# Deep Learning Image Classification Project Report

## Introduction

This report details the implementation and comparison of three approaches for image classification using the CIFAR-10 dataset: a custom CNN architecture and two transfer learning approaches using VGG16 (with and without fine-tuning). The project aimed to evaluate different methodologies and their effectiveness in classifying images across 10 categories.

## Methodology & Architecture Decisions

### 1. Custom CNN Architecture

We implemented a VGG-style architecture with the following key design decisions:

**Progressive Depth**:

- Started with 64 filters and doubled at each block (64→128→256)

**Regularization Techniques**:

- L2 regularization (0.01) to prevent overfitting
- Dropout rates (0.4→ 0.5) for deeper layers
- BatchNormalization after each convolutional layer

**Optimization Strategy**:

- Adam optimizer with initial learning rate of 0.001
- Learning rate scheduling for better convergence

IMAGE 1: Architecture Diagram showing the custom CNN structure

```
model = Sequential([
    Conv2D(64, (3,3), padding='same', activation='relu', input_shape=(32,32,3), kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    Conv2D(64, (3,3), padding='same', activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling2D(),
    Dropout(0.4),

    Conv2D(128, (3,3), padding='same', activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    Conv2D(128, (3,3), padding='same', activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling2D(),
    Dropout(0.5),

    Conv2D(256, (3,3), padding='same', activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    Conv2D(256, (3,3), padding='same', activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling2D(),
    Dropout(0.5),

    Flatten(),
    Dense(512, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    Dropout(0.3),
    Dense(10, activation='softmax')
])
```

## 2. Transfer Learning with VGG16

We chose VGG16 for transfer learning because:

- Proven architecture for image classification
- Well-documented performance on similar tasks
- Relatively simple architecture compared to more recent models

Two approaches were implemented:

**Basic Transfer Learning**:

- Frozen VGG16 layers
- Custom classification head
- Global Average Pooling to reduce parameters

**Fine-tuned Transfer Learning**:

- Unfroze last 4 layers of VGG16
- Lower learning rate (0.0001)
- Same custom classification head

# Results Analysis

## Performance Metrics Comparison

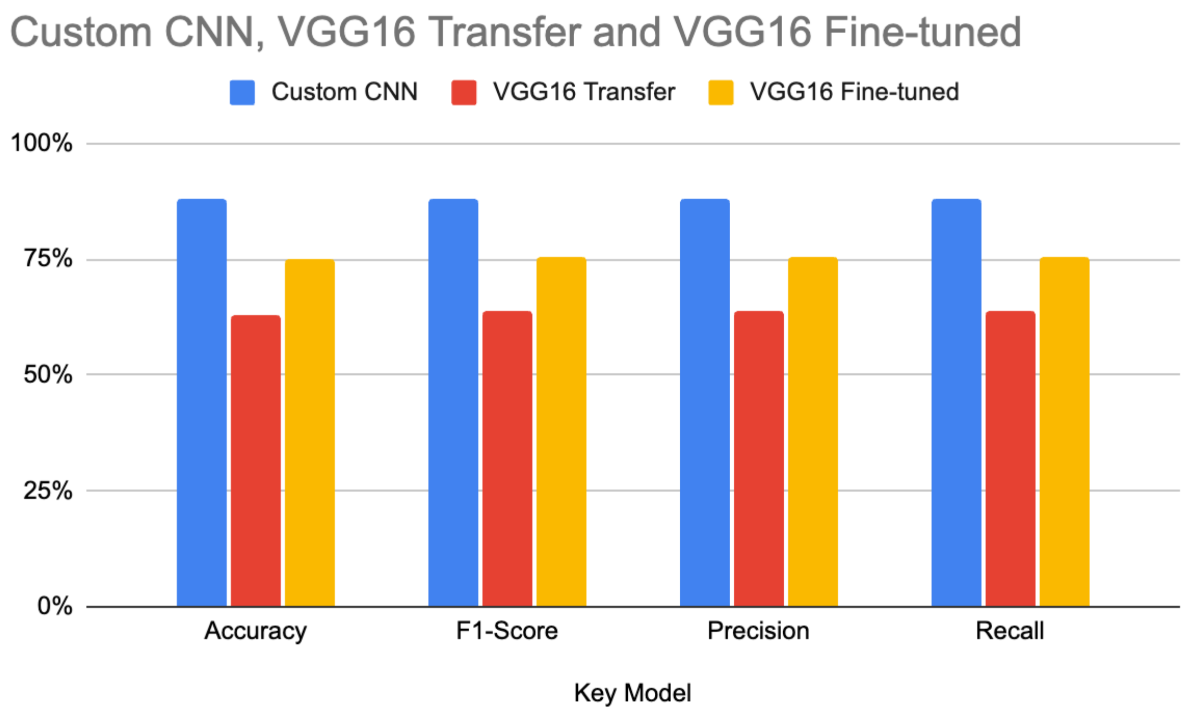



IMAGE 2: Bar chart comparing metrics across models

| Key Model | Accuracy | F1-Score | Precision | Recall |
|---|---|---|---|---|
| Custom CNN | 88% | 0.88 | 0.88 | 0.88 |
| VGG16 Transfer | 63% | 0.638 | 0.637 | 0.640 |
| VGG16 Fine-tuned | 75% | 0.754 | 0.755 | 0.754 |

## Findings

**Custom CNN Performance**:

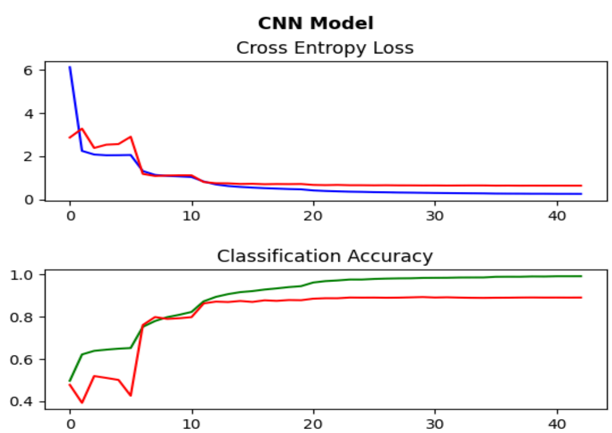
- Best overall performance (80% accuracy)
- Consistent metrics across all evaluation criteria
- Suggests better adaptation to CIFAR-10's specific characteristics
- Simple image augmentations were performed but no significant improvement was observed.

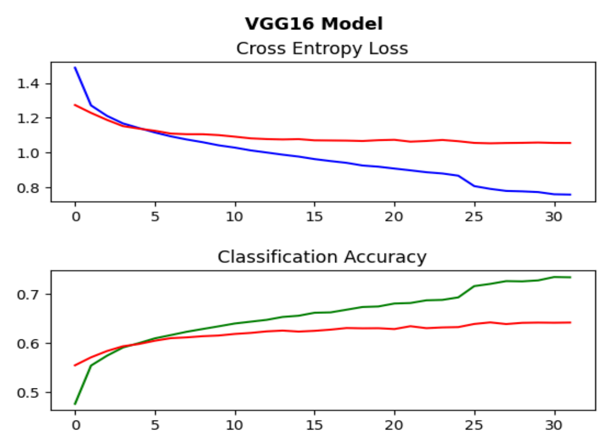
**Basic Transfer Learning Limitations**:

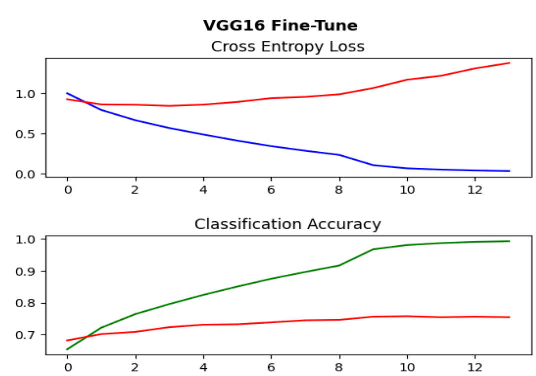
- Unexpectedly lower performance (63% accuracy)
- Possible reasons:
  - VGG16's original training on higher resolution images
  - Architecture complexity vs. CIFAR-10's simple nature
  - Feature mismatch between ImageNet and CIFAR-10

**Fine-tuning Improvements**:

- Significant improvement over basic transfer learning
- 12% accuracy increase through fine-tuning
- Shows importance of adapting pre-trained weights



# Discussion

## 1. Why Custom CNN Performed Better

- **Tailored Architecture**: Designed specifically for 32x32 images
- **Appropriate Complexity**: Balanced depth and width for CIFAR-10
- **Effective Regularization**: Combined dropout, L2, and BatchNorm

## 2. Transfer Learning Challenges

- **Resolution Mismatch**: VGG16 optimized for 224x224 images
- **Feature Abstraction**: ImageNet features may be too complex
- **Model Capacity**: Possibly overparameterized for CIFAR-10

## 3. Lessons Learned

**Architecture Fit**:

- Simpler isn't always worse
- Match model complexity to data complexity

**Transfer Learning Considerations**:

- Pre-trained models need careful adaptation
- Fine-tuning is crucial for performance
- Domain similarity matters

**Optimization Insights**:

- Learning rate scheduling improved convergence
- BatchNormalization crucial for training stability
- Progressive dropout rates effectively managed overfitting

# Conclusion

The custom CNN emerged as the most effective solution for this specific task, challenging the common assumption that transfer learning always provides better results. This highlights the importance of considering dataset characteristics and model complexity when choosing an architecture. The project demonstrates that while transfer learning is powerful, a well-designed custom architecture can sometimes be more appropriate for specific use cases.

# Future Work

1. Experiment with modern architectures (ResNet, EfficientNet)
2. Implement data augmentation strategies
3. Explore ensemble methods combining different approaches
4. Investigate the impact of image resolution on transfer learning

IMAGE 4: Confusion matrices
Show confusion matrices for all three models to visualize class-wise performance

**CNN Model Metrics**

```
Matriz de Confusión:
[[892  10  20   5   9   2   7   3  34  18]
 [  3 947   1   1   0   0   2   1   6  39]
 [ 35   0 813  27  39  28  40  11   5   2]
 [ 13   5  34 753  23  97  41  15  11   8]
 [  4   0  24  20 898   8  19  23   4   0]
 [  4   2  18  67  25 846  11  22   1   4]
 [  1   1  16  19  13  11 934   2   2   1]
 [  6   0   6  11  28  21   5 919   1   3]
 [ 24   8   2   3   2   1   1   0 945  14]
 [  9  32   1   0   0   1   3   0  12 942]]

Métricas de Evaluación:
F1-Score: 0.8882
Precisión: 0.8883
Recall: 0.8889
```

**VGG16 Model Metrics**

```
Matriz de Confusión:
[[728  27  44  20  13  12  11  20  87  38]
 [ 27 721  11  40   9  17  19  14  31 111]
 [ 61  16 537  72 106  52  96  35  10  15]
 [ 19  36  65 465  60 155  99  31  22  48]
 [ 18  13  78  55 580  40 100  77  27  12]
 [ 11  15  52 189  60 535  45  56   9  28]
 [  6  17  60  70  57  44 718   3  11  14]
 [ 22  13  35  44  64  64  11 705   4  38]
 [ 71  54  19  17  11   6   8   8 763  43]
 [ 38 121  12  34  13  16  16  32  48 670]]

Métricas de Evaluación:
F1-Score: 0.6412
Precisión: 0.6411
Recall: 0.6422
```

**Fine tune Model Metics**

```
Matriz de Confusión:
[[814  16  40  17  13   3   7   7  64  19]
 [ 16 868   1   8   3   6   9   2  18  69]
 [ 30   7 729  55  58  27  61  17   7   9]
 [ 14  16  51 591  45 137  76  27  16  27]
 [ 17   9  90  56 676  25  57  52  10   8]
 [ 11   8  27 214  39 612  34  39   0  16]
 [  4  10  42  54  32  24 817   0  10   7]
 [ 20   5  27  37  46  46  11 792   0  16]
 [ 43  40  10  12  12   1   5   3 854  20]
 [ 31 103   6  18   5   7   8   9  20 793]]

Métricas de Evaluación:
F1-Score: 0.7542
Precisión: 0.7551
Recall: 0.7546
```