734863, Precision: 0.9888888597488403, Recall: 0.967391312122345
Fold 2 - Validation Loss: 0.04049786552786827, Accuracy: 0.9800000190734863, Precision: 0.9907407164573669, Recall: 0.9727272987365723
Fold 3 - Validation Loss: 0.05196678638458252, Accuracy: 0.9800000190734863, Precision: 0.9780219793319702, Recall: 0.9780219793319702
Fold 4 - Validation Loss: 0.10881250351667404, Accuracy: 0.9599999785423279, Precision: 0.96875, Recall: 0.9489796161651611
Fold 5 - Validation Loss: 0.09234807640314102, Accuracy: 0.9750000238418579, Precision: 0.9814814925193787, Recall: 0.9724770784378052
```

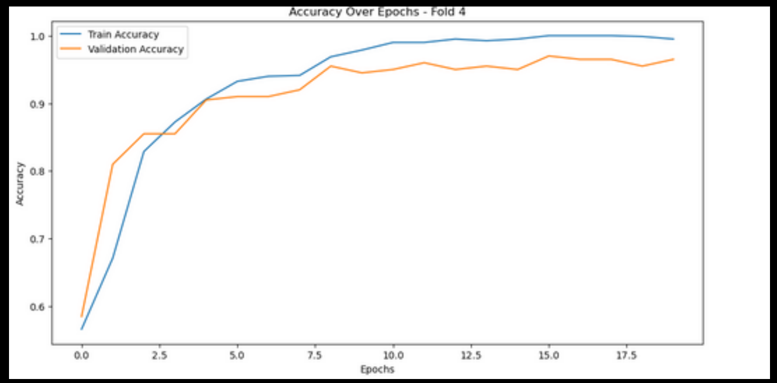
Our model
Acc=97%

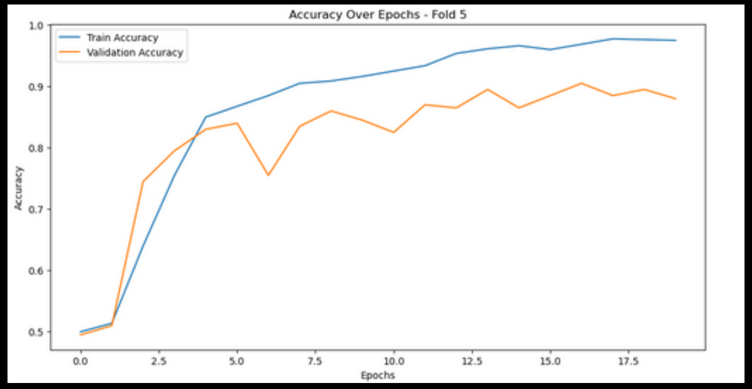
Baseline model
Acc=94%



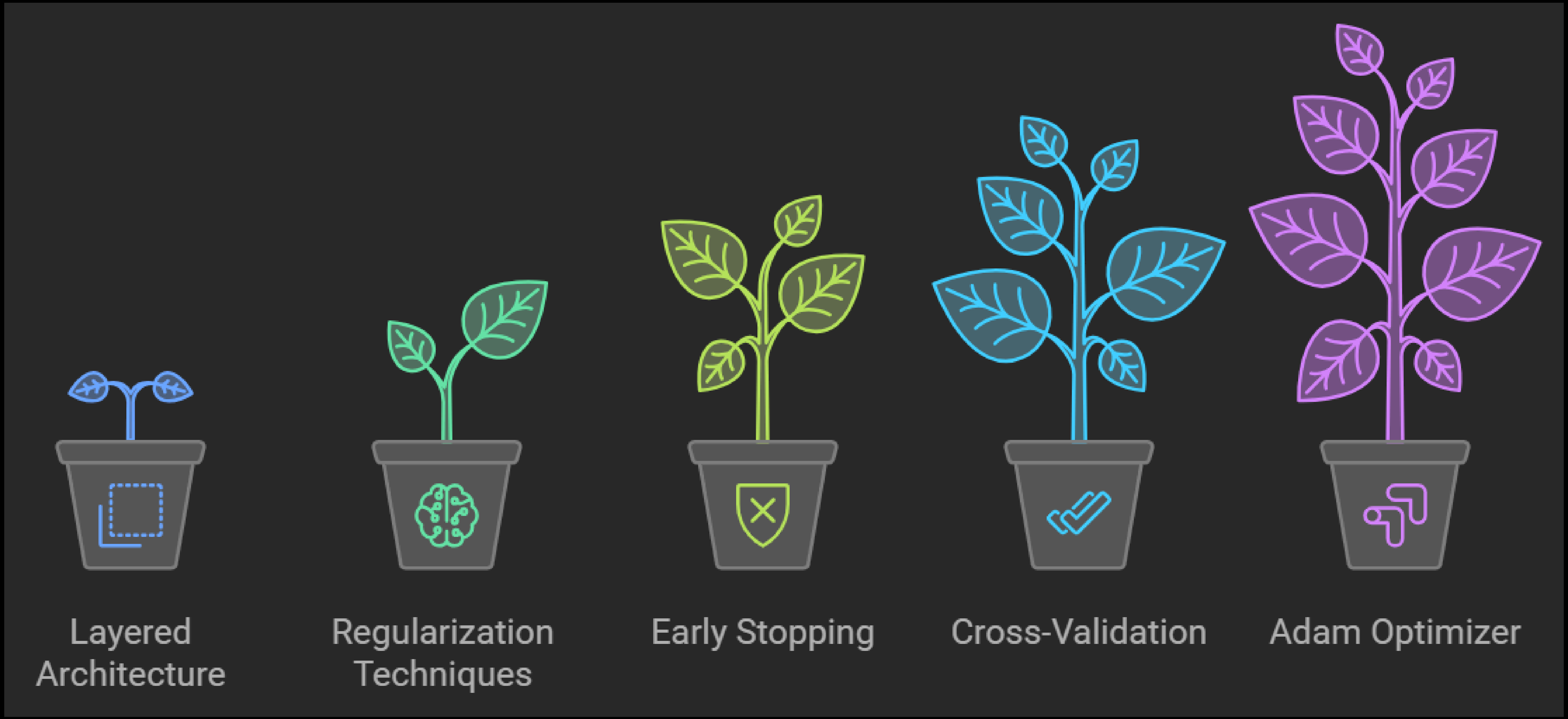








Strengths


Layered Architecture
Regularization Techniques
Early Stopping
Cross-Validation
Adam Optimizer

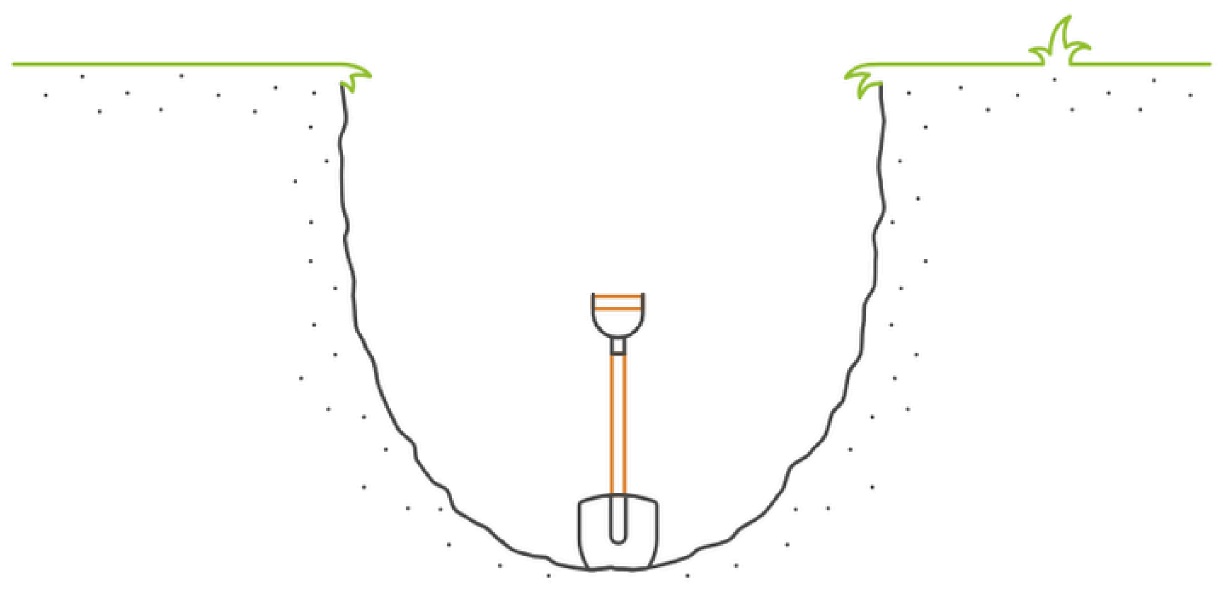
Limitations

A potential limitation of our CNN model lies in its complexity, which may not be well-suited to the relatively small size of our dataset, consisting of only 1,000 images. The model's intricate architecture, with multiple convolutional layers and a large number of units in the fully connected layer, increases the risk of overfitting, as it may learn noise rather than generalizable patterns due to the limited data.

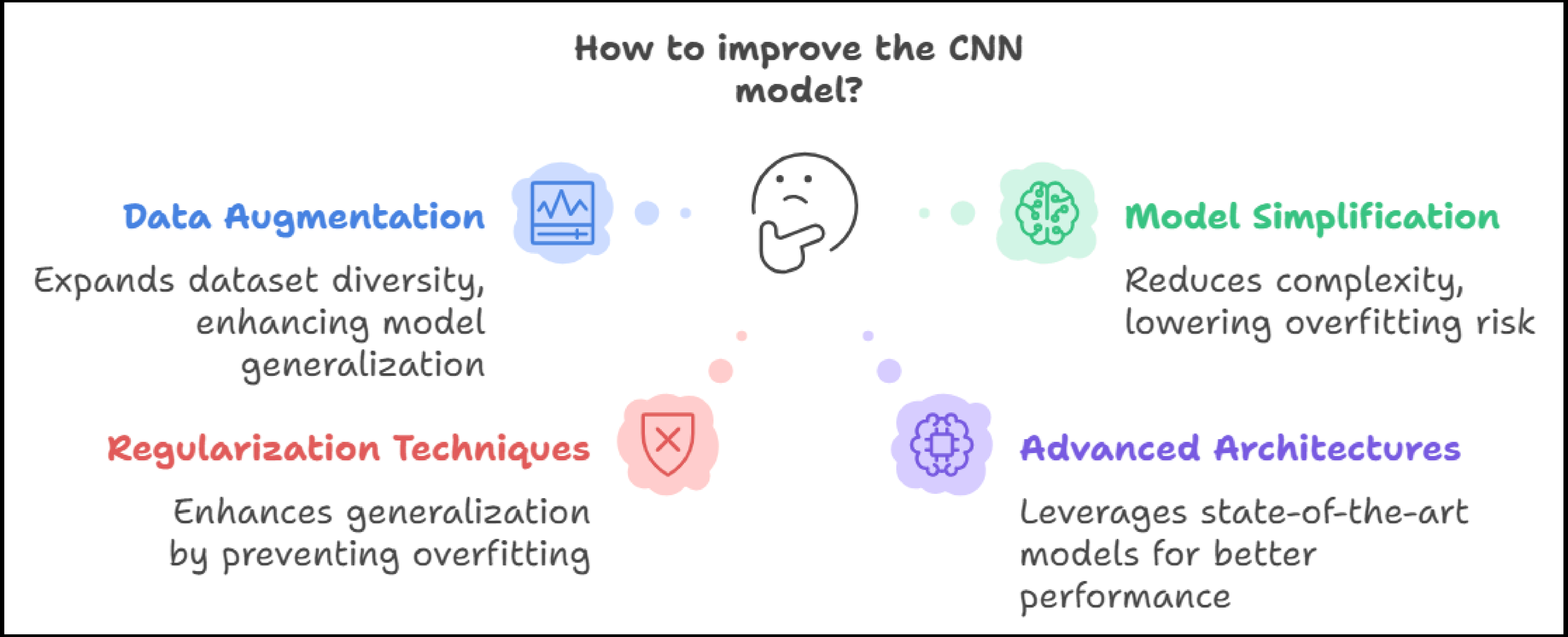
While the model achieved strong accuracy on both the training and test sets, there were instances where the validation accuracy was lower than the training accuracy during certain epochs, suggesting overfitting. However, the model still showed consistently good validation accuracy across all epochs, indicating reasonable generalization despite its complexity.

In conclusion, while the model's complexity posed a risk of overfitting due to the small dataset size, the regularization techniques helped mitigate this issue, allowing for good generalization.

Model complexity risks overfitting, affecting generalization.



Future improvements



How to improve the CNN model?

Data Augmentation

Expands dataset diversity, enhancing model generalization

Model Simplification

Reduces complexity, lowering overfitting risk

Regularization Techniques

Enhances generalization by preventing overfitting

Advanced Architectures

Leverages state-of-the-art models for better performance