



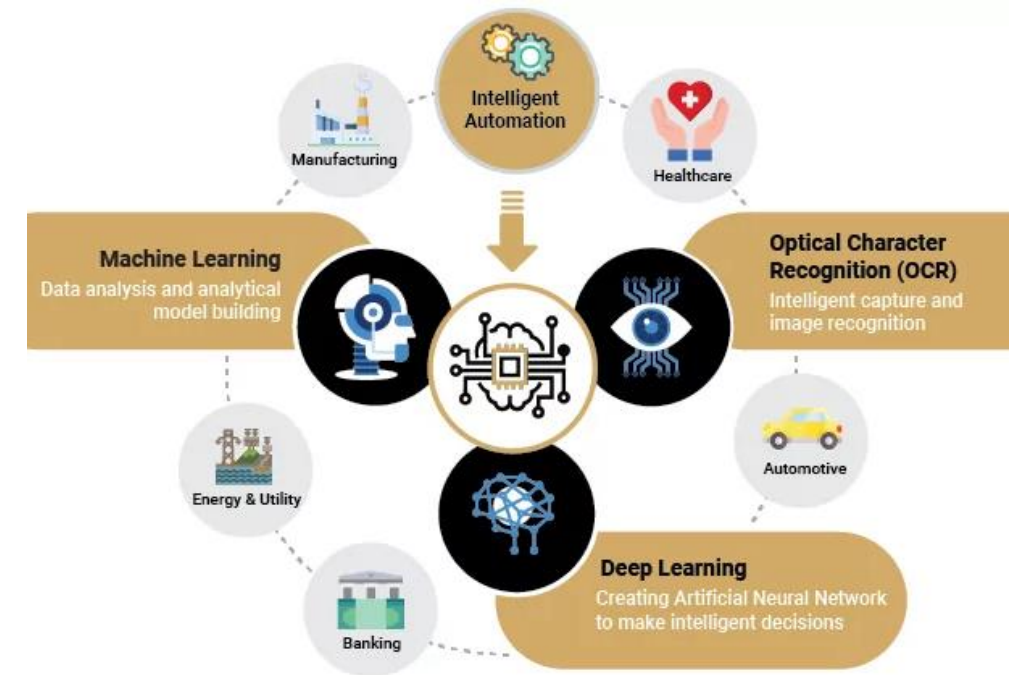
Learning Journey portfolio

- Lionel Silva
- 08/04/2024
- ITAI 1378 Computer Vision
- Summer 2024

Professor: Patricia McManus | CRN:12461

Introduction

- The objective of this course is to provide you with the knowledge and skills necessary to achieve your professional goals in computer vision . You will learn the basics of computer vision and its fundamental platforms, as well as the most recent cameras, lidar sensors, high resolution, 3D and stereo vision, and light capture. Furthermore, you will gain an understanding of the algorithms and techniques that enable computers to process images and videos, and how the Machine Learning process aids in the progress of image understanding from in immense set of data.



GitHub Portfolio Presentation Link

- https://github.com/Leon87551/Pf_LionelSilva_ITAI-1378.git

Module 2 Camera & Sensors

- We learned about the different types of cameras and sensors and their applications.

- Primary computer cameras utilize image sensors and algorithms to capture and explain visual information. Their applications include robotics, automation, and surveillance.

- RGB cameras, being popular computer vision cameras, are widely used in a variety of applications such as object detection, facial recognition, and traffic monitoring.

- Depth cameras create 3D maps of the surroundings by calculating the space among objects in the scene. They are used in gesture recognition, robotics, and augmented reality applications.

- Thermal cameras capture images by detecting the heat emitted by objects and are utilized in applications like firefighting, medical imaging, and surveillance. Stereo cameras utilize two or several cameras to capture 3D images and are employed in applications such as autonomous vehicles and mapping.

- LiDAR (Light Detection and Ranging) is a remote sensing technology that utilizes laser pulses to calculate length and make 3-D maps of the surroundings. It is usually in self-driving cars and other robotics applications.
- Radar (Radio Detection and Ranging) is a remote sensing technology that utilizes radio waves to reveal and find objects. It prevents collision and changeable cruise control in automotive applications.

Module 5: Machine Learning for computer vision

- This module covered the following:
- Data Types: Tabular, Image, and Text.
- Features and Labels in ML: Structuring Tabular Data for ML Tasks.
- Supervised vs. Unsupervised Learning: Concepts and Applications
- ML Lifecycle: problem definition, data processing, model Training, and Model evaluation and comparison.
- What I learned in this module is the three types of ML :
- Supervised learning needs training data with labels. While unsupervised learning does not need labeled data. Moreover, Reinforcement Learning uses a trial-and- error method to learn.
- Activities: L05 Image Classification with Nearest Neighbors.

Module 5: Machine Learning for computer vision

Supervised and Unsupervised learning

Supervised learning

Data is provided *with labels*.

The model learns by looking at these examples.

Unsupervised learning

Data is provided *without labels*.

The model finds patterns in the data.

11

Defining the problem

Real-life problem

Problem formulation

Data processing

Model training/
tuning/testing

Evaluation

Deployment

Define the objective:

A specific outcome that you want to use ML to achieve. For example, to help someone achieve something or some business goal.

Increase revenue by ensuring all products in catalog have prices & can be surfaced to customer.

Translate the objective to an ML problem:

Express the objective in ML terms; for example, predicting or calculating the probability.

Predict the price (continuous numerical value) for all products in the catalog where the price is missing.

Module 6: Basics of Neural Network

- This module covered the following:
- Artificial Neuron
- Machine Learning basics
- Neural Network components
- Training Neural Networks
- Training Neural Networks
- Neural Network Architectures

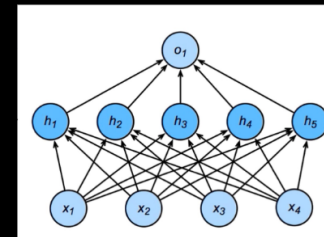
- A neural network is a computational model stimulated by the form and role of the human brain.

I learned about the training process initialization, feedforward, loss calculation, backpropagation, weight update, iteration, and evaluation. In addition, I learned the architecture of neural networks. There were single-neuron output and multi-neuron output. In a single-neuron output, regression or binary classification is made. In contrast, the multi-neuron output makes multiclass classification.

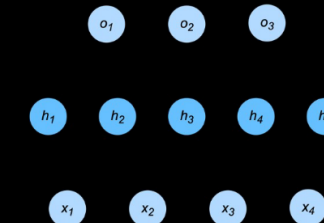
THE LEARNING PROCESS

- **Initialization:** Start with random weights for connections between neurons.
- **Feedforward:** Input data is fed into the network and processed through each layer to get the output.
- **Loss Calculation:** Determine the error of the output by comparing it with the expected result using a loss function.
- **Backpropagation:** Calculate the gradient of the loss function with respect to each weight by the chain rule, moving from the output layer backward to input layers.
- **Weight Update:** Adjust the weights of the connections to minimize the loss, typically using optimization algorithms like gradient descent.
- **Iteration:** Repeat the feedforward, loss calculation, backpropagation, and weight update steps until the model performs satisfactorily.
- **Evaluation:** Validate the model using a separate dataset not seen by the network during training to test its generalization capability.

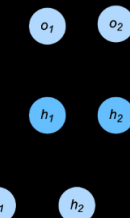
NEURAL NETWORK ARCHITECTURES (1 OF 2)



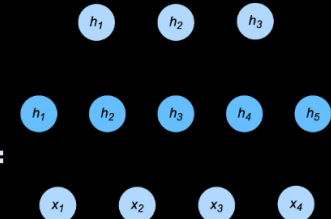
Single-neuron output:
Make regression or binary classification



Multineuron output:
Make multiclass classification



Two-neuron output:
Make binary classification



Module 6: Basics of Neural Network

- Assignments:
- A06 –TensorFlow Playground
- Lab 06 Chihuahua or Muffin ?
- In A06 I gained understanding on how parameters affect the performance of the networks. In the lab 06, we learned how to import data to classify, distinguish, and identify images. We attained knowledge on how to develop a deep neural network model using the Chihuahua or muffin. By using datasets to train, test, and validate, we were able to understand how deep learning and neural networks work in conjunction.
- Challenges encountered:
- I had a few challenges with this assignment. To start with, I had to learn how to use AWS SageMaker Studio Lab. I overcome this challenge by watching a YouTube video tutorial for beginners. This made it easy to navigate the studio and complete my lab.
- Potential real-world applications:
- This model is useful in real-world applications where objects must have to identified certain features for quality control. For example, this model can be used in manufacturing to detect defect parts. As well to detect a good part. This model is useful in real world applications to validate only good parts. This model helps manufacturing plants to meet their quality standards.
- My reflection on this lab is to be knowledgeable in deep learning and neural networks, you must understand what each code dataset you are imputing means.



L06



A06



L06



```

train progress
    def __init__(self, model, data_loader, validation_data_loader, patience):
        self.model = model
        self.data_loader = data_loader
        self.validation_data_loader = validation_data_loader
        self.patience = patience
        self.best_loss = None
        self.counter = 0

    def train(self):
        for epoch in range(1, self.patience + 1):
            # Training phase
            train_loss, train_acc = self.train_epoch()

            # Validation phase
            val_loss, val_acc = self.validate_epoch()

            # Check for early stopping
            if self.best_loss is None or val_loss < self.best_loss:
                self.best_loss = val_loss
                self.counter = 0
            else:
                self.counter += 1

            # Print progress
            print(f'Epoch {epoch}: train_loss={train_loss}, val_loss={val_loss}, train_acc={train_acc}, val_acc={val_acc}')

        return self.model

    def train_epoch(self):
        # Set model to training mode
        self.model.train()

        # Iterate over the training data loader
        for data, targets in self.data_loader:
            # Forward pass
            outputs = self.model(data)

            # Calculate loss
            loss = self.criterion(outputs, targets)

            # Backward pass and optimization
            self.optimizer.zero_grad()
            loss.backward()
            self.optimizer.step()

        # Calculate average loss and accuracy for the epoch
        train_loss, train_acc = self.calculate_metrics(self.data_loader)

        return train_loss, train_acc

    def validate_epoch(self):
        # Set model to evaluation mode
        self.model.eval()

        # Iterate over the validation data loader
        for data, targets in self.validation_data_loader:
            # Forward pass
            outputs = self.model(data)

            # Calculate loss
            loss = self.criterion(outputs, targets)

            # Calculate accuracy
            _, predicted = torch.max(outputs.data, 1)
            acc = sum(predicted == targets).item() / targets.size(0)

        # Calculate average loss and accuracy for the epoch
        val_loss, val_acc = self.calculate_metrics(self.validation_data_loader)

        return val_loss, val_acc

    def calculate_metrics(self, data_loader):
        # Calculate average loss and accuracy for the data loader
        total_loss = 0
        total_acc = 0
        total_size = 0

        for data, targets in data_loader:
            # Forward pass
            outputs = self.model(data)

            # Calculate loss
            loss = self.criterion(outputs, targets)

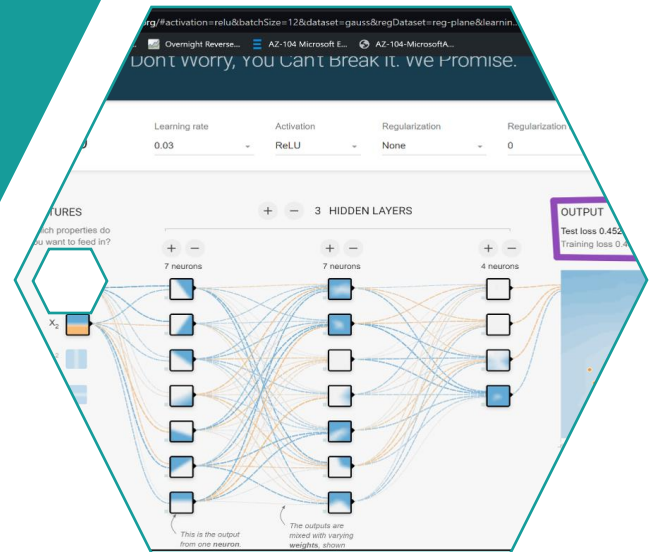
            # Calculate accuracy
            _, predicted = torch.max(outputs.data, 1)
            acc = sum(predicted == targets).item() / targets.size(0)

            # Accumulate metrics
            total_loss += loss.item()
            total_acc += acc
            total_size += targets.size(0)

        # Calculate average metrics
        avg_loss = total_loss / total_size
        avg_acc = total_acc / total_size

        return avg_loss, avg_acc

```



Module : 7 Convolutional Neural Networks

- This module covered the following:
 - Introduction to Convolutional Neural Networks.
 - Core components of CNNs and how they process images.
 - CNN training and leveraging libraries
 - Applications, challenges, and future research.
 - Computer Vision datasets.
 - A convolutional neural network (CNN) is a deep learning algorithm that extracts features and identifies patterns. CNNs outperform traditional methods at comprehending spatial structure in images.
- I learned about the training CNNs, libraries, applications, challenges, future research, core components and cv datasets. The core concepts included:
- Convolutional layers feature extraction.
 - Pooling layers minimize dimensionality.
 - Fully connected layers classification.

Module 7: Convolutional Neural Networks

- Assignments:
- A07 –ITAI 1378 Manual CNN
- L07 Chihuahua or Muffin with CNN
- In A07, I gained an understanding of the task of Convolutional layers and filters in image processing. In the L07, I learned that CNN, when it extracts significant features from images, does it with enhanced precision.
- Challenges encountered:
- The challenges I faced were when I had to fix the errors in the program's cells. I encountered errors in the program that I could not solve, but going through the program was still a rewarding experience. I gained a solid understanding of the code in the program, and I am confident in my ability to become an expert programmer.
- My reflection on this lab is it provides a solid understanding of why CNN outperforms traditional neural networks. Additionally, it explored ethical concerns in areas like deep fake creation, invasion of privacy, and image manipulation.



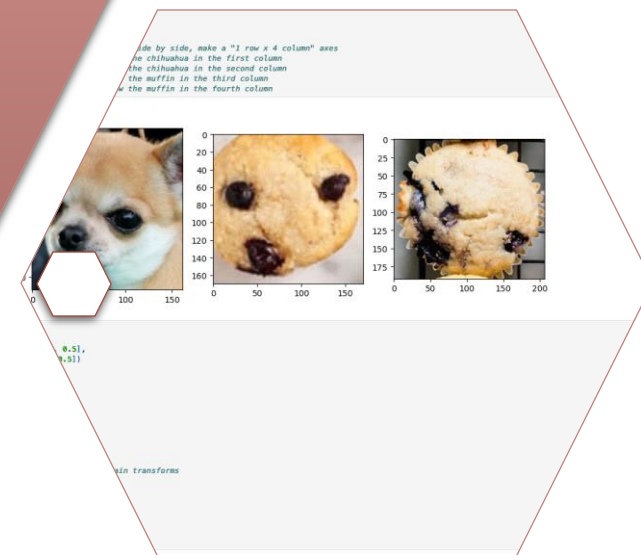
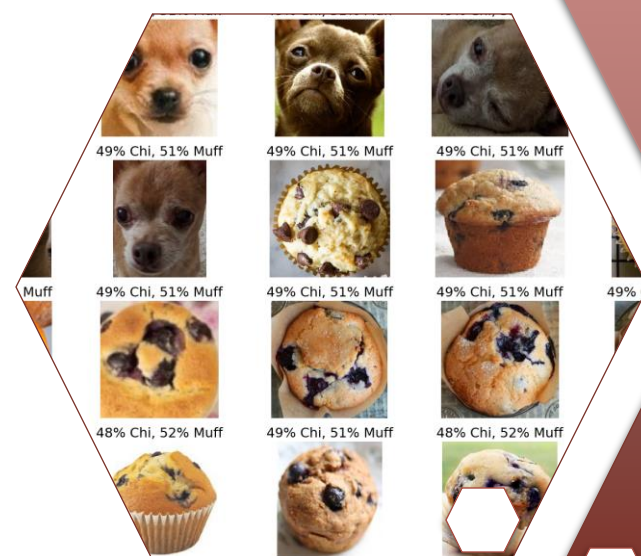
L07



L07



L07



Module : 8 CNN Basic Architecture And Transfer Learning

This module covered the following:

- Convolutional Neural Networks (CNNs) Recap
- Basic concepts and applications
- Image datasets
- Transfer Learning
- Concept and motivation
- Approaches and techniques
- Pre-trained models and toolkits
- Basic CNN Architectures
- Historical perspective
- Key innovations
- Deep dive into:
 - LeNet
 - AlexNet

I learned about the importance of transfer learning, which includes time/resource efficiency and performance. I also learned how CNN architecture has evolved over time. Lastly, I gained knowledge on AlexNet, which is used to artificially enlarge training datasets.

Module : 8 CNN Basic Architecture And Transfer Learning

- Assignments:
- A08 –ITAI 1378 Manual cv
- In A08 we learned about the key concepts, which include bounding boxes, annotations, IoU, and confidence score. A *Bounding box* is a virtual box encompassing an object in an image. *Annotations* are the format used to label images given to a dataset. The confidence score indicates the possibility of an object being present in the bounding box. In addition, we also cover the algorithms like R-CNN, Faster-R-CNN, Fast R-CNN, and SSD. These algorithms play a crucial role in object detection for example Faster-R-CNN evaluates a region-based object classification (ROI) by using the feature maps given through the convolutional layer. In SSD the algorithm executes real-time detection in numerous objects in an image with high precision. R-CNN uses a discriminating search technique to identify Rols in the input images.
- Challenges encountered:
- No challenges for this assignment.
- My reflection on this assignment:
- This cheat sheet will benefit us in the future when performing object detection tasks because we have gained enough knowledge of the steps to take when starting one. We have also gained knowledge of the fundamentals of key concepts, common challenges and solutions, and the libraries and tools needed for the task.

Object Detection Cheat Sheet

Reference Guide for Object
Tasks
Silva Hac-King-Do ITAI 1378 CV
va
o Benson
row
am



A08

A08

A08

Concepts

A bounding box is a virtual box utilized to encompass an object of interest in a video. It serves as a reference point for the object recognition process to determine the position and dimension of the object within the frame.

Annotations: labels images in a provided dataset to train machine learning models. A list of annotation formats (including bounding boxes) XML, JSON, and Pickle.

Intersection over Union (IoU) evaluates the similarity between two bounding boxes by computing the ratio of their intersection to their union.

Object Detection Algorithms

Fast R-CNN uses a discriminating search technique to identify Regions of Interest (RoIs) in the input image. It employs a region-wise classifier based on DCN (Deep Convolutional Neural Network) to classify the RoIs.

Fast R-CNN is a region-based convolutional neural network is an object detection method that improves on (Fast-R-CNN) by using region proposal network (RPN). The system performs region-based object classification (ROI pooling) by using the features mapped through the convolutional layer.

Mask R-CNN is an upgraded version of the R-CNN that collects CNN features from a region of interest (ROI) into one forward pass across the image. A Single Shot Detector is a CV and ML algorithm that is used to perform real time detection of numerous objects in a video.

Module : 09 Advanced CNN Architecture And Object Detection & Recognition

- This module covered the following:

- Basic CNN architectures

- Transfer Learning

Advanced CNN Architectures

Object Detection

- Bounding Boxes

- Two stage detection

- One stage detection

I learned about the differences between one-stage and two-stage detectors. In one-stage detectors, speed is prioritized, and in two-stage detectors, accuracy is prioritized. I also learned that bounding boxes are important for determining an object's location.

Module : 09 Advanced CNN Architecture And Object Detection & Recognition

- Assignments:
 - L09 Object detection using Transfer learning and Pascal VOC 207 Dataset
 - Object detection challenge
- Challenges encountered:
 - The Pascal VOC dataset had incorrect labeling. CIFAR-10 images were constantly out of focus, and COCO caused timeouts. In contrast, the Oxford – IIIT dataset proved to be the most effective dataset for our needs. The images were clear, labeled properly, and processed efficiently.
- Reflection on these assignments:
 - This experience testing different kinds of datasets will help you pick the one that is appropriate for your application.

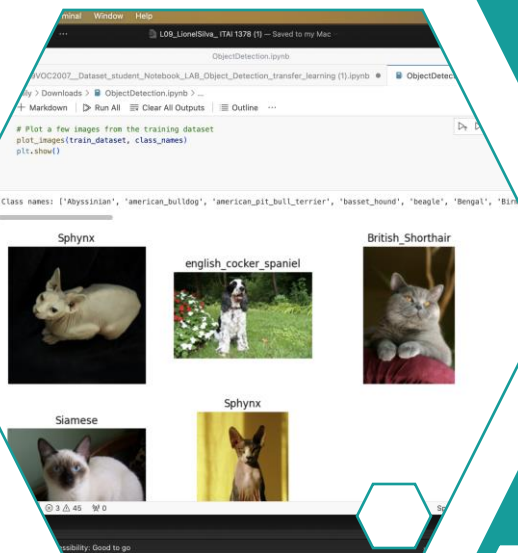
L09

L09

Object Detection Challenge

```
plt.legend()
plt.show()

# take a few examples from the training set
for example in train_dataset.take(2): # Process 2 images
    run_detector_and_visualize(example)
```



```
autogluon-multimodal 0.8.3 requires pandas<1.6,>=1.4.1, but you have pandas 1.6.0 which is incompatible.
autogluon-multimodal 0.8.3 requires pytorch-lightning<1.10.0,>=1.9.0, but you have pytorch-lightning 2.0.9 which is incompatible.
autogluon-multimodal 0.8.3 requires scikit-learn<1.4.1,>=1.1, but you have scikit-learn 1.4.2 which is incompatible.
autogluon-multimodal 0.8.3 requires torch<1.14,>=1.9, but you have torch 0.0.post304 which is incompatible.
autogluon-multimodal 0.8.3 requires torchmetrics<0.12.0,>=0.11.0, but you have torchmetrics 1.0.3 which is incompatible.
autogluon-multimodal 0.8.3 requires torchvision<0.15.0, but you have torchvision 0.15.2a0+ab7b3e6 which is incompatible.
Successfully installed array-record-0.5.1 dm-tree-0.1.8 docstring-parser-0.16 etils-1.7.0 immutabledict-4.2.0 keras-3.4.1 libclang-18.1.1 ml-dtypes-0.4.0 namex-0.0.8 optree-0.12.1 promise-2.3 protobuf-3.20.3 simple-parsing-0.1.5 tensorboard-2.17.0 tensorflow-2.17.0 tensorflow-datasets-4.9.6 tensorflow-hub-0.16.1 tensorflow-io-gcs-filesystem-0.37.1 tensorflow-metadata-1.15.0 tf-keras-2.17.0
Note: you may need to restart the kernel to use updated packages.
```

```
# Install OpenCV
!pip install opencv-python-headless

# Import necessary libraries
import tensorflow as tf
import tensorflow_hub as hub
import tensorflow_datasets as tfds
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.patches as patches
import cv2
from PIL import Image
import requests
from io import BytesIO

print("TensorFlow version:", tf.__version__)
print("TensorFlow Hub version:", hub.__version__)
print("OpenCV version:", cv2.__version__)
```