Technical Report – Deep Convolutional Neural Networks & Transfer Learning

Introduction

The increasing need for automation and intelligence in visual tasks has made computer vision one of the most active fields of machine learning. Convolutional Neural Networks (CNNs) have proven to be the cornerstone of success in image classification, object detection, and semantic segmentation.

This project aimed to develop a series of deep learning applications using CNNs for real-world image recognition problems. The study was carried out in **two main phases**:

1. **From-Scratch CNNs**: Custom-designed CNN architectures were built using TensorFlow/Keras without any transfer learning or pre-trained weights.
2. **Transfer Learning CNNs**: High-performance pre-trained models like **EfficientNetB0** and **ResNet50** were fine-tuned on the same datasets to improve accuracy and reduce training time.

The project incorporated **eight separate applications** with Streamlit interfaces for real-time testing. The tasks involved:

- **Image classification** of rice varieties, fruits, vegetables, and grape diseases.
- **Image segmentation** using the BSDS500 dataset.

Each application required training a model, evaluating its performance, saving the model, and deploying it in an interactive web interface.

Analysis

Datasets Used

1. **Rice Image Dataset**
   Contains categorized images of different rice grain varieties. This dataset is ideal for multi-class classification problems. Images were well-structured and labeled by folder.

2. **Fruit and Vegetable Dataset**
   A comprehensive set containing images of various fruits and vegetables, separated into `train`,

`validation`, and `test` folders. The diversity in lighting and background makes this a robust classification challenge.

3. **Grape Disease Dataset**
   This dataset features leaves of grapevines classified into healthy and various disease classes. Early and accurate detection is crucial in agriculture, making this a high-impact use case.

4. **BSDS500 Segmentation Dataset**
   Designed for image segmentation research, this dataset contains images and `.mat` files representing ground truth masks. The segmentation task involves identifying object boundaries within the images.

Preprocessing and Data Augmentation

Data preprocessing was tailored per task:

- All images were **resized** (100×100, 128×128, or 224×224).
- **Normalization** was applied by dividing pixel values by 255.
- **One-hot encoding** was used for categorical labels in classification.
- For segmentation, `.mat` files were loaded and converted into binary masks.
- Transfer learning models incorporated **data augmentation** using `ImageDataGenerator` with:
  - Rotation
  - Zoom
  - Width/height shifts
  - Horizontal flipping
  - Validation split

These steps ensured the model saw a variety of perspectives, improving generalization.

Methods

1. Custom CNN Architectures

Implemented using `Sequential` models from Keras. Each architecture adhered to the following:

- At least **5 convolutional layers**
- Minimum **3 max pooling layers**
- Dropout layers for regularization
- Final `Dense` layer with softmax activation for classification

Example structure:

Conv2D(32) → MaxPooling2D → Conv2D(64) → MaxPooling2D → Conv2D(128) → ... → Flatten → Dense → Dropout → Output

Used in:

- `rice_image.py`
- `fruitsandveg_image.py`
- `grapevine_cnn.py`

2. Custom U-Net for Segmentation

The model in `bsds500_segmentation_cnn.py` followed a **U-Net** architecture:

- **Encoder**: Downsampling using `Conv2D` + `MaxPooling`
- **Decoder**: Upsampling with `Conv2DTranspose`
- **Skip Connections**: Concatenated encoder layers with decoder layers to preserve spatial information

3. Transfer Learning Models

Used pretrained networks (EfficientNetB0, ResNet50) as feature extractors:

- The convolutional base was frozen to retain ImageNet-trained weights.
- On top: `GlobalAveragePooling`, `Dense`, and output softmax layers.

EfficientNetB0 was used for:

- `rice_transfer.py`
- `fruits_transfer_learning.py`

ResNet50 was used for:

- `grape_transfer_resnet.py`
- `bsds500_transfer_learning.py`

For BSDS500, a **ResNet50 + U-Net decoder** hybrid was created to combine pretrained backbone with segmentation logic.

All models were compiled with:

- Optimizer: `Adam`
- Loss: `categorical_crossentropy` or `binary_crossentropy`

- Metrics: `accuracy`, `precision`, `recall`, `MeanIoU` (for segmentation)

Training was conducted for 10–100 epochs, depending on model type and convergence.

Results

Here is a comparative summary of model performance:

Dataset	Model Type	Architecture	Approx. Accuracy / Metric	Notes
Rice Classification	From Scratch	CNN	~90%+	High accuracy with 5 conv layers
Rice Classification	Transfer Learning	EfficientNetB0	~95%	Faster training, better generalization
Fruit/Veg Detection	From Scratch	CNN	~85%	More variability in data
Fruit/Veg Detection	Transfer Learning	EfficientNetB0	~92–93%	Better feature representation
Grape Disease	From Scratch	CNN	~88%	Challenging due to subtle differences
Grape Disease	Transfer Learning	ResNet50	~93–95%	Excellent results with pretrained model
BSDS500	From Scratch	U-Net	MeanIoU ~0.6	Decent segmentation
BSDS500	Transfer Learning	ResNet50 + U-Net	MeanIoU ~0.7+	More precise segmentation masks

In each case, transfer learning models achieved **better performance** with **less training time**. From-scratch CNNs still performed well, particularly with simpler datasets like rice classification.

Visual predictions and segmentation results were verified using interactive **Streamlit apps**, which allowed image upload and real-time inference.

Reflection

This project was an excellent exercise in deep learning engineering, blending **theory, implementation, and deployment**. Key takeaways include:

- **Transfer learning is extremely powerful**, especially when labeled data is limited. Models like EfficientNet and ResNet50 significantly boost performance.
- **Custom CNNs**, while more complex to tune, are still valuable to understand low-level network design, parameterization, and overfitting.
- **Segmentation tasks** require careful handling of ground truth data and benefit from architectures like U-Net or DeepLabV3.
- Integrating models with **Streamlit** and deploying them via HuggingFace or other platforms allows for real-world application and user interaction.
- Future improvements may include:
  - Hyperparameter optimization using Keras Tuner or Optuna
  - Model explainability with Grad-CAM
  - Trying other pretrained architectures like MobileNetV3 or ViT (Vision Transformers)

This assignment deepened my understanding of convolutional architectures, regularization strategies, and the practical use of transfer learning in industry-grade problems.

What Was Submitted

- **4 CNN models from scratch**:
- `rice_image.py`
- `fruitsandveg_image.py`
- `grapevine_cnn.py`
- `bsds500_segmentation_cnn.py`
- **4 Transfer Learning models**:
- `rice_transfer.py`
- `fruits_transfer_learning.py`
- `grape_transfer_resnet.py`
- `bsds500_transfer_learning.py`
- All models saved as `.h5` files
- Streamlit UI apps for each model
- This technical report

- Ready for HuggingFace deployment or GitHub submission
