Waste Classification using Deep Learning

Khatuna Kakhiani

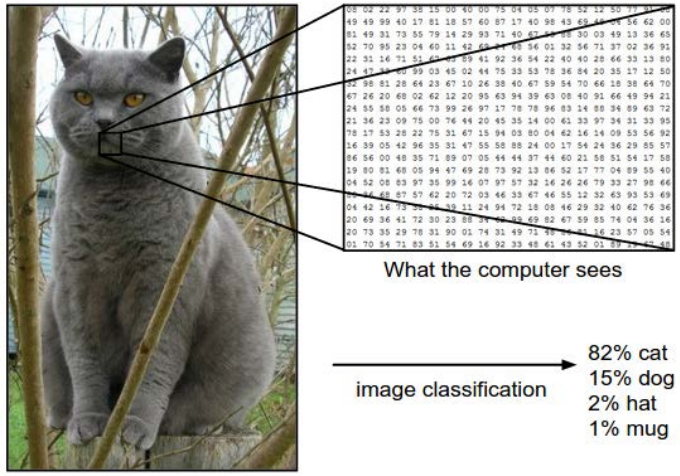
Outline

- Recap of CNN
- Application with real data

Convolutional Neural Network

CNN

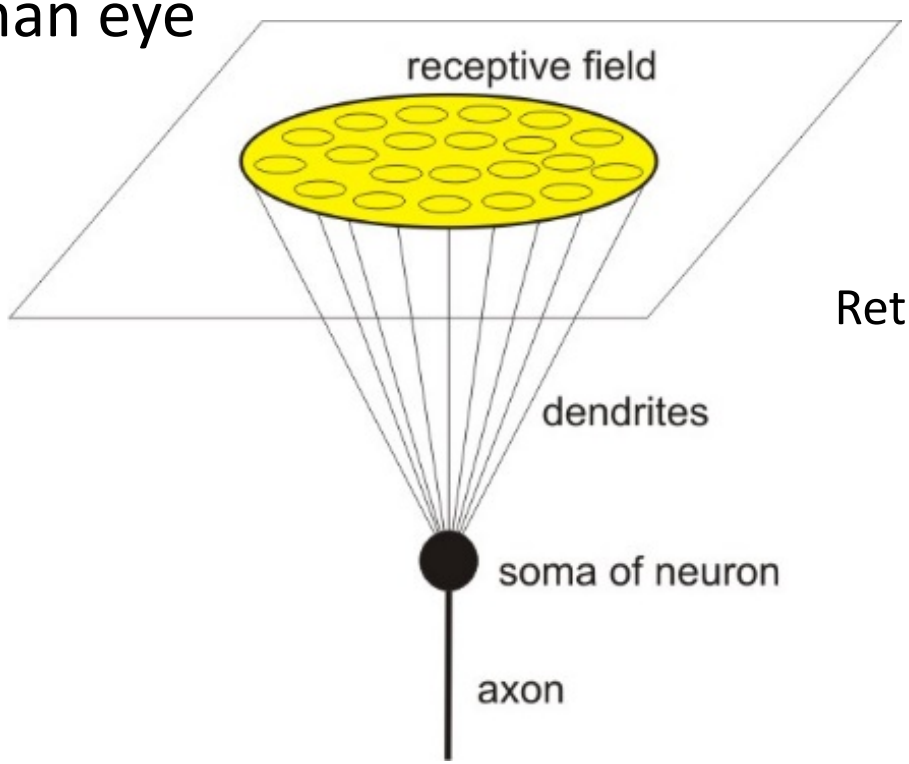
- Human perception is very accurate
- Computers see images as 2D arrays of pixels
- Algorithms need to be trained on lots of images



Source: http://cs231n.github.io/classification/

CNN

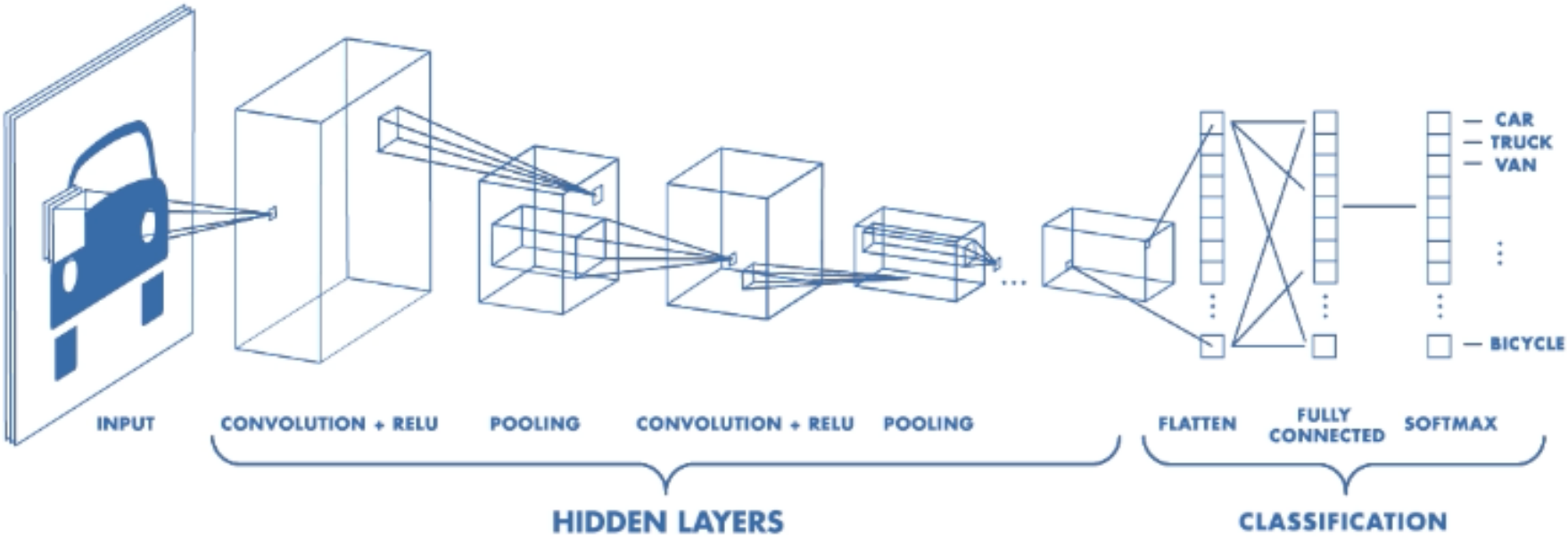
CNN mimics human eye



Retina

A a single sensory neuron's receptive field.
Source: http://cs231n.github.io/convolutional-networks/

Components of CNN



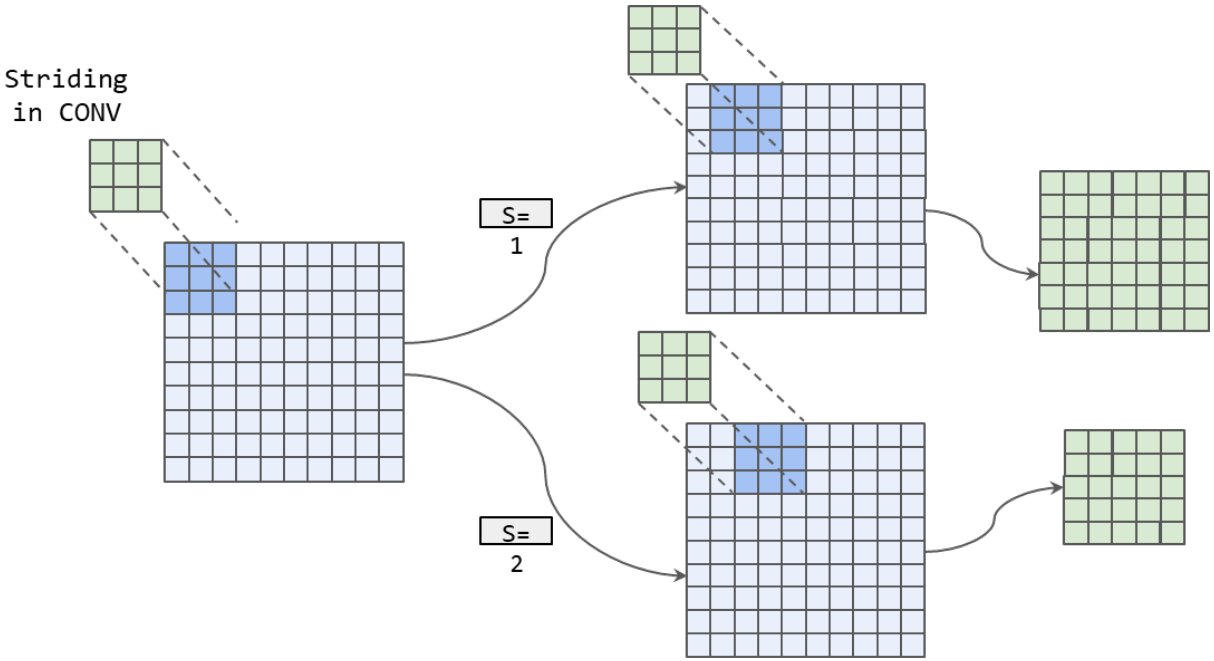
Source: https://www.mathworks.com/videos/introduction-to-deep-learning-what-are-convolutional-neural-networks--1489512765771.html

CNN

- Convolution
- Pooling operation
- Activation functions
- Dropout
- Backpropagation

Convolution Operation

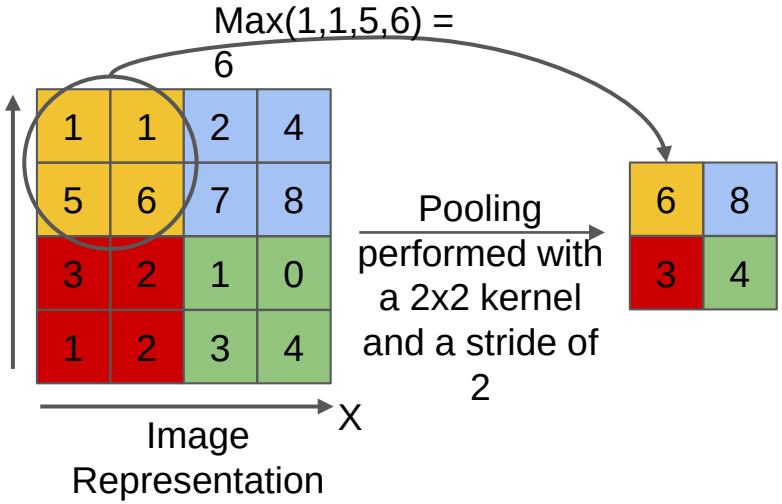
- Combination of 2 functions to produce a third function
- Input, kernel (e.g. 3x3), feature map (output)
- Stride kernel across the input and compute matrix multiplication to produce output



Source https://github.com/dair-ai/ml-visuals

Pooling Operation

- Summarizes the output of a region
- Helps reduce the effect of invariants (small changes to the input)
- Max vs mean-pooling

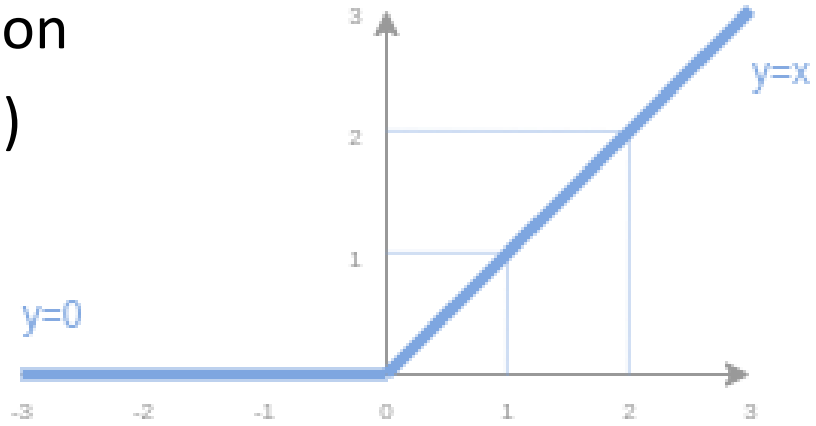


Activation Functions - Sigmoid

- Non-linear transformation of input to allow complex tasks
  - Sigmoid = squish weighted sum of neurons into range (0,1)
    - Problem: vanishing gradient (smaller) and sparsity (dense neurons)
    - Solution: ReLU (constant value and sparse activations)

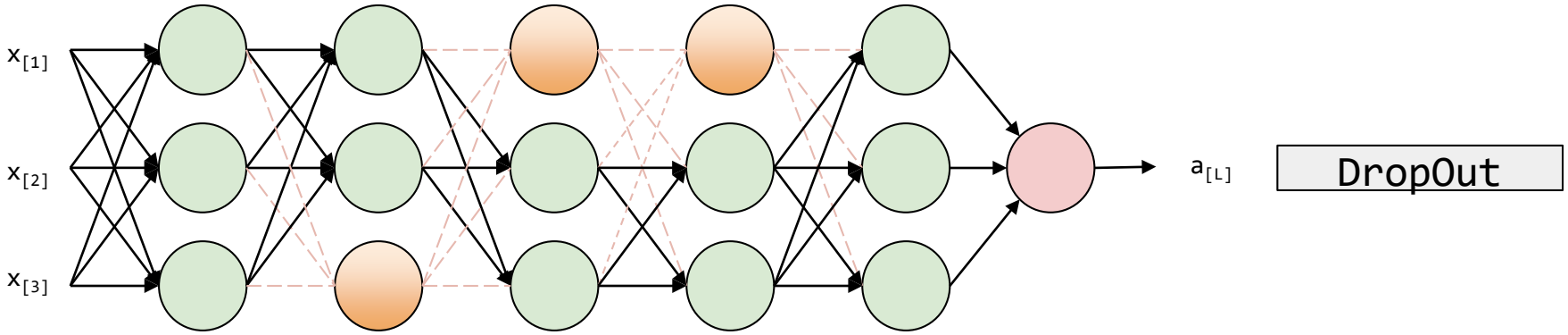
Activation Functions - RELU

- Rectified linear unit activation function
- Fast convergence (sparse activations)
- Constant values
- Negative values do not get activated
- Dying ReLU: neurons get stuck at 0
  - Can lead to model not learning
  - Solution: Leaky ReLU w/ small slope for negatives



Dropout

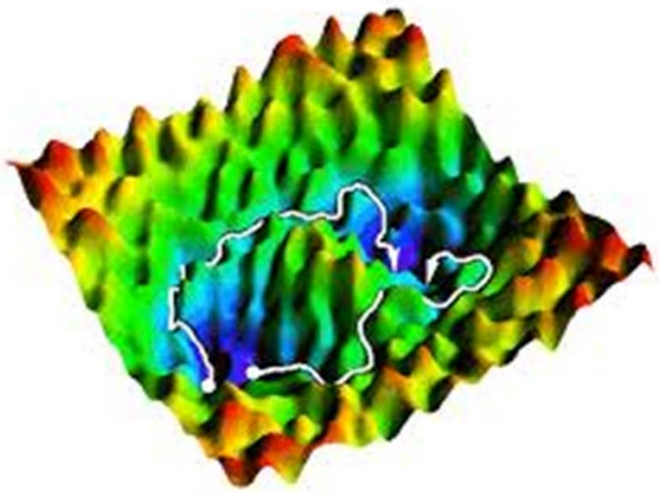
- Regularization technique
- Prevents overfitting making memorization difficult
- Method: randomly throw activations away (e.g. p=0.5),
- Early dropout coupled with RELU – preventive



Source: https://github.com/dair-ai/ml-visuals

Backpropagation

- Based on the **gradient descent** algorithm
- Minimization of error through optimization of function
- Moving in the opposite direction of the gradient
- Backpropagation adjusts parameters (backward) given the error



Application

- Problem
- Data
  - Description
  - Pre-processing
- HPC
  - Vulcan
  - Practical

Problem

- Huge amount of plastic waste
- Impact to wildlife and on humans
- can we help build robots to classify waste automatically?

Source: https://www.onegreenplanet.org/

Data

- TACO dataset- http://tacodataset.org/
- Contains images of waste in the wild
- User annotated
- Size: 1500 annotated images
- 60 categories of objects and 28 super categories


Pop tab
Drink can


Single-use carrier bag


Plastic straw


Plastic Film


Paper cup
Plastic lid


Disposable plastic cup


Other plastic wrapper
Glass bottle


Plastic Film
Meal carton


Foam food container


Plastic drink bottle
Plastic bottle cap
Wrapping paper


Drink can

Data Pre-Processing

Input parameters:

- Number of images
- Image height, image width
- Number of channels - 3 channels of data (RGB)
- Pixel range [0,255]

Data Pre-Processing

- Convert to gray scale (dimensional
- Downscale to fixed size
- Scale the pixel values to the range [0,1]

Practical

Vulcan - log in and set up

PRACTICAL

- Log in into your course account

  > ssh …

- Recover your workspace

  > ws_list

  > cd $MYSCR (**ws** path is stored/full path)

Vulcan - log in and set up

PRACTICAL

- Copy data, scripts and python codes for the litter classification (**TC directory**) to your workspace:
  > scp -r /lustre/nec/ws2/ws/hpckkakh exchange20200708T232151/TC ./

- In **TC** directory unpack **data.tar.gz** archive:
  > tar -zxvf data.tar.gz

https://github.com/khatuka31/Waste_Classification

Vulcan – explore

PRACTICAL

Explore main directory (**TC**) with subdirectories:

- **data*** - for Litter classification (*.json & co)
- **exc** - (python ***.py** and job submission scripts ***.pbs**)
- **solutions** - are included (do not have to use it)

[*] http://tacodataset.org (dataset)

Vulcan - log in and set up

PRACTICAL

- Log in into **exc** directory
- Execute

  > python test.py

```
Traceback (most recent call last):
  File "test.py", line 5, in <module>
    import tensorflow as tf
ImportError: No module named tensorflow
```

Vulcan

PRACTICAL ?

- New session (log in, new job, compute node)
  - default HPC* environment
- Check for modules

  > **module avail**

  > **module list**

* https://kb.hlrs.de/platforms/index.php/NEC_Cluster_Software_Environment (vulcan)

Vulcan

PRACTICAL

- Load modules

> **module load python-site/3.6**

> **module load /opt/hlrs/unsupported-modulefiles/tensorflow**

Vulcan – Python SCRIPT

PRACTICAL

- Check python version
  > which python
  → /opt/…/full path of python-site/…/**3.6.6**/

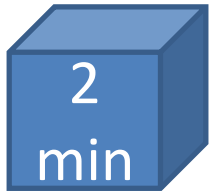
- Execute test python script
  > python **test.py**
  → Tensorflow version: 1.14.0
  Numpy version: 1.14.5

Vulcan – JOB SCRIPT

PRACTICAL

- Correct errors in batch job script

job_test_queues.pbs




```
#!/bin/bash
#PBS -N mytest
#PBS -l select=-----:node_type=------
#PBS -l walltime=00:00:00

cd $PBS_O_WORKDIR

# Load modules
----------------------
type python

python ------------ > test.out
```

Directives

Loading software

User scripting

(1 node, queue: test, max: 3 min)

Vulcan – JOB SCRIPT

PRACTICAL

job_test_queues.pbs

```
#!/bin/bash
#PBS -N test
#PBS -l select=1:node_type=hsw
#PBS -l walltime=00:03:00

cd $PBS_O_WORKDIR

# Load modules
module load python-site/3.6
module load /opt/hlrs/unsupported-modulefiles/tensorflow

type python
python test.py > test.out
```

Directives

Loading software

User scripting

Vulcan – SUBMITT BATCH JOB

PRACTICAL

- Submit job:
  - **qsub -q queue job_test_queues.pbs**
  - **qsub job_test_queues.pbs**

- Check status:

> qstat **# status of your job**
> qstat –q **# available queues**

Practical on Vulcan

Waste classification using Deep Learning

Khatuna Kakhiani
HLRS, Universität Stuttgart Online course: Deep Learning and GPU programming using OpenACC
17.07.2020

Waste sorting is a more complex task than just assigning the material labels: Plastic, Glass, Metal, Paper or Composite. In this example we will learn litter clasification using a small dataset[1]. Litter consists of waste products that have been discarded incorrectly, without consent, at an unsuitable location.

In [1]:
```
#Import libraries and modules

import warnings
warnings.filterwarnings('ignore',category=FutureWarning)    # ignores warnings about future version of numpy

#For JSON data
import json

#For interacting with operating system
import os

#For copying files
import shutil
```

Vulcan – SUBMITT BATCH JOB

PRACTICAL 1 – Preprocess.py

Please complete **tasks** in code

- Load annotations: **annotations.json**
- Explore annotations dictionary
- Create new **python dictionary**
- **Split data** into training, validation, and test
- Preprocess: resize, RGB, gray scale
- Create the **numpy arrays** of images and labels
- Submit job step by step

✓ **Easy solution: /solutions/work_data_2/*. npy**

Vulcan – SUBMITT BATCH JOB

PRACTICAL 2 – train.py

Please complete **tasks** for model training

- Load saved numpy arrays (3 images, 3 labels)
- Summarize training, validation, and test data.
- Normalize, scale, experiment
- **Configure model –** explore **keras.Sequential**
- During experiment limit number of epochs

✓ **Hint: /solutions/saved_model/my_model**

Requirements and libraries

For JSON data
- import json

For interacting: system/files
- import os
- import shutil

For visualization
- TensorBoard
- Pillow
- Matplotlib

For modeling
- import **tensorflow** as tf
- from tensorflow import keras

For vector/array operations
- import **numpy** as np
- from numpy import asarray
- from random import sample
- import random
- from random import shuffle
- import math