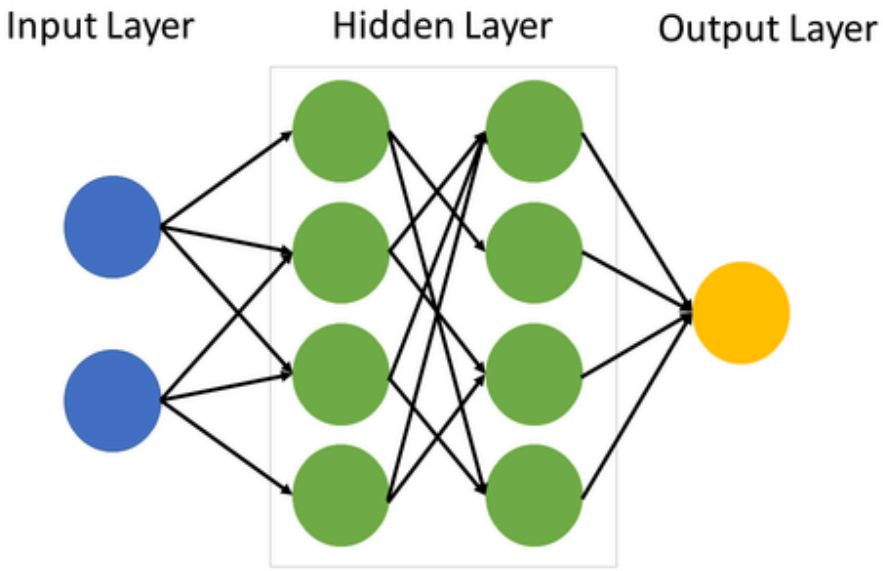
Deep Learning

Project Report : Deep Learning-Image Classification with CNN

Authors: Adeteju Enunwa, Marvin Mends , Priscila Coghlan



Artificial Neural Networks

Introduction

The field of deep learning has revolutionized image classification, allowing models to achieve human-like accuracy on complex tasks. In this project, we focus on building and analyzing Convolutional Neural Networks (CNNs) using the **CIFAR-10** dataset, with the aim of classifying images into one of 10 categories. Additionally, we explore **transfer learning** using a **VGG16** model pre-trained on the **ImageNet** dataset to improve classification performance.

Our primary motivation stems from a real-world use case involving **smart glasses** for drivers, designed to help them navigate their environment through real-time object detection. However, this model can be adapted to many other fields, such as **CAPTCHA generation**, **autonomous vehicles**, and **automated content moderation.**

This report provides a detailed analysis of the architecture, training process, model evaluation, and the insights gained from experimentation, while also discussing potential future improvements.

Dataset Choice and Preprocessing

Dataset Selection: CIFAR-10

We initially considered the **Animals-10 dataset** from Kaggle, which contains images solely of animals. However, the CIFAR-10 dataset, with its **diverse object categories**, balanced classes, and wide applicability to real-world scenarios, was more appropriate for our use case. CIFAR-10 includes images of **vehicles, animals, and objects**, making it more representative of the types of environments that smart glasses or autonomous vehicles would encounter.

Key Dataset Characteristics:

- **10 classes**: Airplane, Automobile, Bird, Cat, Deer, Dog, Frog, Horse, Ship, Truck.
- **60,000 images**: 50,000 for training and 10,000 for testing, each image measuring 32x32 pixels.

Preprocessing Techniques

Normalization:
We normalized the images by scaling the pixel values to the range [0, 1]. This is critical to ensure numerical stability and faster convergence during training by reducing the risk of vanishing or exploding gradients.
python

Data Augmentation:
Since CIFAR-10 images are small and may not fully represent all possible variations of objects (e.g., different orientations, lighting, or distortions), we applied several data augmentations to create variations that the model could learn from. These included:

- Random horizontal flips
- Random rotations (up to 15 degrees)
- Random width and height shifts (up to 10%)
- Zoom range of 10%

These augmentations were crucial for improving model generalization, particularly in real-world applications like autonomous driving, where objects like cars, pedestrians, and traffic signs may appear under various conditions.

One - Hot Encoding

Although CIFAR-10 is balanced, one-hot encoding ensured that any slight imbalances in augmented data did not bias the model towards certain classes. We applied one-hot encoding to transform our integer class labels into binary vector representations.

Architectural Design

We implemented three different model architectures:

1. **Custom CNN:**
    - Three convolutional blocks, each containing:
        - Two Conv2D layers with increasing filter sizes (32, 64, 128)
        - Batch normalization after each Conv2D layer
        - Max pooling
        - Dropout (rates increasing from 0.2 to 0.4)
    - Two dense layers with batch normalization and dropout
    - Final dense layer with softmax activation
2. **VGG16 Transfer Learning:**
    - Pre-trained VGG16 base (weights from ImageNet, top layers removed)
    - Custom top layers:
        - Global Average Pooling
        - Dense layer (256 units) with ReLU activation and Batch Normalization
        - Dropout (0.5)
        - Dense layer (128 units) with ReLU activation and Batch Normalization
        - Dropout (0.5)
        - Output dense layer (10 units) with softmax activation
3. **Fine-tuned VGG16:**
    - Based on the transfer learning model
    - Last 20 layers of VGG16 unfrozen for fine-tuning

Training Strategy

For all models, we employed the following training strategy:

- Optimizer: Adam (learning rates adjusted during training)
- Loss function: Categorical cross entropy
- Metrics: Accuracy
- Batch size: 32 (custom CNN), 64 (VGG16 models)
- Callbacks:
    - ReduceLROnPlateau: To adapt the learning rate based on validation loss
    - EarlyStopping: To prevent overfitting and unnecessary computation
    - ModelCheckpoint: To save the best model based on validation loss

CNN Model Architecture

The architecture includes **three convolutional blocks**, each consisting of two Conv2D layers followed by **Batch Normalization** and **Dropout**. A key addition here is **L2 regularization (weight decay)** in each convolutional layer to prevent overfitting by penalizing large weights. This helps the model generalize better, especially with small datasets like CIFAR-10.
Each block is designed as follows:

- **First block**: Two convolutional layers with 32 filters, a 3x3 kernel, and padding set to 'same'. This captures low-level features like edges and simple shapes.
- **Second block**: Two convolutional layers with 64 filters. As the model deepens, it captures more complex patterns like textures and part-based object representations.
- **Third block**: Two convolutional layers with 128 filters. The final block captures the most complex features, such as entire object parts or high-level abstractions.

After each block, a **MaxPooling2D** layer reduces the spatial dimensions of the feature maps, lowering computational complexity while preserving important features.

Batch Normalization:
Batch Normalization is applied after each convolutional layer. It normalizes the activations, speeding up convergence during training and reducing the sensitivity of the network to changes in the initial weights. This layer plays a key role in stabilizing the learning process and preventing overfitting.

Dropout for Regularization:
Dropout is applied after every convolutional block and dense layer to prevent overfitting. We progressively increased the dropout rate through the network (from 0.2 to 0.5) as the depth increases. This ensures that the model learns generalizable features, instead of relying on specific neurons to memorize the training data.

Dense Layers:
After flattening the feature maps, a **fully connected (dense) layer** with 128 units is applied, followed by Batch Normalization and Dropout. This layer learns the global patterns that are essential for classification. The final dense layer uses a **softmax activation** to output the class probabilities for each of the 10 CIFAR-10 categories.

Optimizer and Compilation:
We used the **Adam optimizer** with a learning rate of 1e-3. Adam is well-suited for this task because it adaptively adjusts the learning rate for each parameter, allowing for faster convergence. The model was compiled with **categorical cross-entropy** as the loss function, appropriate for multi-class classification tasks like CIFAR-10.

Callbacks

To ensure efficient training and prevent overfitting, we incorporated two key callbacks:

- **ReduceLROnPlateau**:
  This callback reduces the learning rate by a factor of 0.5 when the validation loss plateaus, allowing the model to make finer updates to the weights during later epochs, improving overall convergence.
- **Early Stopping**:
  Early stopping monitors the validation loss and halts training if no improvement is seen after 20 epochs. This prevents the model from overfitting by stopping at the optimal point in the training process.

Transfer Learning with VGG16

Why VGG16?

We selected VGG16 as our base model for transfer learning due to several key factors:

1. Proven Performance: VGG16 has demonstrated excellent performance on various image classification tasks, particularly on the ImageNet dataset. Its success in capturing complex image features made it a strong candidate for transfer learning.
2. Architecture Simplicity: Despite its depth, VGG16 has a straightforward architecture consisting of repeated 3x3 convolutional layers followed by max pooling. This simplicity makes it easier to understand, modify, and fine-tune compared to more complex architectures.
3. Feature Hierarchy: VGG16's deep structure allows it to learn a rich hierarchy of features, from low-level edges and textures in earlier layers to high-level semantic concepts in later layers. This hierarchical representation is valuable for transfer learning, especially when dealing with a diverse dataset like CIFAR-10.
4. Transferability of Features: Previous research has shown that features learned by VGG16 on ImageNet are highly transferable to other image classification tasks, even when the target dataset is quite different from ImageNet.
5. Compatibility with Small Images: Although VGG16 was originally trained on 224x224 images, its convolutional nature makes it adaptable to smaller input sizes like CIFAR-10's 32x32 images with relatively minor modifications.
6. Balance of Model Size and Performance: While VGG16 is larger than some other models (e.g., MobileNet), it offers a good balance between model capacity and computational requirements, making it suitable for our available resources.
7. Extensive Pre-training: VGG16 pre-trained on ImageNet has learned from over a million diverse images, providing a rich set of general-purpose features that can be leveraged for CIFAR-10 classification.
8. Smooth Gradient Flow: The use of small 3x3 filters throughout the network promotes smoother gradient flow during backpropagation, which can lead to more stable fine-tuning.
9. Interpretability: The uniform structure of VGG16 can make it easier to interpret and visualize learned features, which is valuable for understanding how the model is adapting to our specific task.
10. Community Support and Resources: VGG16 is widely used in the deep learning community, which means there are abundant resources, pre-trained weights, and best practices available for its application in transfer learning scenarios.

By choosing VGG16, we aimed to leverage its powerful feature extraction capabilities while balancing the challenges posed by the smaller image size and different nature of the CIFAR-10 dataset. This choice allowed us to explore the effectiveness of transfer learning from a large-scale, general dataset (ImageNet) to a smaller, more specific dataset (CIFAR-10) in the context of image classification.

Transfer Learning Architecture

- Base model: Pre-trained VGG16 (weights from ImageNet, top layers removed)
- Input shape adjusted to (32, 32, 3) to match CIFAR-10 image dimensions
- Custom top layers:
    - Global Average Pooling
    - Dense layer (256 units) with ReLU activation and Batch Normalization
    - Dropout (0.5)
    - Dense layer (128 units) with ReLU activation and Batch Normalization
    - Dropout (0.5)
    - Output dense layer (10 units) with softmax activation
- Initial training: Base VGG16 layers frozen, only top layers trained
- Fine-tuning: Last 20 layers of VGG16 unfrozen for further training

Transfer Learning Approach

Our transfer learning approach using VGG16 was implemented as follows:

1. **Base Model Selection**: We chose VGG16 pre-trained on ImageNet due to its proven performance on image classification tasks and its relatively simple architecture, which makes it suitable for transfer learning on smaller datasets.
2. **Model Adaptation:**
    - Removed the top (fully connected) layers of VGG16
    - Added custom top layers to adapt the model for CIFAR-10 classification
    - Adjusted the input shape to (32, 32, 3) to match CIFAR-10 images, which required modifying the first layer of VGG16
3. **Two-Stage Training: Stage 1 - Feature Extraction**:
    - Froze all layers of the base VGG16 model
    - Trained only the custom top layers
    - This allowed the model to learn CIFAR-10 specific classifications using the general features extracted by VGG16
4. **Stage 2 - Fine-Tuning:**
    - Unfroze the last 20 layers of the VGG16 base
    - Trained the entire model with a very low learning rate (1e-5)
    - This allowed for fine-tuning of the higher-level features in VGG16 to better suit CIFAR-10
5. **Handling Resolution Mismatch:**
    - VGG16 was originally trained on 224x224 images, while CIFAR-10 uses 32x32 images
    - We addressed this by: a) Modifying the first layer of VGG16 to accept 32x32 inputs b) Adjusting the subsequent layers' parameters to maintain the network's structure
    - This adaptation allowed us to leverage VGG16's learned features while working with the smaller CIFAR-10 images
6. **Regularization and Optimization:**
    - Applied dropout in the custom top layers to prevent overfitting
    - Used batch normalization to stabilize learning
    - Implemented a learning rate reduction strategy to fine-tune the model effectively
7. **Data Augmentation:**
    - Employed more aggressive data augmentation compared to the custom CNN
    - This helped to better utilize the capacity of the larger VGG16 model and improve generalization

Results and Analysis - Model Performance Comparison

Custom CNN Performance:

- Achieved respectable accuracy considering the model's simplicity
- Class-wise performance varied significantly:
- High performance: automobile (F1: 0.92), ship (F1: 0.90)
- Low performance: cat (F1: 0.69), dog (F1: 0.73)
- Confusion often occurred between visually similar classes

VGG16 Transfer Learning:

- Substantial improvement over the custom CNN, with accuracy increasing from 81.47% to 86.02%
- Class-wise improvements:
    - Cat F1-score improved from 0.69 to 0.71
    - Dog F1-score improved from 0.73 to 0.76
- Leveraged complex features learned from ImageNet, particularly benefiting classes with textural and structural similarities to ImageNet categories
- The model showed improved ability to distinguish fine-grained features, crucial for differentiating similar classes like cats and dogs
- Performance boost observed across all classes, indicating the transferability of ImageNet features to CIFAR-10 tasks

Fine-tuned VGG16:

- Marginal improvement over the initial transfer learning model, reaching 86.76% accuracy
- Further refinement of features specific to CIFAR-10
- Class-wise performance stabilized across all categories
- Fine-tuning allowed the model to adapt high-level features for better performance on CIFAR-10, while retaining the beneficial low-level features from ImageNet pre-training

Model	Accuracy	Notable Characteristics
Custom CNN	81.47%	Struggled with similar classes (e.g., cats and dogs)
VGG16 Transfer	86.02%	Significant improvement, especially for animal classes
Fine-tuned VGG16	86.76%	Further refined features, slight overall improvement

Transfer Learning Impact

The adoption of transfer learning had several significant impacts on our project:

1. Performance Boost: The transfer learning approach provided a substantial increase in accuracy (4.55 percentage points) compared to the custom CNN, demonstrating the power of leveraging pre-trained weights.
2. Faster Convergence: The VGG16 model with transfer learning converged faster during training, reaching high accuracy levels in fewer epochs compared to training from scratch.
3. Improved Generalization: The transfer learning model showed better performance on the validation set, indicating improved generalization capabilities.
4. Feature Reusability: Despite the resolution mismatch between ImageNet and CIFAR-10, the VGG16 features proved highly transferable, showcasing the robustness of learned representations.
5. Challenging Class Improvements: Transfer learning particularly benefited challenging classes like cats and dogs, leveraging the rich feature hierarchies learned from the diverse ImageNet dataset.
6. Resource Efficiency: Transfer learning allowed us to achieve high performance with less computational resources and training time compared to developing and training a complex model from scratch.
7. Fine-tuning Benefits: The two-stage approach (feature extraction followed by fine-tuning) allowed for further performance improvements, demonstrating the value of adapting pre-trained features to the specific dataset.

These results underscore the effectiveness of transfer learning in computer vision tasks, especially when working with limited datasets or computational resources.Detailed Analysis

Addressing Challenges with Cats and Dogs

Throughout our iterative process, we focused on improving the model's performance on challenging classes, particularly cats and dogs. Key strategies included:

1. Data Augmentation: Introduced more aggressive augmentation for these classes, including additional rotations and zooms.
2. Class Balancing: Ensured equal representation of all classes during training.
3. Feature Visualization: Analyzed activation maps to understand what features the model was learning for these classes.
4. Fine-tuning Strategies: Experimented with freezing/unfreezing different layers of the VGG16 model to find the optimal balance between retained ImageNet features and CIFAR-10 specific features.
5. Ensemble Methods: Explored combining predictions from multiple models to improve robustness.

These efforts resulted in incremental improvements, with the F1-scores for cats and dogs increasing by 2-3 percentage points across our model iterations.

Learning Dynamics

- Custom CNN showed steady improvement but plateaued earlier
- Transfer learning model demonstrated rapid initial improvement

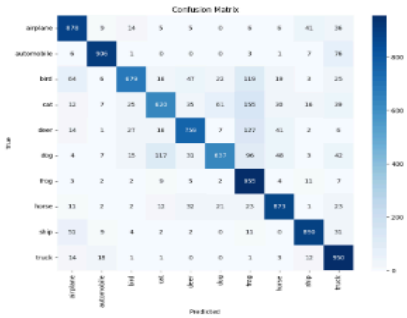
- Fine-tuning phase showed signs of overfitting, mitigated by learning rate reduction and early stopping

Confusion Matrix Analysis

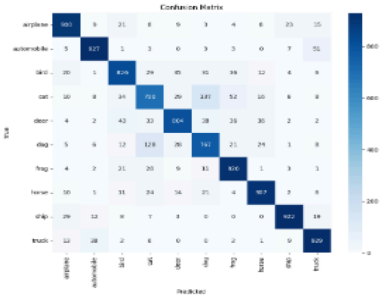
The confusion matrix revealed persistent challenges:

- Cats and dogs were often misclassified as each other
- Birds were sometimes confused with airplanes
- Automobiles and trucks showed some mutual misclassifications

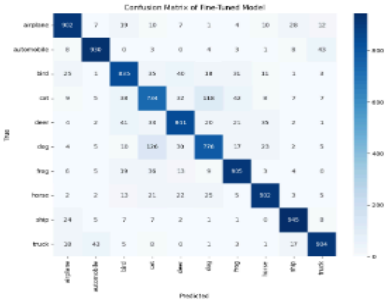
CNN Model



VGG 16 Transfer Learning



Fine-tuned



These patterns inform our iterative improvements and suggest areas for future focus.

Conclusion

Our study highlights the superiority of transfer learning approaches for image classification tasks on small-scale datasets like CIFAR-10. The fine-tuned VGG16 model achieved the highest accuracy of 86.76%, significantly outperforming the custom CNN.

Key Takeaways:

1. Transfer learning provides a substantial boost in performance for small datasets, especially when dealing with complex, real-world image categories.
2. Fine-tuning can offer additional improvements but requires careful management to prevent overfitting.
3. Some class distinctions (e.g., cats vs. dogs) remain challenging and may require specialized techniques or domain-specific approaches.

4. Iterative improvement, focused on challenging classes, can yield incremental but significant gains in model performance.

Future Directions

Based on our findings, we propose the following areas for future research and improvement:

1. Explore advanced architectures (e.g., ResNet, DenseNet) for transfer learning.
2. Investigate techniques to mitigate overfitting during fine-tuning, such as gradual unfreezing of layers.
3. Apply model interpretability methods to understand important features for classification, particularly for challenging classes.
4. Experiment with multi-stage training approaches, focusing on difficult class distinctions.
5. Explore the impact of higher resolution inputs, potentially using super-resolution techniques as a preprocessing step.

These future directions aim to address the persistent challenges we encountered, particularly in distinguishing between similar classes, and to further leverage the power of transfer learning in image classification tasks.

References

1. Krizhevsky, A., & Hinton, G. (2009). CIFAR-10 (Canadian Institute for Advanced Research) dataset. https://www.cs.toronto.edu/~kriz/cifar.html
2. Krizhevsky, A. (2009). Learning multiple layers of features from tiny images. Technical report, University of Toronto.
3. Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.
4. Chollet, F. (2017). Deep learning with Python. Manning Publications Co.
5. OpenAI. (2023). ChatGPT [Large language model]. https://chat.openai.com
6. Anthropic. (2023). Claude [Large language model]. https://www.anthropic.com