**COURSE:** DSA4050

**COURSE TITLE:** Deep Learning for Computer Vision

**INSTRUCTOR:** DR. EDWARD OMBUI

**Week 5 Lesson Notes: Image Classification and Object Detection**

**Overview**

This week, we will explore the fundamental principles of **image classification** and **object detection**, two essential tasks in **computer vision**. You will learn how deep learning models classify images based on features and patterns, and how object detection extends classification by identifying **both the object’s category and its location in an image**.

We will cover the following topics:

1. **Image Classification:** Understanding how neural networks categorize objects in images.
2. **Object Detection:** Comparing **R-CNN, Fast R-CNN, Faster R-CNN, and YOLO** frameworks for object detection.
3. **Practical Implementation:** Training an **object detection model** and evaluating its performance using **precision, recall, and mean Average Precision (mAP)**.

By the end of this week, you will be able to describe image classification principles, compare object detection models, and implement an object detection model on a dataset.

**1. Image Classification: Principles and Techniques**

**What is Image Classification?**

Image classification is the task of assigning a label to an image based on its content. It involves training a model to recognize patterns and features from images and categorize them into predefined classes.

📌 **Key Concepts in Image Classification:**

* **Feature Extraction:** Identifying patterns such as edges, shapes, and textures.
* **Convolutional Neural Networks (CNNs):** Using **convolutional layers** to detect hierarchical features.
* **Softmax Activation:** Assigning probabilities to each class and selecting the most likely one.

**Steps in Image Classification using CNNs**

1. **Preprocessing:** Resize images, normalize pixel values, and augment data.
2. **Feature Learning:** Convolutional layers extract spatial features.
3. **Classification:** Fully connected layers assign class probabilities.
4. **Evaluation:** Accuracy, precision, recall, and F1-score measure performance.

**Popular Image Classification Models**

| **Model** | **Key Features** | **Best For** |
| --- | --- | --- |
| **AlexNet** | First deep CNN, 8 layers | General image classification |
| **VGGNet** | 16-19 layers, uniform filters | High accuracy but slow |
| **ResNet** | Uses residual connections | Very deep networks |
| **EfficientNet** | Balances accuracy and efficiency | Lightweight applications |

📖 **Reference:** Krizhevsky et al. (2012). *ImageNet Classification with Deep Convolutional Neural Networks.*

**2. Object Detection: Techniques and Frameworks**

**What is Object Detection?**

Object detection extends image classification by identifying **both the class and location** of objects in an image. It outputs **bounding boxes** around detected objects and assigns labels to them.

📌 **Key Concepts in Object Detection:**

* **Region Proposals:** Identifying regions in an image that may contain objects.
* **Feature Extraction:** Using CNNs to extract spatial features.
* **Bounding Box Regression:** Predicting object location.

**Comparison of Object Detection Frameworks**

| **Framework** | **Key Features** | **Pros** | **Cons** |
| --- | --- | --- | --- |
| **R-CNN** (Region-based CNN) | Generates region proposals & classifies them | High accuracy | Very slow |
| **Fast R-CNN** | Improves R-CNN by sharing computations | Faster than R-CNN | Still slow |
| **Faster R-CNN** | Uses a Region Proposal Network (RPN) | Faster than Fast R-CNN | Computationally expensive |
| **YOLO** (You Only Look Once) | Treats detection as a regression problem | Extremely fast | Lower accuracy for small objects |

📖 **Reference:** Redmon et al. (2016). *You Only Look Once: Unified, Real-Time Object Detection.*

**3. Training an Object Detection Model**

**Dataset Preparation**

Before training an object detection model, we need a labeled dataset.

* **Common Datasets:** COCO (Common Objects in Context), PASCAL VOC, and custom datasets.
* **Annotations:** Images need labeled bounding boxes for objects.

**Step-by-Step Implementation in TensorFlow**

**Step 1: Import Libraries**

import tensorflow as tf

from tensorflow.keras.models import Model

from tensorflow.keras.applications import EfficientNetB0

from tensorflow.keras.layers import Dense, Flatten

from tensorflow.keras.preprocessing.image import ImageDataGenerator

**Step 2: Load a Pre-trained Model**

base\_model = EfficientNetB0(weights='imagenet', include\_top=False, input\_shape=(224, 224, 3))

# Freeze base model layers

for layer in base\_model.layers:

layer.trainable = False

**Step 3: Add Custom Layers for Object Detection**

x = Flatten()(base\_model.output)

x = Dense(256, activation='relu')(x)

x = Dense(4, activation='sigmoid')(x) # Bounding box coordinates

x\_class = Dense(10, activation='softmax')(x) # Classification output

model = Model(inputs=base\_model.input, outputs=[x, x\_class])

**Step 4: Compile the Model**

model.compile(optimizer='adam', loss=['mse', 'categorical\_crossentropy'], metrics=['accuracy'])

**Step 5: Train the Model**

history = model.fit(train\_images, [train\_boxes, train\_labels], epochs=10, validation\_data=(test\_images, [test\_boxes, test\_labels]))

**Step 6: Evaluate the Model**

model.evaluate(test\_images, [test\_boxes, test\_labels])

model.save("object\_detection\_model.h5")

**Evaluation Metrics**

* **Precision:** Measures how many detected objects were correct.
* **Recall:** Measures how many actual objects were detected.
* **mAP (mean Average Precision):** Combines precision and recall for overall accuracy.

📖 **Reference:** Lin et al. (2014). *Microsoft COCO: Common Objects in Context.*

**4. Learning Outcomes**

By the end of this session, you should be able to:  
✅ **Describe** the principles of **image classification** and **object detection**.  
✅ **Compare** different object detection frameworks (**R-CNN, Fast R-CNN, Faster R-CNN, YOLO**).  
✅ **Train an object detection model** on a dataset and evaluate its performance.

**5. Activities**

📌 **Discussion Topics:**

1. **What are the differences between image classification and object detection?**
2. **When would you choose Faster R-CNN over YOLO for object detection?**
3. **How can transfer learning improve object detection performance?**

📌 **Hands-on Task:**

* Train a **YOLO-based object detection model** on a real-world dataset.
* Fine-tune a **pre-trained Faster R-CNN model** and compare results.

**6. Conclusion**

This week, we explored:  
✅ **How image classification works** and its role in computer vision.  
✅ **The differences between object detection frameworks** like R-CNN and YOLO.  
✅ **How to train and evaluate an object detection model** using deep learning.

📌 **Next Steps:**

* Experiment with **different object detection frameworks**.
* Use **custom datasets** for training models.
* Explore **real-time applications** like self-driving cars and medical imaging.

**References**

1. **Krizhevsky, A., Sutskever, I., & Hinton, G. (2012).** *ImageNet Classification with Deep Convolutional Neural Networks.*
2. **Girshick, R. (2015).** *Fast R-CNN.*
3. **Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016).** *You Only Look Once: Unified, Real-Time Object Detection.*
4. **Lin, T. Y., Maire, M., Belongie, S., Hays, J., & Perona, P. (2014).** *Microsoft COCO: Common Objects in Context.*