Introduction to Machine Based Image Processing & Deep Learning

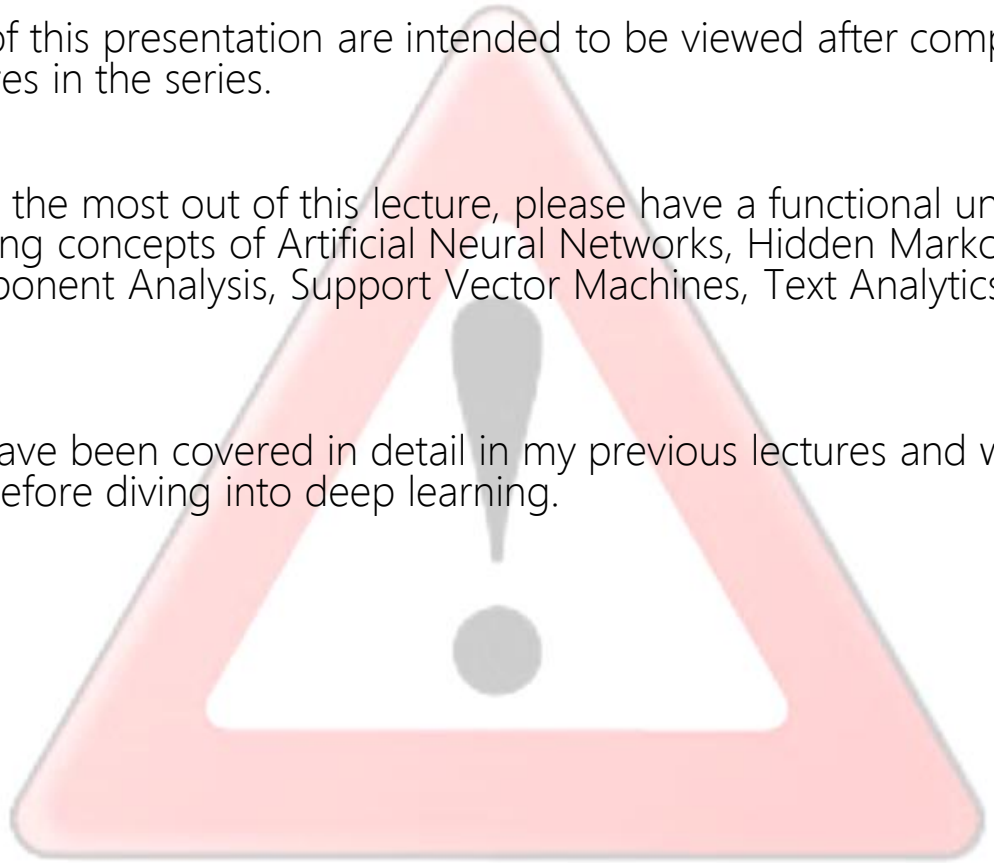
Presented by: Derek Kane

Overview of Topics

- Image Processing for Machine Learning
- Practical Application Example
  - Where is Waldo?
  - Test Grading
- Introduction to Deep Learning
- Additional Modern Architectures
  - Restricted Boltzmann Machines
  - Deep Belief Networks
  - Deep Boltzmann Machines
  - Convolutional Neural Networks
- Practical Application Example
  - MNIST Hand Writing Recognition with Deep Learning

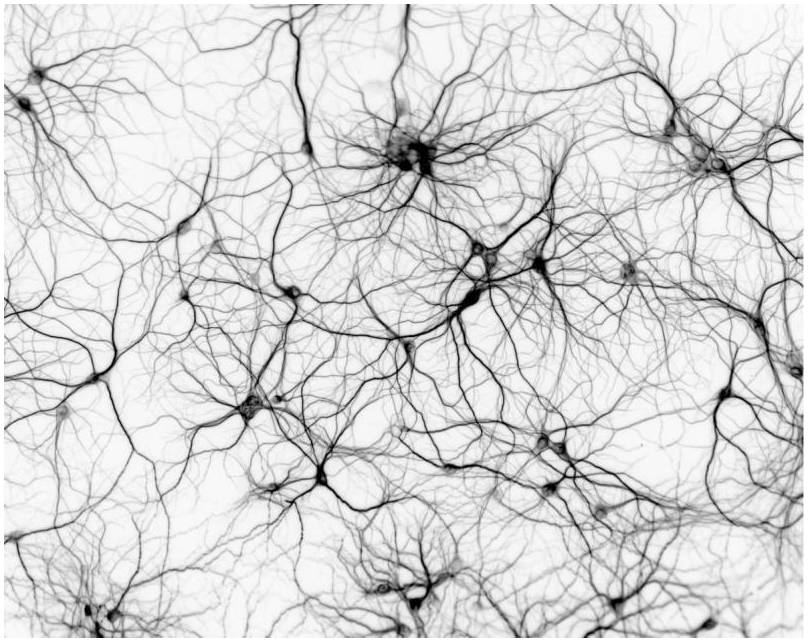
Preface

- The contents of this presentation are intended to be viewed after completing the previous lectures in the series.
- In order to get the most out of this lecture, please have a functional understanding of the underlying concepts of Artificial Neural Networks, Hidden Markov Models, Principal Component Analysis, Support Vector Machines, Text Analytics, and Clustering.
- These topics have been covered in detail in my previous lectures and would be a good primer before diving into deep learning.



Structure of Presentation

- In previous lectures we have discussed the topics of machine learning & artificial intelligence.
- A very powerful machine learning tool in our toolkit called, Artificial Neural Network or ANN, mimics the biological process of our brain.
- This machine implementation of neurons firing (perceptron) can create powerful models which identifies the underlying structure of the phenomenon studied.
- Our discussion later will build on these foundational ideas into a more robust version of ANN's that are referred as Deep Neural Network's or Deep Learning.



Structure of Presentation

- When I think about how my brain works and relate this back to Machine Learning and AI, I cant help but to think about the 5 senses we perceive.
- While all of the senses plays a significant role in the human experience, the sense of sight seems to dominate my personal experience.
- I don’t want to diminish the importance of the other senses but if we can master the extraction of useful information from visual data, we will be poised to advance artificial intelligence significantly.

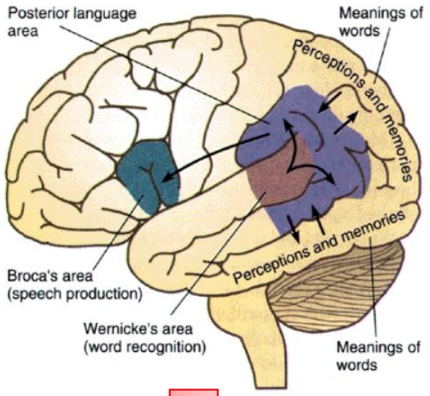
Transforming the Senses

Here is an example of dictating spoken text which hopefully showcases why I believe this to be true:

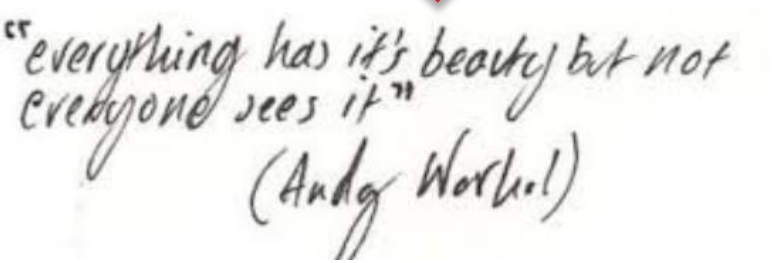
Sound is produced

The auditory system interprets the sound wave

The brain processes the sound wave into language and text



A picture is created which contains information in the form of handwritten text.



Transforming the Senses

- If we think about modern technological advances, we can directly translate the audio sound into a text string.

Sound is produced

Audio signal is processed in a computer.

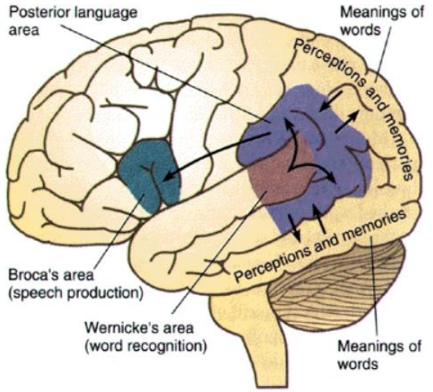
Signal is converted into unstructured text.

Transforming the Senses

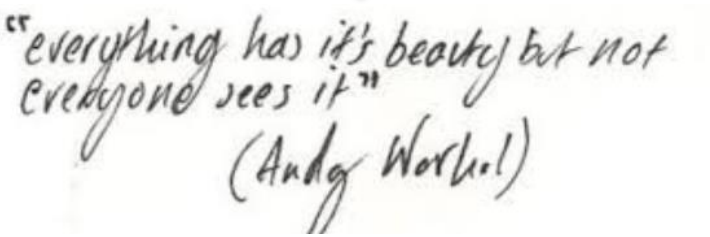
- After the audio has been turned into text, the results are then displayed back to the dictator.

The text string is shown to the dictator.

The visual signal is processed by the brain.



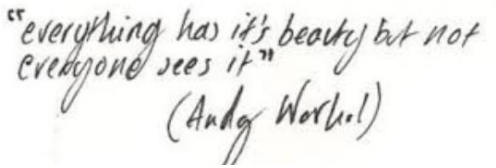
A picture is created which contains information in the form of handwritten text.



Transforming the Senses

- Lets now flip this idea on its side and focus on extracting information from the handwritten text into a machine learning model.

Visual image of text



Conversion to computer text data string

Prepare data for analytics*

Gender	already	also
Male	0	0
Male	0	0
Male	1	2
Male	0	1
Female	0	0
Male	0	0

Note: Please refer back to my lecture on text analytics for details on how to prepare the data for analytics.

Transforming the Senses

- This simple example shows that we can effectively combine information that has been encoded in a sound form into a visual format.
- The visual representation of text data can then be deconstructed into structured strings of text which then can be processed with our full arsenal of machine learning techniques.
- Advances are currently being explored to integrate the other senses (Ex. smell) into a single computing ecosystem.

Image Processing

- Now that we have the basic mechanics outlined lets now focus on how we can actually deconstruct visual images to be processed in machine learning applications.

- This is an emerging field in machine learning and can be extremely difficult. Take the images below as an example. We can clearly identify the text in the image however this is challenging for a computer to detect accurately.

Image Processing

- To tackle these challenges in a practical sense, lets begin assessing and analyzing some image data.
- We will be drawing from the packages in R: EBImage and Raster for this section of the analysis and will focus more on the mechanics of manipulating images.
- The EBImage package allows for us to work with image formats of JPEG, GIF, and TIFF.

```
library("EBImage")
f = system.file("images",
"sample.png", package="EBImage")
img = readImage(f)
display(img)
```

- The display function will show a black and white image with pixel densities ranging from 0 (black) to 1 (white).

Image Processing

- Color images or images with multiple frames can also be read with readImage.

```
library("EBImage")
imgc = readImage(system.file("images",
"sample-color.png", package="EBImage"))
display(imgc)
nuc = readImage(system.file('images',
'nuclei.tif', package='EBImage'))
display(nuc)
```

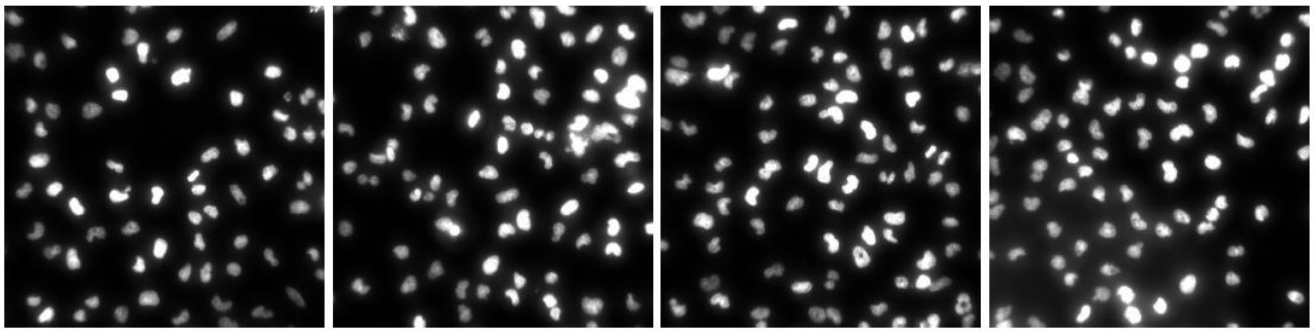


Image Processing

- We can also write off processed images back into a file:

```
library("EBImage")
writeImage(img,  'img.jpeg', quality=85)
writeImage(imgc, 'imgc.jpeg', quality=85)
```

- The package EBImage uses the class Image to store and process images. Images are stored as multi-dimensional arrays containing the pixel intensities. All EBImage functions are also able to work with matrices and arrays.

```
  print(img)
Image
  colorMode    : Grayscale
  storage.mode : double
  dim          : 768 512
  frames.total : 1
  frames.render: 1

imageData(object)[1:5,1:6]
          [,1]      [,2]      [,3]      [,4]      [,5]      [,6]
[1,] 0.4470588 0.4627451 0.4784314 0.4980392 0.5137255 0.5294118
[2,] 0.4509804 0.4627451 0.4784314 0.4823529 0.5058824 0.5215686
[3,] 0.4627451 0.4666667 0.4823529 0.4980392 0.5137255 0.5137255
[4,] 0.4549020 0.4666667 0.4862745 0.4980392 0.5176471 0.5411765
[5,] 0.4627451 0.4627451 0.4823529 0.4980392 0.5137255 0.5411765
```

Image Processing

- As matrices, images can be manipulated with all R mathematical operators. This includes + to control the brightness of an image, * to control the contrast of an image or ^ to control the gamma correction parameter.

```
library("EBImage")

img1 = img+0.5
img2 = 3*img
img3 = (0.2+img)^3
```

Other operators include [to crop images, < to threshold images or t to transpose images.

```
img4 = img[299:376, 224:301]
img5 = img>0.5
img6 = t(img)
print(median(img))
```

Image Processing

- Images with multiple frames are created using combine which merges images.

```
library("EBImage")
imgcomb = combine(img, img*2,
img*3, img*4)
display(imgcomb)
```

Color Management

- As an example, the color image imgc is a 512x512x3 array, with a colormode slot equals to Color. The object is understood as a color image by EBImage functions.

```
  print(imgc)
Image
  colorMode    : Color
  storage.mode : double
  dim          : 768 512 3
  frames.total : 3
  frames.render: 1

imageData(object)[1:5,1:6,1]
          [,1]      [,2]      [,3]      [,4]      [,5]      [,6]
[1,] 0.4549020 0.4784314 0.4941176 0.5137255 0.5294118 0.5529412
[2,] 0.4588235 0.4784314 0.4941176 0.4980392 0.5215686 0.5490196
[3,] 0.4705882 0.4823529 0.4980392 0.5137255 0.5294118 0.5372549
[4,] 0.4666667 0.4745098 0.5058824 0.5137255 0.5450980 0.5647059
[5,] 0.4705882 0.4705882 0.4901961 0.5137255 0.5372549 0.5647059
```

Color Management

- The function colorMode can access and change the value of the slot colormode, modifying the rendering mode of an image.
- The Color image imgc with one frame is changed into a Grayscale image with 3 frames, corresponding to the red, green and blue channels. The function colorMode does not change the content of the image but changes only the way the image is rendered by EBImage.

```
colorMode(imgc) = Grayscale
display(imgc)
```

- Above: imgc, rendered as a Color image and as a Grayscale image with 3 frames (red channel, green channel, blue channel)

- The color mode of image imgc is reverted back to Color.

```
colorMode(imgc) = Color
```

Image Filtering

- Images can be linearly filtered using “filter2”. “filter2” convolves the image with a matrix filter.
- Linear Filtering is useful to perform low-pass filtering (to blur images, remove noise, …) and high-pass filtering (to detect edges, sharpen images, …).
- Various filter shapes can be generated using makeBrush.

```
flo = makeBrush(21, shape='disc', step=FALSE)^2
flo = flo/sum(flo)
imgflo = filter2(imgc, flo)
fhi =  matrix(1, nc=3, nr=3)
fhi[2,2] = -8
imgfhi = filter2(imgc, fhi)
```

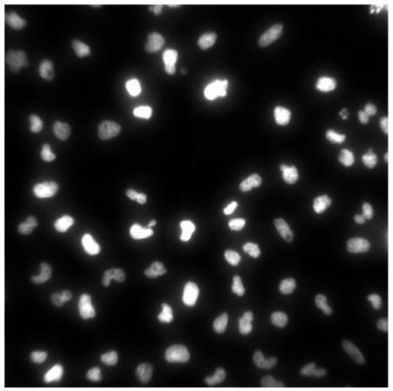
Morphological Operations

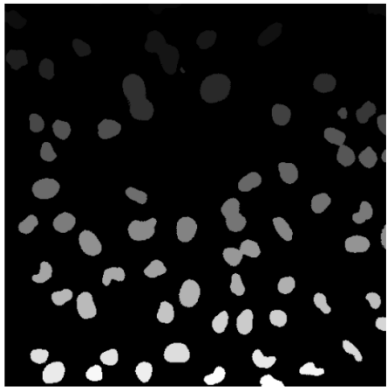
- Binary images are images where the pixels of value 0 constitute the background and the other ones constitute the foreground.
- These images are subject to several non-linear mathematical operators called morphological operators, able to erode and dilate an image.

```
library("EBImage")
ei = readImage(system.file('images',
'shapes.png', package='EBImage'))
ei = ei[110:512,1:130]
display(ei)
kern = makeBrush(5, shape='diamond')
eierode = erode(ei, kern)
eidilat = dilate(ei, kern)
```

Segmentation

- Segmentation consists in extracting objects from an image.
- The function “bwlabel” is a simple function able to extract every connected sets of pixels from an image and relabel these sets with a unique increasing integer.
- “bwlabel” can be used on binary images and is useful after thresholding.
- Since the image nuclabel ranges from 0 to the number of object they contain (given max(nuclabel)).
- This has to be divided by these number before displaying, in order to have the [0,1] range needed by display.

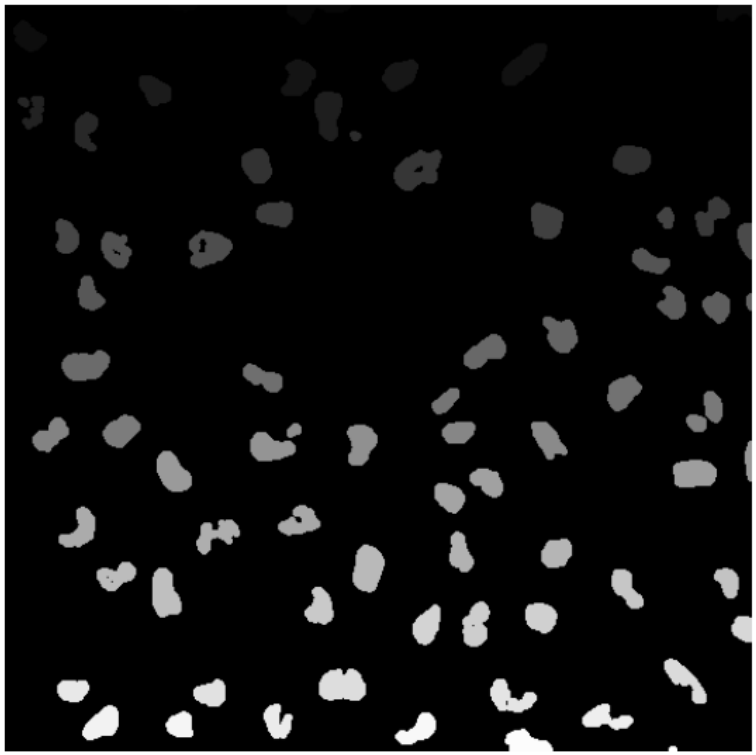




```
library("EBImage")
nuct = nuc[,,1]>0.2
nuclabel = bwlabel(nuct)
cat('Number of nuclei=', max(nuclabel),'\n')
```

```
Number of nuclei= 74
```

Segmentation



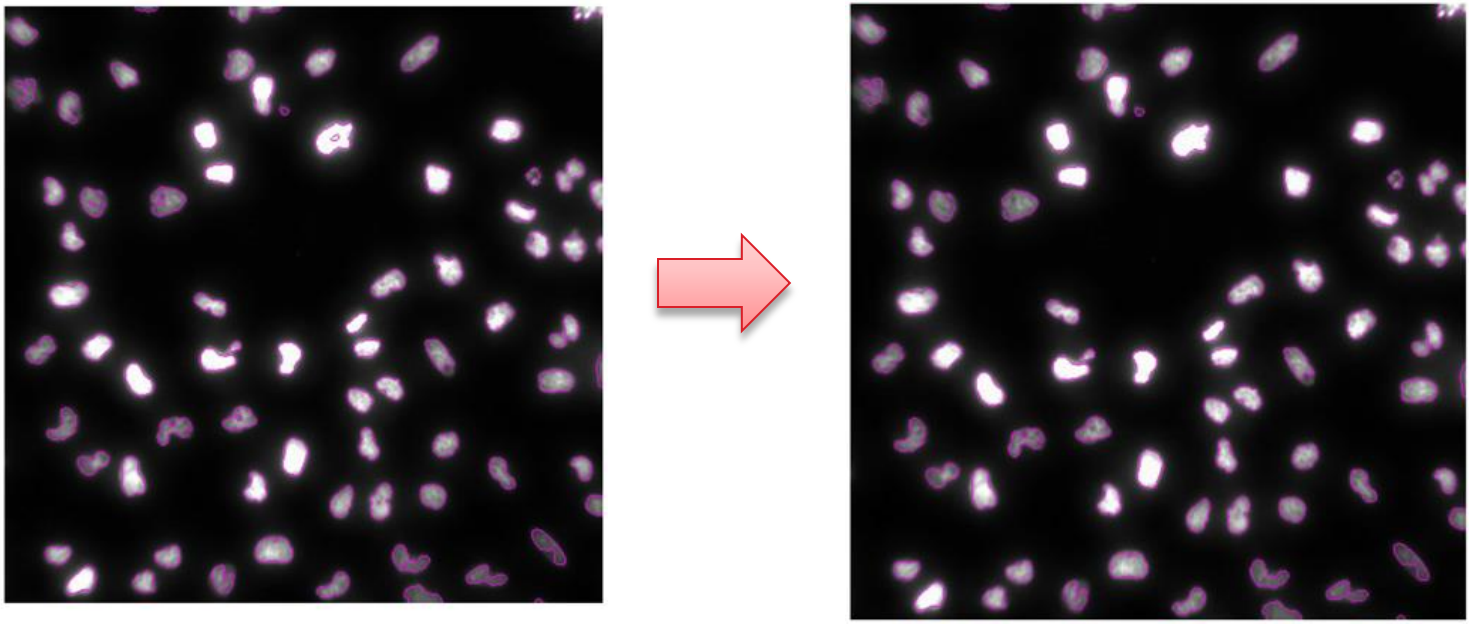
- Adaptive thresholding consists in comparing the intensity of pixels with their neighbors, where the neighborhood is specified by a filter matrix.
- The function thresh performs a fast adaptive thresholding of an image with a rectangular window while the combination of filter2 and < allows a finer control.
- Adaptive thresholding allows a better segmentation when objects are close together.

```
nuct2 =  thresh(nuc[,,1], w=10, h=10, offset=0.05)
kern = makeBrush(5, shape='disc')
nuct2 = dilate(erode(nuct2, kern), kern)
nuclabel2 = bwlabel(nuct2)
cat('Number of nuclei=', max(nuclabel2),'\n')
```

Object Manipulation

- Objects, defined as sets of pixels with the same unique integer value can be outlined and painted using paintObjects.

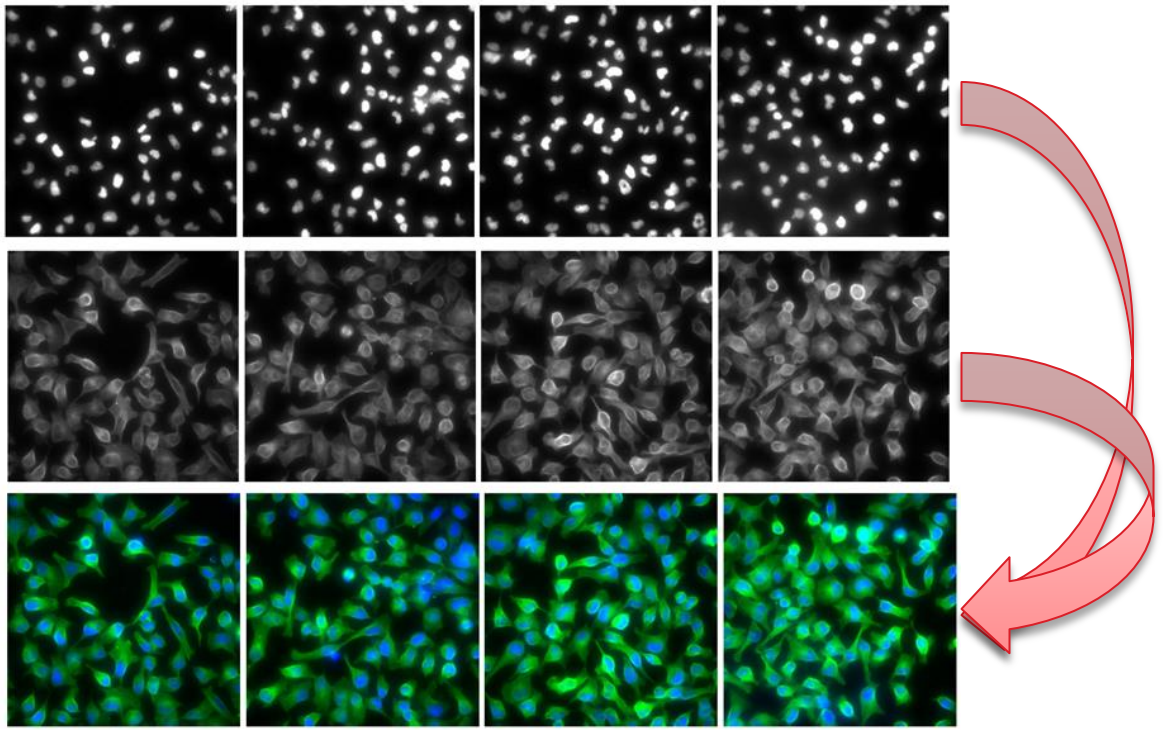
- Some holes are present in objects of nuclabel2 which can be filled using fillHull.



```
nucgray = channel(nuc[,,1], 'rgb')
nuch1 = paintObjects(nuclabel2, nucgray, col='#ff00ff')
nuclabel3 = fillHull(nuclabel2)
nuch2 = paintObjects(nuclabel3, nucgray, col='#ff00ff')
```

Image Processing

- Lets now look at an example that ties all of the concepts together and is able to perform a veroni based segmentation. Lets load some images and then combine them into a composite image.

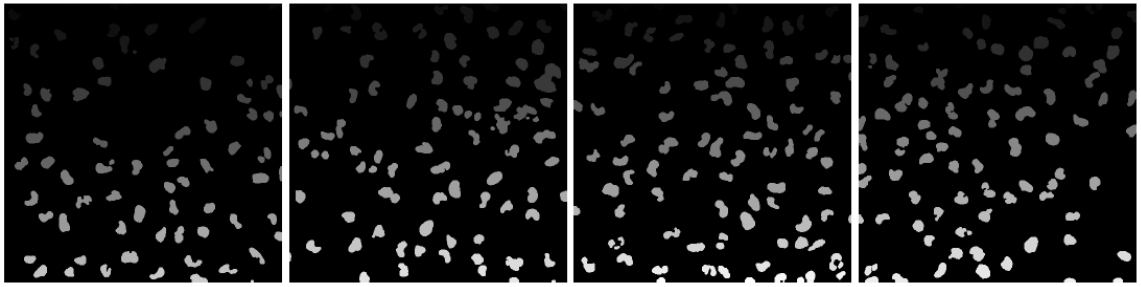


```
nuc = readImage(system.file('images', 'nuclei.tif', package='EBImage'))
cel = readImage(system.file('images', 'cells.tif', package='EBImage'))
img = rgbImage(green=1.5*cel, blue=nuc)
```

Image Processing

- Nuclei are first segmented using thresh, fillHull, bwlabel and opening, which is an erosion followed by a dilatation.

```
nmask = thresh(nuc, w=10, h=10, offset=0.05)
nmask = opening(nmask, makeBrush(5, shape='disc'))
nmask = fillHull(nmask)
nmask = bwlabel(nmask)
```



- Cell bodies are segmented using propagate.

```
ctmask = opening(cel>0.1, makeBrush(5, shape='disc'))
cmask = propagate(cel, seeds=nmask, mask=ctmask)
```

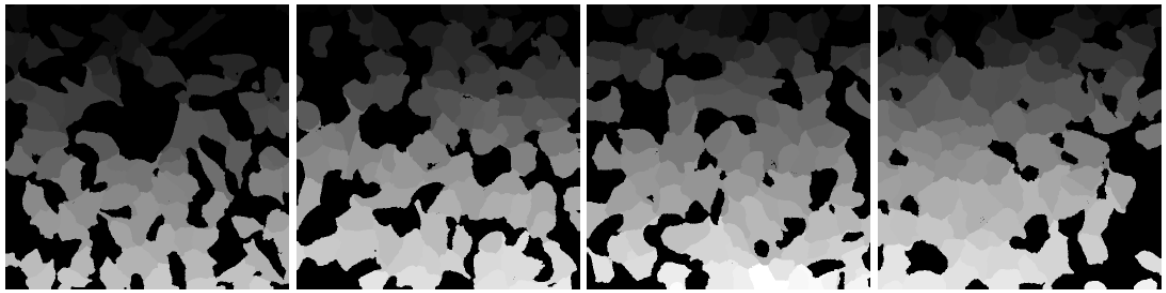
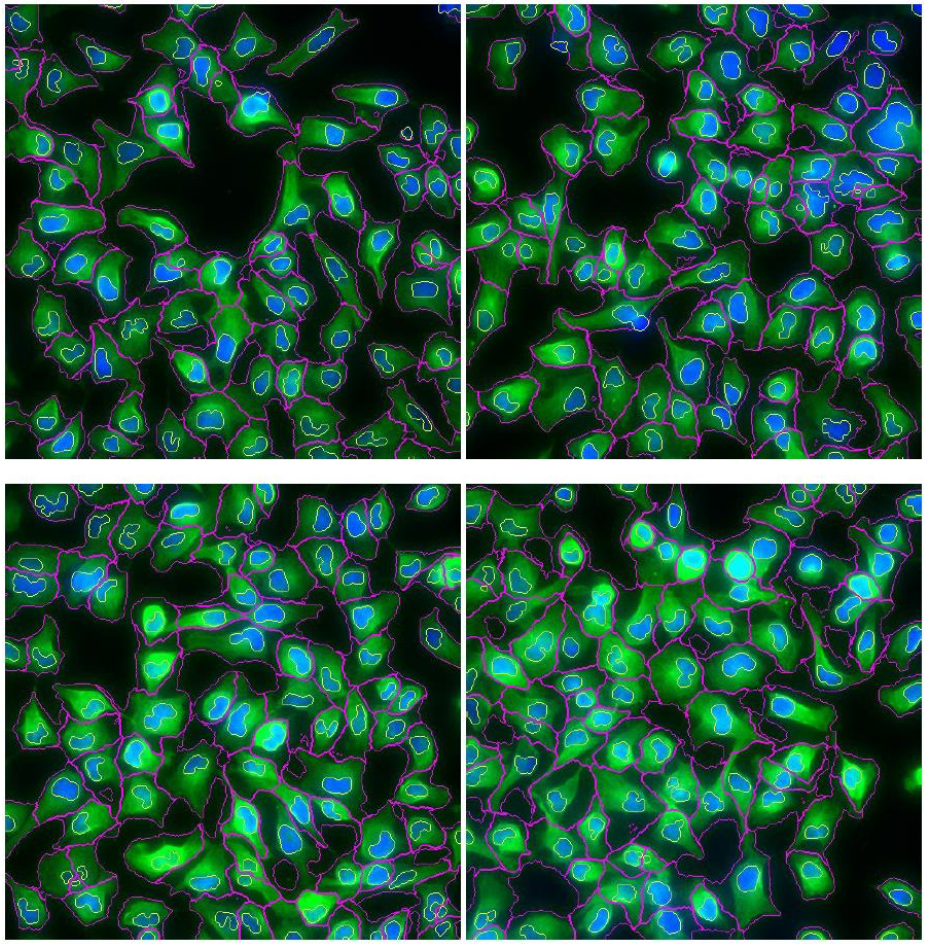


Image Processing

- Finally, the cells are outlined using paintObjects.

```
res = paintObjects(cmask, img, col='#ff00ff')
res = paintObjects(nmask, res, col='#ffff00')
```

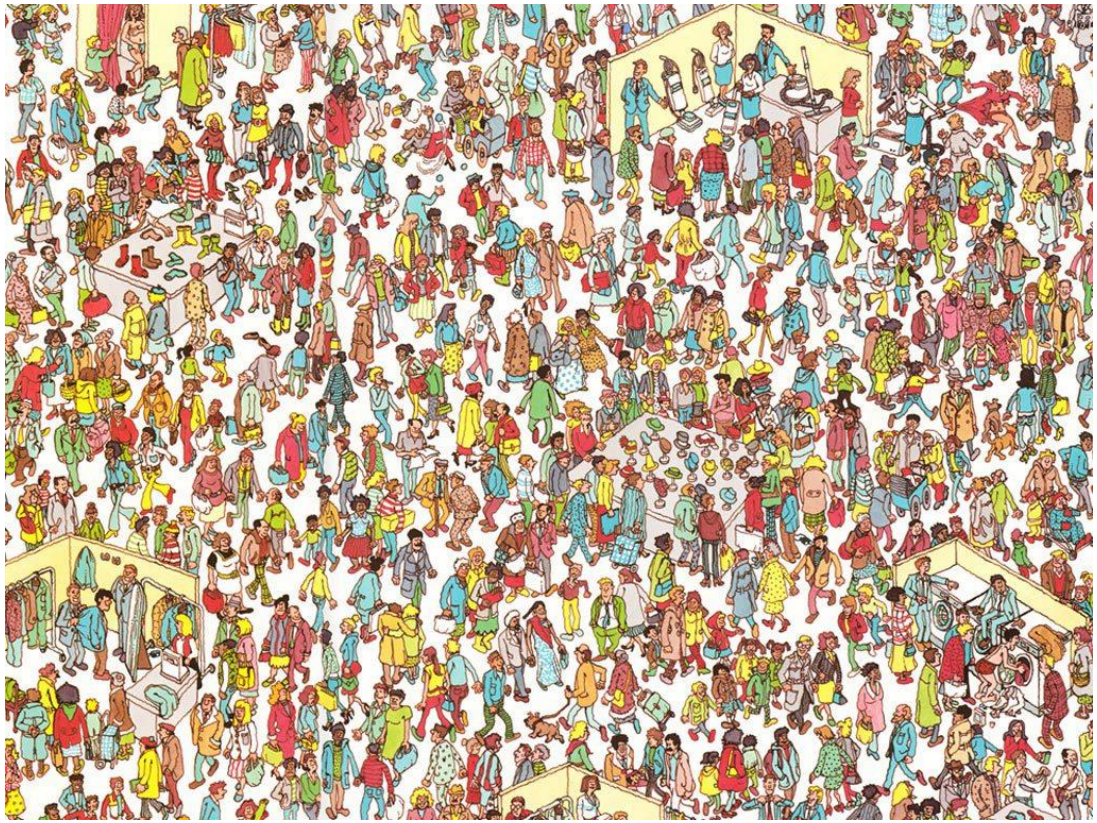


Practical Example – Where is Waldo?

Where is Waldo?

- In this example, we are going to walk through the challenges one will face when trying to employ image recognition techniques to try find the notoriously elusive Waldo.

- Where is Waldo anyway?



Where is Waldo?

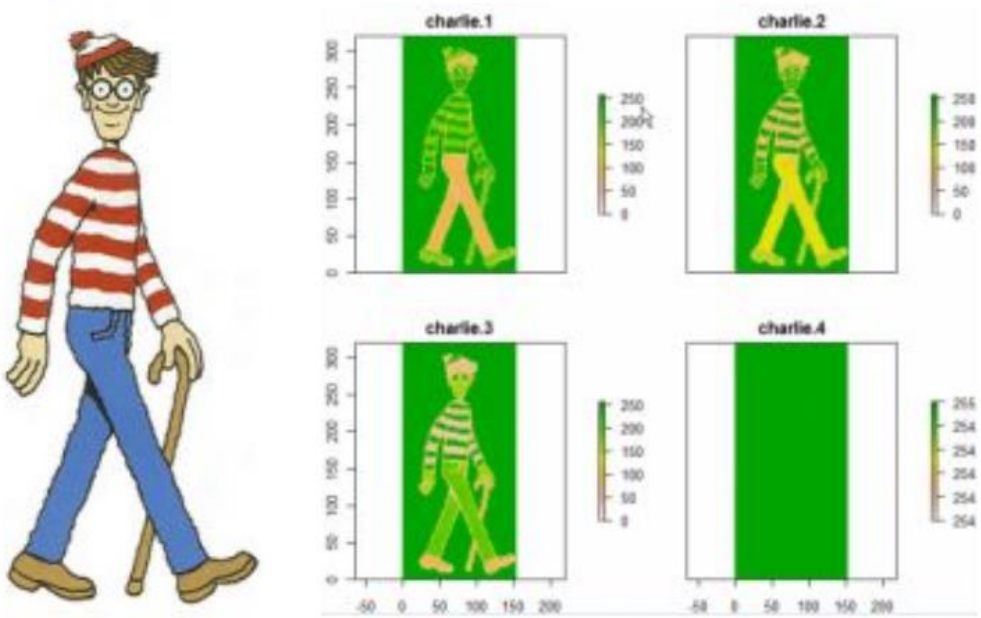
- Lets first start off by loading the raster package in R and it is possible to import the picture, and to extract the colors (based on an RGB decomposition).

```
library(raster)
waldo=brick(system.file("DepartmentStoreW.grd",
+ package="raster"))
waldo
class       : RasterBrick
dimensions  : 768, 1024, 786432, 3 (nrow,ncol,ncell,nlayer)
resolution  : 1, 1  (x, y)
extent      : 0, 1024, 0, 768  (xmin, xmax, ymin, ymax)
coord. ref. : NA
values      : C:\R\win-library\raster\DepartmentStoreW.grd
min values  : 0 0 0
max values  : 255 255 255
```

- Our strategy here is fairly simple. We will try to spot areas with white and red stripes (horizontal stripes).

Where is Waldo?

- In order to get a better understanding of what could be done, let us start with something much more simple. Here, it is possible to extract the three colors (red, green and blue).

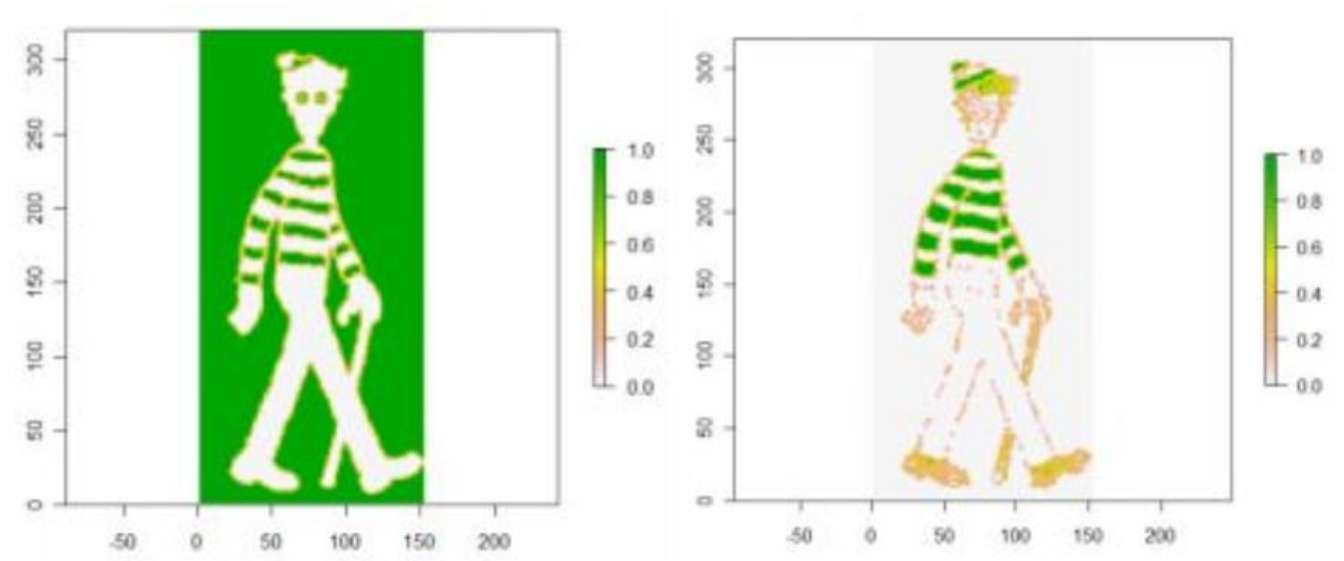


- It is possible to extract the red zones (already on the graph above, since red is a primary color), as well as the white ones (green zones on the graphs means a white region on the picture, on the left)

```
# white component
white = min(waldo[[1]] , waldo[[2]] , waldo[[3]])>220
focalswhite = focal(white, w=3, fun=mean)
plot(focalswhite,useRaster=FALSE)

# red component
red = (waldo[[1]]>150)&(max(  waldo[[2]] , waldo[[3]])<90)
focalsred = focal(red, w=3, fun=mean)
plot(focalsred,useRaster=FALSE)
```

Where is Waldo?

- Here we have the graphs below, with the white regions, and the red ones:

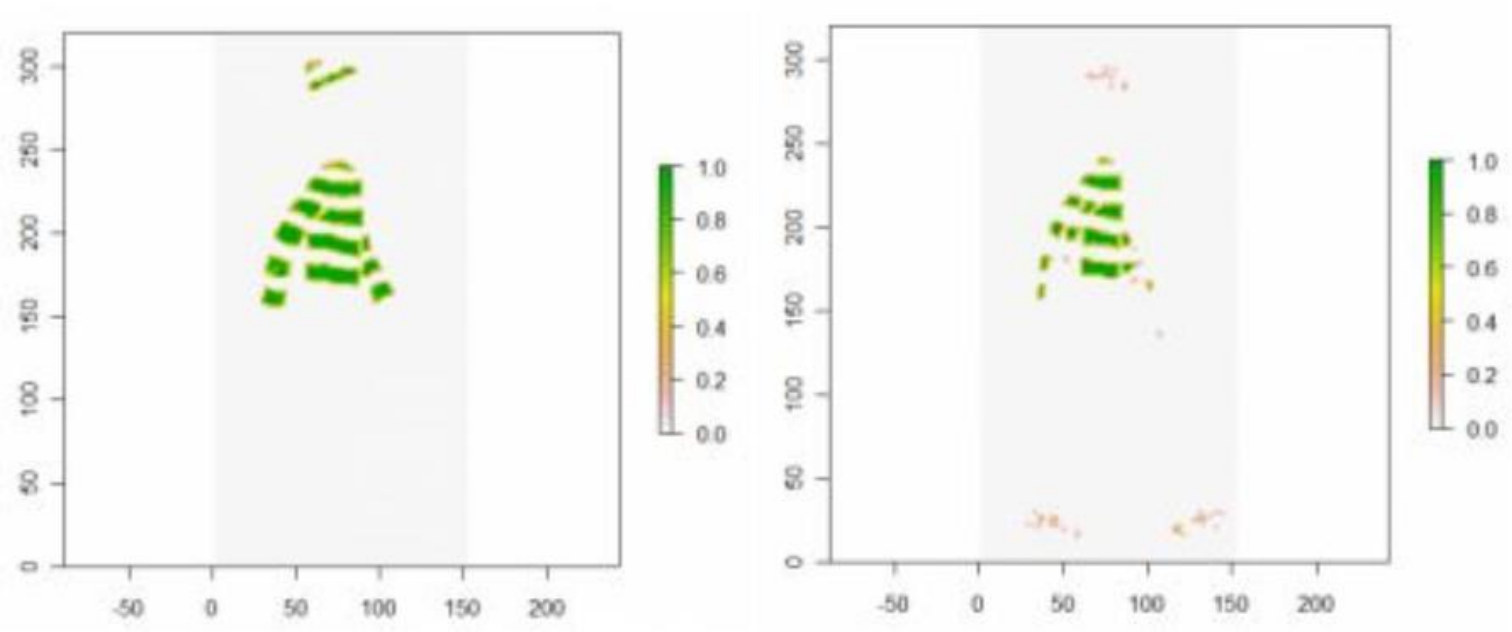


- From those two parts, it has been possible to extract the red-and-white stripes from the picture, i.e. some regions that were red above, and white below (or the reverse).

```
# striped component
striped = red; n=length(values(striped)); h=5
values(striped)=0
values(striped)[(h+1):(n-h)]=(values(red)[1:(n-2*h)]==
TRUE)&(values(red)[(2*h+1):n]==TRUE)
focalsstriped = focal(striped, w=3, fun=mean)
plot(focalsstriped,useRaster=FALSE)
```

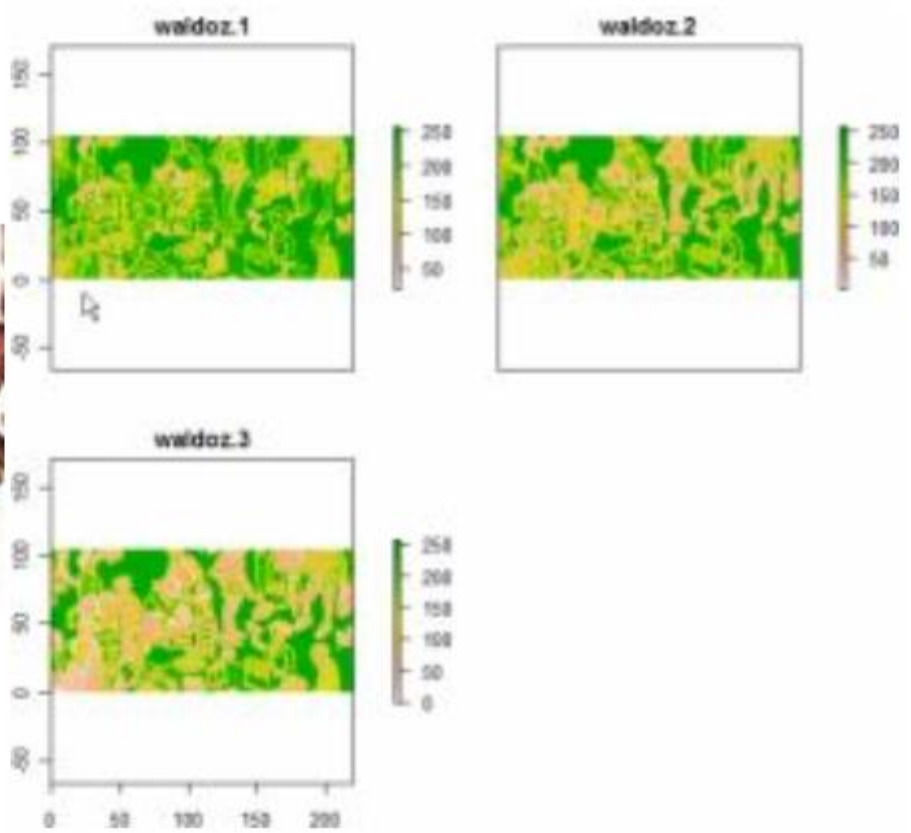
Where is Waldo?

- So here, we can easily spot Waldo, i.e. the guy with the red-white stripes (with two different sets of thresholds for the RGB decomposition).



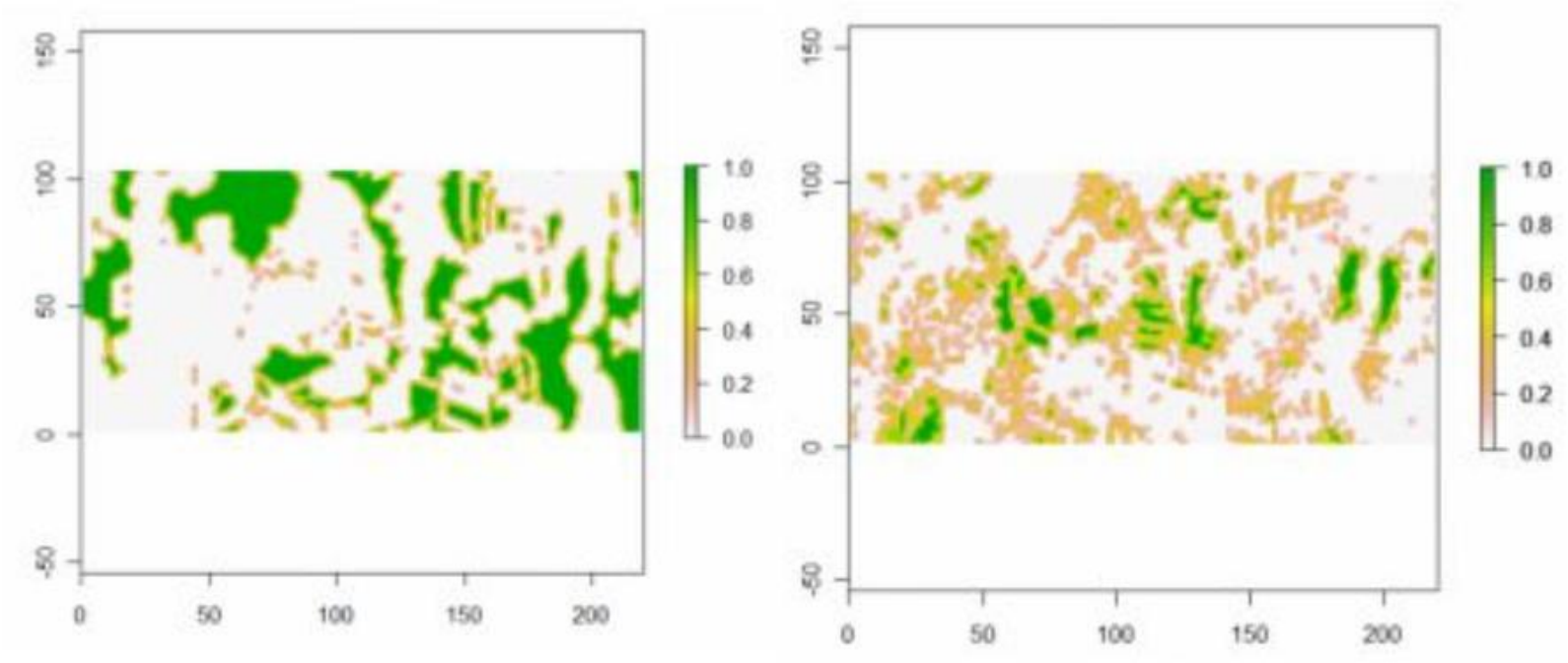
Where is Waldo?

- Let us try something slightly more complicated, with a zoom on the large picture of the department store (since, to be honest, I know where Waldo is…).



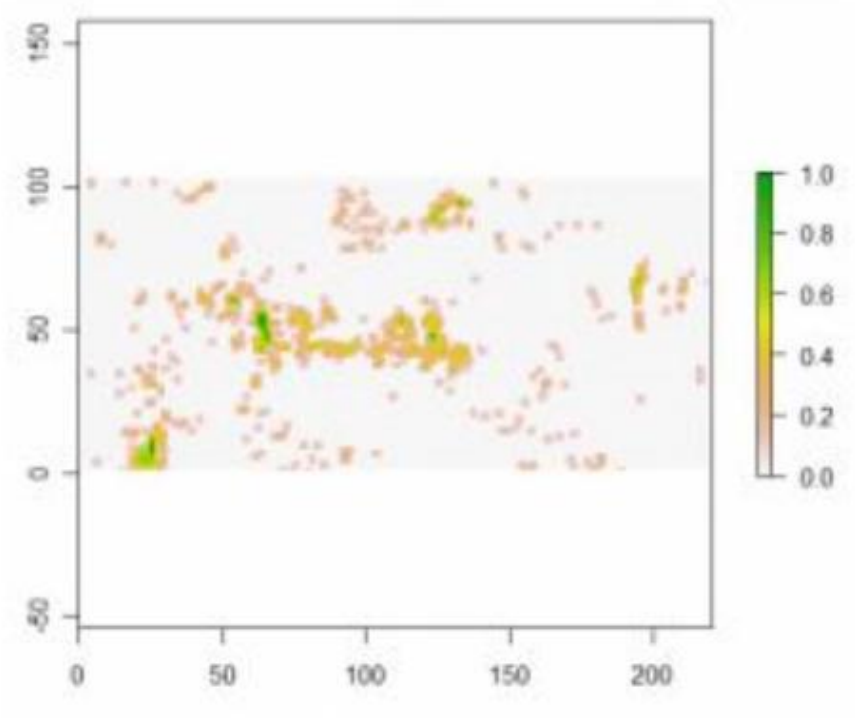
Where is Waldo?

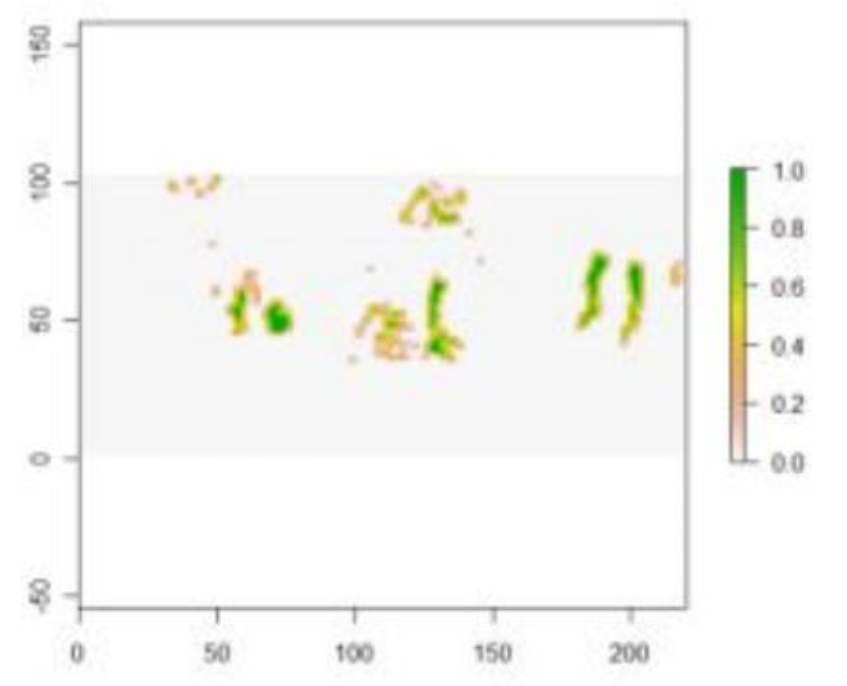
- Here again, we can spot the white part (on the left) and the red one (on the right), with some thresholds for the RGB decomposition.



Where is Waldo?

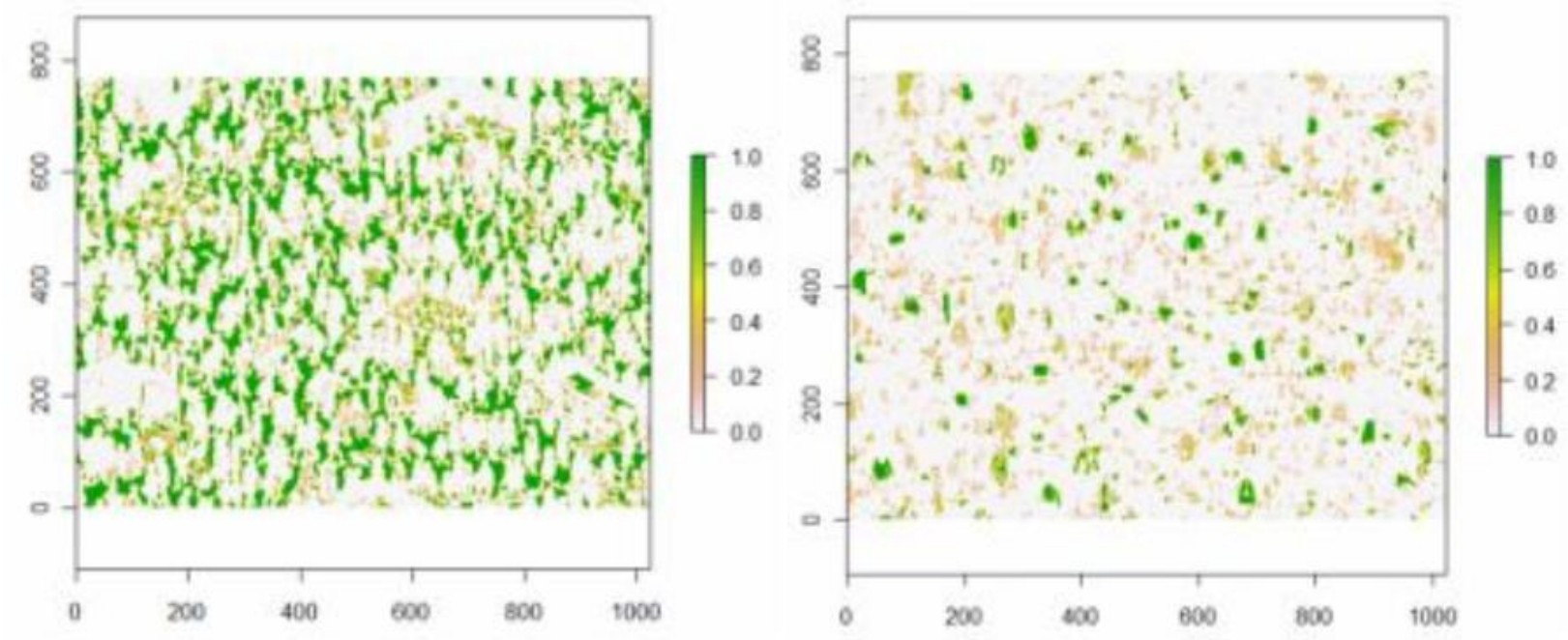
- Note that we can try to be (much) more selective, playing with threshold. Here, it is not very convincing: I cannot clearly identify the region where Waldo might be (the two graphs below were obtained playing with thresholds).





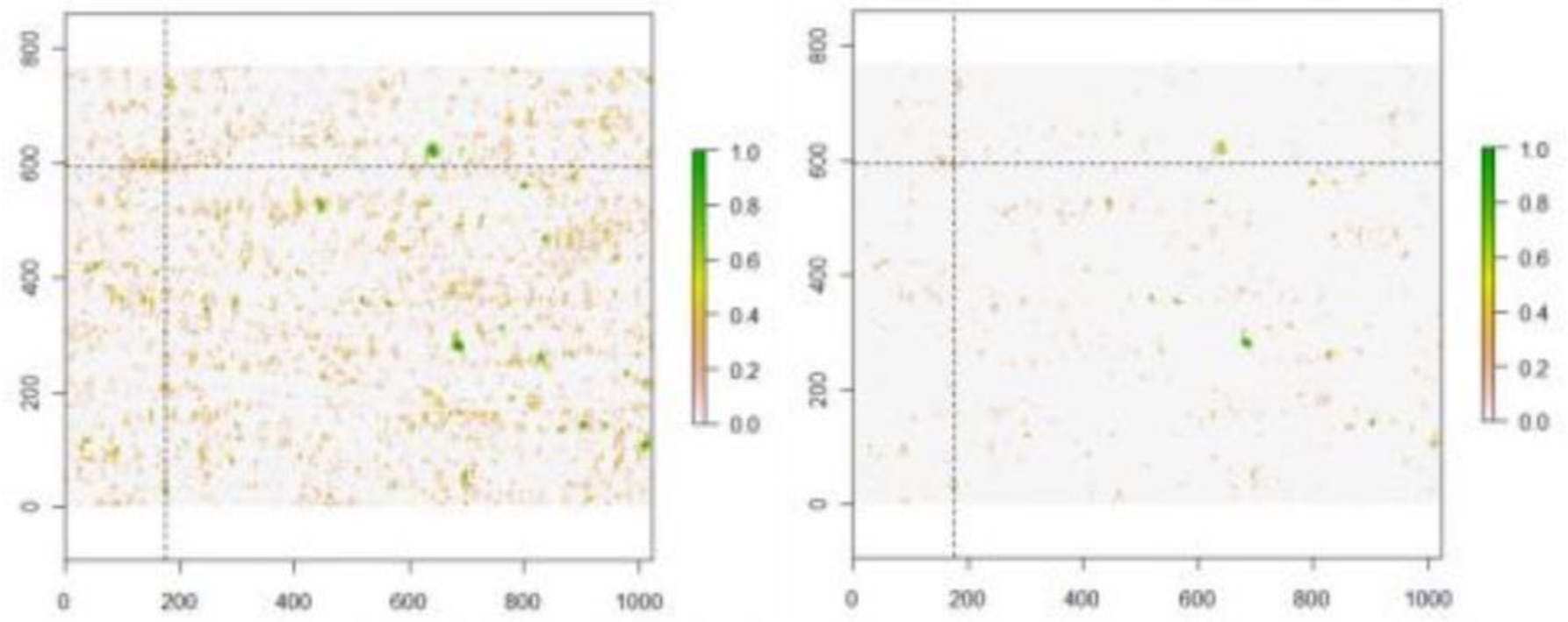
Where is Waldo?

- And if we look at the overall pictures, it is worst. Here are the white zones, and the red ones:



Where is Waldo?

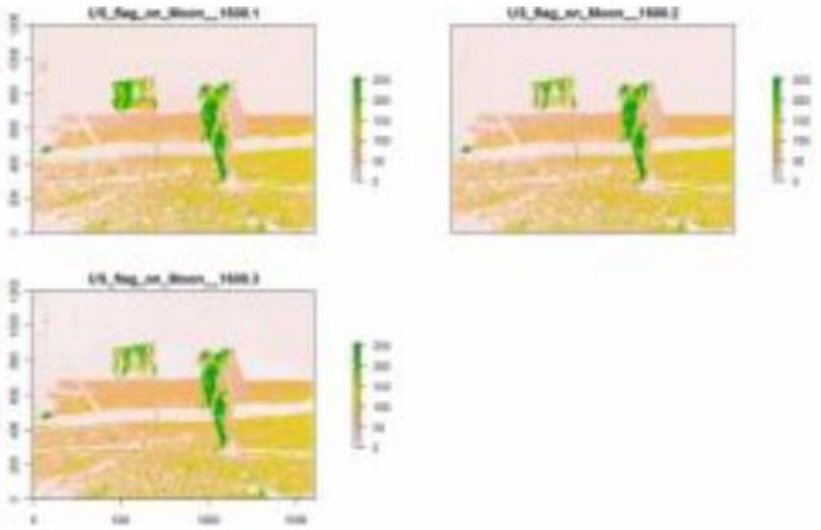
- and again, playing with RGB thresholds, I cannot spot Waldo:



- Key Takeaway: This example showcases the difficulties machine learning practitioners are faced with when trying to identify isolated objects out of the complex images. Not easy.

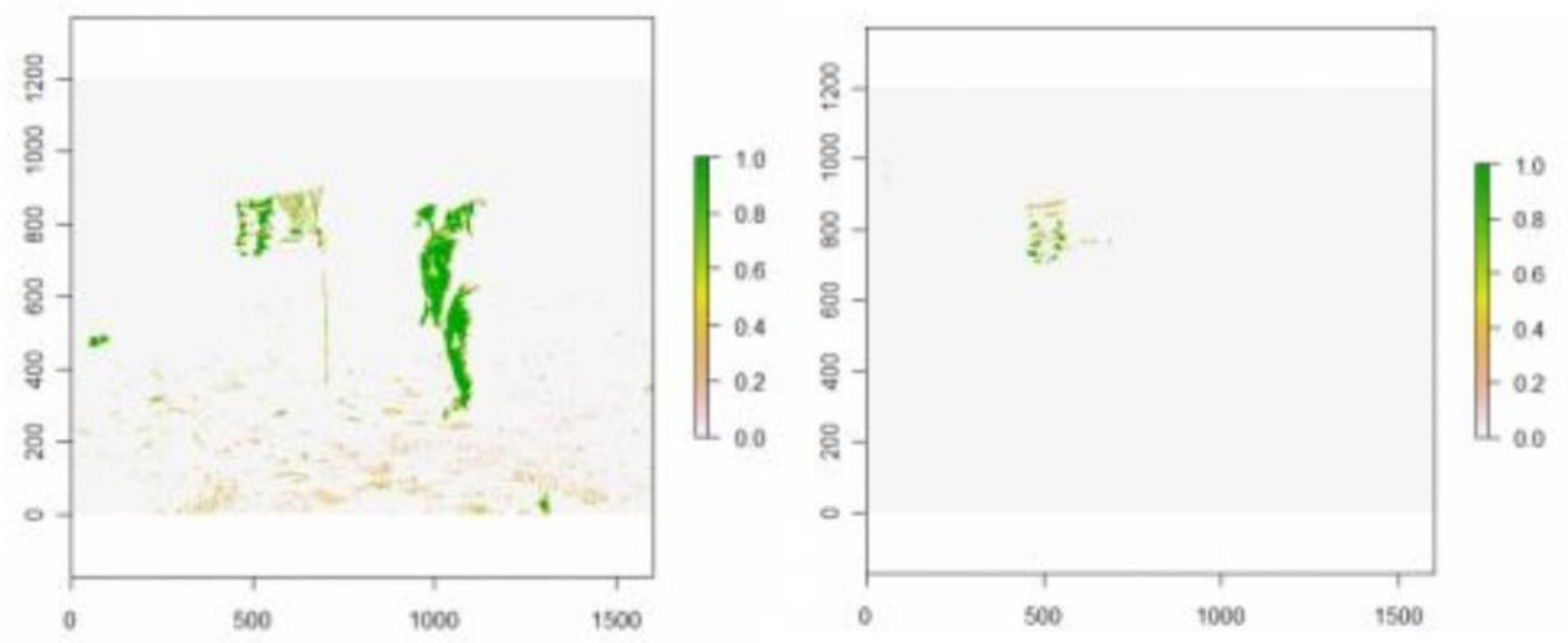
Flag on the Moon

- Let us try something more simple, like finding a flag on the moon. Consider the picture below on the left, and let us see if we can spot an American flag:



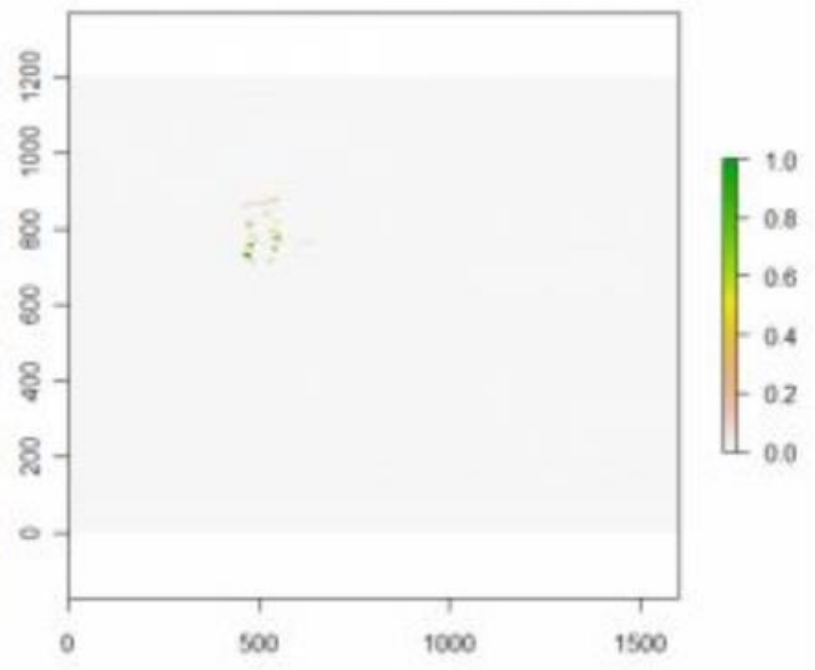
Flag on the Moon

- Again, on the left, let us identify white areas, and on the right, red ones:



Flag on the Moon

- Then as before, let us look for horizontal stripes:



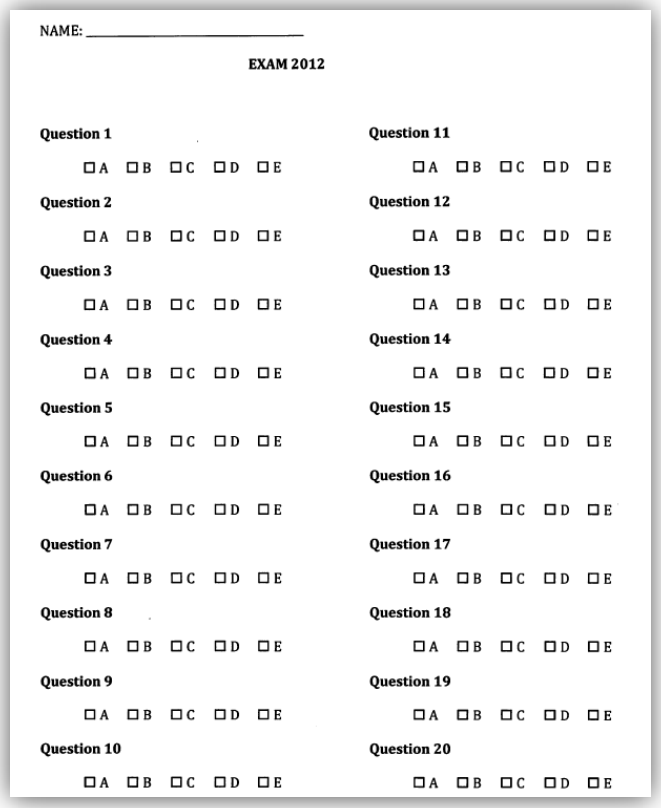
- SUCCESS!!!

Practical Example – Grading Exams

Grading an Exam

- Lets now see how we can apply this technique to automate the grading of an examination. Take the following examples, a template and a filled copy.

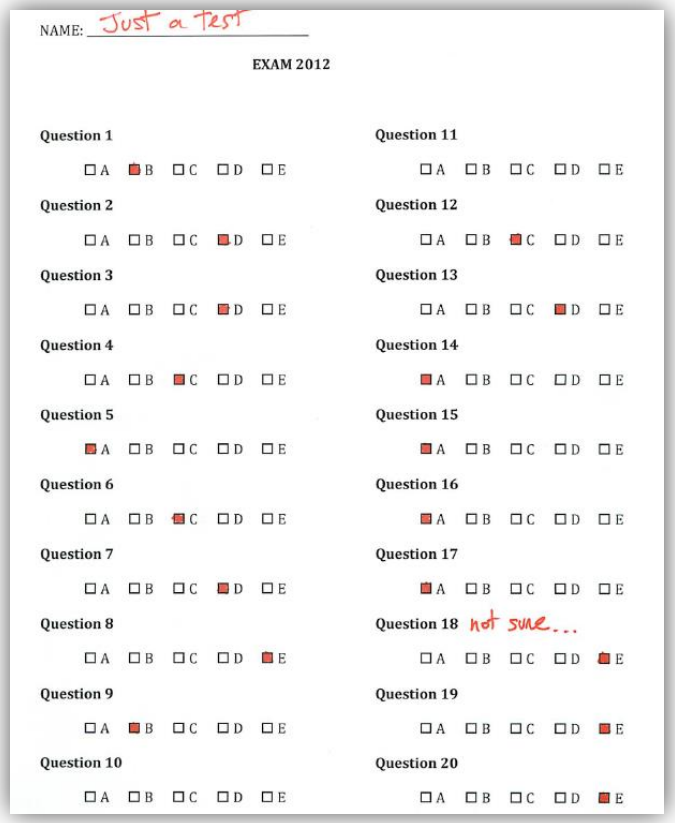
Template


NAME: _____

EXAM 2012

Question 1 ☐A ☐B ☐C ☐D ☐E
Question 2 ☐A ☐B ☐C ☐D ☐E
Question 3 ☐A ☐B ☐C ☐D ☐E
Question 4 ☐A ☐B ☐C ☐D ☐E
Question 5 ☐A ☐B ☐C ☐D ☐E
Question 6 ☐A ☐B ☐C ☐D ☐E
Question 7 ☐A ☐B ☐C ☐D ☐E
Question 8 ☐A ☐B ☐C ☐D ☐E
Question 9 ☐A ☐B ☐C ☐D ☐E
Question 10 ☐A ☐B ☐C ☐D ☐E
Question 11 ☐A ☐B ☐C ☐D ☐E
Question 12 ☐A ☐B ☐C ☐D ☐E
Question 13 ☐A ☐B ☐C ☐D ☐E
Question 14 ☐A ☐B ☐C ☐D ☐E
Question 15 ☐A ☐B ☐C ☐D ☐E
Question 16 ☐A ☐B ☐C ☐D ☐E
Question 17 ☐A ☐B ☐C ☐D ☐E
Question 18 ☐A ☐B ☐C ☐D ☐E
Question 19 ☐A ☐B ☐C ☐D ☐E
Question 20 ☐A ☐B ☐C ☐D ☐E

Exam

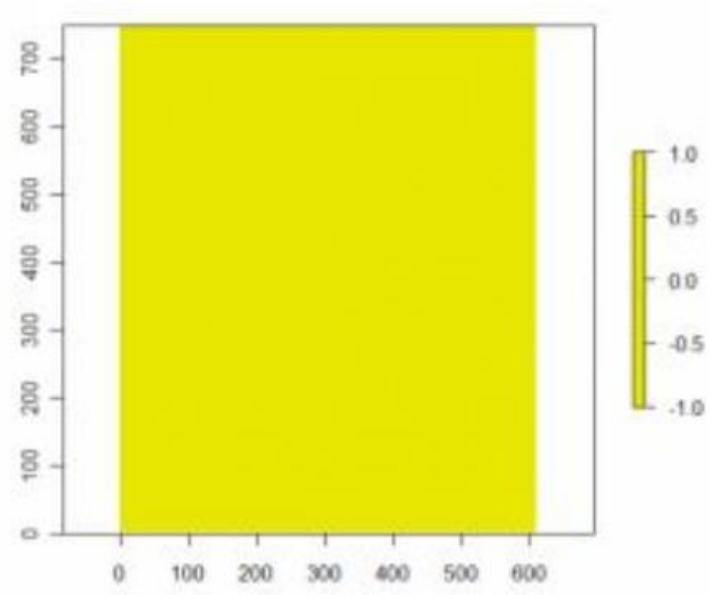

NAME: Just a Test

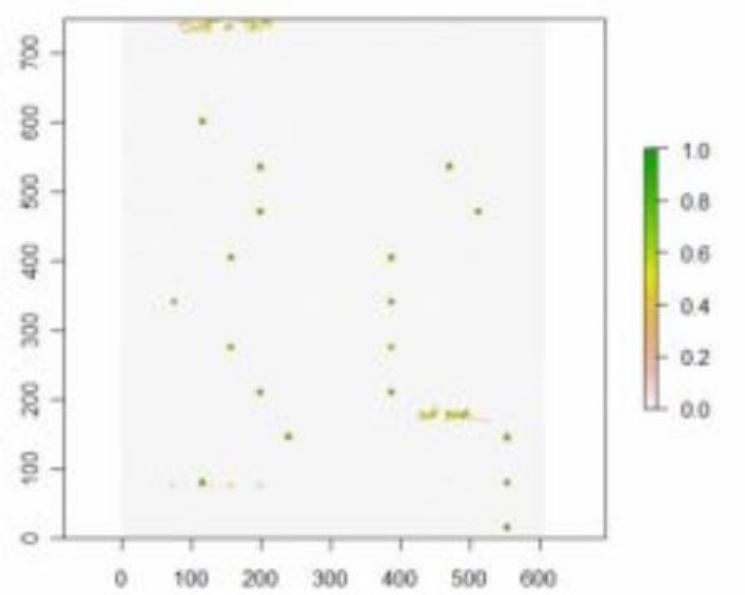
EXAM 2012

Question 1 ☐A ■B ☐C ☐D ☐E
Question 2 ☐A ☐B ☐C ■D ☐E
Question 3 ☐A ☐B ☐C ■D ☐E
Question 4 ☐A ☐B ■C ☐D ☐E
Question 5 ■A ☐B ☐C ☐D ☐E
Question 6 ☐A ☐B ■C ☐D ☐E
Question 7 ☐A ☐B ☐C ■D ☐E
Question 8 ☐A ☐B ☐C ☐D ■E
Question 9 ☐A ■B ☐C ☐D ☐E
Question 10 ☐A ☐B ☐C ☐D ☐E
Question 11 ☐A ☐B ☐C ☐D ☐E
Question 12 ☐A ☐B ■C ☐D ☐E
Question 13 ☐A ☐B ☐C ■D ☐E
Question 14 ■A ☐B ☐C ☐D ☐E
Question 15 ■A ☐B ☐C ☐D ☐E
Question 16 ■A ☐B ☐C ☐D ☐E
Question 17 ■A ☐B ☐C ☐D ☐E
Question 18 not sure... ☐A ☐B ☐C ☐D ■E
Question 19 ☐A ☐B ☐C ☐D ■E
Question 20 ☐A ☐B ☐C ☐D ■E

Grading an Exam

- A first step is to identify regions where we expect to find some "red" part (I assume here that students have to use a red pencil). Let us start to check on the template and the filled form if we can identify red areas:



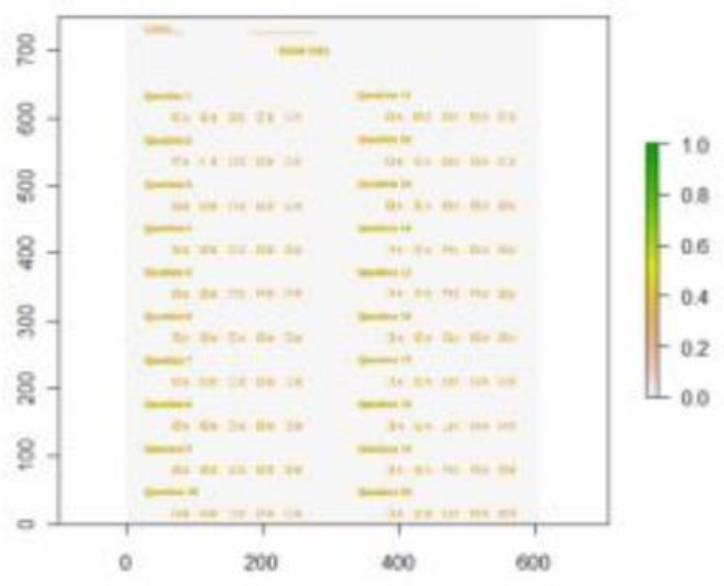


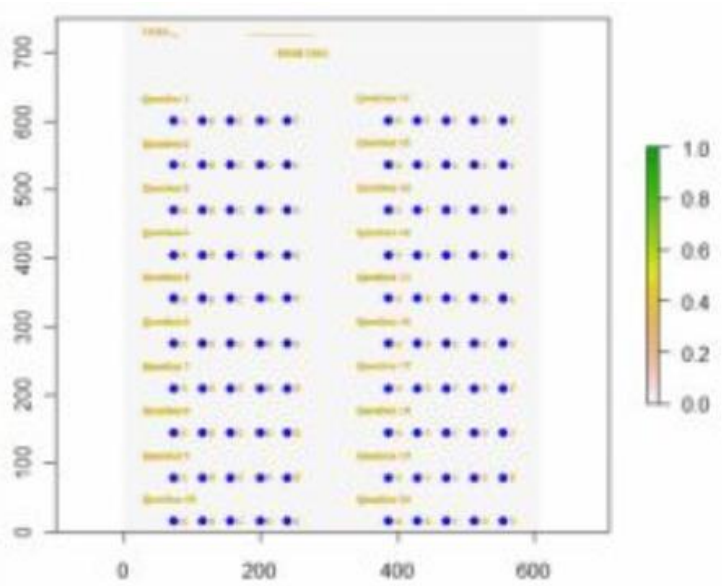

```
library(raster)
exam = stack("C:\\Users\\exam-blank.png")
red = (exam[[1]]>150)&(max(  exam[[2]] , exam[[3]])<150)
focalsred = focal(red, w=3, fun=mean)
plot(focalsred,useRaster=FALSE)
```

```
exam = stack("C:\\Users\\exam-filled.png")
red = (exam[[1]]>150)&(max(  exam[[2]] , exam[[3]])<150)
focalsred = focal(red, w=3, fun=mean)
plot(focalsred,useRaster=FALSE)
```

Grading an Exam

- First, we have to identify areas where students have to fill the blanks. So in the template, identify black boxes, and get the coordinates (here manually)





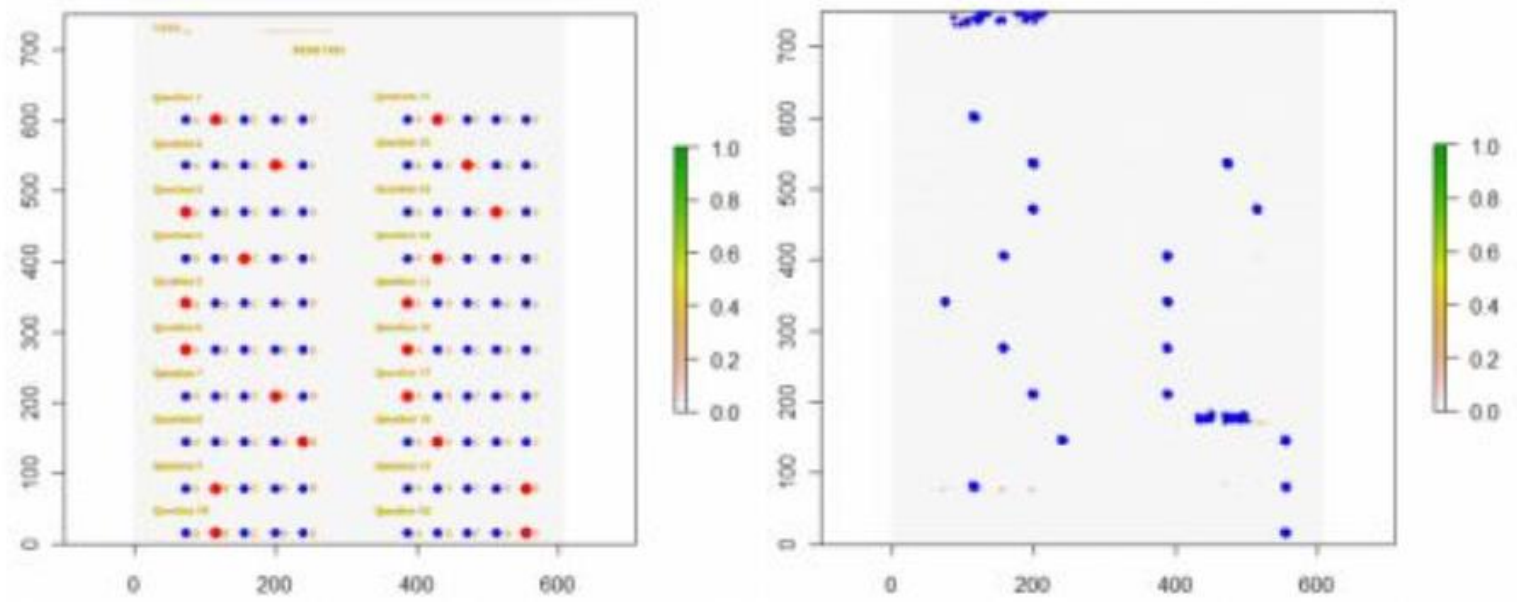

```
exam = stack("C:\\Users\\exam-blank.png")
black = max(  exam[[1]] ,exam[[2]] , exam[[3]])<50
focalsblack = focal(black, w=3, fun=mean)
plot(focalsblack,useRaster=FALSE)
correct=locator(20)
coordinates=locator(20)
X1=c(73,115,156,199,239)
X2=c(386,428.9,471,510,554)
Y=c(601,536,470,405,341,276,210,145,79,15)
LISTX=c(rep(X1,each=10),rep(X2,each=10))
LISTY=rep(Y,10)
points(LISTX,LISTY,pch=16,col="blue")
```

Grading an Exam

- The blue points above are where we look for students' answers. Then, we have to define the vector of correct answers:

```
CORRECTX=c(X1[c(2,4,1,3,1,1,4,5,2,2)],
X2[c(2,3,4,2,1,1,1,2,5,5)])
CORRECTY=c(Y,Y)
points(CORRECTX, CORRECTY,pch=16,col="red",cex=1.3)
UNCORRECTX=c(X1[rep(1:5,10)[-(c(2,4,1,3,1,1,4,5,2,2)
+seq(0,length=10,by=5))]],
X2[rep(1:5,10)[-(c(2,3,4,2,1,1,1,2,5,5)
+seq(0,length=10,by=5))]])
UNCORRECTY=c(rep(Y,each=4),rep(Y,each=4))
```

- Now, let us get back on red areas in the form filled by the student, identified earlier:

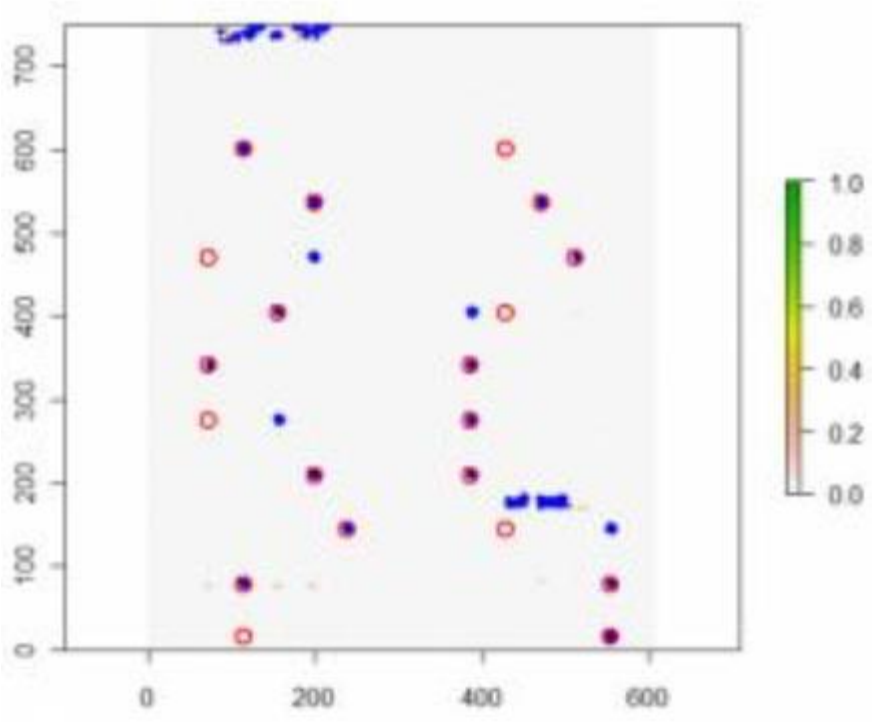



```
exam = stack("C:\\Users\\exam-filled.png")
red = (exam[[1]]>150)&(max(  exam[[2]] , exam[[3]])<150)
focalsred = focal(red, w=5, fun=mean)
```

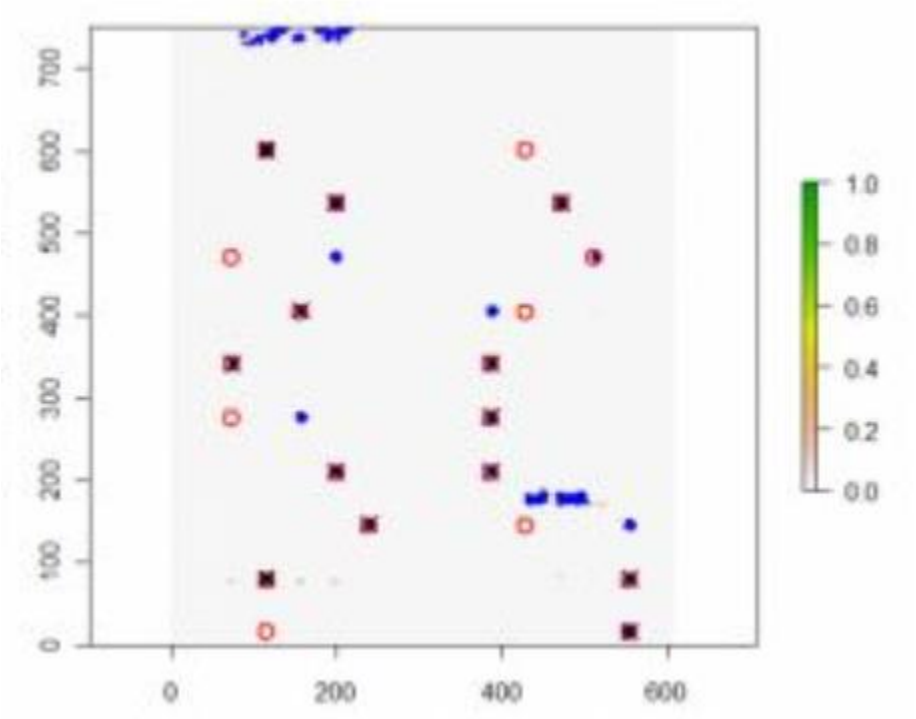
Grading an Exam

- Here, we simply have to compare what the student answered with areas where we expect to find some red in.

```
ind=which(values(focalsred)>.3)
yind=750-trunc(ind/610)
xind=ind-trunc(ind/610)*610
points(xind,yind,pch=19,cex=.4,col="blue")
points(CORRECTX, CORRECTY,pch=1,
col="red",cex=1.5,lwd=1.5)
```



Grading an Exam



- Crosses on the graph on the left are the answers identified as correct (here 13).

```
icorrect=values(red)[(750-CORRECTY)*
+ 610+(CORRECTX)]
points(CORRECTX[icorrect], CORRECTY[icorrect],
+ pch=4,col="black",cex=1.5,lwd=1.5)
sum(icorrect)
[1] 13
```

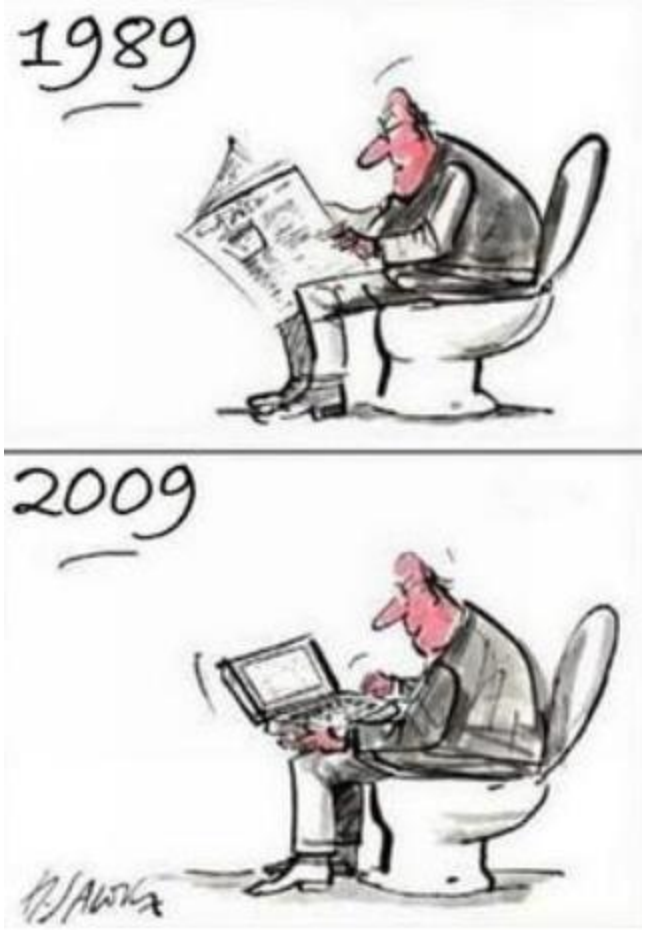
- In the case there are negative points for non-correct answer, we can count how many incorrect answers we had. Here 4.

```
iuncorrect=values(red)[(750-UNCORRECTY)*610+
+ (UNCORRECTX)]
sum(iuncorrect)
[1] 4
```

Deep Neural Networks & Deep Learning

Modern Advancements in ML

- We live in an exciting time where significant advancements have been made in regards to machine learning.
- These advances have been possible due to the persistence of human ingenuity and building upon proven concepts.
- This can be easily taken for granted based upon how rapidly the technology has been adapted within our consumer based culture.



Modern Advancements in ML

To showcase some of the amazing advancements, lets us look at the following:

Google

- Able to process speech and translate into a target language.
- Self driving car has autonomously driven 700,000 accident free miles in 2014, is able to stop at railroad crossings, & avoid cyclists.
- Project Glass introduced an integration of google technologies into an innovative hands free ecosystem.

Google Translate

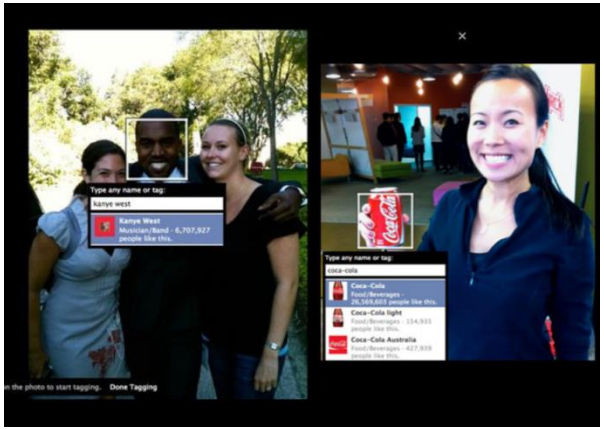
Break through language barriers

Modern Advancements in ML

WordLens – Identifies text within images, translates into a foreign language, and superimposes the results back on the image.

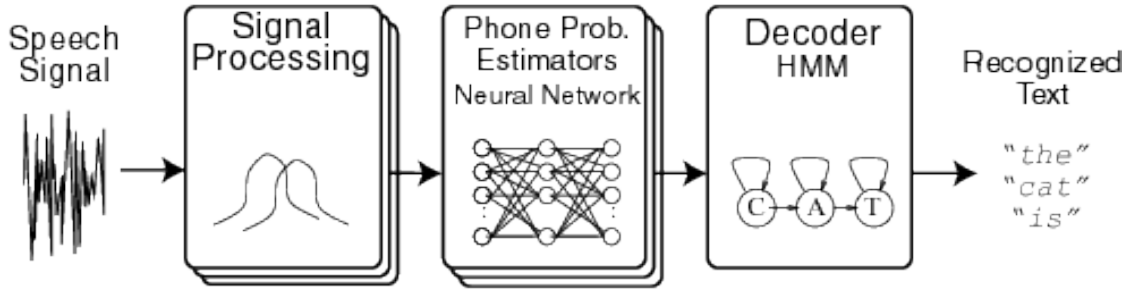
Modern Advancements in ML

- Facebook has incorporated dynamic facial / object recognition tagging functionalities, Apple's Siri improved the user interaction with voice recognition and search technologies, IBM's Watson is a jeopardy champion, and the list goes on and on…



Modern Advancements in ML

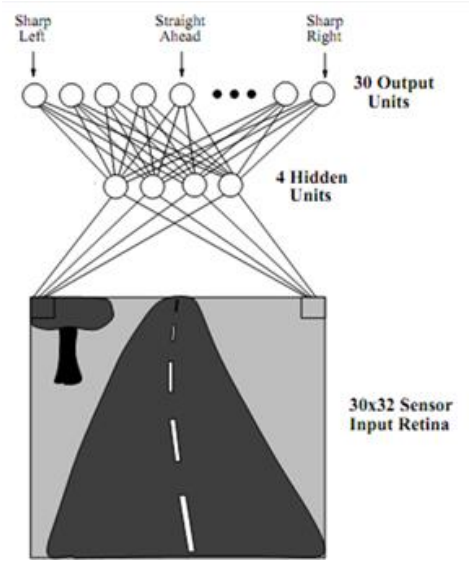
- As a curious data scientist, I first think about these life altering enhancements in terms of how the data is constructed.
- For example: How do we turn an image into a piece of usable data for machine learning?
- I also had an epiphany when I realized that a video file is just a series of images in sequence and a corresponding audio file.
  - Perhaps the image/audio can be processed using a Hidden Markov Model?
  - The audio signal can also analyzed within a Fast Fourier Transform and content can be derived through text analytics (including NLP).



Modern Advancements in ML

- Additionally, when I think about the Google self driving automobile as an extension the ALVINN Project. (Autonomous Land Vehicle In a Neural Network)

- ALVINN was based off of an artificial neural network with 5 hidden nodes connected an output layer of 30 units that symbolizes different directions.



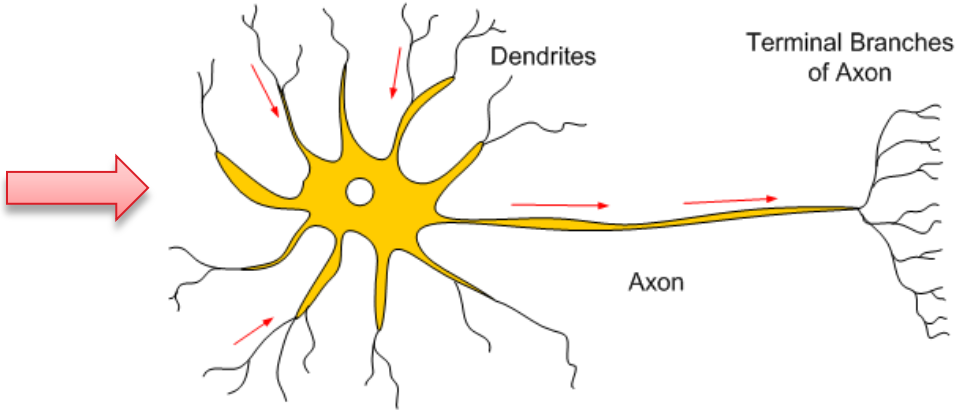
- ALVINN has successfully driven autonomously at speeds of up to 70 mph, and for distances of over 90 miles on a public highway north of Pittsburgh

Modern Advancements in ML

- However, I am still left with the observation that the majority of these advancements have been in fairly recent years.
- So what has changed? Why do we have all of these technologies now?

Return to Neural Networks

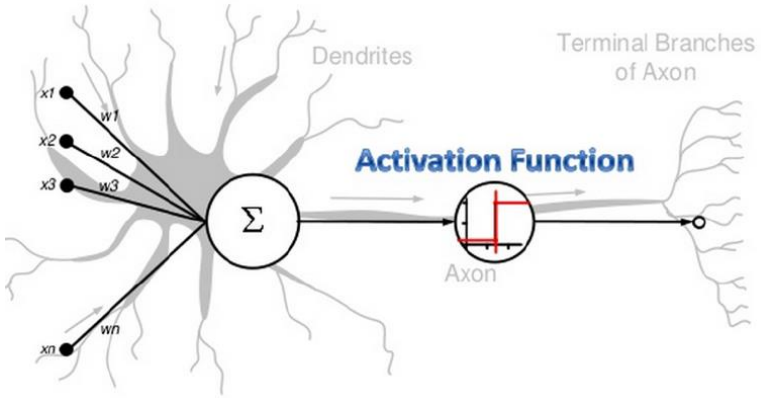
- Before we dive into this topic in detail, lets revisit one of the most powerful techniques in our ML arsenal: The Feedforward Artificial Neural Network trained through Back Propagation.



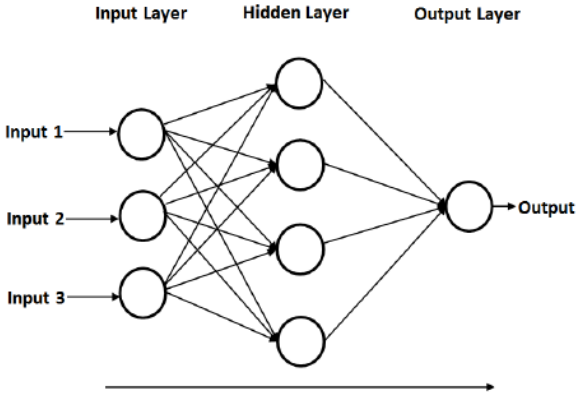
Return to Neural Networks

- Now lets get reacquainted with the Machine Learning ANN technique.

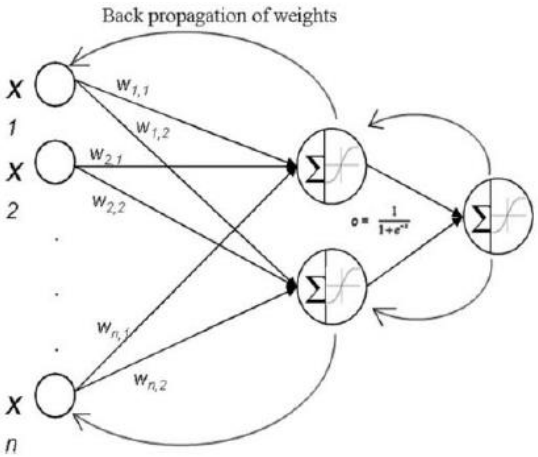
Perceptron



Single Hidden Layer

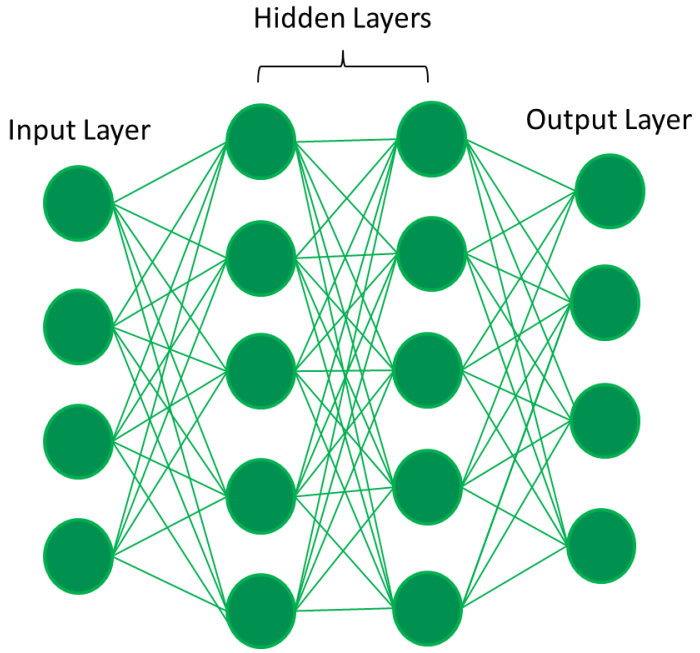


Back Propagation of Weights



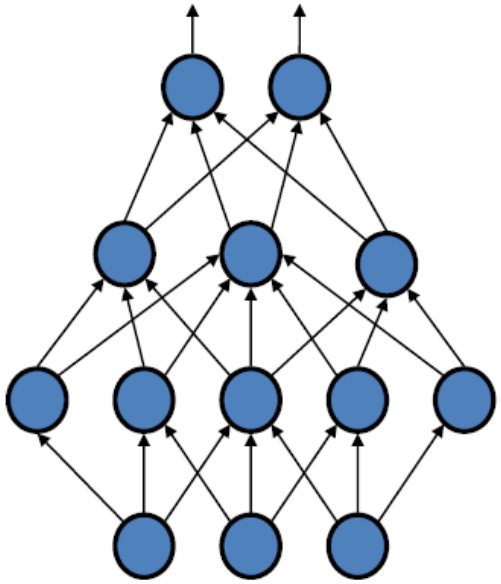
Deep Neural Networks – Gen 1

- The use of a single hidden node within the development of the ANN is the characteristic of the model.
- However, ML researchers in the 1990's looked into expanding the number of hidden layers into a new ML structure called Deep Neural Networks (DNN).
- These DNN also approached learning through the back propagation of weights at each node.



Deep Neural Networks – Gen 1

Unfortunately, these Deep Architectures repeatedly failed to train well.



- Here is the standard learning strategy utilized.
  - Randomly initializing the weights of the network.
  - Applying gradient descent using backpropagation.
- But, backpropagation does not work well (if randomly initialized)

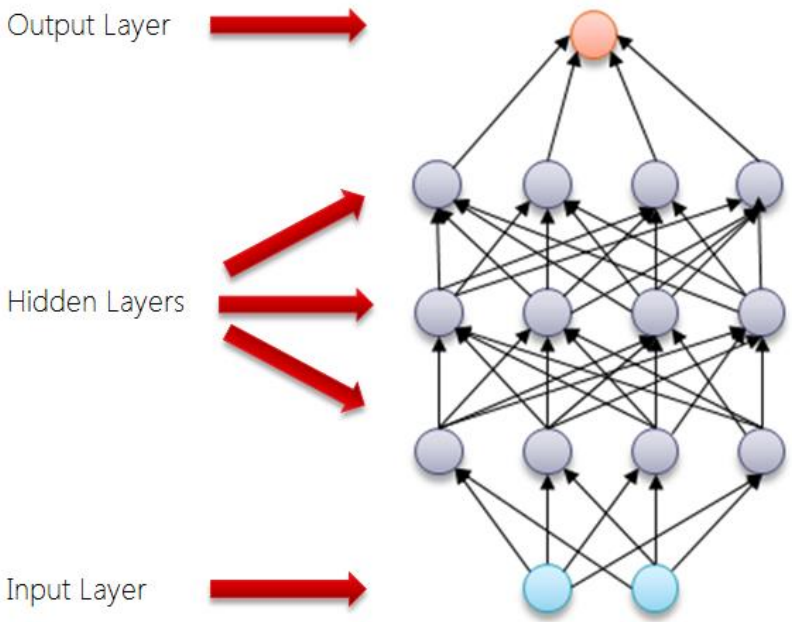
	train.	valid.	test
DBN, unsupervised pre-training	0%	1.2%	1.2%
Deep net, auto-associator pre-training	0%	1.4%	1.4%
Deep net, supervised pre-training	0%	1.7%	2.0%
Deep net, no pre-training	.004%	2.1%	2.4%
Shallow net, no pre-training	.004%	1.8%	1.9%

(Bengio et al., NIPS 2007)

Deep Neural Networks – Gen 1

Problems with Back Propagation

- Deep networks trained with back-propagation (without unsupervised pre-train) perform worse than shallow networks.
- Gradient descent is progressively getting more dilute across each hidden layer.
  - Below top few layers, correction signal is minimal
- The technique gets stuck in local minima
  - Especially since they start out far from 'good' regions (i.e., random initialization)
- In usual conditions, we can use only labeled data (supervised learning).
  - Almost all data is unlabeled!
  - The brain can learn from unlabeled data



The Deep Learning Breakthrough

- The discipline of machine learning had received a significant boost in neural network training in 2006.

- Hinton, Osindero, & Teh, "A Fast Learning Algorithm for Deep Belief Nets", Neural Computation, 2006
- Bengio, Lamblin, Popovici, & Larochelle, "Greedy Layer Wise Training of Deep Networks", NIPS 2006.
- Ranzato, Poultney, Chopra, & LeCun, "Efficient Learning of Sparse Representations with Energy Based Model" NIPS 2006

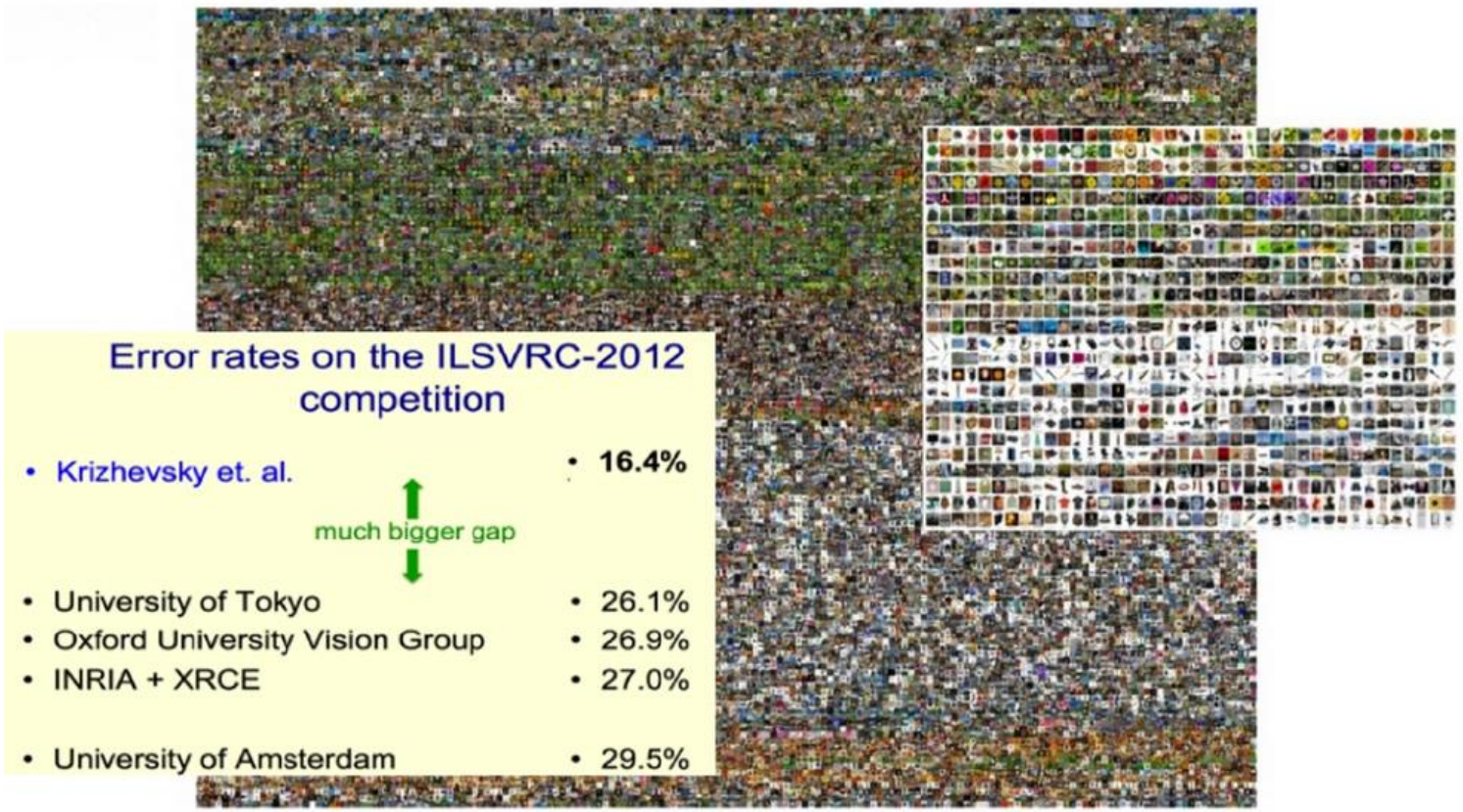
The Deep Learning Breakthrough

- The subtle shift from a standard deep neural network training of the perceptron weights through backpropagation into a layer wise greedy method was the most significant advancement in Machine Learning in the past 15 years.
- This is what we now refer to as "Deep Learning" but could also be referred to as Deep Neural Networks by some folks.
- I would recommend that we be careful of ambiguity here and simply refer to the technique as "Deep Learning".

"I've worked all my life in machine learning, and I've never seen one algorithm knock over benchmarks like deep learning." – Andrew Ng , ML Expert

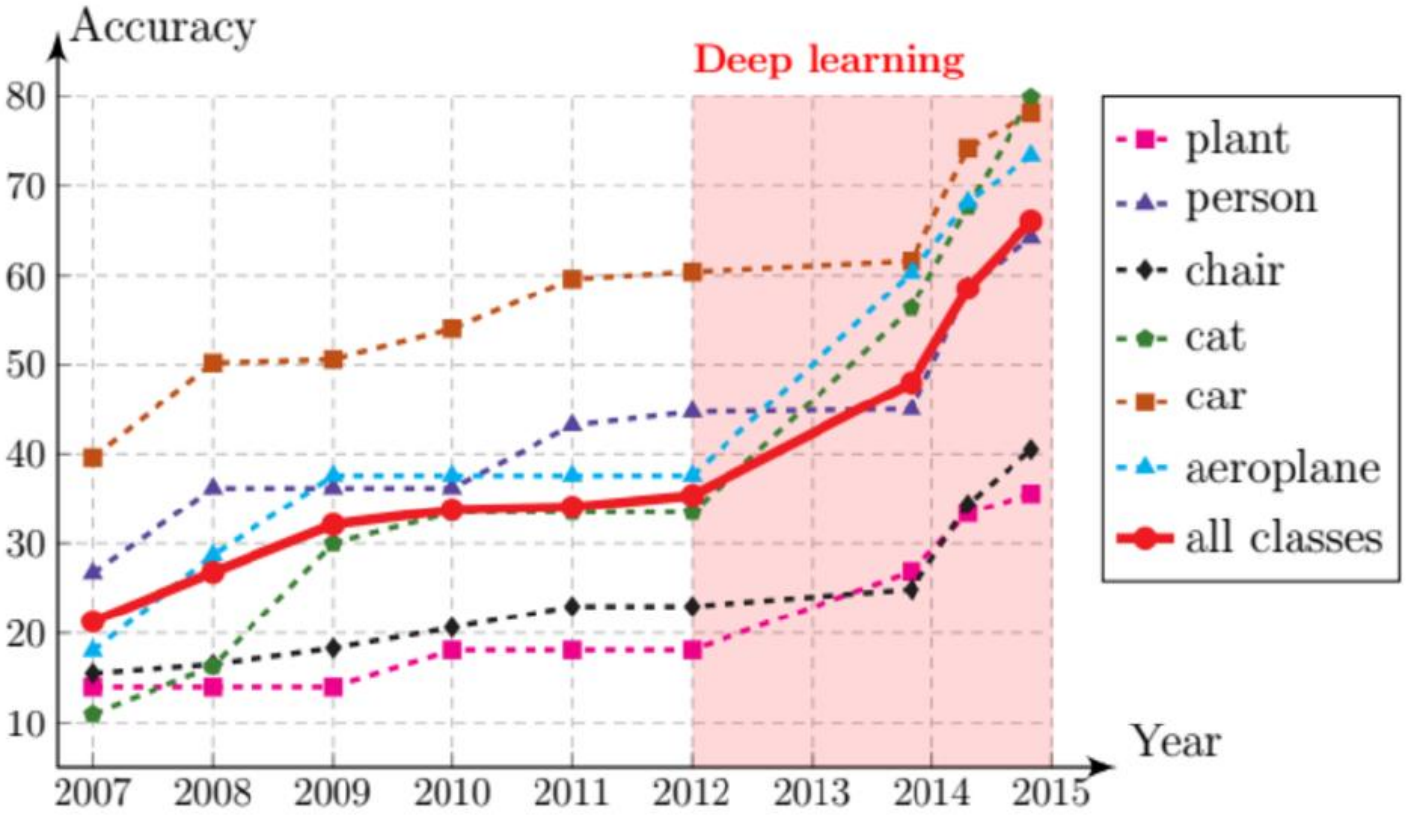
Impact on Computer Vision

- ImageNet Challenge 2012
- 1.2 Million Images with 1000 object categories.



Impact on Computer Vision

- ImageNet Challenge 2012



Impact on Audio Processing

- First public breakthrough with Deep Learning in 2010 - Dahl et al. (2010)

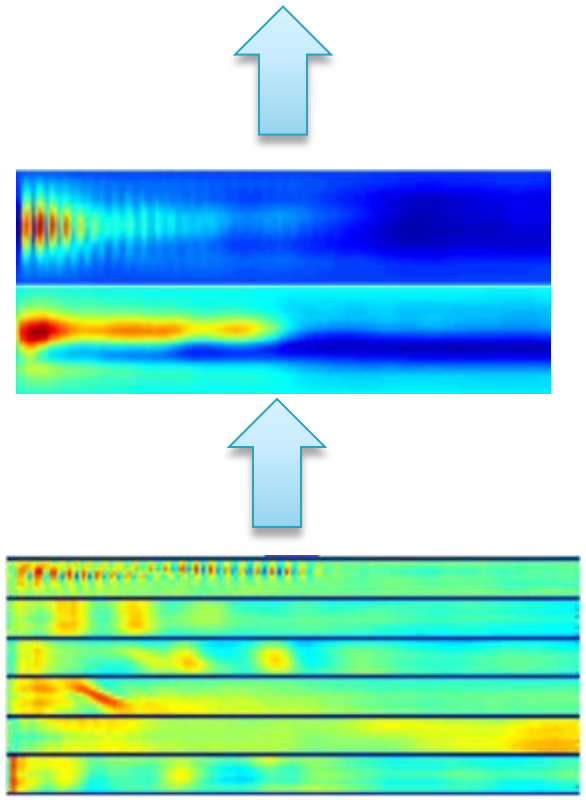
IEEE TRANSACTIONS ON AUDIO, SPEECH, AND LANGUAGE PROCESSING, VOL. 20, NO. 1, JANUARY 2012

Context-Dependent Pre-Trained Deep Neural Networks for Large-Vocabulary Speech Recognition

George E. Dahl, Dong Yu, *Senior Member, IEEE*, Li Deng, *Fellow, IEEE*, and Alex Acero, *Fellow, IEEE*

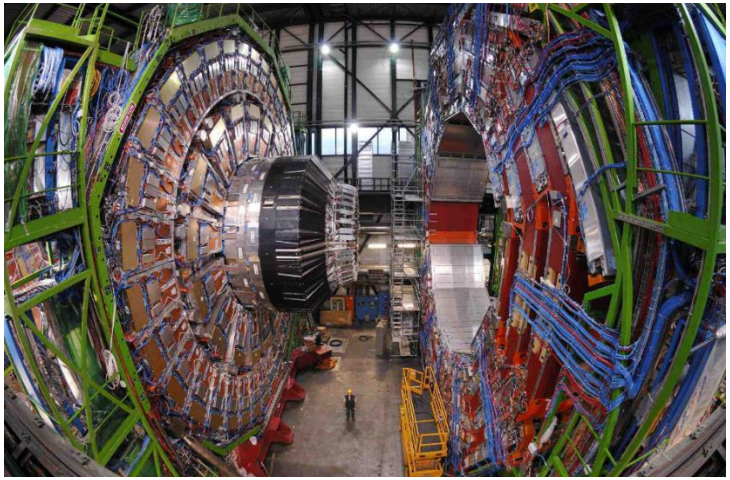
Acoustic model	Recog \ WER	RT03S FSH	Hub5 SWB
Traditional features	1-pass –adapt	**27.4**	**23.6**
Deep Learning	1-pass –adapt	**18.5**	**16.1**

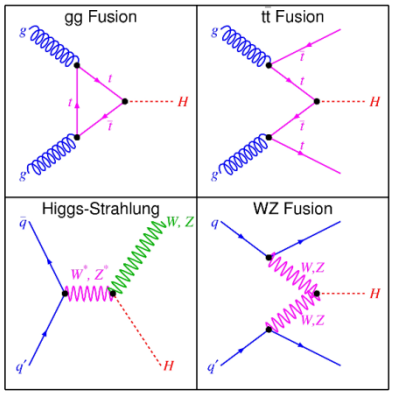
Improvement of 33% !!!

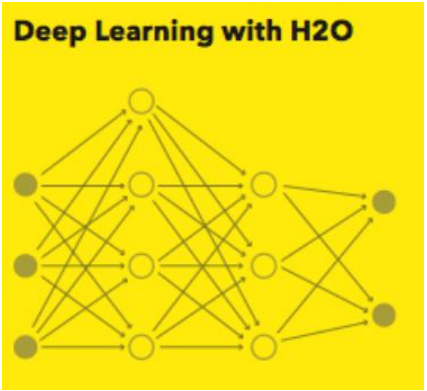


Impact on Scientific Research

- The LHC is the largest experiment of mankind today. $13+ Billion, 16.8 miles long, 120 megawatts, 1 Petabyte of data daily, etc…
- The Higgs boson discovery in July 2012 led to the 2013 Nobel Prize and further validates the Standard Model of particle physics, mankind's greatest triumph.
- Deep learning algorithms produced results with a statistical significance of 5σ to the mean (compared to Neural Networks 3.7σ). This 5σ is the generally accepted threshold for a discovery.
- H2o's implementation of Deep Learning in R claims to be able to produce the base line which the scientists confirmed the discovery.







Deep Learning in the Media

- This technique is receiving a lot of mainstream attention recently.

Baidu Hires Coursera Founder Andrew Ng to Start Massive Research Lab

‹re/code›

/ GENERAL

Andrew Ng

By Liz Gannes | @LizGannes | EMAIL | ETHICS

May 16, 2014, 9:00 AM PDT

Chinese search giant Baidu has hired Andrew Ng, the noted artificial intelligence researcher and co-founder of online education startup Coursera, to be its chief scientist, as it kicks off a new Baidu Research initiative with labs in Beijing and Sunnyvale, Calif., expecting to hire 150 to 200 people by the end of 2015.

NYU "Deep Learning" Professor LeCun Will Head Facebook's New Artificial Intelligence Lab

Posted Dec 9, 2013 by *Josh Constine* (@joshconstine)

By teaching a computer to think, Facebook hopes to better understand how its users do too. So today the company announced that one of the world's leading deep learning and machine learning scientists, NYU's Professor Yann LeCun, will lead its new artificial intelligence laboratory.

MIT Technology Review first reported that Facebook would launch an Artificial Intelligence lab back in September, but now it has something of a celebrity scientist at its helm. Facebook's AI research will be split across its Menlo Park headquarters, London office, and a new AI lab built just a block from NYU's campus in Manhattan.

LeCun has been pioneering artificial intelligence breakthroughs since the 1980s when he developed an early version of the "back-propagation algorithm" that became the top way to train artificial neural networks. He went on to work for AT&T Bell Laboratories where he created the "convolutional network model" that mimics the visual cortex of living beings to create a pattern recognition system for machines. This model was used for optical character recognition and handwriting recognition that powered how many banks read checks in the late 1990s and early 2000s.

LeCun's expertise is in "deep learning" speech and image recognition systems has driven his research in building visual navigation systems for self-driving cars, autonomous ground robots, drones, and more.

Deep Learning in the Media

Deep Learning in the Press…

Hinton Ng Zuckerberg LeCun Kurzweil

Google Hires Brains that Helped Supercharge Machine Learning.
Wired 3/2013.

Facebook taps 'Deep Learning' Giant for New AI Lab.
Wired 12/2013.

Is "Deep Learning" A Revolutions in Artificial Intelligence?
New Yorker 11/2012.

The Man Behind the Google Brain: Andrew Ng and the Quest for the New AI.
Wired 5/2013.

New Techniques from Google and Ray Kurzweil Are Taking Artificial Intelligence to Another Level.
MIT Technology Review 5/2013.

Big Data **EXCLUSIVE** VB

Snapchat is quietly building a research team to do deep learning on images, videos

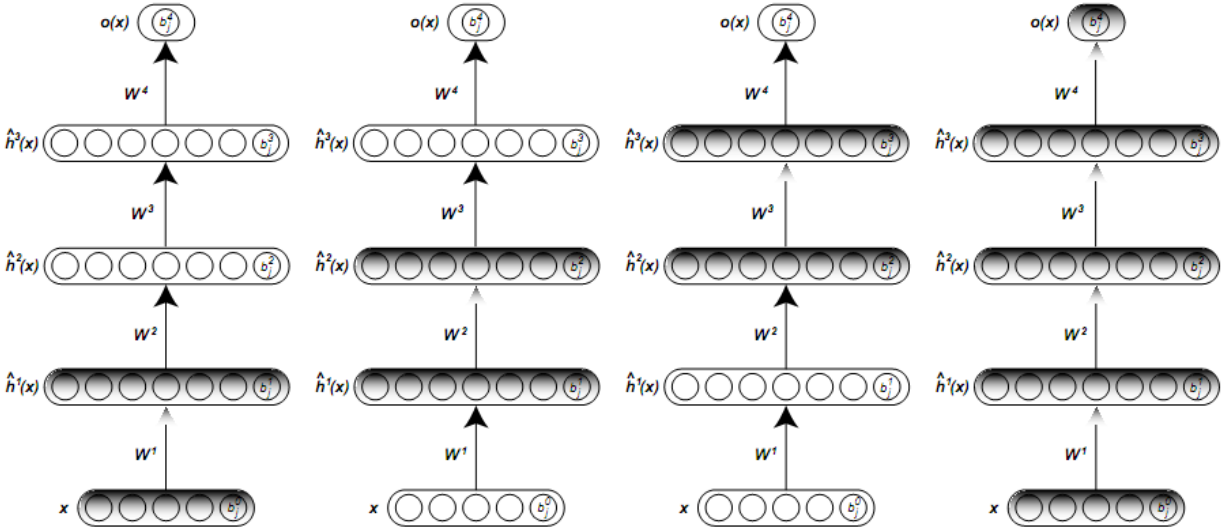
April 8, 2015 3:15 PM
Jordan Novet

Snapchat, that well-funded Los Angeles startup providing a popular ephemeral-mobile-messaging app, has been slowly developing a research arm to run sophisticated algorithms on user data like images and videos, VentureBeat has learned.

Unsupervised Greedy Layer Wise Training

- We will now go through the mechanics of the deep learning approach to better understand how it works and more importantly, what can we ultimately achieve with the technique.

- Here is a basic pictorial representation of the training procedure for various deep learning architectures.

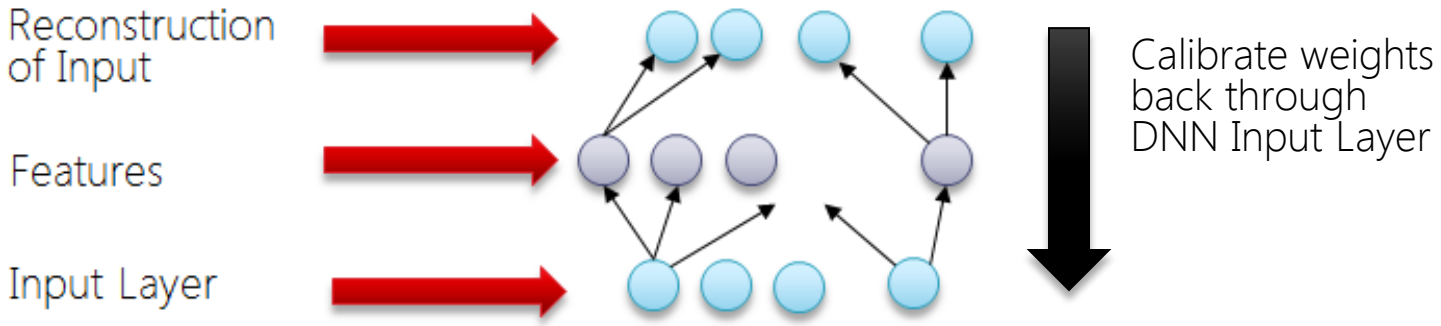


(a) First hidden layer pre-training (b) Second hidden layer pre-training (c) Third hidden layer pre-training (d) Fine-tuning of whole network

Unsupervised Greedy Layer Wise Training

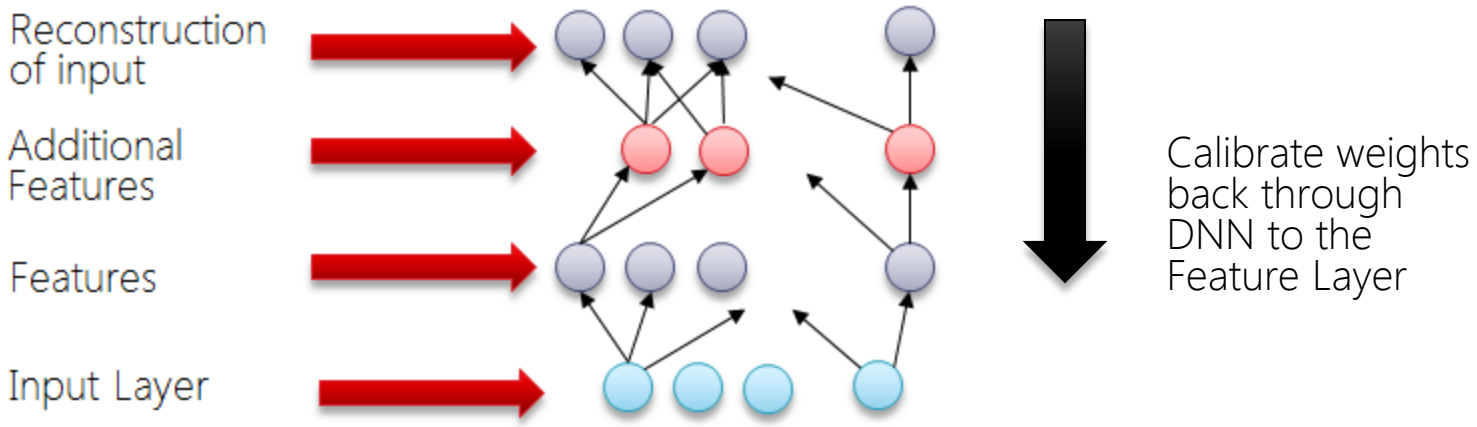
- Lets now walkthrough the concept step by step so we can see how the algorithm is working:

Step 1: Train the first hidden layer of the DNN and reconstruct the input based upon the hidden layers weights.



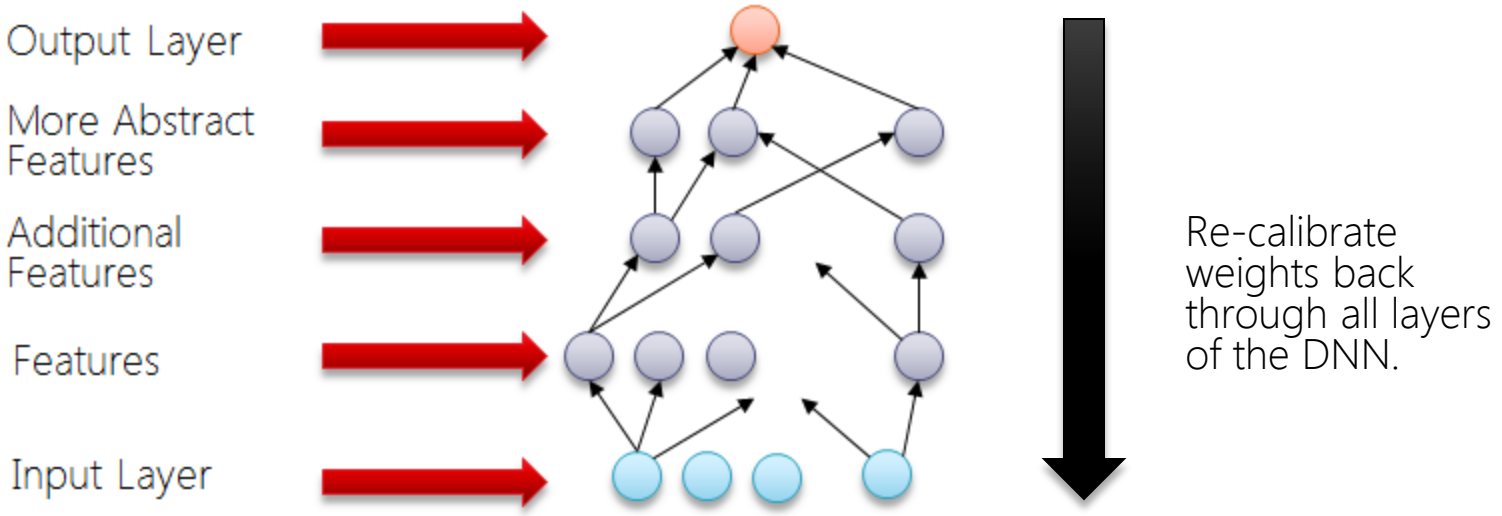
Unsupervised Greedy Layer Wise Training

- Step 2: We now take the next hidden layer of "Additional Features" and train the layer using the inputs from the "Features" and reconstruct the Feature layer from the inputs.

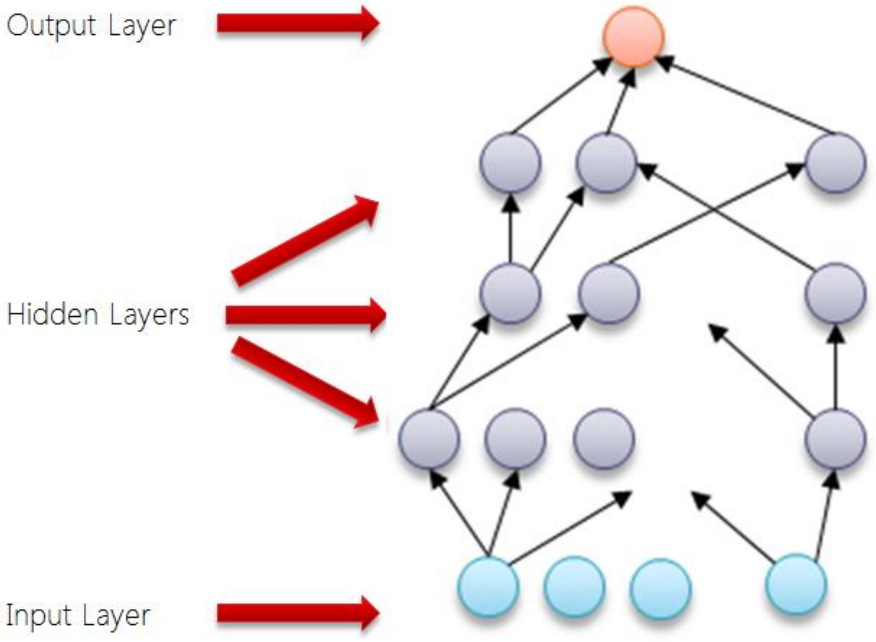


Unsupervised Greedy Layer Wise Training

- **Step 3:** We continue to go through each hidden layer as described in step 2 until we reach the final output layer.



Modern Deep Neural Net Architecture



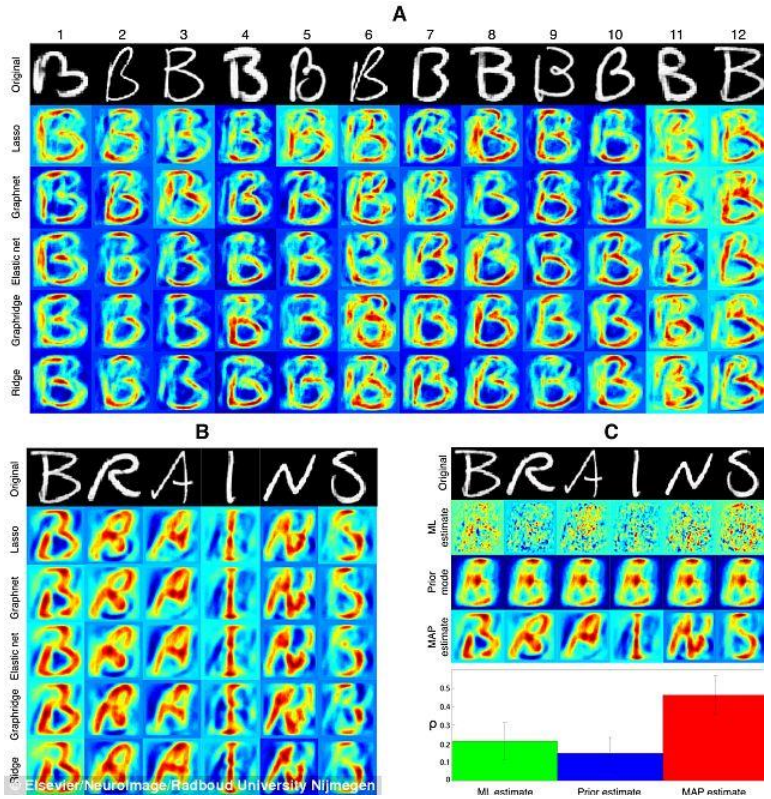
Here is a depiction of Partial Feature Sharing with mixed mode learning and composition of functions.

Unsupervised Greedy Layer Wise Training Benefits:

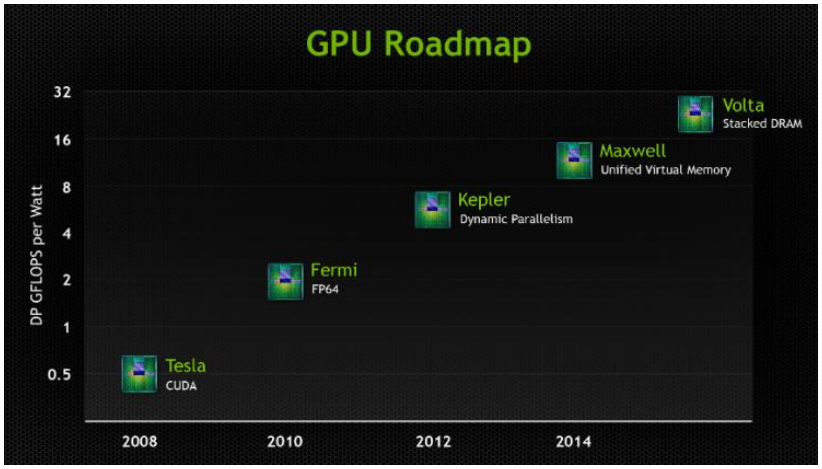
- Allows abstraction to develop naturally from one layer to another.
- Help the network initialize with good parameters
  - Perform supervised top-down training as final step
- Refine the features (intermediate layers) so that they become more relevant for the task.

What does Deep Learning do?

- Deep learning is a form of representation learning. This approach attempts to automatically learn good features or representations from the data.
- To be successful with this technique:
  - We need more data to train the models. This is the essence of the "Big Data" era for me.
  - Faster hardware: (GPU & Multicore CPU)
- Deep learning attempts to learn multiple levels of representation of increasing complexity/abstraction. (More on this later)



What does Deep Learning do?



In 2008, the fastest computer in the world was the PlayStation.

Importance of GPU's:

- GPUs power video cards and draw pictures on the screen. They are very parallel, fast, cheap, and low power.
- Advancements to the GPU have opened up the capabilities of the greedy learning algorithm to be performed in a reasonable time frame.
- Without this hardware breakthrough, deep learning would not be possible.
- To showcase the performance gains, here are a couple of performance benchmarks.
  - PC CPU: 1-3 Gflop / s average
  - GPU: 100 Gflop / s average

What does Deep Learning do?

- For supervised learning tasks where label information is readily available in training, deep learning promotes a principle which is very different than traditional methods of machine learning which incorporate feature engineering.

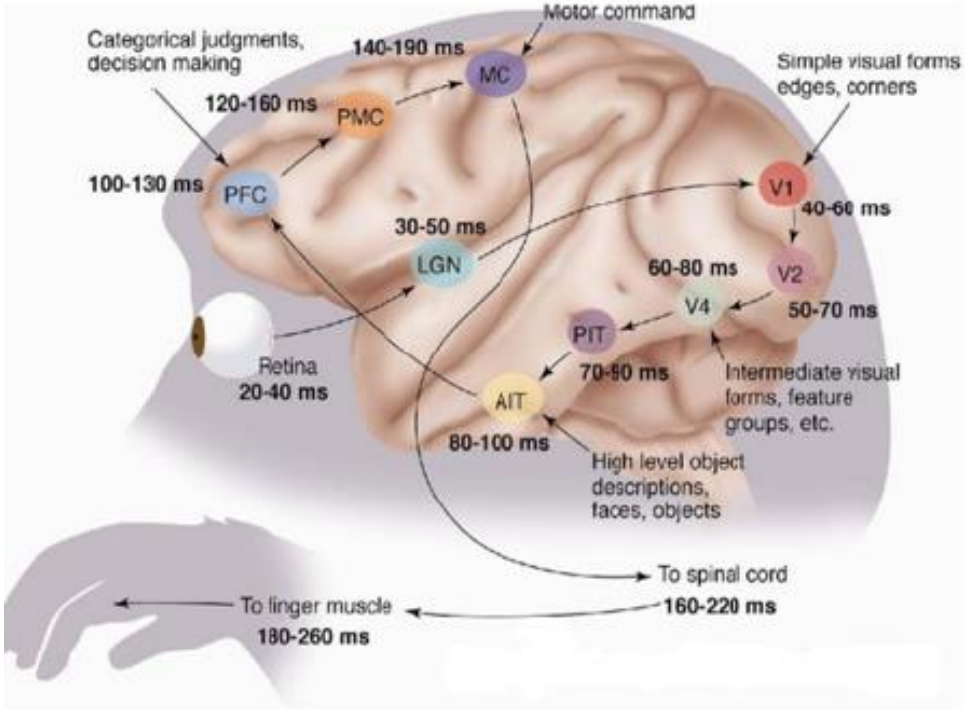
- Feature engineering is the process of using domain knowledge of the data to create features that make machine learning algorithms work better. Coming up with features is difficult, time-consuming, requires expert knowledge, and varies from one task to another .

- Rather than focusing on feature engineering which is often labor-intensive, deep learning methods are focused on end-to-end learning based on raw features.

- Deep learning moves away from feature engineering to a maximal extent possible.

What does Deep Learning do?

- Deep Learning is truly inspired by the brain.
- Audio/Visual Cortex has multiple stages which implies that our thought process is hierarchical in nature.



- Deep Learning can "theoretically" learn any function.

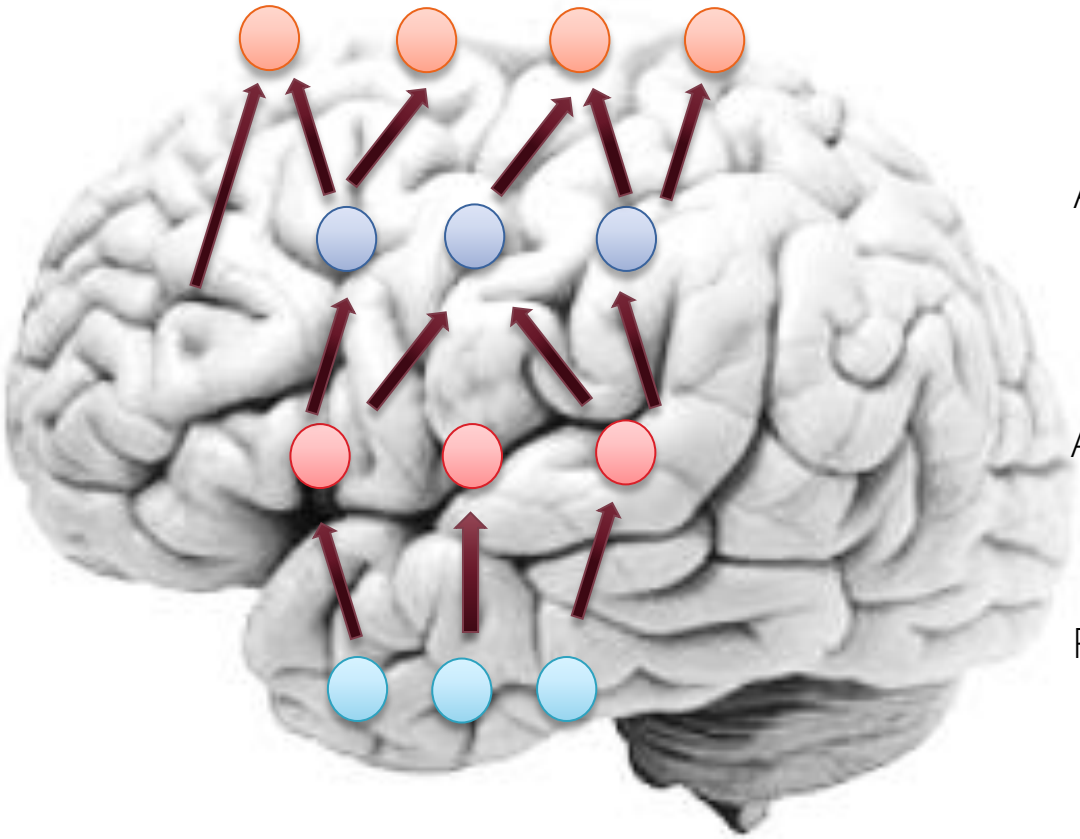
Different Levels of Abstraction

Higher level visual abstractions

Primitive Shape Detectors

Edge Detectors

Pixels



Area V2

Area V1

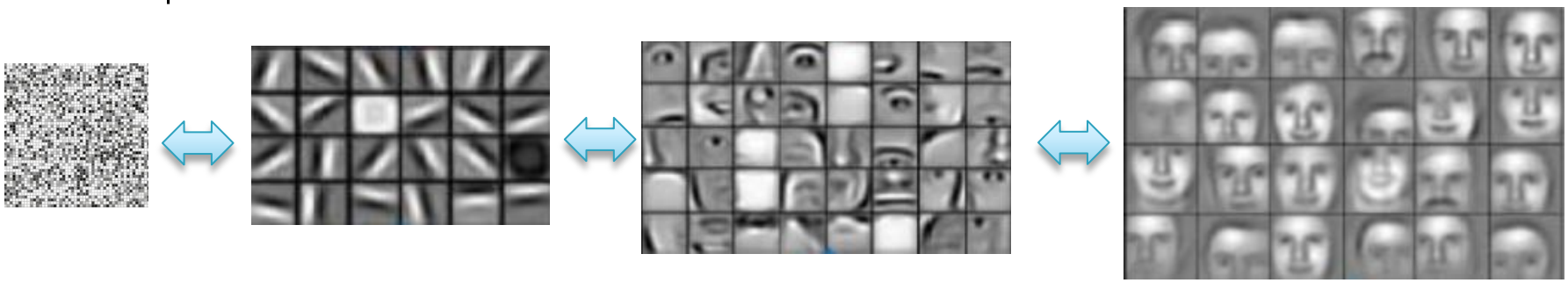
Retina

Different Levels of Abstraction

Hierarchical Learning

- Natural progression from low level to high level structure as seen in natural complexity.
- Information is being processed and learned at each stage of the hierarchy (each hidden layer) and used to guide the machine to better subspaces
- A good lower level representation can be used for many distinct tasks.

Feature Representation



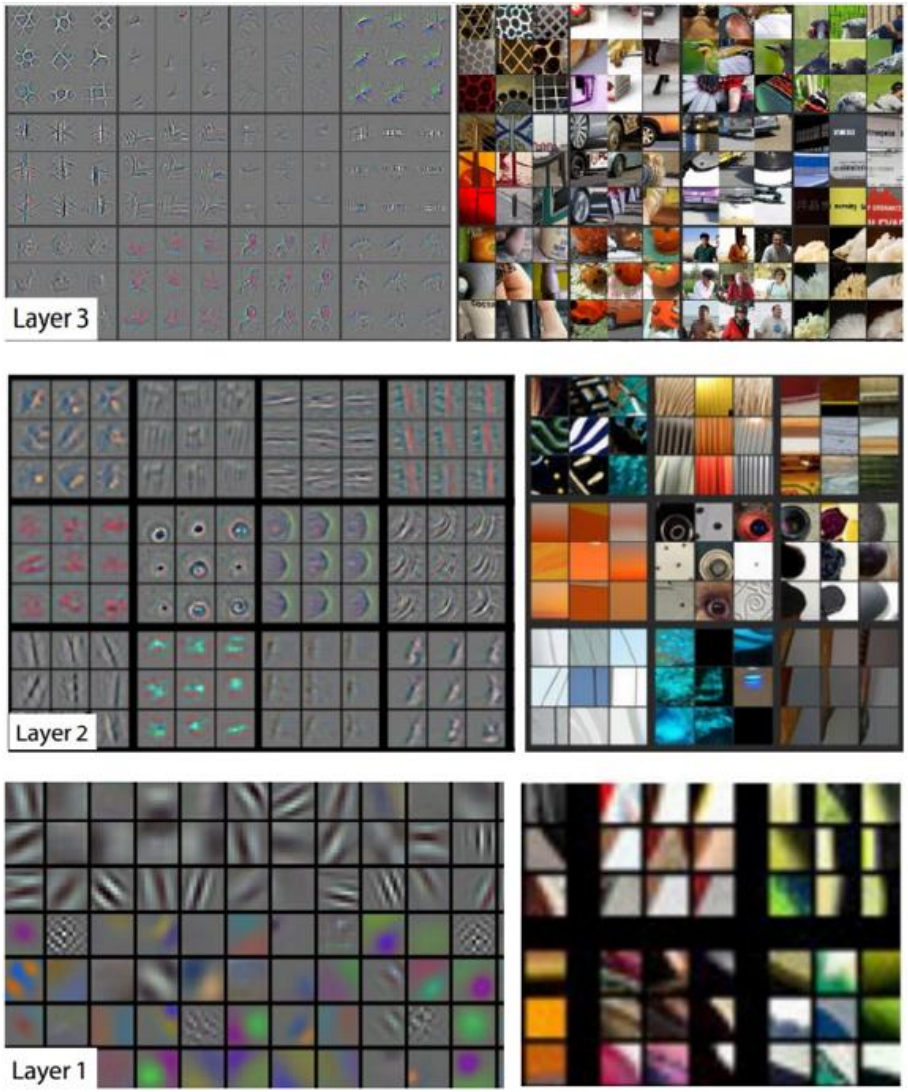
Pixels

1st Layer
"Edges"

2nd Layer
"Object Parts"

3rd Layer
"Objects"

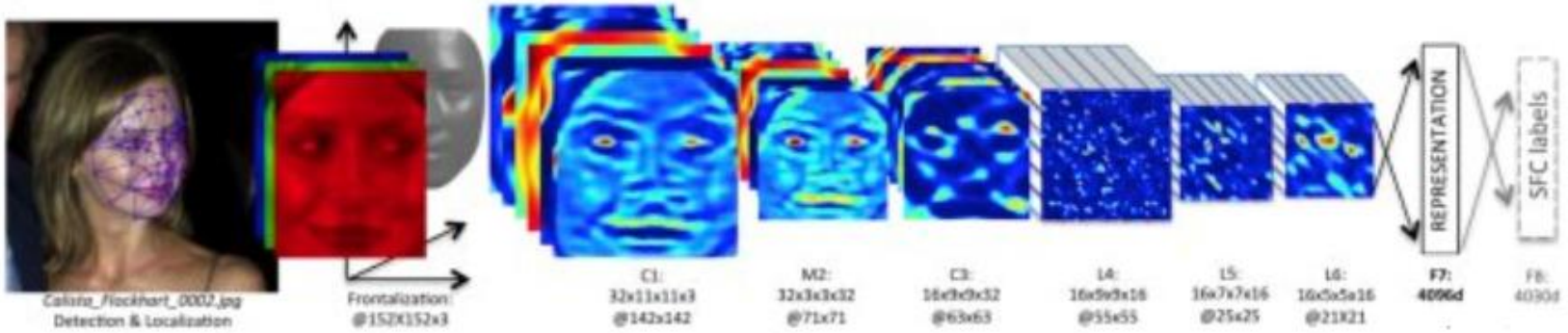
Different Levels of Abstraction



Deep Learning Examples

- These different levels of abstraction are ultimately how the modern day machine learning advancements have occurred:

Facial Recognition:



Deep Learning Examples

- Google's secretive X lab utilized 16,000 computers to create a simulation of the brain using deep learning.
- Researchers took 10 million randomly selected thumbnails from YouTube videos over the course of 3 days.
- One of the neurons in the artificial neural network learned to detect cats.
- The network had never been told what a cat was, nor was it ever given a single image labeled as a cat.

Is this image the birth of true Artificial Intelligence?

Deep Learning Examples

- With multiple output nodes each representing a specific physical object, we can see how this technology could be used to identify a number of unidentified objects in an unsupervised manner.

Original Image

Expected Labels

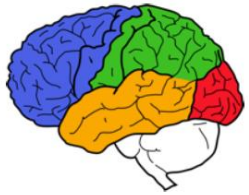
Deep Learning Model

Deep Learning Examples

- Combine this concept with hard or soft clustering and unstructured text analytics (NLP) and we can see the near unlimited potential.

Real Time Scene Parsing

Deep Learning Examples



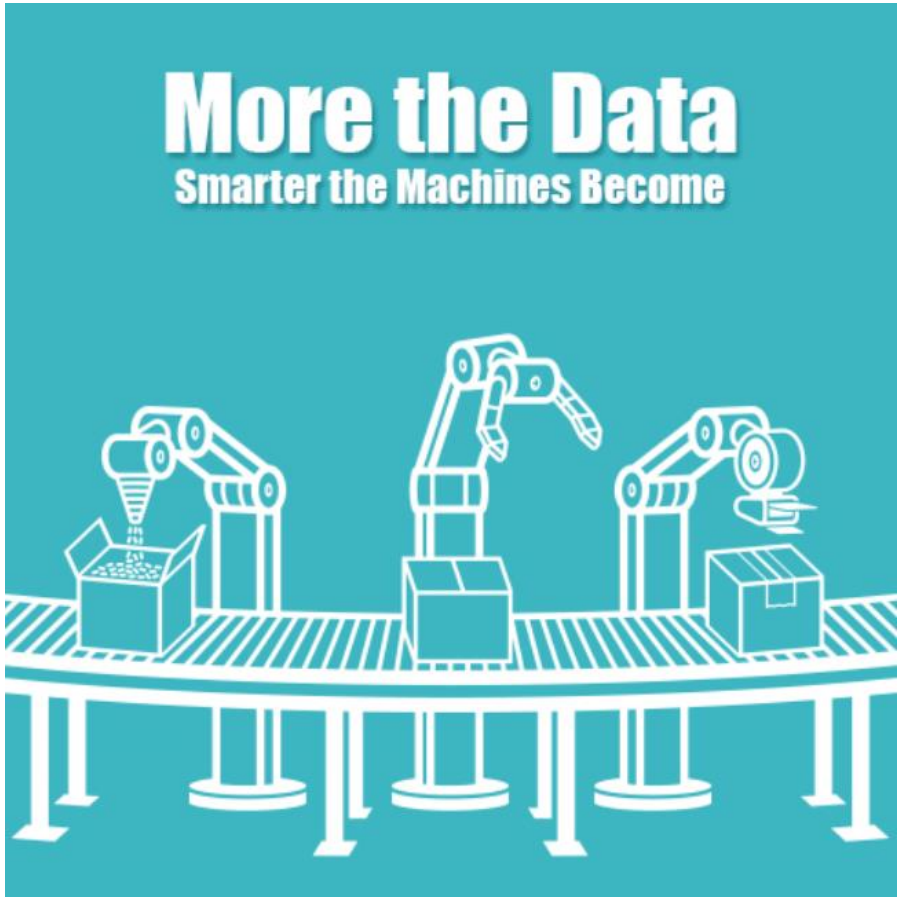
- To showcase this further, Jeff Dean and his colleagues at Google are actively working on Large Scale Deep Learning Applications that automatically create image captions from the pixel content of the digital image. Incredible!!!

Human: A young girl asleep on the sofa cuddling a stuffed bear.

Model sample 1: A close up of a child holding a stuffed animal.

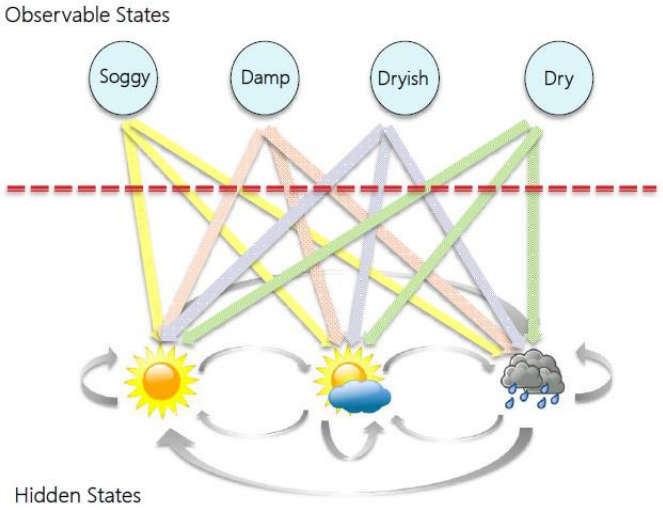
Model sample 2: A baby is asleep next to a teddy bear.

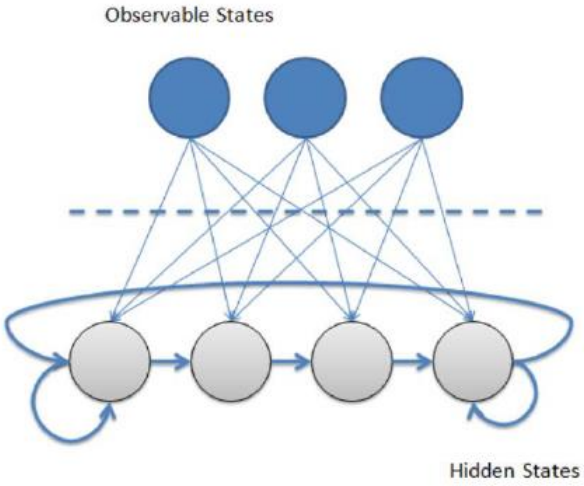
Deep Learning Example



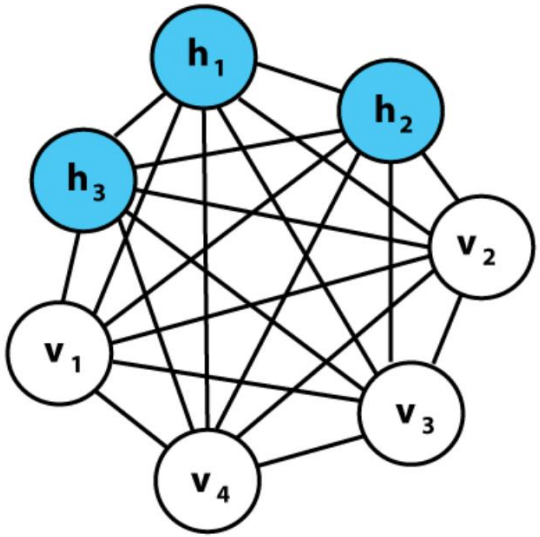
Expanding the Deep Learning Architecture

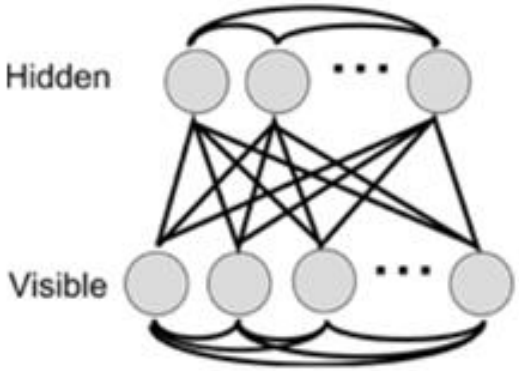
- Now that we have spent some time discussing Deep Learning in depth, lets focus on some variations of the technique for different DNN architectures.
- Before we begin, we will revisit the topic of Hidden Markov Model's. (See previous lecture for in depth review)
- Lets recall the example of the weather prediction hermit. The observation of the seaweeds condition is hinting at the underlying hidden markov process of the change of weather.
- This example of the hidden markov process can be thought of as a hidden layer, similar to the Neural Network and Deep Learning Architecture.





Boltzmann Machines

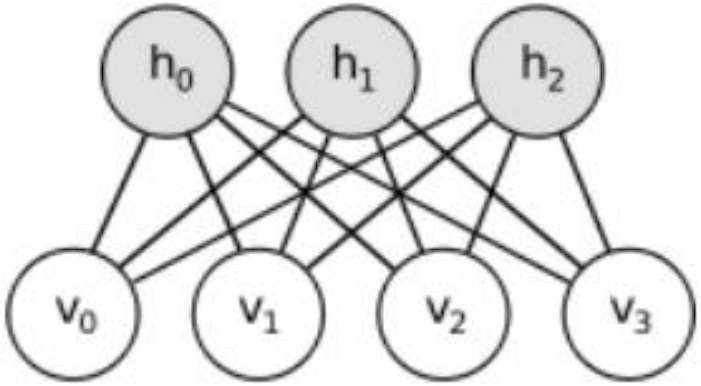




- There is a special form the log-linear Markov Random Field which produces what is called a Boltzmann Machine.
- A key feature is that the log-linear Markov field's energy function is linear in its free parameters.
- To make the Boltzmann Machine's powerful enough to represent complicated distributions (i.e., go from the limited parametric setting to a non-parametric one), we consider that some of the variables are never observed (they are called hidden).
- By having more hidden variables (also called hidden units), we can increase the modeling capacity of the Boltzmann Machine.

Restricted Boltzmann Machines

- Restricted Boltzmann Machines further restrict Boltzmann Machines to those without visible-visible and hidden-hidden connections.
- One example of a practical application of Restricted Boltzmann machines is the performance improvement of speech recognition software.
- A graphical depiction of an Restricted Boltzmann machine is shown below.



- Question: Do you notice the distinct similarity to the Hidden Markov Model?

Restricted Boltzmann Machines

- The energy function $E(v, h)$ of an RBM is defined as:

$$E(v, h) = -b'v - c'h - h'Wv$$

- where W represents the weights connecting hidden and visible units and b, c are the offsets of the visible and hidden layers respectively.

- This translates directly to the following free energy formula:

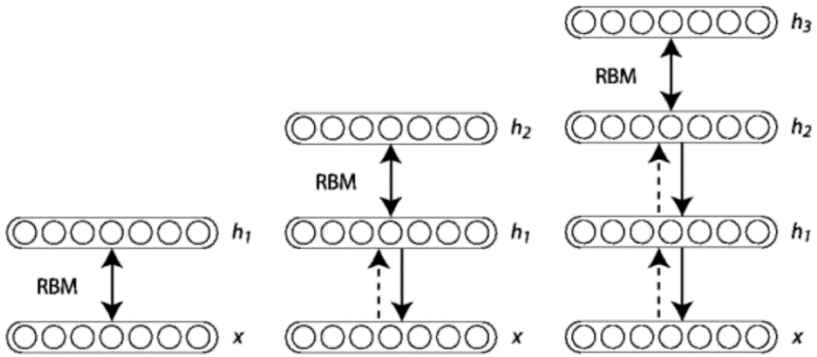
$$\mathcal{F}(v) = -b'v - \sum_i \log \sum_{h_i} e^{h_i(c_i + W_i v)}.$$

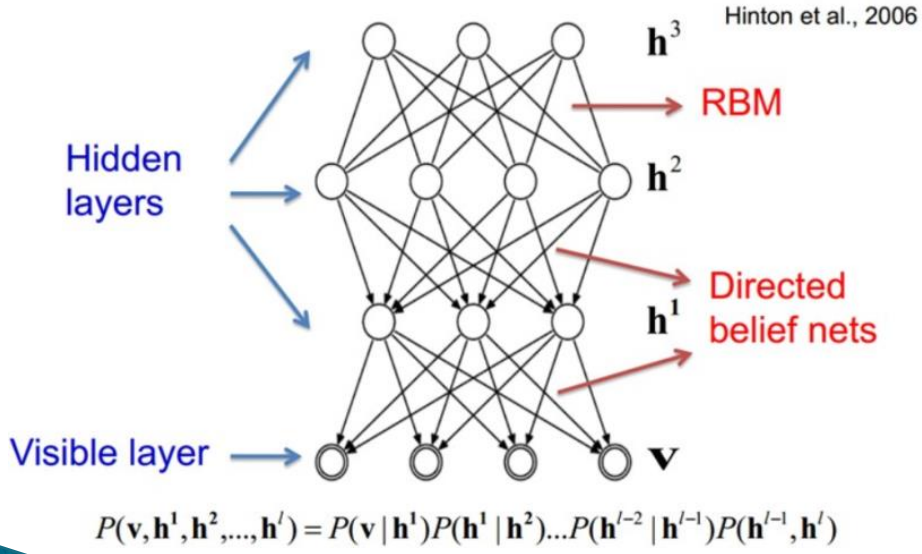
- Because of the specific structure of RBMs, visible and hidden units are conditionally independent given one-another. Using this property, we can write:

$$p(h|v) = \prod_i p(h_i|v)$$

$$p(v|h) = \prod_j p(v_j|h).$$

Deep Belief Networks



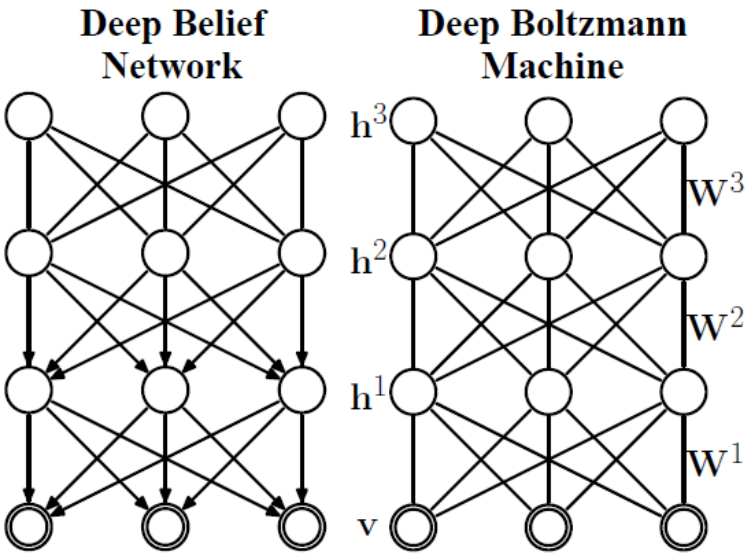


$$P(\mathbf{v},\mathbf{h}^1,\mathbf{h}^2,...,\mathbf{h}^l) = P(\mathbf{v}\,|\,\mathbf{h}^1)P(\mathbf{h}^1\,|\,\mathbf{h}^2)...P(\mathbf{h}^{l-2}\,|\,\mathbf{h}^{l-1})P(\mathbf{h}^{l-1},\mathbf{h}^l)$$

- The hidden layer of a Restricted Boltzmann Machine can contain additional hidden layer(s) stacked on top, and trained using the greedy manner.
- This produces what we now call a Deep Belief Network (DBN). I like to think of them as a hybrid of Hidden Markov Model's and Deep Learning.
- DBN's are graphical models which learn to extract a deep hierarchical representation of the training data.
- We can think of RBM's as being generative auto encoders; if we want a deep belief network we should be stacking RBMs and not plain auto encoders.

Deep Boltzmann Machines

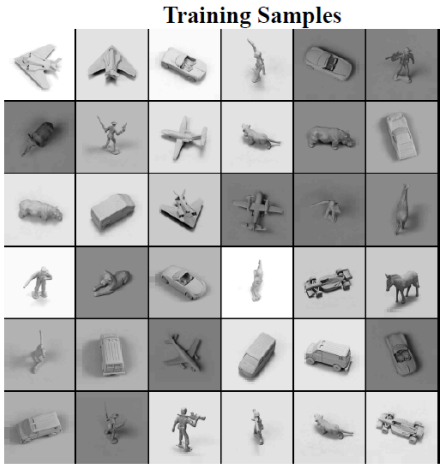
- With the Deep Belief Network, we trained the stacked Restricted Boltzmann Machines using the greedy layer wise algorithm.
- The use of this training algorithm essentially no longer makes this a true Boltzmann Machine.
- It is a hybrid generative model that has undirected connections between its top two layers and downward directed connections between all its lower layers.
- A Deep Boltzmann Machine resolves this issue by altering the training procedure for the top two layers.

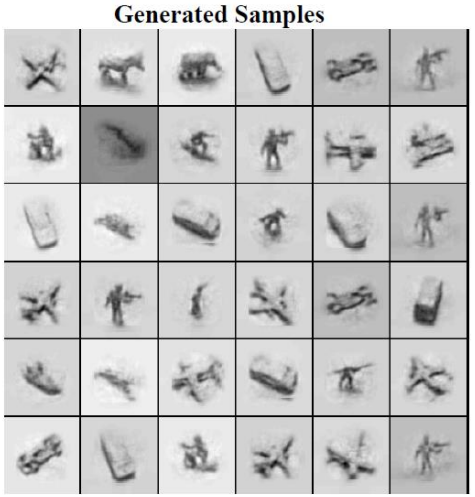


Deep Boltzmann Machines

Deep Boltzmann Machines are interesting for several reasons:

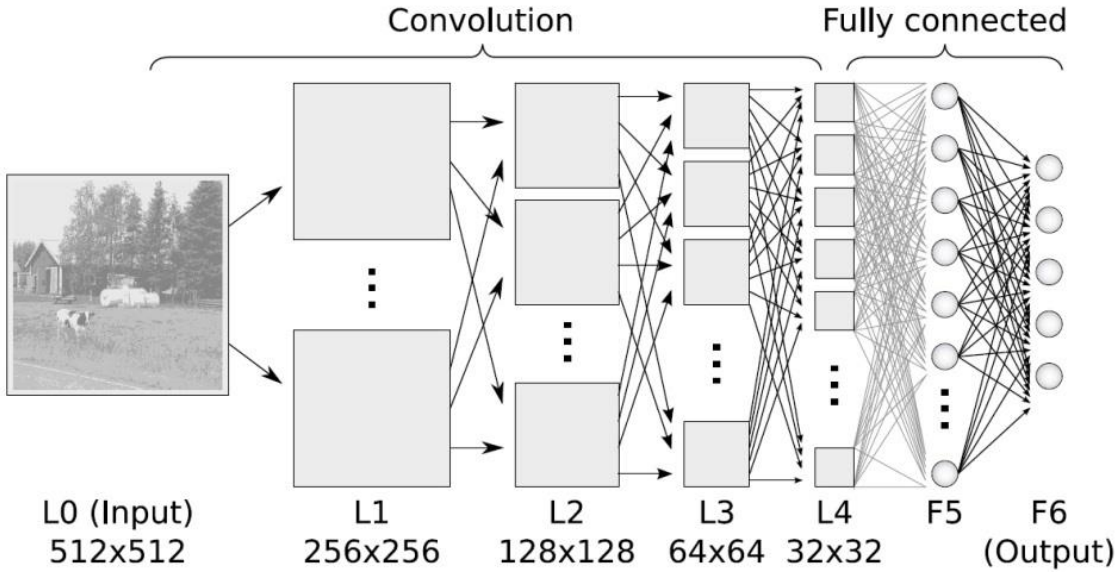
- DBM's (like Deep Belief Networks) have the potential of learning internal representations that become increasingly complex, which is considered to be a promising way of solving object and speech recognition problems.
- High-level representations can be built from a large supply of unlabeled sensory inputs and very limited labeled data can then be used to only slightly fine-tune the model for a specific task at hand.
- Unlike Deep Belief Network's, the approximate inference procedure, in addition to an initial bottom-up pass, can incorporate top-down feedback, allowing Deep Boltzmann Machine's to better propagate uncertainty about, and hence deal more robustly with, ambiguous inputs.





Convolutional Neural Networks

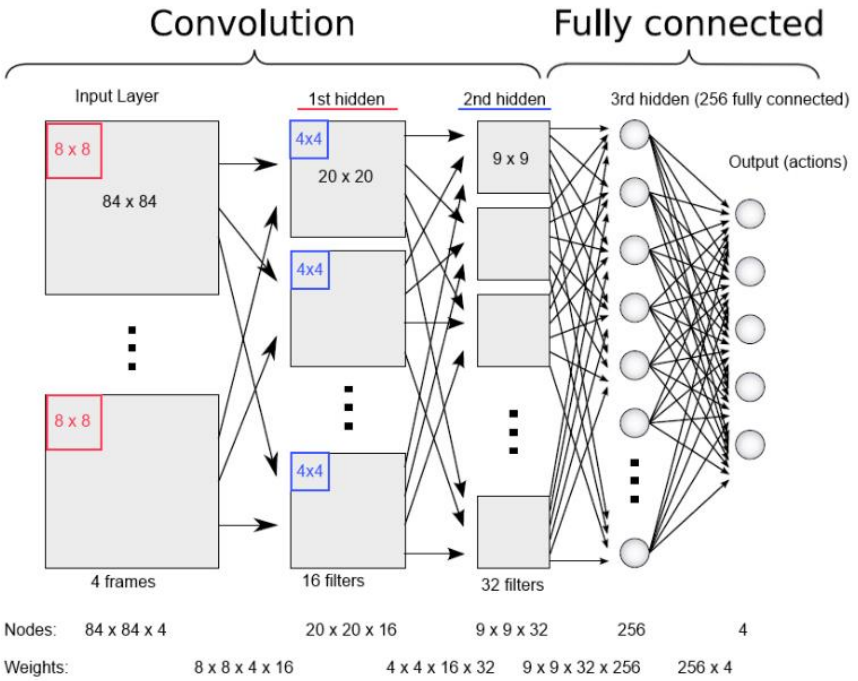
- A CNN or Convolutional Neural Network is a type of feed-forward artificial neural network where the individual neurons are tiled in such a way that they respond to overlapping regions in the visual field.



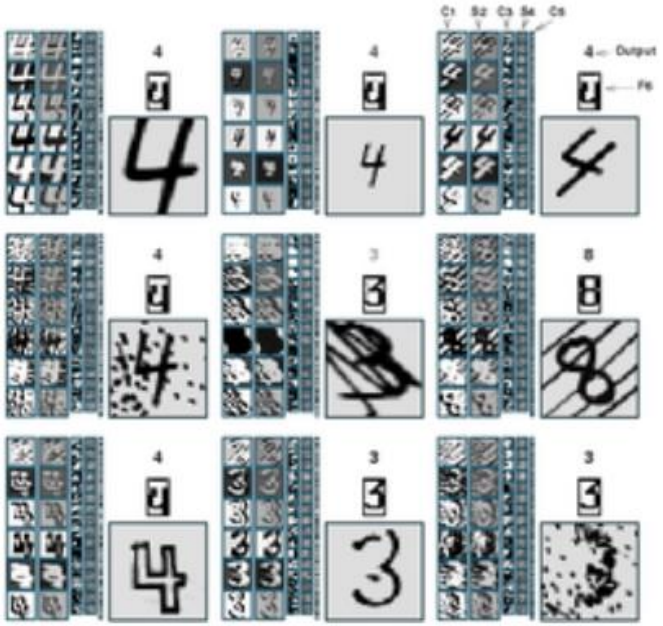
- Convolutional Neural Networks may include local or global pooling layers, which combine the outputs of neuron clusters.

Convolutional Neural Networks

- CNN's are widely used models for image and video recognition.
- When used for image recognition, CNN's consist of multiple layers of small neuron collections which look at small portions of the input image, called receptive fields.
- The results of these collections are then tiled so that they overlap to obtain a better representation of the original image; this is repeated for every such layer.
- Because of this, they are able to tolerate translation of the input image.



Convolutional Neural Networks



- Convolutional Neural Networks were inspired by biological processes and are variations of multilayer perceptron's which are designed to use minimal amounts of preprocessing.
- Convolutional Neural Network's have performed better than DBN's by themselves in current literature on benchmark computer vision datasets such as MNIST.
- If the dataset is not a computer vision one, then DBN's can most definitely perform better.
- In theory, DBN's should be the best models but it is very hard to estimate joint probabilities accurately at the moment.

Building on these Ideas

- An area of active research is within combining the strength of Convolutional Neural Networks with the strength of greedy layer wise learning procedure of the Deep Belief Networks into a Hybrid approach called "Convolutional Deep Belief Networks".
- This algorithm is showing a lot of promise and is at the forefront of modern machine learning.
- This is the essence of Deep Learning.

Unsupervised feature learning for audio classification using convolutional deep belief networks

Honglak Lee **Yan Largman** **Peter Pham**
Computer Science Department
Stanford University
Stanford, CA 94305

Convolutional Deep Belief Networks for Scalable Unsupervised Learning of Hierarchical Representations

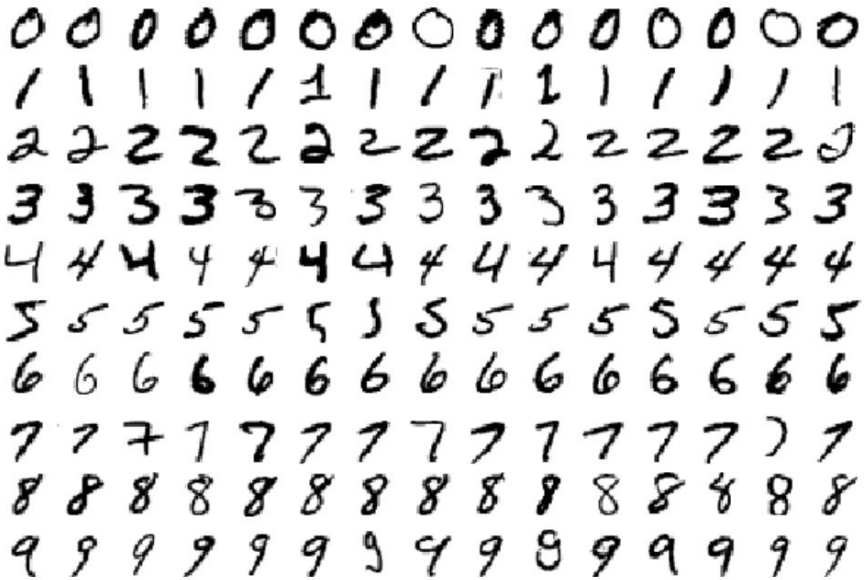
Honglak Lee HLLEE@CS.STANFORD.EDU
Roger Grosse RGROSSE@CS.STANFORD.EDU
Rajesh Ranganath RAJESHR@CS.STANFORD.EDU
Andrew Y. Ng ANG@CS.STANFORD.EDU
Computer Science Department, Stanford University, Stanford, CA 94305, USA

Practical Example – Hand Writing Recognition

Bringing It All Together

- Before we begin, I would like to revisit the start of the presentation where we went over techniques to manipulate images.
- This was intended to introduce us to the concept on learning how to transfer over an image into a format that is compatible with machine learning.
- It is important to note that we cannot just dump in an image or a video into the "black box" and hope that the machine will take care of the rest. It has to be converted and potentially reworked in order to increase the machine learning techniques effectiveness in prediction.
- The ultimate goal is to get the data into a structured format where we can begin to apply all of our tools and tricks (Ex. Dimensionality Reduction, Data munging, etc…) to get the maximum performance.
- The upcoming exercise already has the data in a useable format, however, I am certain that we can create this on our own using the R package EBImage.

Handwriting Recognition



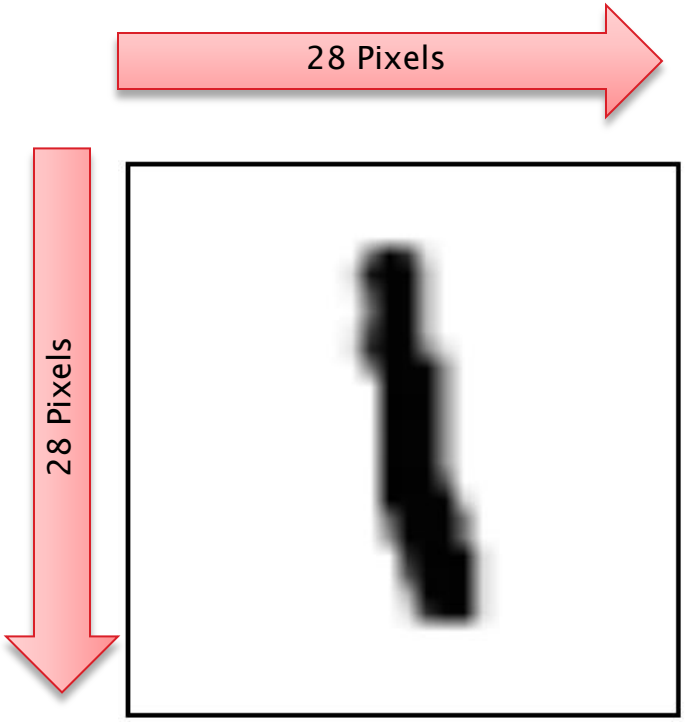
- A well known and studied database for image processing is called the MNIST – Mixed National Institute of Standards and Technology).
- This database contains 70,000 images of handwritten numbers and is widely used for machine learning and training various image processing systems.

Our goal is to devise:

- Build a Deep Learning model under the H2O Framework in R and assess the models performance.

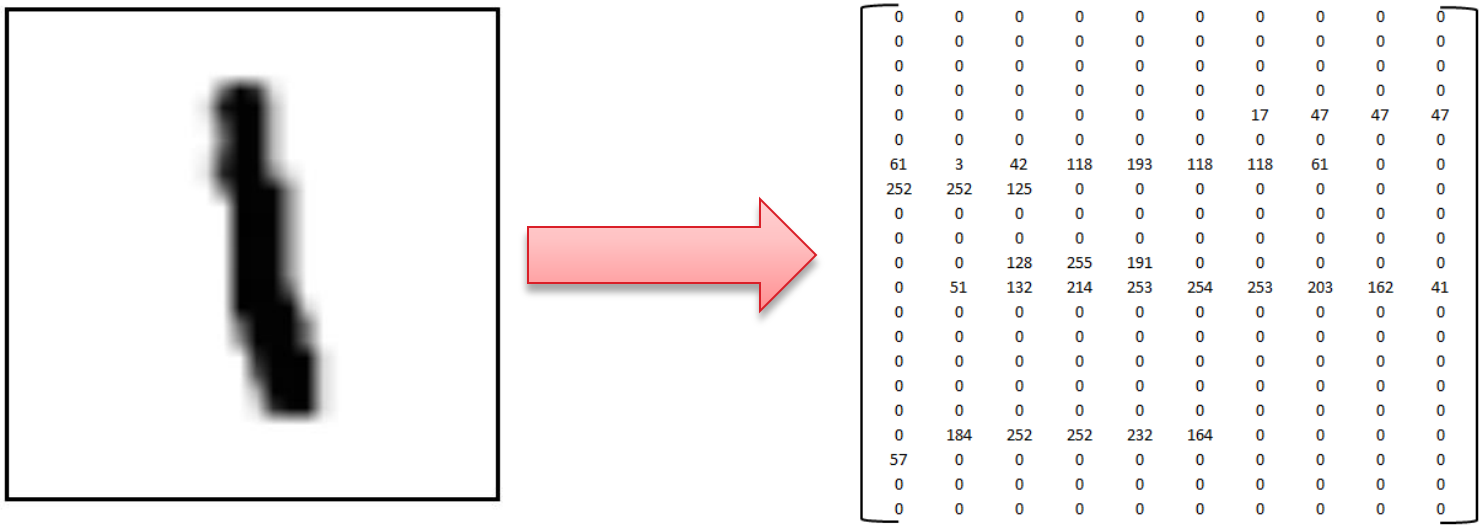
Understanding the Data

- The dataset which we are working with has already been pre-processed and ready for performance.
- However, lets dig into understanding how this came to be in order to understand the shape of the dataset (data frame).
- We know that each image is 28 x 28 pixels in size.



Understanding the Data

- Since each image has 28 by 28 pixels, we get a 28x28 array.
- Each component of the vector is a value between 0 - 255 describing the intensity of the pixel.



0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	17	47	47	47
0	0	0	0	0	0	0	0	0	0
61	3	42	118	193	118	118	61	0	0
252	252	125	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	128	255	191	0	0	0	0	0
0	51	132	214	253	254	253	203	162	41
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	184	252	252	232	164	0	0	0	0
57	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

Understanding the Data

- We can flatten each array into a 28∗28=784 dimensional vector and turn this into a data frame.

- Therefore, we should think of MNIST as being a collection of 784-dimensional vectors.

Pixel 1X1	Pixel 2X1	Pixel 3X1	Pixel 4X1	Pixel 5X1	etc...	Pixel 28X28
0	0	0	0	0	etc...	0
0	0	0	0	0	etc...	0
0	0	0	0	0	etc...	0
0	0	0	0	0	etc...	0
0	0	0	0	0	etc...	0
0	0	0	0	0	etc...	0
0	0	0	0	0	etc...	0
0	0	0	0	0	etc...	0
0	0	0	0	0	etc...	0
0	0	0	0	0	etc...	0
0	0	0	0	0	etc...	0
0	0	0	0	0	etc...	0
0	0	0	0	0	etc...	0
0	0	0	0	0	etc...	0

- If we look at each image, a vast majority of the pixels are white which has a value of 0.

Understanding the Data

- When we inspect the .csv dataset we see that there is 785 columns in total.

- The additional column contains the numeric representation of the digit, as determined by a human.

Digit	Pixel 1X1	Pixel 2X1	Pixel 3X1	Pixel 4X1	Pixel 5X1	etc…	Pixel 28X28
7	0	0	0	0	0	etc…	0
2	0	0	0	0	0	etc…	0
1	0	0	0	0	0	etc…	0
0	0	0	0	0	0	etc…	0
4	0	0	0	0	0	etc…	0
1	0	0	0	0	0	etc…	0
4	0	0	0	0	0	etc…	0
9	0	0	0	0	0	etc…	0
5	0	0	0	0	0	etc…	0
9	0	0	0	0	0	etc…	0
0	0	0	0	0	0	etc…	0
6	0	0	0	0	0	etc…	0
9	0	0	0	0	0	etc…	0
0	0	0	0	0	0	etc…	0

- Therefore, it should be apparent that we are looking into a supervised form of model building; we know the value of the digit that we are trying to predict.

H2o Preparation & Initialization

H2o Initialization and Data Import:

- The first step is to start an instance of H2o. Using the "max_mem_size" parameter in the h2o.init function we can set aside the amount of RAM we want to use.
- I have 4 GB of RAM on my machine so I allocated 1 GB to H2o.
- The "digit" variable I have must first be converted into a categorical variable through the factor function.

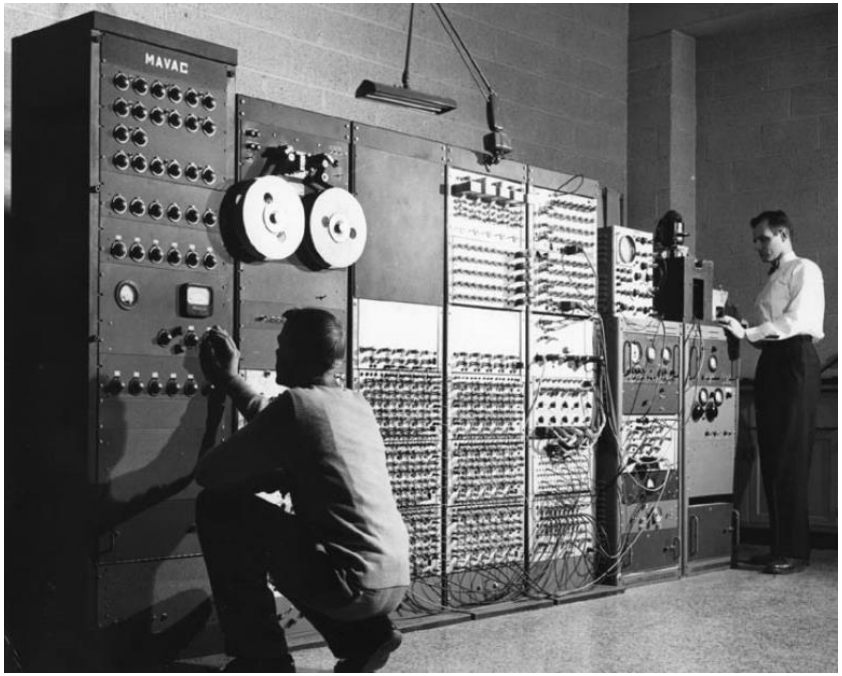
```
library(h2o)

## Start a local cluster with 1GB RAM

localH2O = h2o.init(ip = "localhost", port = 54321,
                    startH2O = TRUE, max_mem_size = '1g')

############################################################
# Load the data
############################################################

## Import MNIST CSV data as H2O

traindata<- read.csv("Deep Learning/mnist_train.csv")
testdata<- read.csv("/Deep Learning/mnist_test.csv")

# Convert categorical variable to factor

traindata$X5 <- as.factor(traindata$X5)
testdata$X7 <- as.factor(testdata$X7)

############################################################
#  Convert data table into H2O Format
############################################################

train_h2o <- as.h2o(localH2O, traindata)
test_h2o <- as.h2o(localH2O, testdata)
```

Deep Learning Model Building

Building the Model

- The demonstration model I settled on has the following attributes:
  - Tanh with dropouts as the activation function
  - Input dropout ratio of 0.2
  - Hidden dropout ratio of 0.5
  - Neuron architecture of 50, 50, 50 (784 inputs, 3 hidden layers of 50 neurons each)
  - 10 epochs

- **Note:** When defining parameters for the deep neural network a good reference would be to use many of the suggestions in Geoffrey Hinton's and Alex Krizhevsky's paper.



Deep Learning Model Building

Building the Model

```
#####################################################################
# Develop the Deep Learning Model
#####################################################################


model <- h2o.deeplearning(x = 2:785,  # column numbers for predictors
                          y = 1,   # column number for dependent variable
                          training_frame = train_h2o, # train data in H2O format
                          validation_frame = test_h2o, # test data in H2O Format
                          activation = "TanhWithDropout", # or 'Tanh'
                          input_dropout_ratio = 0.2, # % of inputs dropout
                          hidden_dropout_ratios = c(0.5,0.5,0.5), # % for nodes dropout
                          balance_classes = TRUE,
                          hidden = c(50,50,50), # three layers of 50 nodes
                          epochs = 10) # max. no. of epochs
```

Deep Learning Model Building

Model Diagnostics

- The model diagnostics gives us a sense of the predictive performance of the model through the confusion matrix and MSE on a temporary training frame.

```
H2OMultinomialMetrics: deeplearning
** Reported on training data. **
Description: Metrics reported on temporary training frame with 10117 samples

Training Set Metrics:
=====================
Metrics reported on temporary training frame with 10117 samples

MSE: (Extract with `h2o.mse`) 0.1922967
R^2: (Extract with `h2o.r2`) 0.9766816
Logloss: (Extract with `h2o.logloss`) 0.6741027
Confusion Matrix: Extract with `h2o.confusionMatrix(<model>,train=TRUE)`)
=========================================================================
         X0   X1  X2   X3   X4  X5   X6  X7  X8   X9      Error           Rate
0       938    0   9    8    2   1   26   4   3    2 0.05538771       55 / 993
1         0  893   5    9    3   1    1   3  61    5 0.08970438       88 / 981
2        40    8 745   35   28   2  138  19  38    7 0.29716981    315 / 1,060
3        27   10  25  807    3  25   10  12  20   16 0.15497382      148 / 955
4         1   11   0    0  764   1   34   0   4  159 0.21560575      210 / 974
5       144   56  11  178   72 456   53   4  19   35 0.55642023    572 / 1,028
6        25   13   3    1    4  15  971   0   1    1 0.06092843     63 / 1,034
7         9   14   3    1   13   0    1 814  14  160 0.20894072    215 / 1,029
8        27   47  18  112   24  42   21  16 724   31 0.31826742    338 / 1,062
9        12   12   1   20  104   1    2  31   7  811 0.18981019    190 / 1,001
Totals 1223 1064 820 1171 1017 544 1257 903 891 1227 0.21686271 2,194 / 10,117
```

- The MSE is at 0.192 which is not surprising given the relatively low # of epochs utilized in the training of the model. Improvements are easily made through tweaking the model parameters.

Deep Learning Model Building

Making Predictions on the Test Set

- Here is some additional code which highlights how to make predictions from the Deep Learning model.


```
##########################################
# Predicting with the Deep Learning Model
##########################################

## Using the DNN model for predictions
h2o_Yhat <- h2o.predict(model, test_h2o)

## Converting H2O format into data frame
df_yhat_test <- as.data.frame(h2o_Yhat)

# merge the dataframes for prediction
mydata <- cbind(testdata, df_yhat_test)
```

Model Results

- The Deep Learning prediction creates a probability matrix for the likelihood that the prediction falls between the number 0 – 9.

- This is particularly useful because it allows for us to create ensembles of various predictions from different algorithms (Random Forests, SVM, etc…) in order to improve general overall accuracy.

Final Prediction	Predict = 0	Predict = 1	Predict = 2	Predict = 3	Predict = 4	Predict = 5	Predict = 6	Predict = 7	Predict = 8	Predict = 9
0	98%	0%	0%	0%	0%	1%	0%	0%	0%	0%
4	0%	0%	0%	0%	66%	0%	0%	2%	0%	31%
1	0%	82%	1%	2%	0%	3%	0%	0%	12%	0%
9	0%	0%	0%	0%	7%	0%	0%	3%	0%	89%
2	0%	0%	70%	0%	1%	0%	15%	9%	3%	1%
1	0%	89%	1%	5%	0%	2%	1%	0%	2%	0%
3	0%	0%	0%	92%	0%	4%	0%	0%	3%	0%
1	0%	94%	0%	1%	0%	1%	1%	0%	1%	1%
4	0%	0%	0%	0%	97%	0%	1%	0%	0%	1%
3	0%	0%	0%	97%	0%	2%	0%	0%	1%	0%
1	0%	64%	3%	1%	0%	19%	1%	0%	11%	0%
3	0%	0%	0%	97%	0%	2%	0%	0%	0%	0%
6	0%	0%	1%	0%	1%	0%	98%	0%	0%	0%
1	0%	96%	0%	1%	0%	1%	0%	0%	1%	0%
7	0%	0%	0%	0%	1%	0%	0%	80%	0%	19%
2	1%	0%	92%	0%	0%	0%	5%	0%	1%	0%
8	0%	1%	2%	1%	0%	1%	0%	0%	96%	0%
6	0%	0%	0%	0%	1%	0%	99%	0%	0%	0%
9	0%	3%	0%	0%	5%	1%	0%	23%	0%	68%
4	0%	1%	1%	0%	51%	0%	2%	3%	2%	40%
1	0%	79%	1%	3%	0%	3%	0%	0%	14%	0%
2	8%	0%	55%	9%	0%	7%	0%	0%	21%	0%
3	1%	0%	1%	92%	0%	5%	0%	1%	1%	0%

Model Results

- The resulting predictions are then bound back to the original validation dataset and can be utilized for a variety of purposes.

Predict	Digit	Pixel 1X1	Pixel 1X2	Pixel 1X2	Pixel 1X2	etc…	Pixel 28X28
0	0	0	0	0	0	etc…	48
4	4	0	0	0	0	etc…	0
1	1	0	0	0	0	etc…	0
9	9	0	0	0	0	etc…	0
2	2	0	0	0	0	etc…	0
1	1	0	32	237	253	etc…	252
3	3	0	38	43	105	etc…	255
1	1	0	0	5	63	etc…	197
4	4	0	0	0	0	etc…	0
3	3	103	242	254	254	etc…	254
1	5	0	0	0	0	etc…	0
3	3	155	155	155	155	etc…	131
6	6	0	0	38	178	etc…	252
1	1	0	1	168	242	etc…	28
7	7	0	0	0	0	etc…	0
2	2	0	93	164	211	etc…	250
8	8	0	0	0	0	etc…	0
6	6	0	0	0	0	etc…	75
9	9	0	0	0	0	etc…	0
4	9	0	0	0	0	etc…	0
1	1	0	0	0	0	etc…	0
2	2	49	180	246	253	etc…	253
3	3	9	80	207	255	etc…	254

- Congratulations. We just successfully performed handwriting recognition using Deep Learning!!!

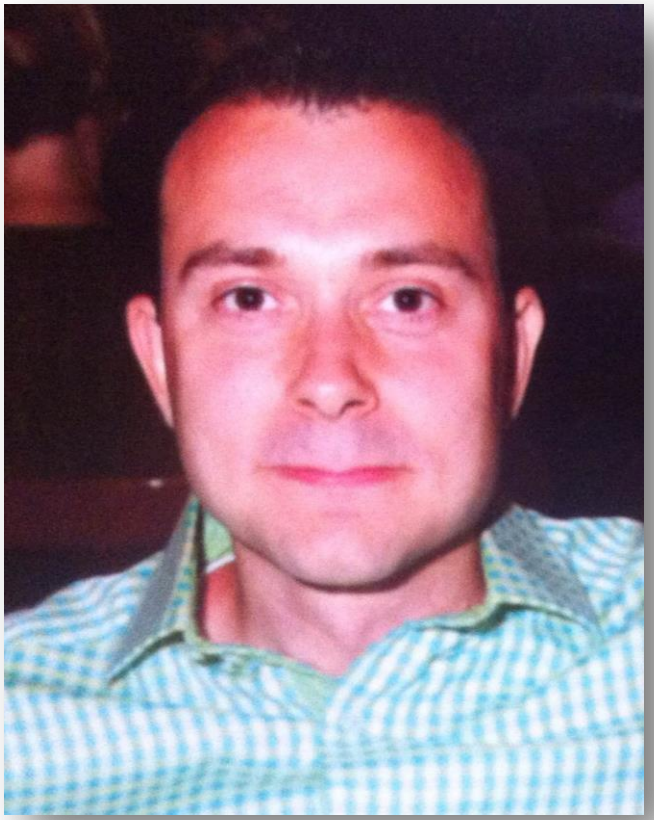
World Record Results on MNIST

- With the parameters shown below, H2O Deep Learning matched the current world record of 0.83% test set error for models without pre-processing, unsupervised learning, convolutional layers or data augmentation after running for 8 hours on 10 nodes:

```
record_model <- h2o.deeplearning(x = 1:784, y = 785, data = train_hex, validation = test_hex,
                                 activation = "RectifierWithDropout", hidden = c(1024,1024,2048)
                                 epochs = 8000, l1 = 1e-5, input_dropout_ratio = 0.2,
                                 train_samples_per_iteration = -1, classification_stop = -1)
record_model@model$confusion
         Predicted
Actual     0    1    2    3   4   5   6    7   8    9 Error
  0      974    1    1    0   0   0   2    1   1    0 0.00612
  1        0 1135    0    1   0   0   0    0   0    0 0.00088
  2        0    0 1028    0   1   0   0    3   0    0 0.00388
  3        0    0    1 1003   0   0   0    3   2    1 0.00693
  4        0    0    1    0 971   0   4    0   0    6 0.01120
  5        2    0    0    5   0 882   1    1   1    0 0.01121
  6        2    3    0    1   1   2 949    0   0    0 0.00939
  7        1    2    6    0   0   0   0 1019   0    0 0.00875
  8        1    0    1    3   0   4   0    2 960    3 0.01437
  9        1    2    0    0   4   3   0    2   0  997 0.01189
  Totals 981 1142 1038 1013 977 891 956 1031 964 1007 0.00830
```

About Me

- Reside in Wayne, Illinois
- Active Semi-Professional Classical Musician (Bassoon).
- Married my wife on 10/10/10 and been together for 10 years.
- Pet Yorkshire Terrier / Toy Poodle named Brunzie.
- Pet Maine Coons' named Maximus Power and Nemesis Gul du Cat.
- Enjoy Cooking, Hiking, Cycling, Kayaking, and Astronomy.
- Self proclaimed Data Nerd and Technology Lover.



Acknowledgements

- http://www.toptal.com/machine-learning/an-introduction-to-deep-learning-from-perceptrons-to-deep-networks
- http://freakonometrics.blog.free.fr/index.php?post/2012/04/18/foundwaldo
- http://cran.r-project.org/web/views/MedicalImaging.html
- http://deeplearning.net/tutorial/rbm.html
- https://en.wikipedia.org/wiki/Deep_learning
- http://googleblog.blogspot.com/2012/06/using-large-scale-brain-simulations-for.html
- https://gigaom.com/2014/07/02/researchers-say-deep-learning-can-ease-discovery-of-higgs-bosons-and-other-particles/
- http://www.utstat.toronto.edu/~rsalakhu/papers/dbm.pdf
- http://colah.github.io/posts/2014-10-Visualizing-MNIST/
- http://didericksen.github.io/deeplearning-r-h2o/
- Object detection using Given Key Words, James Guevara and Ankit Gupta
- Deep Learning and Application in Neural Networks, Andrew Ng, Geoffrey Hinton, et al.
- Deep Machine Learning—A New Frontier in Artificial Intelligence Research, Itamar Arel, Derek Rose, & Thomas Karnowski.
- How to win data science competitions with Deep Learning, Arno Candel
- Tutorial on Deep Learning and Applications, Honglak Lee

Fine