National University of Sciences and Technology
School of Electrical Engineering and Computer Science
Department of Computer Science

CS867: Computer Vision

Fall 2019

Assignment 3

Image Classification using CNN

Announcement Date: 3rd Dec, 2019

Due Date: 11th Dec 2019 at 11:55 pm (on LMS)

Instructor: Dr. Muhammad Moazam Fraz

Background:

The convolutional neural network (CNN) is a class of **deep learning neural networks**. CNNs represent a huge breakthrough in image recognition. They're most commonly used to analyze visual imagery and are frequently working behind the scenes in image classification. They can be found at the core of everything from Facebook's photo tagging to self-driving cars. They're working hard behind the scenes in everything from healthcare to security.
They're fast and they're efficient. But how do they work?
Image classification is the process of taking an **input** (like a picture) and outputting a **class** (like "cat") or a **probability** that the input is a particular class ("there's a 90% probability that this input is a cat"). You can look at a picture and know that you're looking at a terrible shot of your own face, but how can a computer learn to do that?
With a convolutional neural network!
A CNN has

- Convolutional layers
- ReLU layers
- Pooling layers
- a Fully connected layer

A classic CNN architecture would look something like this:
Input ->Convolution ->ReLU ->Convolution ->ReLU ->Pooling -> ReLU ->Convolution ->ReLU ->Pooling ->Fully Connected

A CNN **convolves** (not convolutes…) learned features with input data and uses 2D convolutional layers. This means that this type of network is ideal for processing 2D images. Compared to other image classification algorithms, CNNs actually use very little preprocessing. This means that they can **learn** the filters that have to be hand-made in other algorithms. CNNs can be used in tons of applications from image and video recognition, image classification, and recommender systems to natural language processing and medical image analysis.
CNNs have an input layer, and output layer, and hidden layers. The hidden layers usually consist of convolutional layers, ReLU layers, pooling layers, and fully connected layers.

- Convolutional layers apply a convolution operation to the input. This passes the information on to the next layer.
- Pooling combines the outputs of clusters of neurons into a single neuron in the next layer.
- Fully connected layers connect every neuron in one layer to every neuron in the next layer.

In a convolutional layer, neurons only receive input from a subarea of the previous layer. In a fully connected layer, each neuron receives input from *every* element of the previous layer.
A CNN works by extracting features from images. This eliminates the need for manual feature extraction. The features are not trained! They're learned while the network trains on a set of images. This makes deep learning models extremely accurate for computer vision

tasks. CNNs learn feature detection through tens or hundreds of hidden layers. Each layer increases the complexity of the learned features.

A CNN

- starts with an input image
- applies many different filters to it to create a feature map
- applies a ReLU function to increase non-linearity
- applies a pooling layer to each feature map
- flattens the pooled images into one long vector.
- inputs the vector into a fully connected artificial neural network.
- processes the features through the network. The final fully connected layer provides the "voting" of the classes that we're after.
- trains through forward propagation and backpropagation for many, many epochs. This repeats until we have a well-defined neural network with trained weights and feature detectors.

Task

Background

How can computer vision and deep learning detect natural disasters?

Natural disasters cannot be prevented — *but they can be detected.*
All around the world we use sensors to monitor for natural disasters:

- **Seismic sensors** (seismometers) and **vibration sensors** (seismoscopes) are used to monitor for earthquakes (and downstream tsunamis).
- **Radar maps** are used to detect the signature "hook echo" of a tornado (i.e., a hook that extends from the radar echo).
- **Flood sensors** are used to measure moisture levels while **water level sensors** monitor the height of water along a river, stream, etc.
- **Wildfire sensors** are still in their infancy but hopefully will be able to detect trace amounts of smoke and fire.

Each of these sensors is *highly specialized* to the task at hand — detect a natural disaster early, alert people, and allow them to get to safety.
Using computer vision we can *augment* existing sensors, thereby increasing the accuracy of natural disaster detectors, and most importantly, allow people to take precautions, stay safe, and prevent/reduce the number of deaths and injuries that happen due to these disasters.

The Dataset:

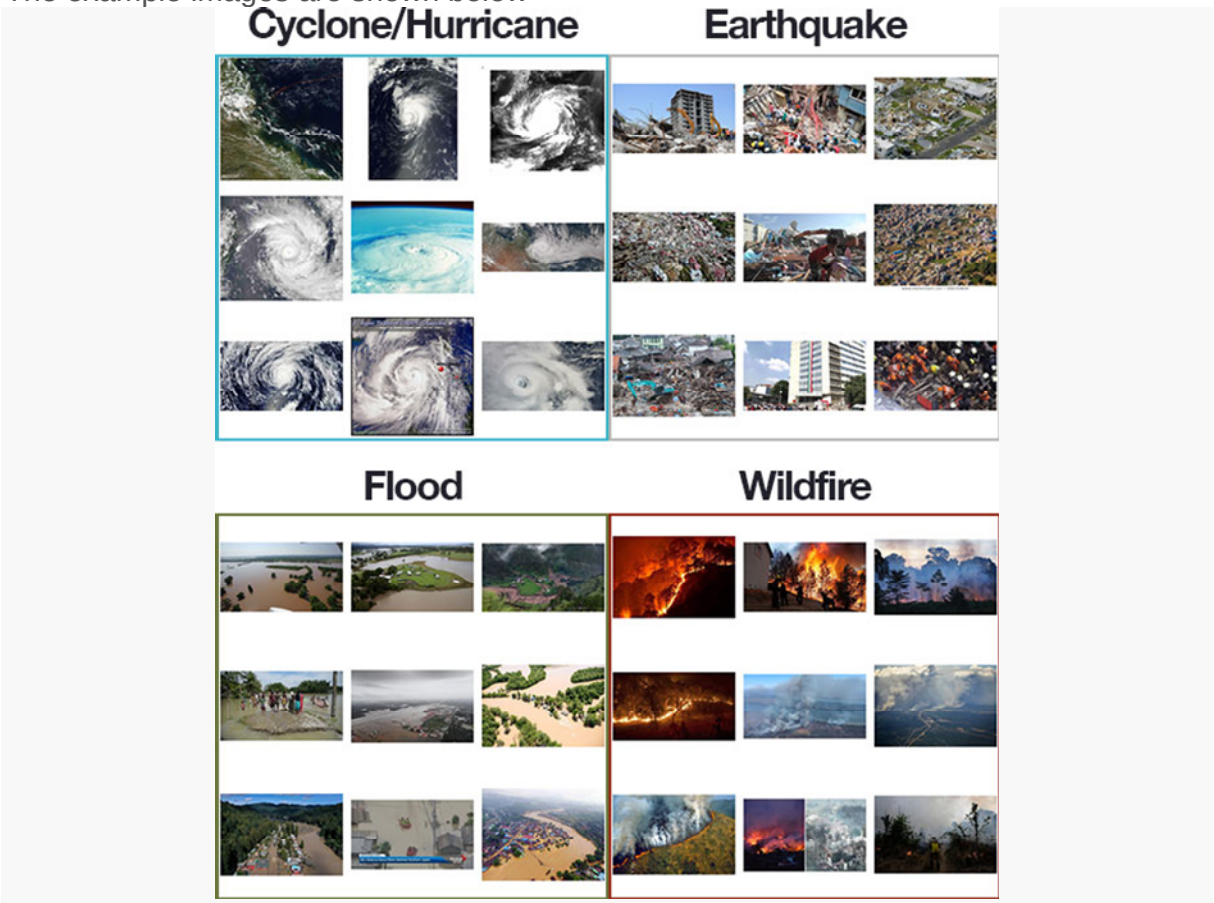
Natural Disaster Dataset: A dataset of images containing fruits and vegetables

Available at : https://drive.google.com/file/d/1NvTyhUsrFbL91E10EPm38IjoCg6E2c6q/view

Natural disaster dataset consists of four classes:

- **Cyclone/Hurricane:** 928 images
- **Earthquake:** 1,350
- **Flood:** 1,073
- **Wildfire:** 1,077

The example images are shown below



Task

Implement an image classification pipeline using CNN to classify the natural disaster images into different classes.

A few tutorials and python notebooks are shared on LMS about implementing a CNN using Keras.

This assignment enable us solve a real-world classification problem for classifying natural disaster videos.
Such an application could be:

- Deployed along riverbeds and streams to monitor water levels and detect floods early.
- Utilized by park rangers to monitor for wildfires.
- Employed by meteorologists to automatically detect hurricanes/cyclones.
- Used by television news companies to sort their archives of video footage.

Submission instruction:

Please submit following in a zip file on LMS.

- Python code
- Report : containing the following
    - Network summary or diagram
    - training settings (Experiment with different hyperparameters)
    - Confusion matrix
    - Training and Validation Performance Graphs of accuracy and loss per epoch. (Do not expect your accuracy will be above 95%)

You can use Amazon AWS / Microsoft Azure / Google Colab for this assignment. Following links may be helpful.

- Getting Started With Google Colab
- Google Colab Free GPU Tutorial

Extra credit: Use of tensor board for displaying graphs of performance measures and use of cyclic learning rate.