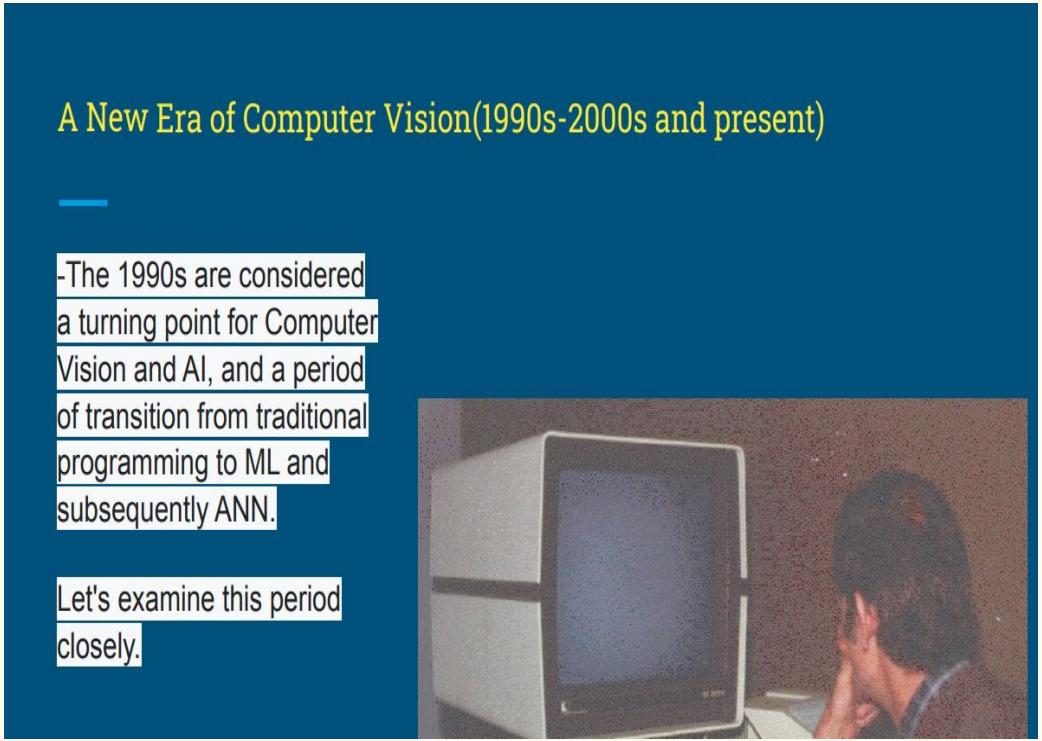
Portfolio Project

—

Mustafa Yucedag
ITAI-1378SUM

MyFirst Powerpoint Slide of Comp. Vis.



Everything has started with this assignment. Firstly, thanks to Professor McManus for the class. Secondly, thanks to my group mates for their effort. We have had an educational and fun semester.

Group Assignment A04(My Part) - One Day in the life of Seth Celtic

Feature Engineering and Image Processing Functions

After an amazing pasta salad for lunch I am now ready for the preliminary steps for this pattern recognition task. I have decided to apply the GaussianBlur function from OpenCV and Canny edge detection to reduce noise. I have done careful research into these algorithms and how they can be applied to this pattern recognition problem introduced in breast cancer detection. While selecting a template it becomes evident to me that Open CV will be used to convert images to grayscale before the edge detection methods can be applied.

Before applying those methods, I did some “theoretical” research on how Image Processing functions work. I found out it is all about converting the images to pixel values and multiplying or convolving them with the very functions we are using with the help of opencv to get the new pixel values meaning “preprocessed images”.

I have checked the images using matplotlib after applying the opencv functions using simple 5 lines of code that goes like this:

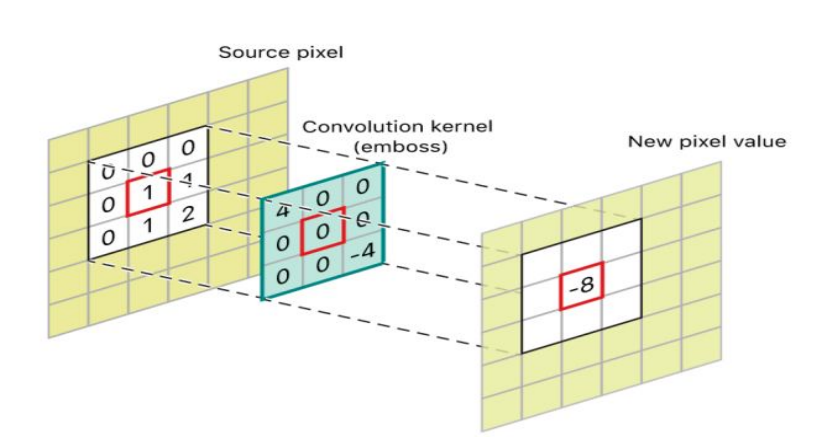
```
sample_img = images[0]

processed_sample = preprocess_image(sample_img)

plt.imshow(processed_sample, cmap='gray')

plt.title('Processed Image')

plt.show()
```



My Journal on L06

The neural network's architecture for this lab is called “MySkynet” and includes 3 linear layers with the dimensions being (128,64)&(64,32)&(32,2). Compared to the neural network in L07, this NN can be called more “primitive”.

I received around %83 accuracy for this task. The training time for this NN was shorter compared to the NN in Lab07 because it has linear layers rather than complicated convolution layers. I trained it for 3 epochs and it took around 10 seconds. The challenge I have gone through was about the defining phase of the NN from scratch. Because this Lab was interactive (I needed to define the dimensions etc.), I had to define MySkynet properties so I got help from my teammates.

I believe binary classification solutions are very useful in the modern age. I remember when COVID-19 happened, a model was developed only to recognize if one wears a mask or not. That was useful for public areas like buses, metros, trains, etc. Binary classifications can be adapted to anywhere, anytime. I have an interest in using image processing algorithms on factory production lines to detect defective products. I found an article that shows the usage of AI and Comp. Vis. for this problem. These models are sensitive in terms of security. So let’s talk about Dongbock Kim’s production line model. Databases where images are kept for such models should be handled as cautiously as possible. If these images fall into the hands of a rival company, huge problems may occur in terms of competition.

A08 - Our Cheatsheet

Object Detection Cheat Sheet

Key Concepts

- Bounding Boxes:
  - Definition: Rectangles that define the location of objects within an image.
  - Purpose: Used to specify the area of interest for object detection models.
- Annotations:
  - Explanation: Labeling objects within images to create a training dataset.
  - Tools: LabelImg, VGG Image Annotator (VIA).
- Confidence Scores:
  - Definition: Measure of how confident the model is that an object exists within a bounding box.
  - Range: Typically between 0 and 1.
- Intersection over Union (IoU):
  - Definition: Metric used to evaluate the accuracy of an object detector by comparing the overlap between predicted and ground truth bounding boxes.
  - Formula: $IoU = \frac{\text{Area of Overlap}}{\text{Area of Union}}$

Object Detection Algorithms

- R-CNN (Region-based Convolutional Neural Networks):
  - Overview: Extracts region proposals and uses CNN to classify them.
  - Workflow: Region proposal → Feature extraction → Classification.
- Fast R-CNN:
  - Improvements: Combines region proposal and feature extraction for faster processing.
  - Workflow: Single-stage detection using ROI pooling.
- Faster R-CNN:
  - Introduction: Adds Region Proposal Network (RPN) for real-time object detection.
  - Workflow: Feature extraction → RPN → ROI pooling → Classification.
- SSD (Single Shot MultiBox Detector):
  - Features: Detects objects in a single forward pass, using multiple feature maps.
  - Advantages: Faster than R-CNN variants.
- YOLO (You Only Look Once):
  - Key Features: Divides image into grid cells, each cell predicts bounding boxes and class probabilities.
  - Advantages: Real-time object detection with high speed.

Object Detection Challenge

My Reflections

Throughout this project, my primary focus was on implementing the NASNetMobile model, Marcus did the fine-tuning part for me. This involved several key tasks, including setting up the NASNetMobile architecture and integrating data augmentation techniques.

The NASNet (Neural Architecture Search Network) architecture is particularly fascinating because it was discovered using neural architecture search (NAS). This architecture basically uses a Reinforcement Learning Model to find the best ANN structure. Reorganizing the code for better readability and maintainability was a critical step. I learned to structure the project in a way that makes it easier for others to understand and contribute. So I have created a Github page with the scripts for each step(training, evaluation, etc.)

This project enhanced my skills in deep learning, particularly in using pre-trained models and documenting them with Github. It also improved my abilities in data preprocessing and augmentation, which are critical for any successful machine learning project.

Object Detection Challenge NASNetMobile Model

Using NASNetMobile Model

+ Code + Markdown

```
# Load pre-trained NASNetMobile model
base_model = tf.keras.applications.NASNetMobile(input_shape=(224, 224, 3), include_top=False, weights='imagenet')
base_model.trainable = False  # Freeze the base model

# Add classification head
model = tf.keras.Sequential([
    tf.keras.layers.Input(shape=(224, 224, 3)),
    base_model,
    tf.keras.layers.GlobalAveragePooling2D(),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(37, activation='softmax')  # 37 classes in the dataset
])

model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
model.summary()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/nasnet/NASNet-mobile-no-top.h5
19993432/19993432 [==============================] - 0s 0us/step
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 NASNet (Functional)         (None, 7, 7, 1056)        4269716

 global_average_pooling2d_1  (None, 1056)              0
  (GlobalAveragePooling2D)
```

Object Detection Challenge Model Training

```
# Train the model
history = model.fit(augmented_train_data, validation_data=test_data, epochs=6)
```

```
Epoch 1/6
115/115 [==============================] - 70s 585ms/step - loss: 1.2454 - accuracy: 0.6707 - val_loss: 0.5522 - val_accuracy: 0.8160
Epoch 2/6
115/115 [==============================] - 67s 587ms/step - loss: 0.5281 - accuracy: 0.8245 - val_loss: 0.4705 - val_accuracy: 0.8471
Epoch 3/6
115/115 [==============================] - 67s 583ms/step - loss: 0.4276 - accuracy: 0.8592 - val_loss: 0.5113 - val_accuracy: 0.8452
Epoch 4/6
115/115 [==============================] - 67s 584ms/step - loss: 0.3388 - accuracy: 0.8954 - val_loss: 0.4818 - val_accuracy: 0.8509
Epoch 5/6
115/115 [==============================] - 66s 574ms/step - loss: 0.3100 - accuracy: 0.8970 - val_loss: 0.5418 - val_accuracy: 0.8392
Epoch 6/6
115/115 [==============================] - 66s 577ms/step - loss: 0.2685 - accuracy: 0.9095 - val_loss: 0.4873 - val_accuracy: 0.8485
```

Object Detection Challenge Predictions

Actual: Bombay
Predicted: Bombay

Actual: Birman
Predicted: Birman

Citations

Kim, Dongbock, et al. "A Study on Sample Size Sensitivity of Factory Manufacturing Dataset for CNN-Based Defective Product Classification." Computation, vol. 10, no. 8, 2022, p. 142. MDPI, https://doi.org/10.3390/computation10080142.

Sharda, Aryaman. "Image Filters: Gaussian Blur." Medium, 20 June 2023, https://aryamansharda.medium.com/image-filters-gaussian-blur-e b36db6781b1. Accessed 25 June 2024.