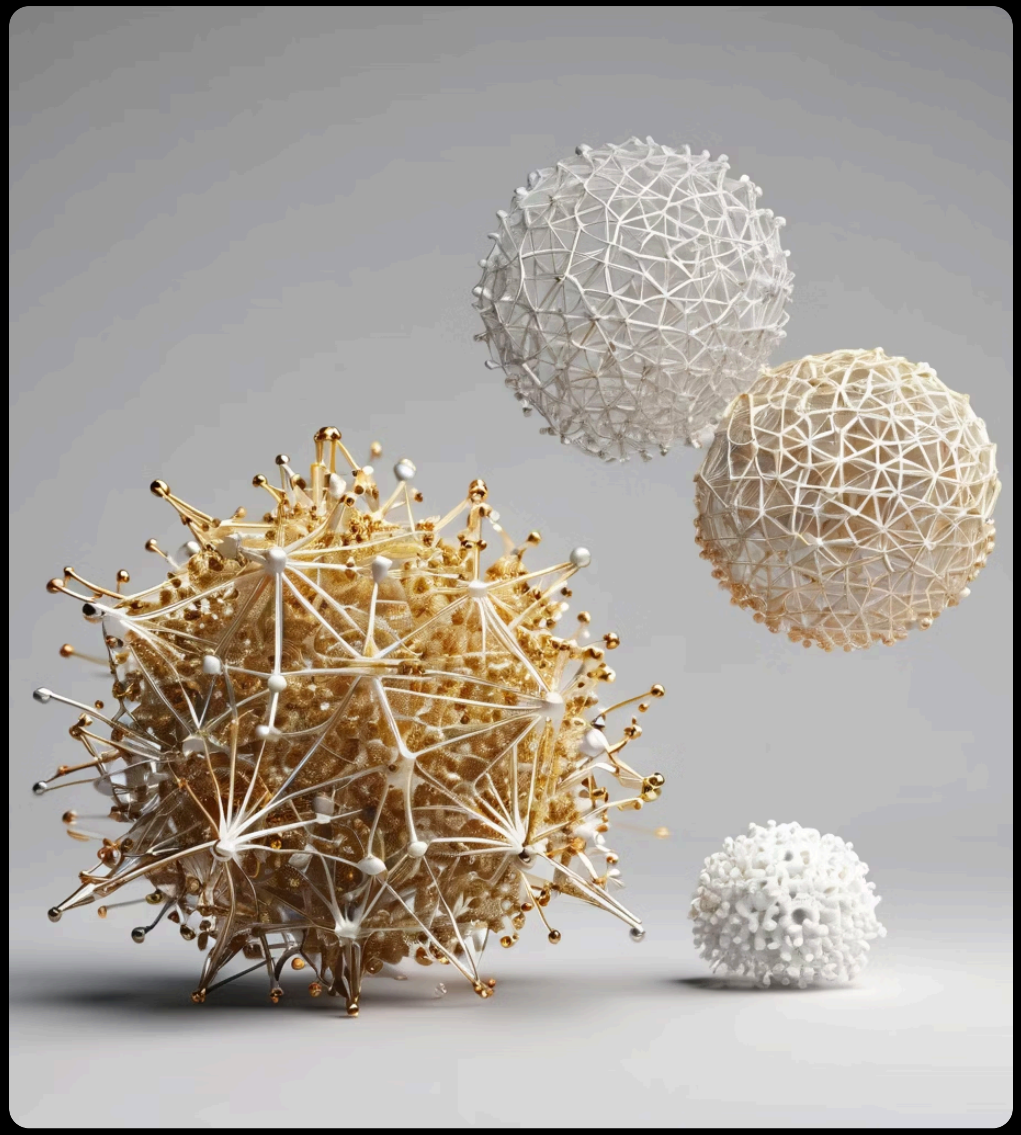
Comparative Analysis of Custom CNN and Pre-trained ResNet50 on CIFAR-10 Dataset



Group No 3

- Gregor Seegers
- Michalis Koskinas
- Jiota Belesi

“CNNs: The Ultimate Animal Spotters, Trained on Frogs and Trucks”

"Move over, we can say to Picasso. CNNs are here to master the fine art of recognizing your breakfast." 😁🤣

Dataset CIFAR-10

Balanced Class Distribution

The CIFAR-10 dataset is evenly distributed across its 10 classes, each containing 6,000 images.

Visual Diversity

Each class in the CIFAR-10 dataset showcases a wide range of visual examples, highlighting the variety within each category.

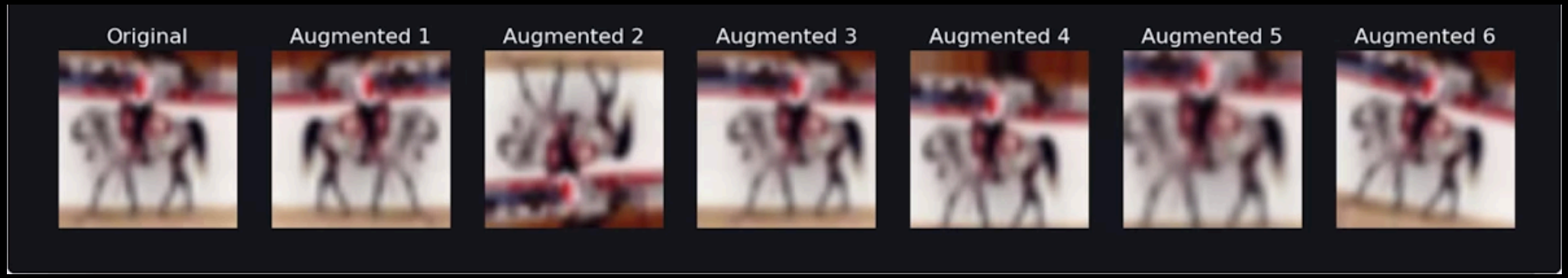
Overview of the problem

Challenges:

- Small image size limits detailed feature extraction
- Class diversity increases complexity
- Similar visual features among classes (e.g., cats and dogs) make differentiation harder

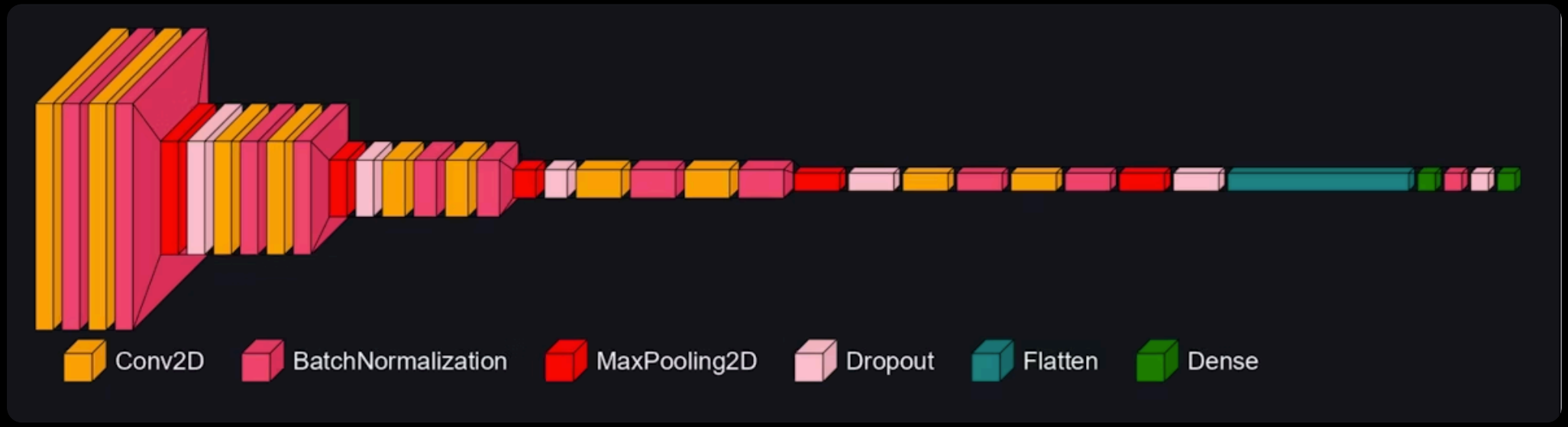
Data Preprocessing

- **One-Hot Encoding**
  - Training labels shape: (50000,) → labels shape: (50000, 10)
  - Test labels shape: (10000,) → labels shape: (10000, 10)
- **Normalization**
  - Image data value transformation: 0.0 (min) - 1.0 (max)
- **Resizing the 32x32** CIFAR-10 **Images**
  - 64x64 custom CNN
  - 96x96 pre-trained ResNet50
- **Data Augmentation**



Designed CNN Architecture

- 5 blocks
- Convolutional layers (Conv2D): 10
  - Batch Normalization
- MaxPooling layers: 5
- DropOut (0.2→0.5)
- Dense (fully connected) layers: 2



Optimisation Techniques

Adam Optimizer

Learning Rate Scheduling

- exponential decay schedule
  - Initial learning rate: 0.001 → ≈ 0.000349
  - decay steps: 10.000
  - decay rate: 0.9

Loss function

- cross entropy loss function

Early Stopping

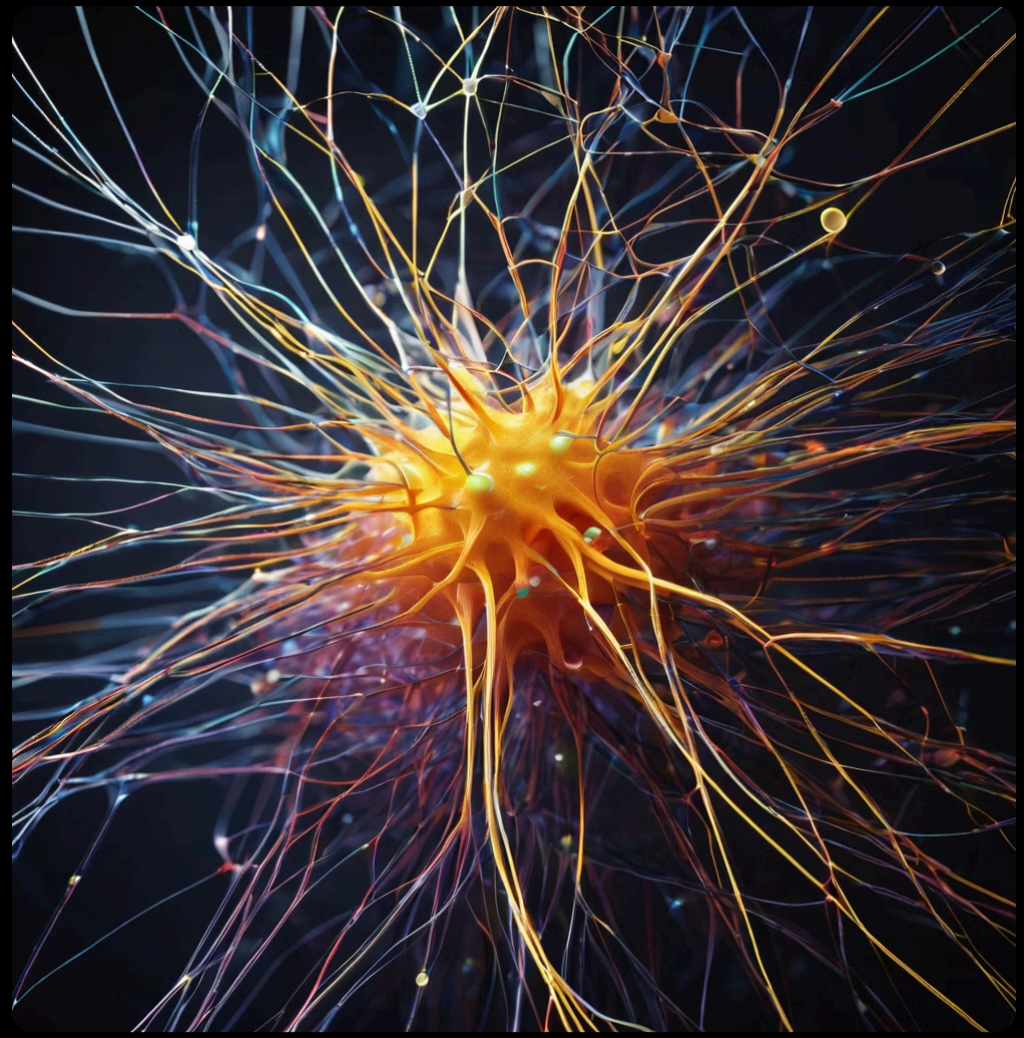
- patience=10

Model saving

- ModelCheckpoint callback

Grid Search

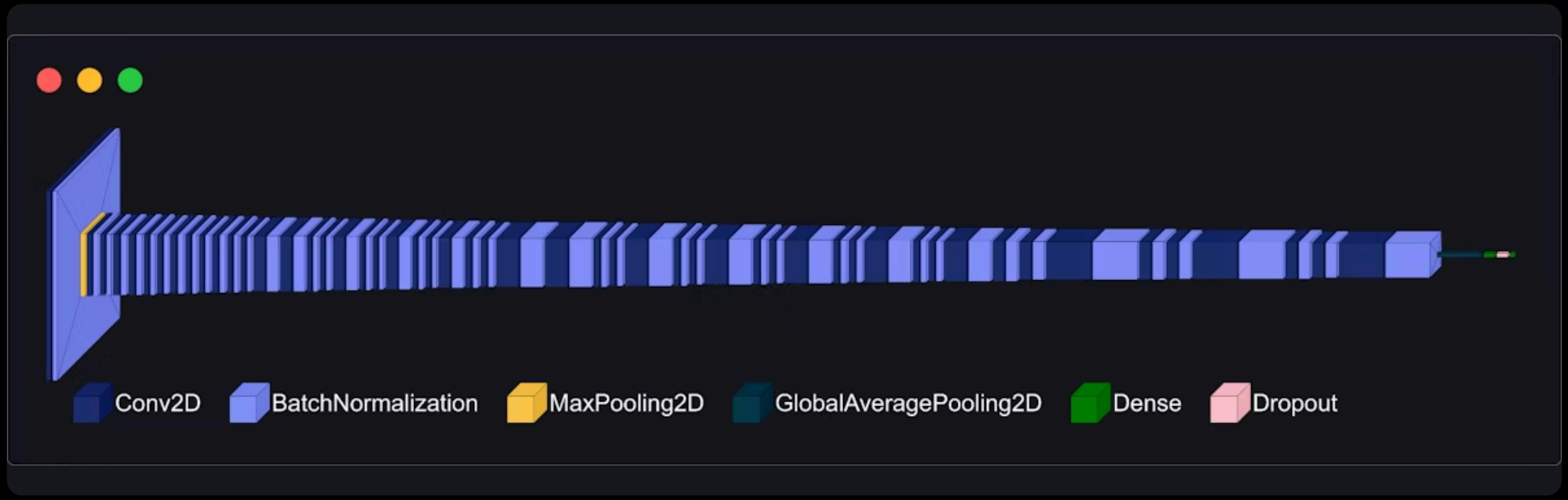
- parameters e.g. dropout rate, batch size, no. of epochs



Pre-trained ResNet50 (ImageNet)

Custom layers added:

GlobalAveragePooling | Dense (512 units, ReLU) | Dropout (0.3) | Final Dense layer (Softmax)



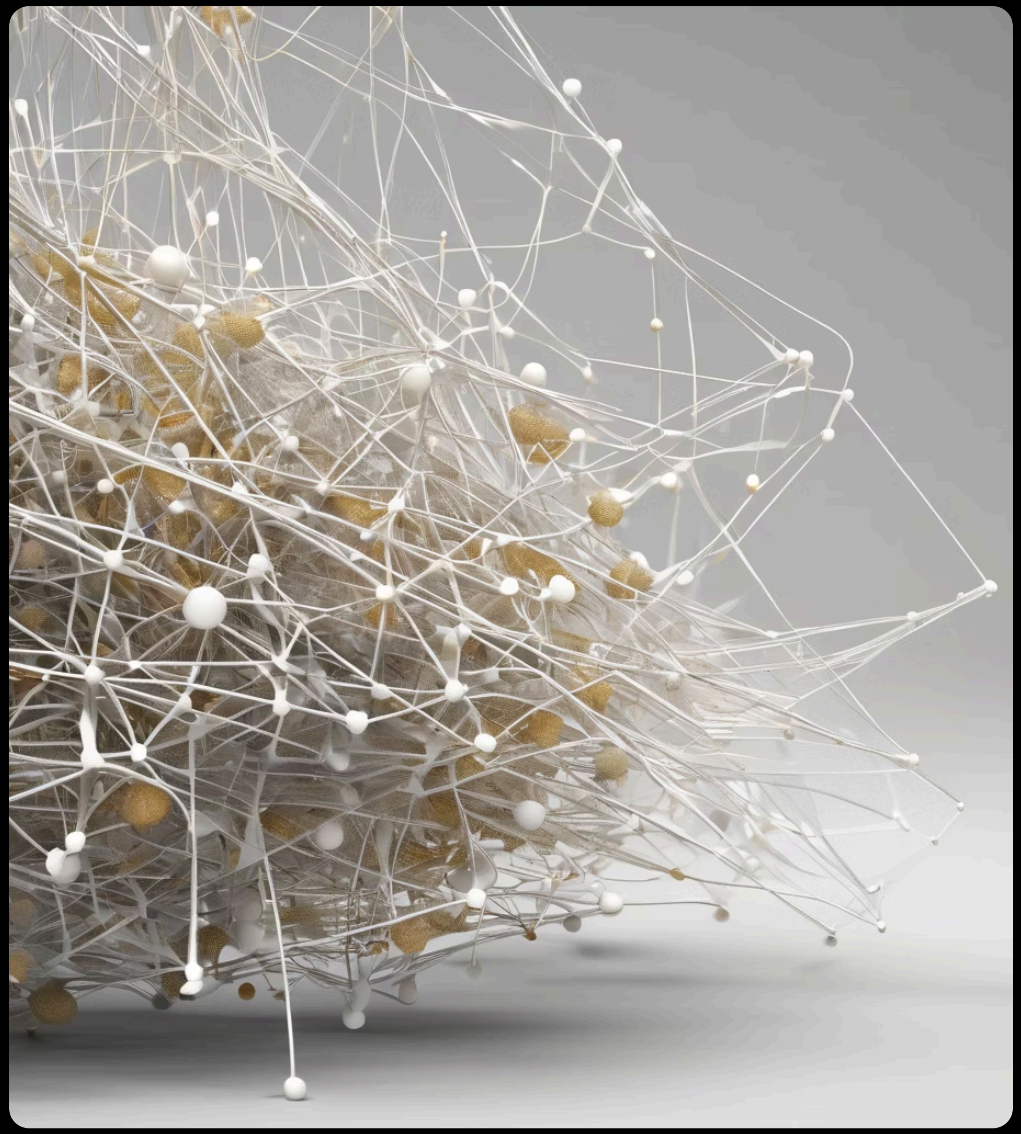
ResNet50 Training

Process Overview

- Conducted in three cycles.
- Saved weights used to continue training.

Optimizer & Hyperparameters

- Adam optimizer, learning rate adjustments to fine-tune.
- Learning rate / unfrozen layers:
  - Cycle 1: 0.0001
  - Cycle 2: 0.00005 (15 unfrozen layers)
  - Cycle 3: 0.00001 (30 unfrozen layers).



Evaluation Metrics for Custom CNN & ResNet50

✅ Accuracy

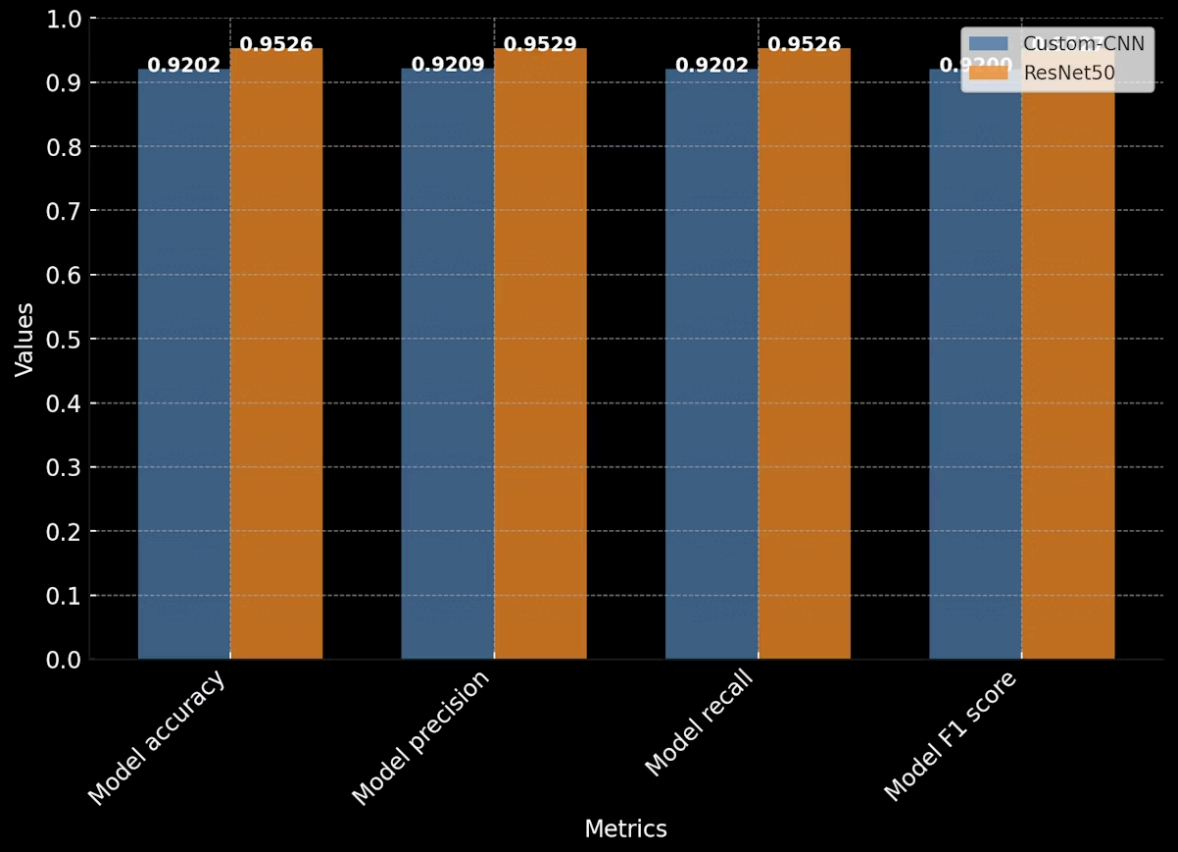
- ResNet50 achieved higher accuracy than the custom CNN, demonstrating better generalization. The custom CNN had slightly lower accuracy.

⚖️ Precision

- The custom CNN had lower precision, resulting in more false positives compared to ResNet50, which minimized false positives more effectively.

➡️ Recall

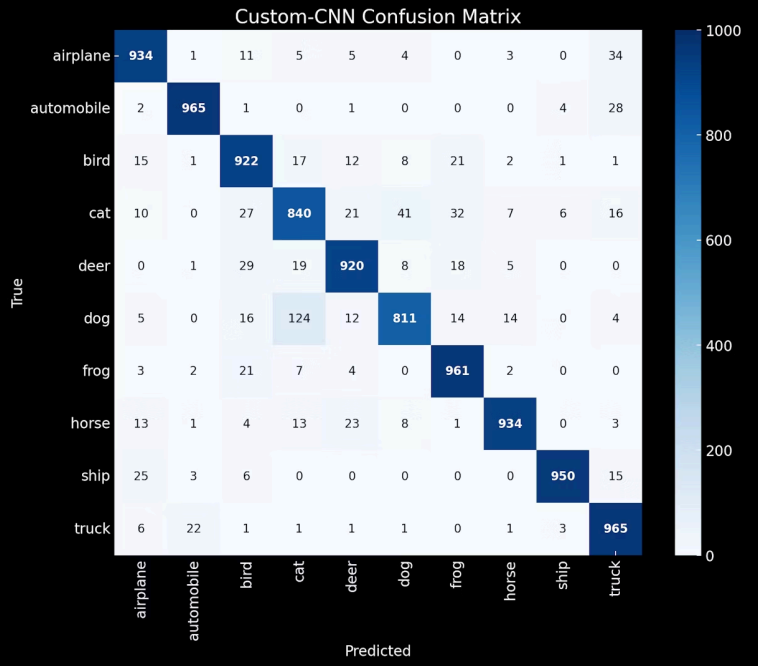
- ResNet50 had stronger recall, capturing more true positives compared to the custom CNN, which missed some instances.

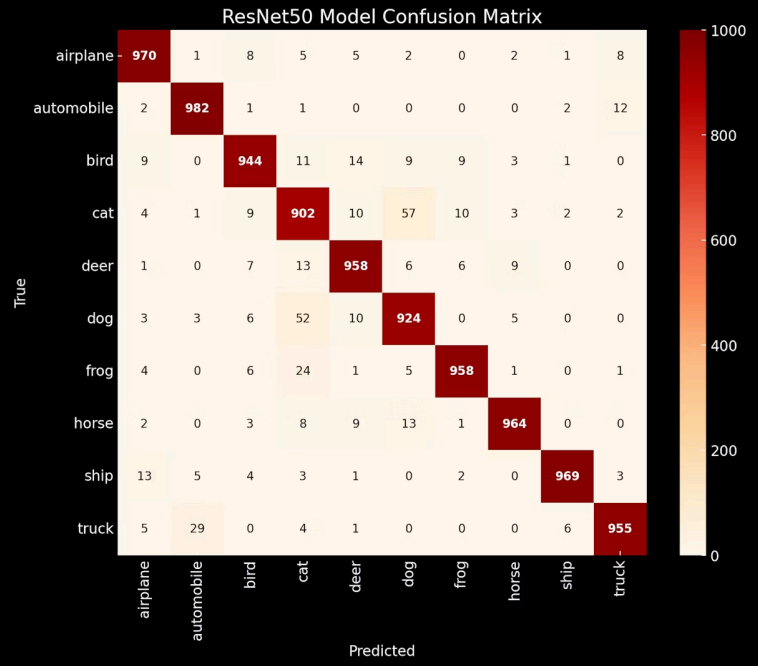


Confusion Matrix - Custom CNN vs ResNet50

The confusion matrices show areas where both models struggled, particularly with visually similar classes like 🐈 and 🐕.

The Custom CNN had more misclassifications, while ResNet50 showed better overall performance with fewer mistakes.





Summary of Evaluation Results

Advantages of Custom CNN:

- Faster training times due to a simpler architecture.
- Greater ease of modification, making it suitable for straightforward scenarios.

Advantages of ResNet50:

- Leveraged transfer learning, resulting in superior performance.
- Deeper architecture enabled better feature extraction.
- Pre-trained on ImageNet, providing a strong foundation for classification tasks.

ResNet50's superior accuracy and robustness stem from pre-training and a deeper architecture.

Custom CNN offered *f*aster training and ease of modification, making it useful for simpler scenarios.

Project Recap

Key Learnings:

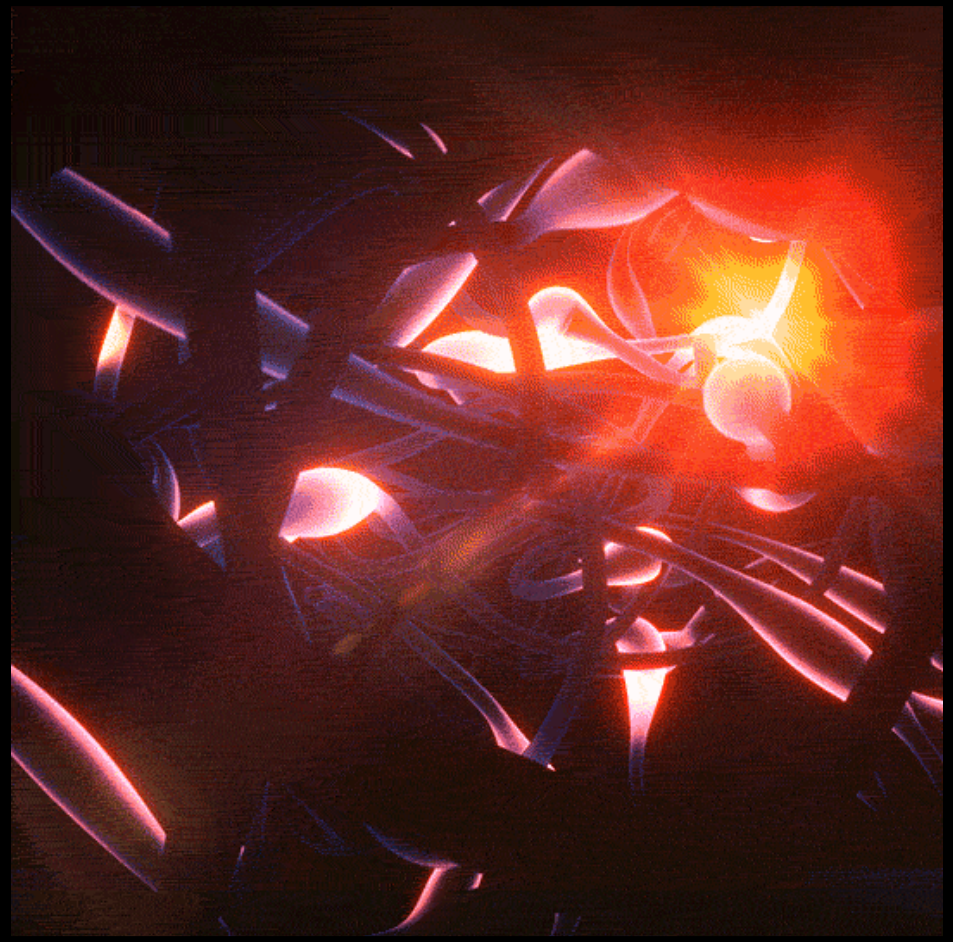
- Custom CNN Flexibility: Offers design control but needs careful tuning.
- ResNet50 Transfer Learning: Boosted accuracy, saved training time.
- Data Augmentation: Improved generalization and reduced overfitting.

Strategies for Success:

- Gradual learning rate schedules and regular validation monitoring.
- Unfreezing layers and using multiple training cycles for improved performance.

Future Directions:

- Explore other advanced architectures like EfficientNet.
- Investigating ensemble methods for even better accuracy.



Thank you for listening! 🙃