Deep Learning with Keras and R

Feb 15, 2018

RajivShah.com
rshah@pobox.com
 github.com/rajshah4/image_keras

 rajcs4


Hotdog!
Share
No Thanks


Not hotdog!
Share
No Thanks

decryptr

https://github.com/decryptr/decryptr

generate startup names

CALLER	A SIMPLE, AND INTUITIVE VISUAL INSTRUCTION, BASED ON 56+ PHOTO PROPERTIES.
DEARDHEAR	BUILDS WORKFLOW & COLLABORATION SOFTWARE FOR SMALL AND MID-MARKET BUSINESSES.
R8VETURE HEALTH	PROVIDES NATIVE VIDEO ADVERTISING EXPERIENCE FOR KIDS, COUPLES AND ATTORNEYS.
STRICWITEA	ENABLES LOCAL GOVERNMENT TO ACCEPT AND CURATE DIGITAL CONTENT.
HIMY	IF YOUR STARTUP IS LOOKING FOR A SHARE OF THEIR TEMPORARY STAFFING NEEDS.
HYPART	WE'RE A B2B MARKETPLACE FOR INSURANCE SECURITIZATION.
LATT	TABLETS ON RESTAURANT TABLES, SO GUESTS CAN ORDER, PAY AND PLAY GAMES FROM THEIR SMARTPHONE IN LESS THAN 5 MINUTES.
AIYDLRMSS	FLOYD IS HEROKU FOR MACHINE LEARNING TO HELP YOU GET FEEDBACK AND ITERATE ON THEIR DESIGNS.
KIPPI	WE BUILD CROWDFUNDING SITES THAT LETS WOMEN RENT DRESSES FROM EACH OTHER.
TATT	CCG EMPOWERS CLINICIANS TO MAKE PR SCALE IN THE REAL WORLD.

goals

History of Keras for R

Use Keras for:

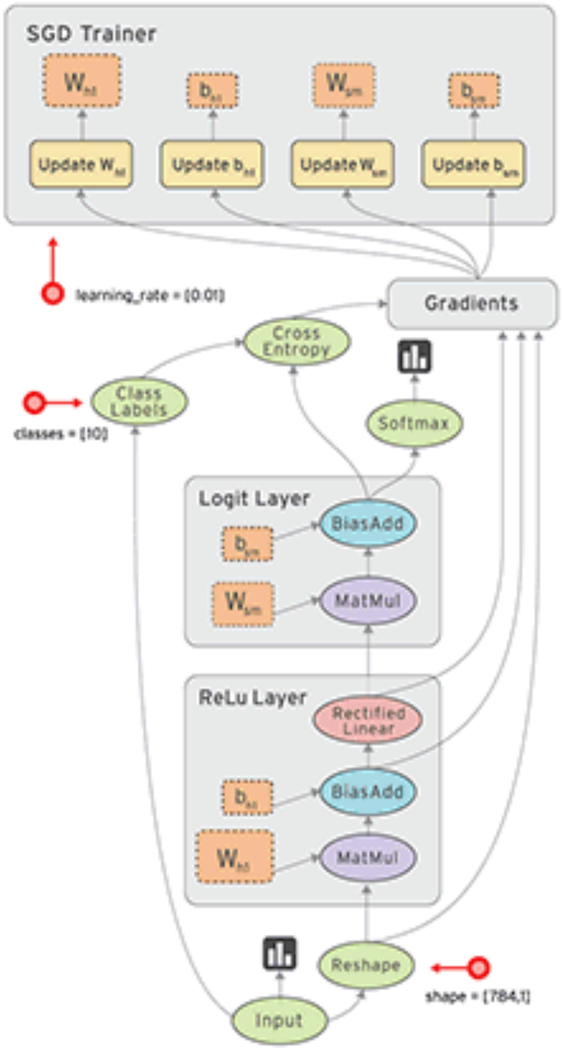
Telling 🐱 from 🐶

😻 . . . 😽 . . . 🐱🐱🐱

Generating new 🐱

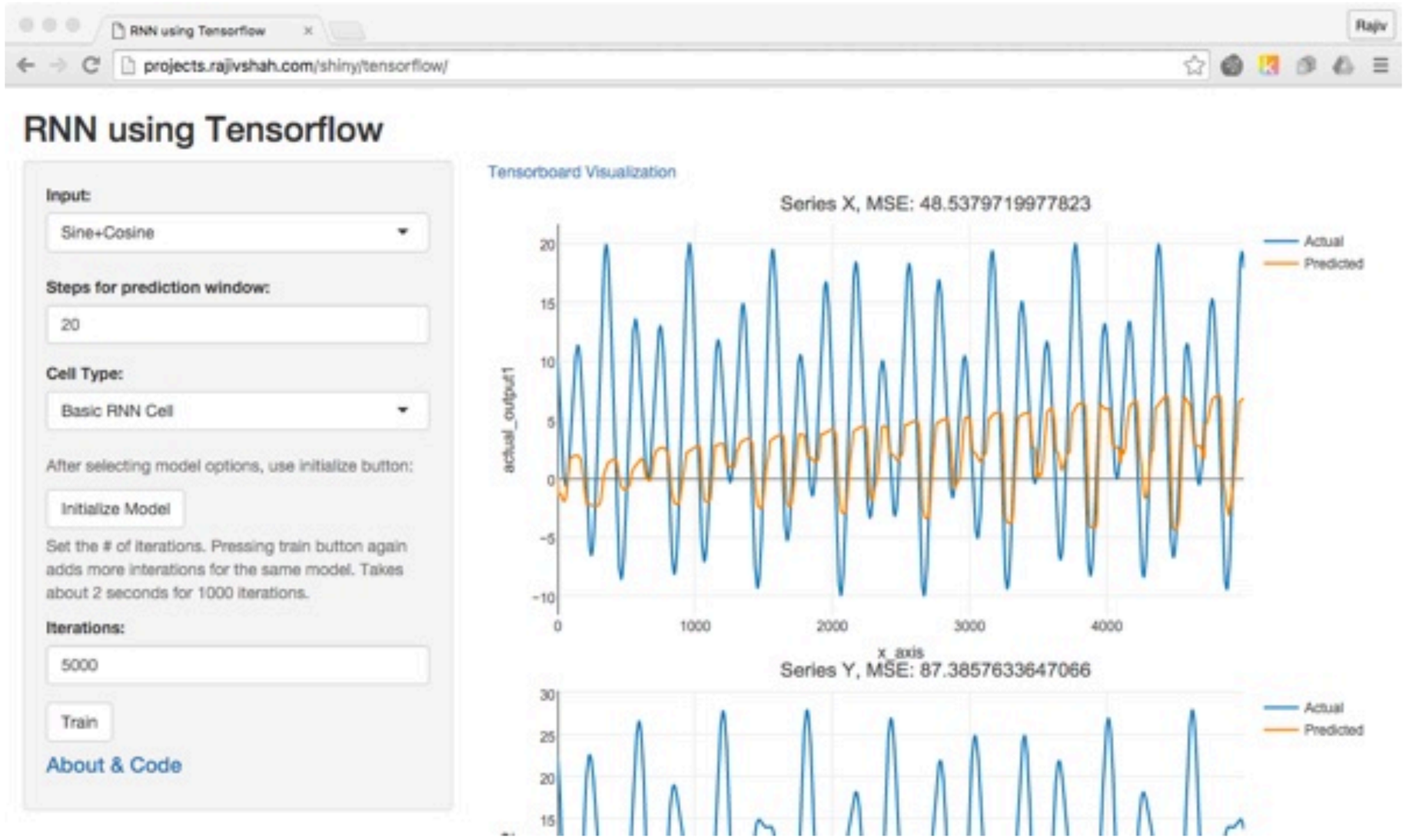
Python & R interfaces

Tensorflow



https://rviews.rstudio.com/2018/02/14/deep-learning-rstudio-conf-2018/

rPython (c. 2010)



https://github.com/rajshah4/tensorflow_shiny/

rPython (c. 2010)

```
iw <- as.numeric(input$input_wave)-1
python.assign( "input_wave", iw )
python.assign( "prlag", input$prlag )
python.exec("
            lag = prlag
            def get_sample():
            global angle1, angle2
            angle1 += 2*pi/float(frequency1)
            angle2 += 2*pi/float(frequency2)
            angle1 %= 2*pi
            angle2 %= 2*pi
            return array([array([
            5 + 5*sin(angle1) + 10*cos(angle2)**input_wave,
            7 + 7*sin(angle2)**input_wave + 14*cos(angle1)])])
            “)
```

Tensorflow for R (Oct. 2016)

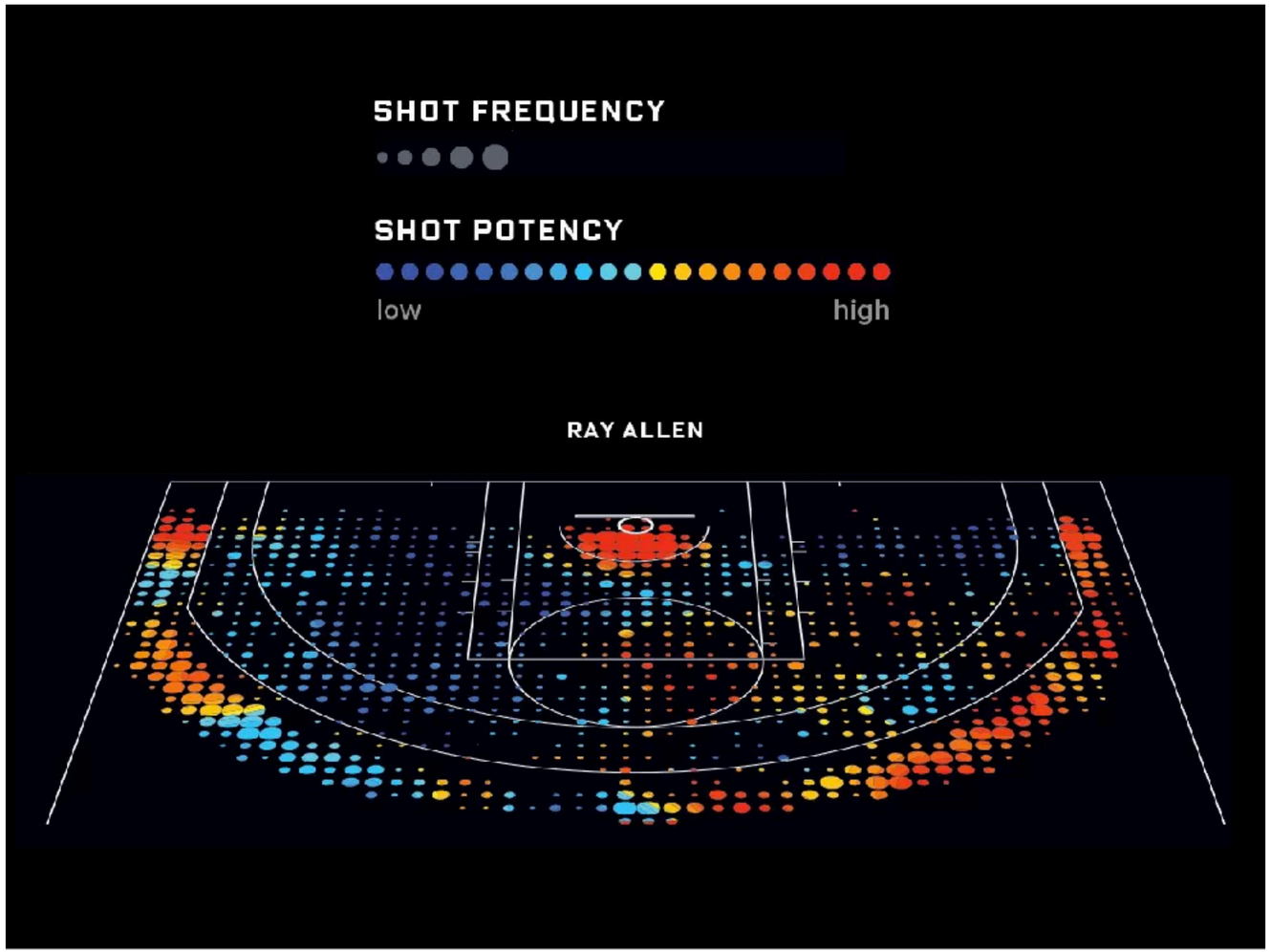
```
cross_entropy <- tf$reduce_mean(-tf$reduce_sum(y_ * tf$log(y_conv),
reduction_indices=1L))

train_step <- tf$train$AdamOptimizer(1e-4)$minimize(cross_entropy)

correct_prediction <- tf$equal(tf$argmax(y_conv, 1L), tf$argmax(y_, 1L))

accuracy <- tf$reduce_mean(tf$cast(correct_prediction, tf$float32))
sess$run(tf$global_variables_initializer())
```

Reticulate (Feb. 2017)

The `reticulate` package provides a very clean & concise interface bridge between R and Python which makes it handy to work with modules that have yet to be ported to R (going native is always better when you can do it). This post shows how to use `reticulate` to create parquet files directly from R using `reticulate` as a bridge to the `pyarrow` module, which has the ability to natively create parquet files.

spacyr: an R wrapper for spaCy

Reticulate (Feb. 2017)



https://randomnerds.com/how-to-create-nba-shot-charts-in-python-just-like-kirk-goldsberry/

Reticulate (Feb. 2017)

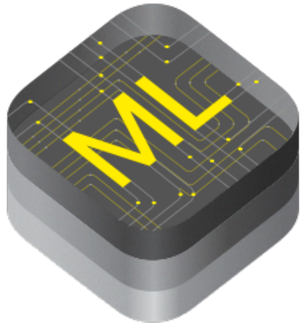
```
library(reticulate)

reticulate::import('goldsberry') -> g
df <- g $
  PlayerList(Season = '2016-17') $
  players() %>%
  bind_rows
```

https://medium.com/@InDataLabs/r-vs-python-729ceaa15620

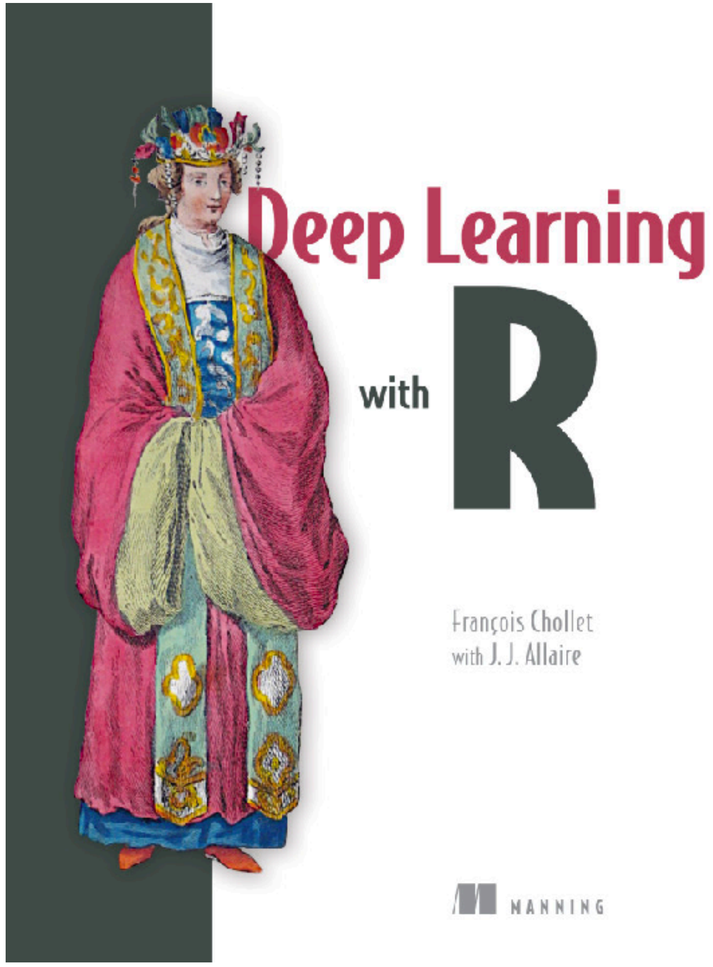
Room for both

Keras

High Level Neural Network API



the book


Deep Learning
with R
François Chollet
with J. J. Allaire
MANNING

faq: keras/r

Install: library(keras)

Works with GPU

You can set locations of python/tensorflow – extensive instructions for installing/ configuring tensorflow

Approximately the same speed as python

Awesome work by J.J. Allaire

Telling  from 

Telling 🐱 from 🐶

In 2013 - 82.7%
CNN - 99%

https://github.com/rajshah4/image_keras

goals:  from 

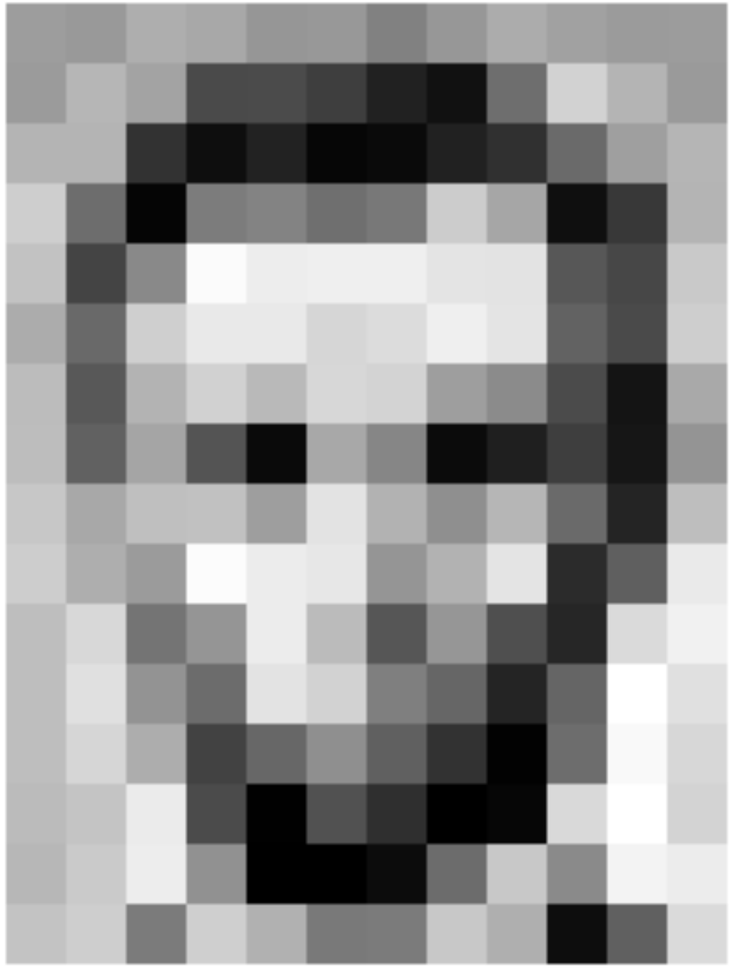
Build a simple convolutional neural network

Augment data

Use a pretrained convolutional neural network

Use transfer learning (fine tuning a pretrained network)

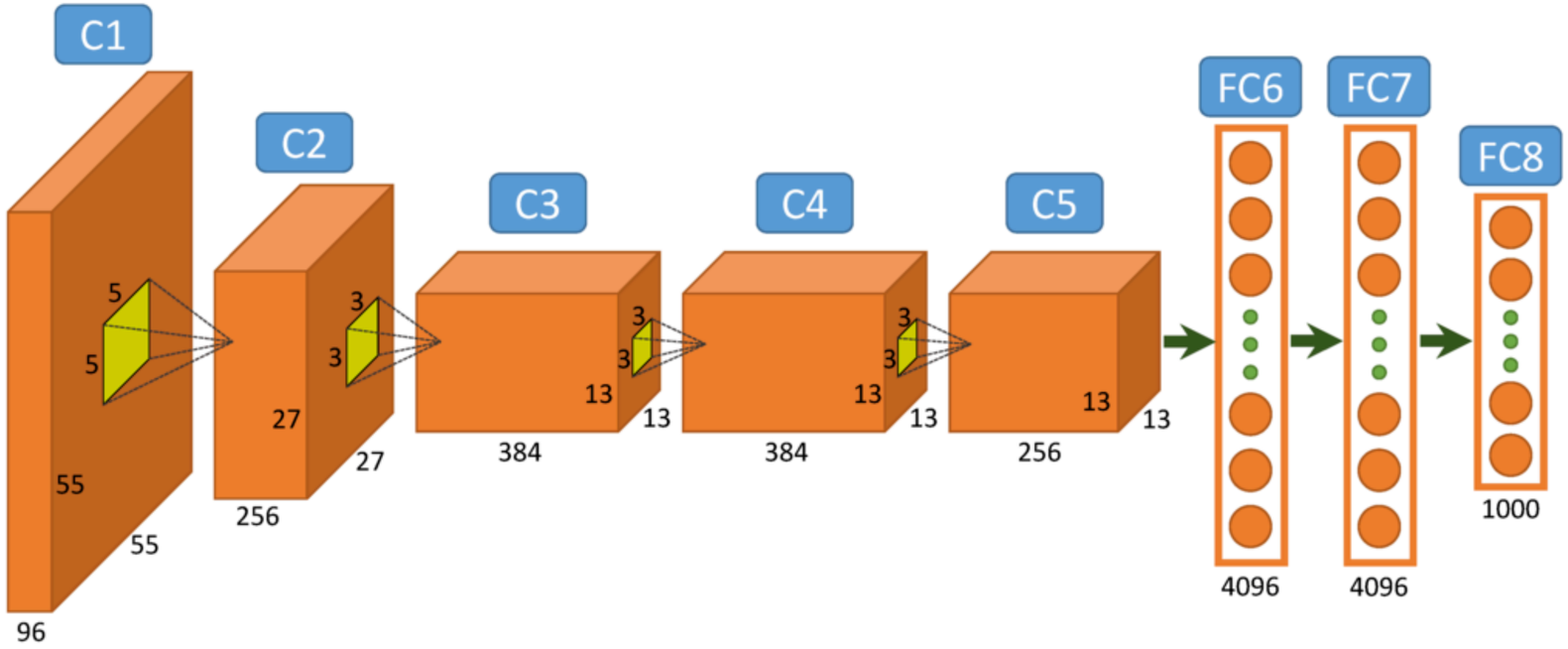
Computer vision:



157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218

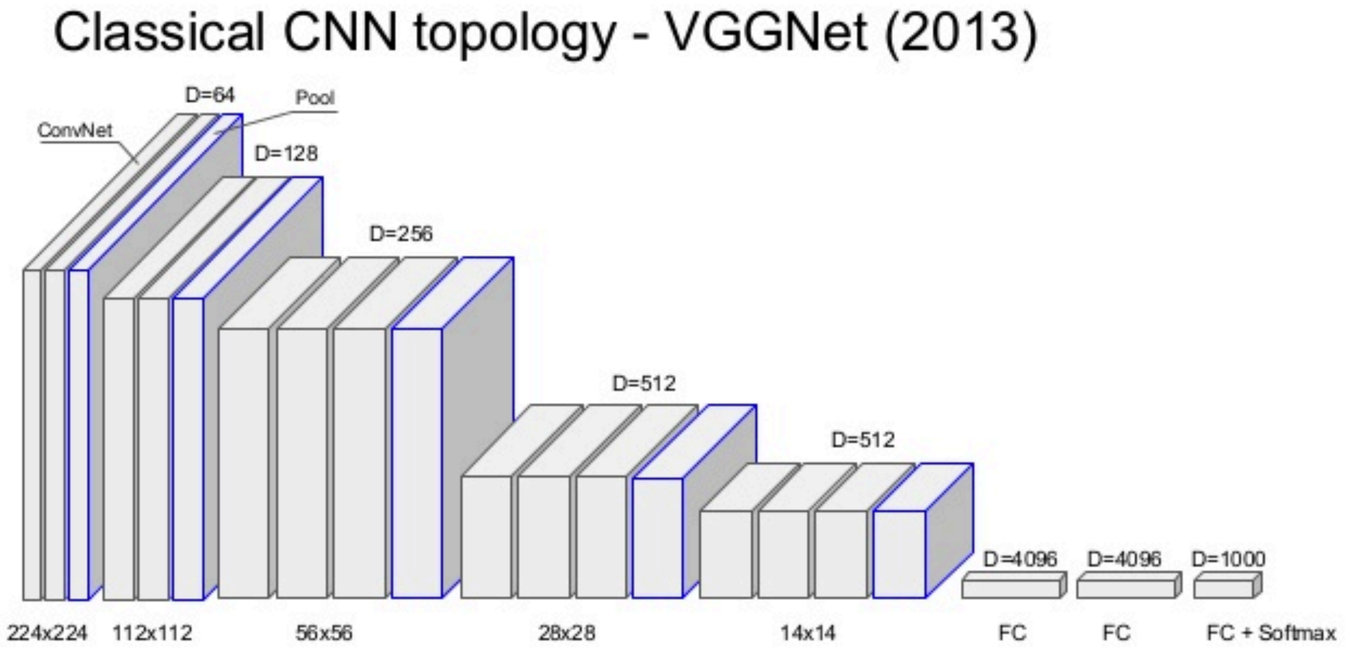
157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218

Alexnet architecture

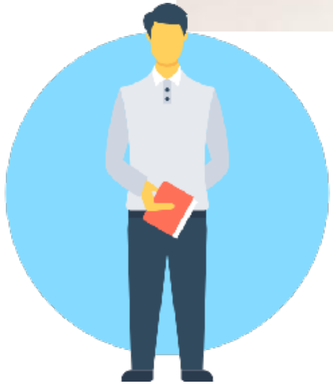


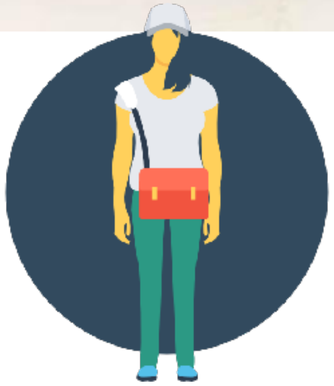
https://www.saagie.com/blog/object-detection-part1

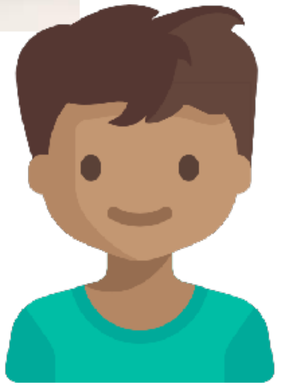
VGG architecture

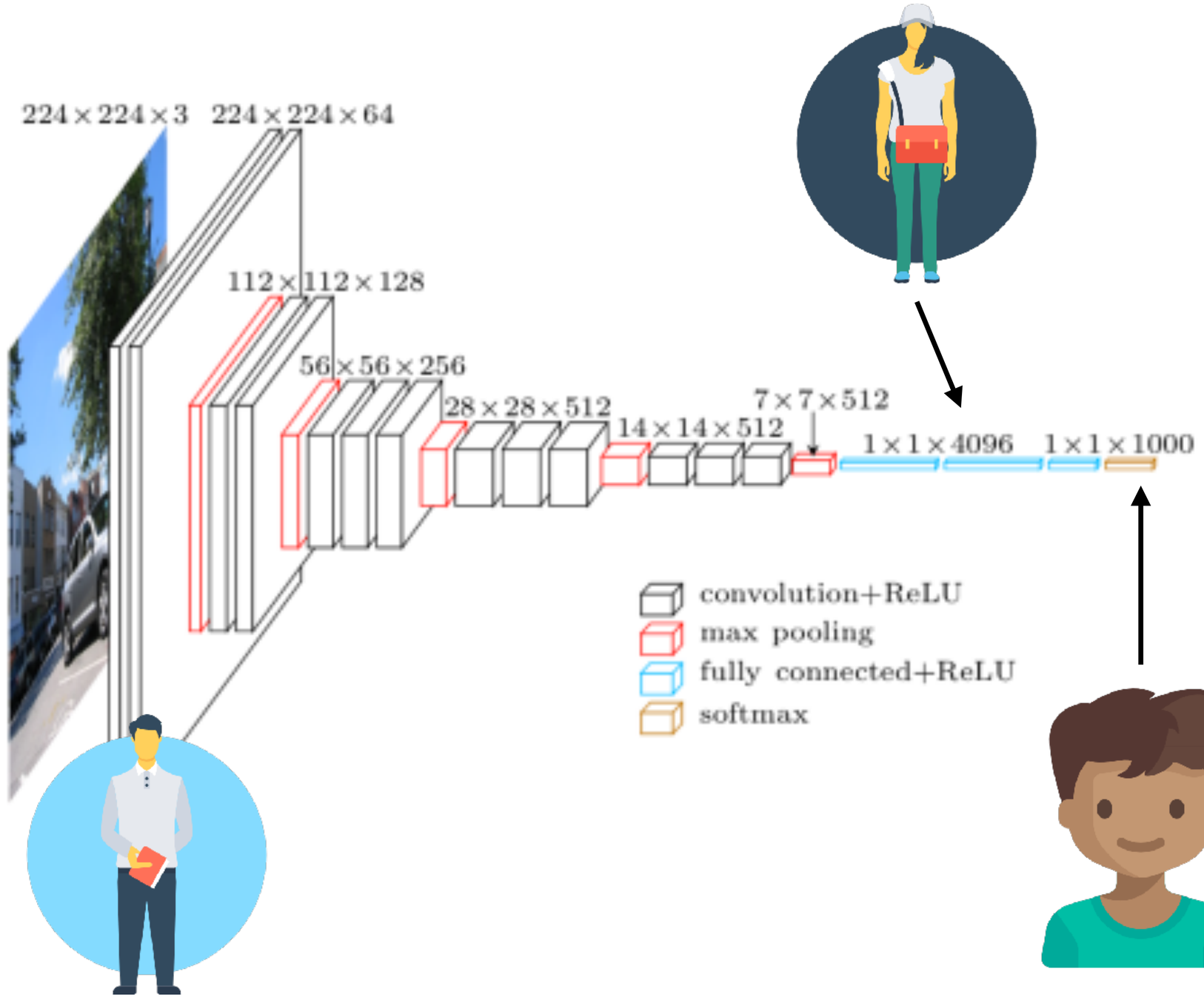


https://www.saagie.com/blog/object-detection-part1

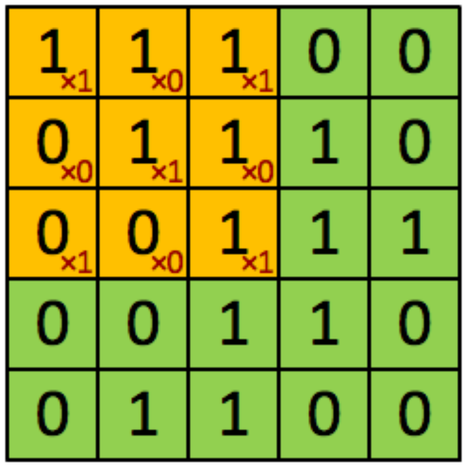


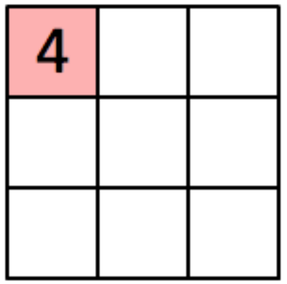





224 × 224 × 3
224 × 224 × 64
112 × 112 × 128
56 × 56 × 256
28 × 28 × 512
14 × 14 × 512
7 × 7 × 512
1 × 1 × 4096
1 × 1 × 1000
convolution+ReLU
max pooling
fully connected+ReLU
softmax

understanding convolution





http://www.wildml.com/2015/11/understanding-convolutional-neural-networks-for-nlp/

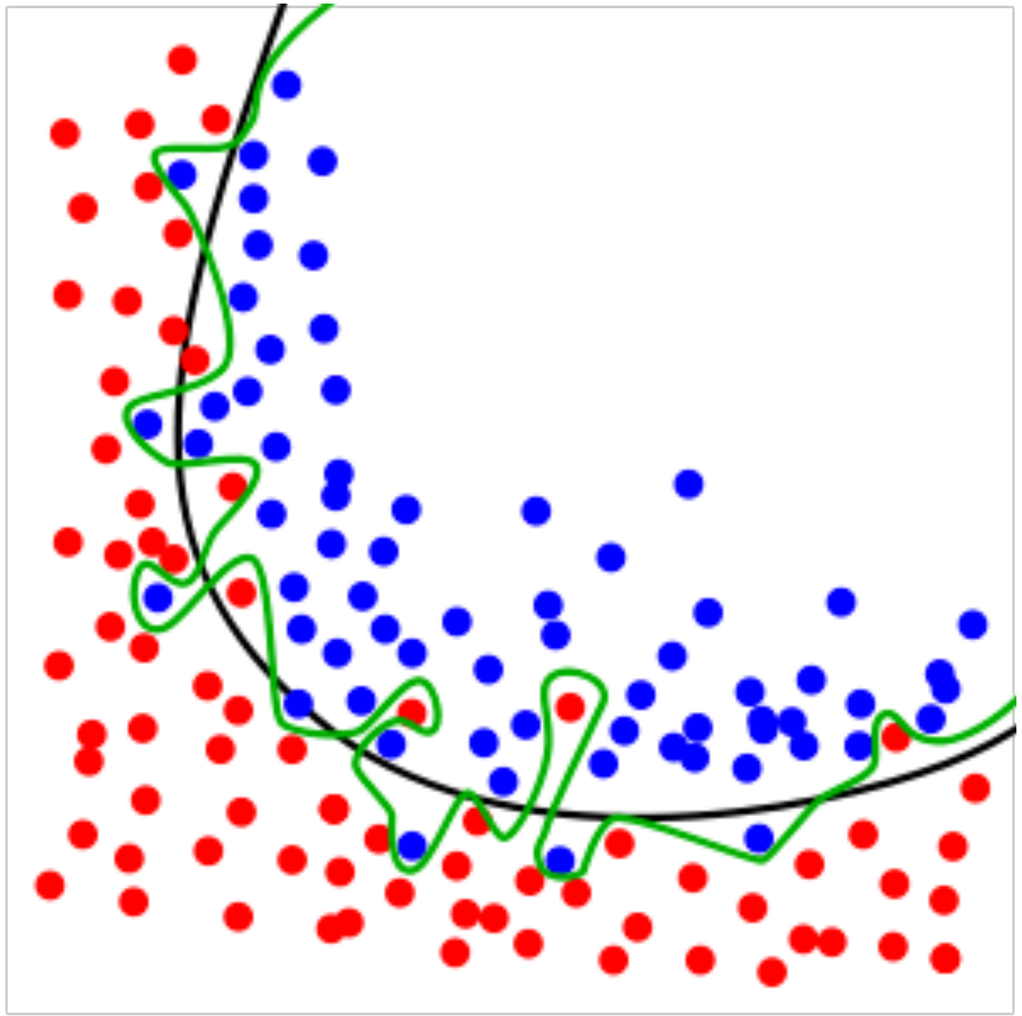
Simple convolutional neural network

```
model %>%
  layer_conv_2d(filter = 32, kernel_size = c(3,3), input_shape = c(img_width,
img_height, 3)) %>%
  layer_activation("relu") %>%
  layer_max_pooling_2d(pool_size = c(2,2)) %>%

  layer_conv_2d(filter = 32, kernel_size = c(3,3)) %>%
  layer_activation("relu") %>%
  layer_max_pooling_2d(pool_size = c(2,2)) %>%

  layer_conv_2d(filter = 64, kernel_size = c(3,3)) %>%
  layer_activation("relu") %>%
  layer_max_pooling_2d(pool_size = c(2,2)) %>%

  layer_flatten() %>%
  layer_dense(64) %>%
  layer_activation("relu") %>%
  layer_dropout(0.5) %>%
  layer_dense(1) %>%
  layer_activation("sigmoid")
```

augmenting data

overfitting (tanks)

overfitting



augmentation

Data augmentation for improving the model

By applying random transformation to our train set, we artificially enhance our dataset with new unseen images.

shear_range: Rotate image by 0.2
zoom_range: Zoom the image by 0.2
horizontal_flip: Randomly flip inputs horizontall
rescale: rescaling factor (1./255)

To augment data:

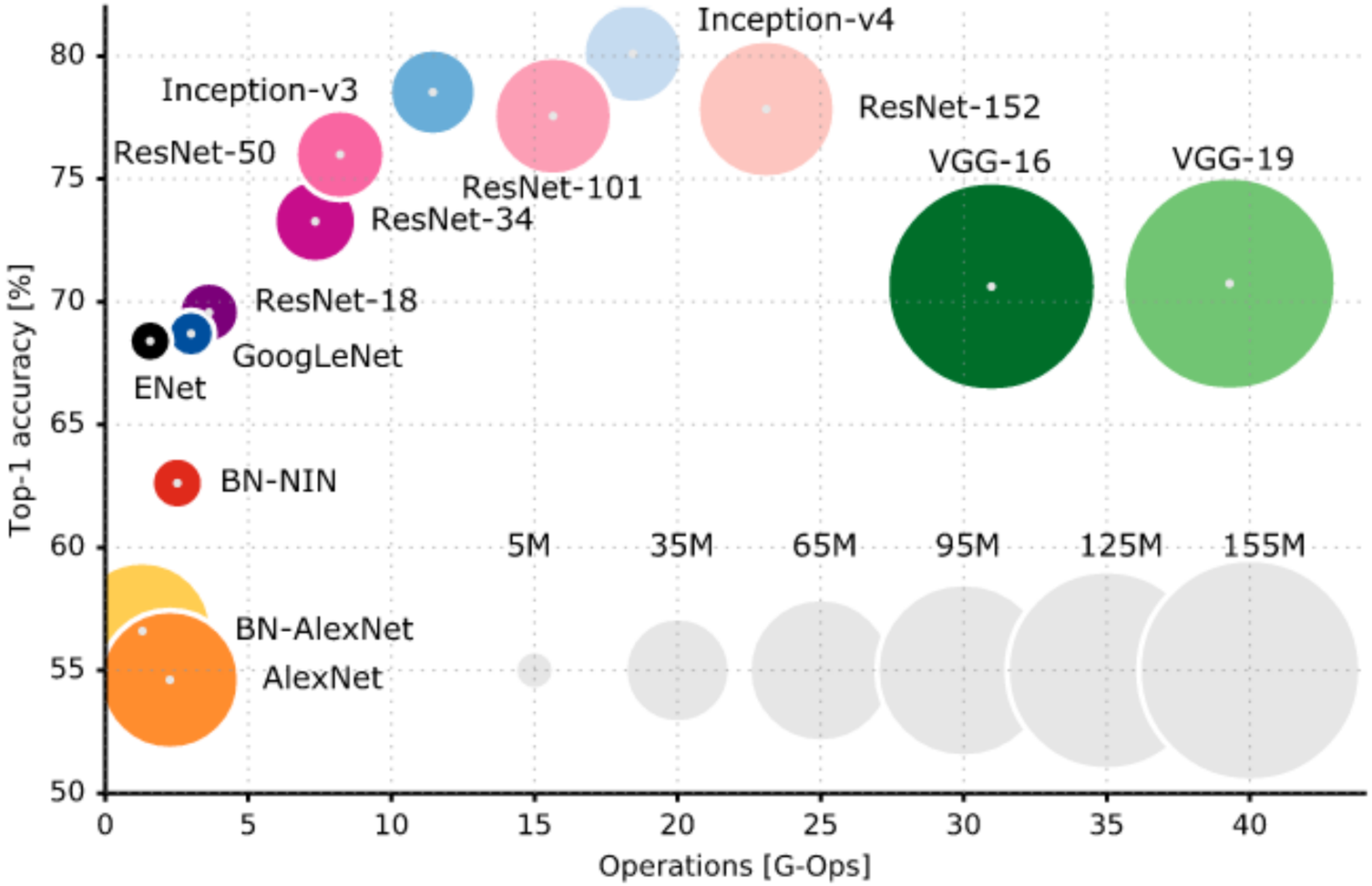
```
augment <- image_data_generator(rescale=1./255,
                                shear_range=0.2,
                                zoom_range=0.2,
                                horizontal_flip=TRUE)
```

using a pre-trained network

IMAGENET

138 GB, 14 million images

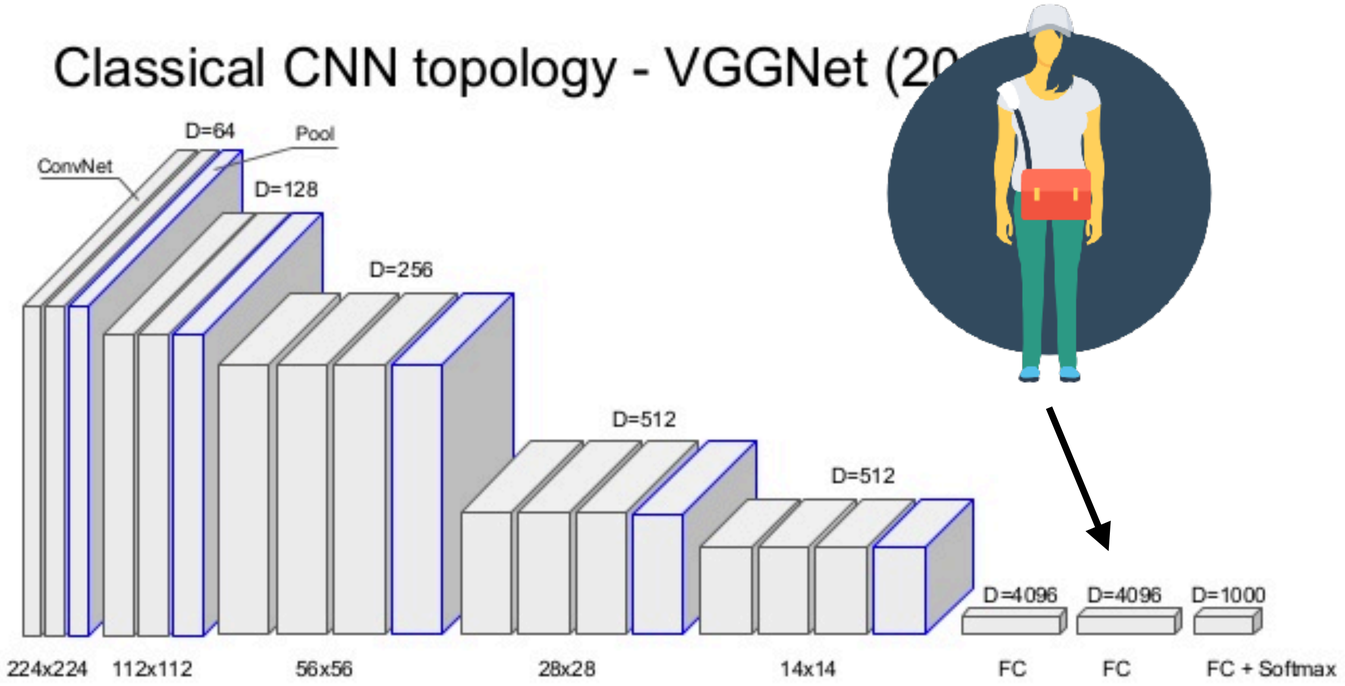
pre-trained networks



Load a pretrained network:

```
model_vgg <- application_vgg16(include_top = FALSE, weights = "imagenet")
```

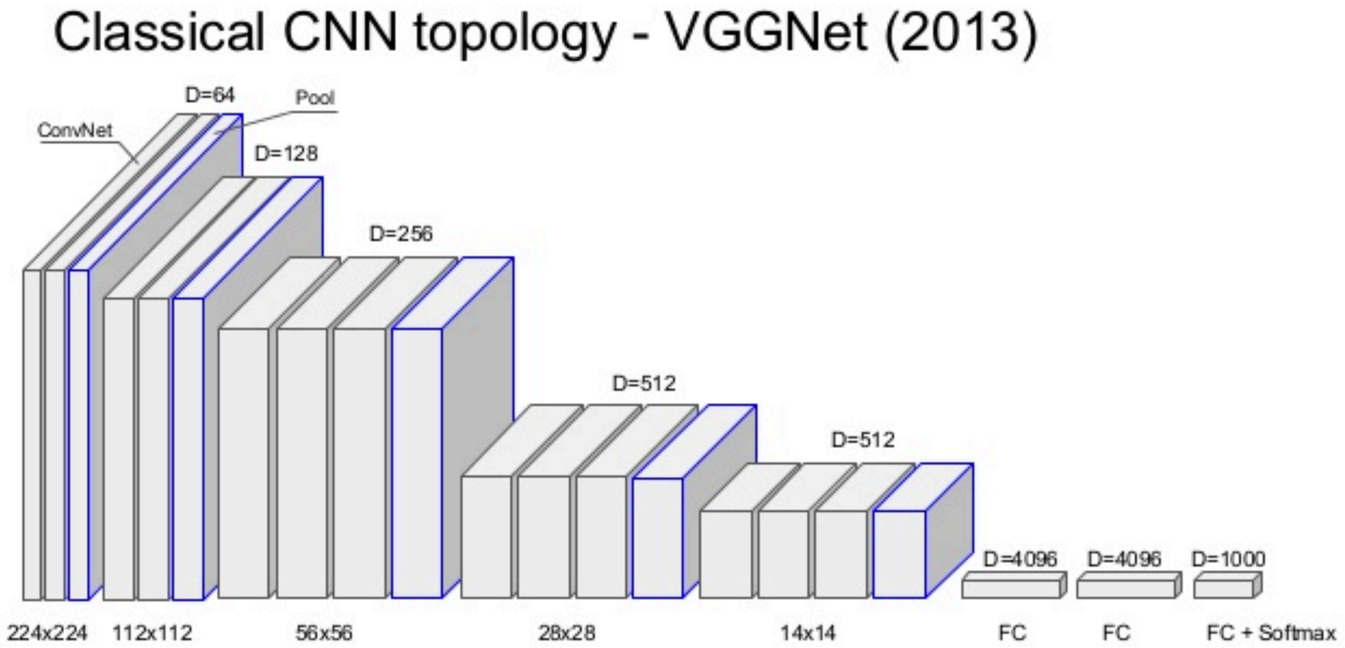
VGG architecture



https://www.saagie.com/blog/object-detection-part1

fine tuning a pre-trained network

VGG architecture



https://www.saagie.com/blog/object-detection-part1

Train only last few layers:

```
for (layer in model_ft$layers[1:16])
 layer$trainable <- FALSE
```

Complete Notebook

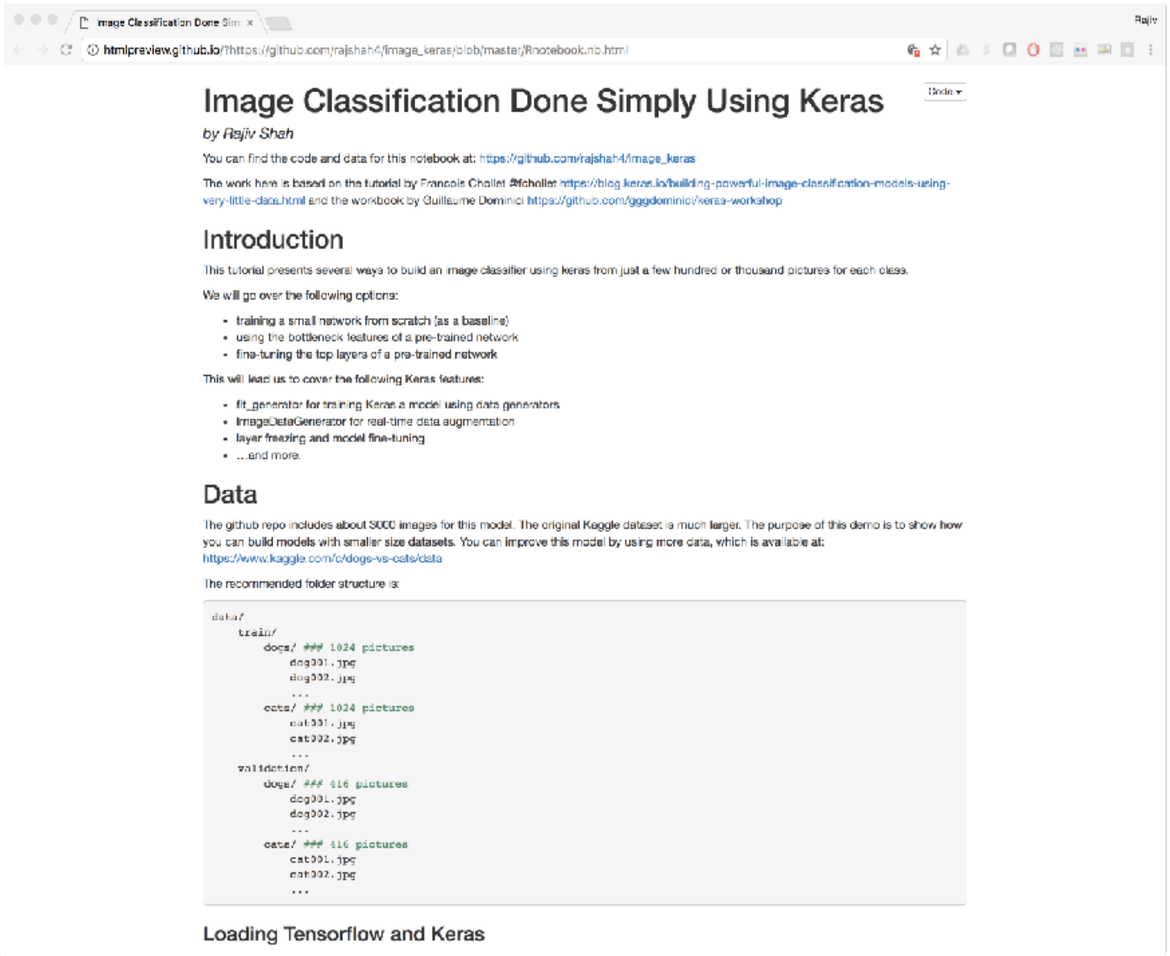


Image Classification Done Simply Using Keras

htmlpreview.github.io/?https://github.com/rajshah4/image_keras/blob/master/Rnotebook.nb.html

Image Classification Done Simply Using Keras

by Rajiv Shah

You can find the code and data for this notebook at: https://github.com/rajshah4/image_keras

The work here is based on the tutorial by Francois Chollet @fchollet https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html and the workbook by Guillaume Dominici https://github.com/gggdominici/keras-workshop

Introduction

This tutorial presents several ways to build an image classifier using keras from just a few hundred or thousand pictures for each class.

We will go over the following options:

- training a small network from scratch (as a baseline)
- using the bottleneck features of a pre-trained network
- fine-tuning the top layers of a pre-trained network

This will lead us to cover the following Keras features:

- fit_generator for training Keras a model using data generators
- ImageDataGenerator for real-time data augmentation
- layer freezing and model fine-tuning
- ...and more.

Data

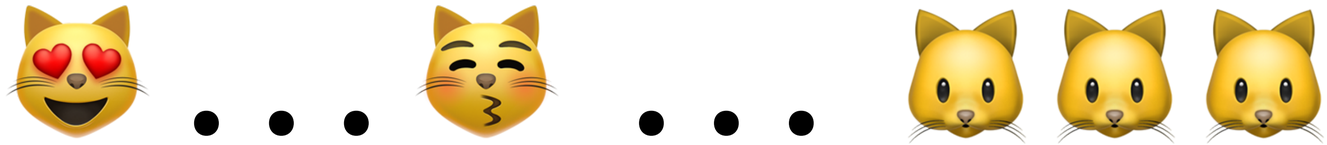
The github repo includes about 3000 images for this model. The original Keggle dataset is much larger. The purpose of this demo is to show how you can build models with smaller size datasets. You can improve this model by using more data, which is available at: https://www.kaggle.com/c/dogs-vs-cats/data

The recommended folder structure is:

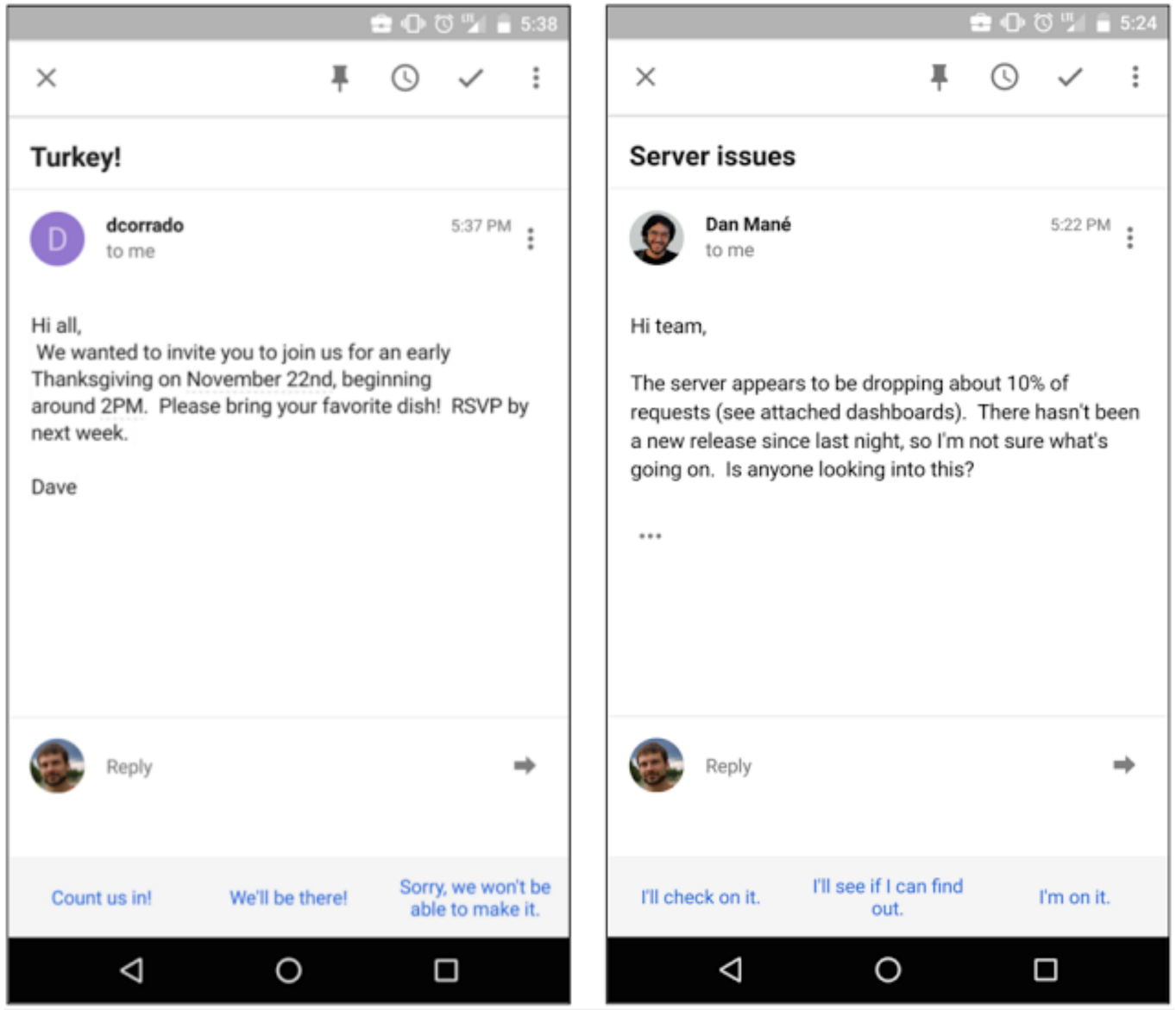
```
data/
    train/
        dogs/ ### 1024 pictures
            dog001.jpg
            dog002.jpg
            ...
        cats/ ### 1024 pictures
            cat001.jpg
            cat002.jpg
            ...
    validation/
        dogs/ ### 416 pictures
            dog001.jpg
            dog002.jpg
            ...
        cats/ ### 416 pictures
            cat001.jpg
            cat002.jpg
            ...
```

Loading Tensorflow and Keras

https://github.com/rajshah4/image_keras

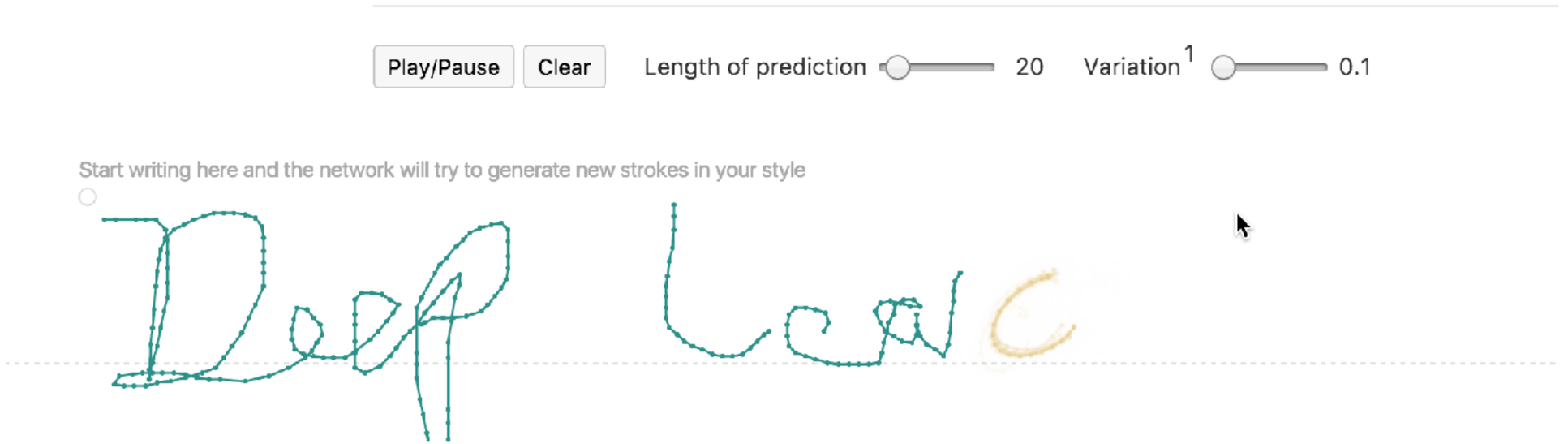


smart reply



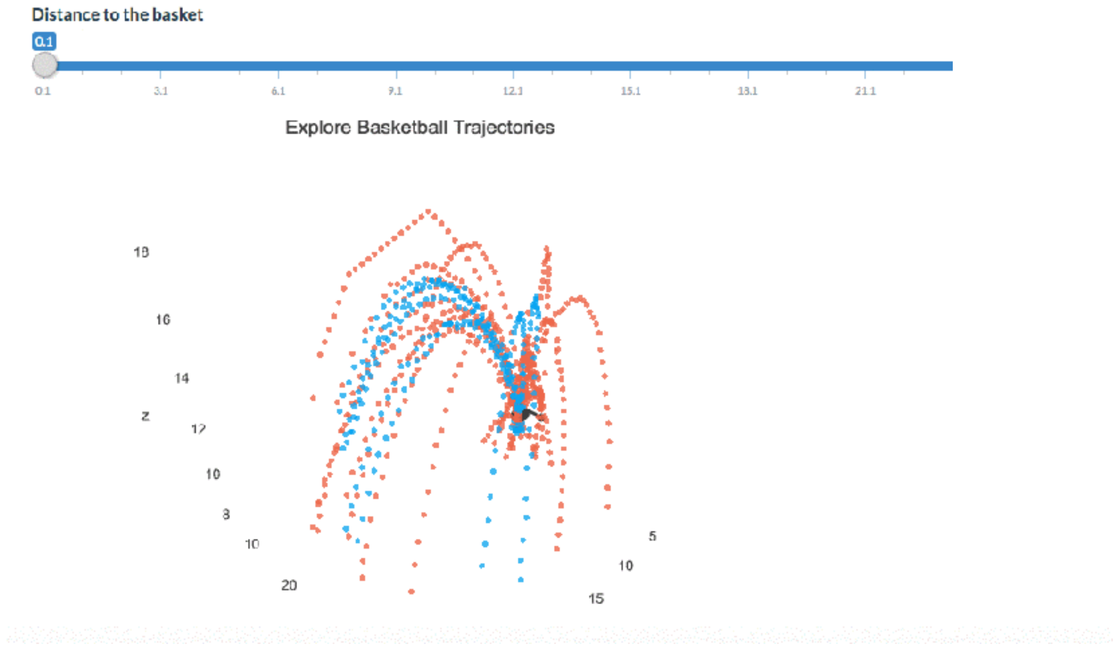
http://googleresearch.blogspot.com/2015/11/computer-respond-to-this-email.html

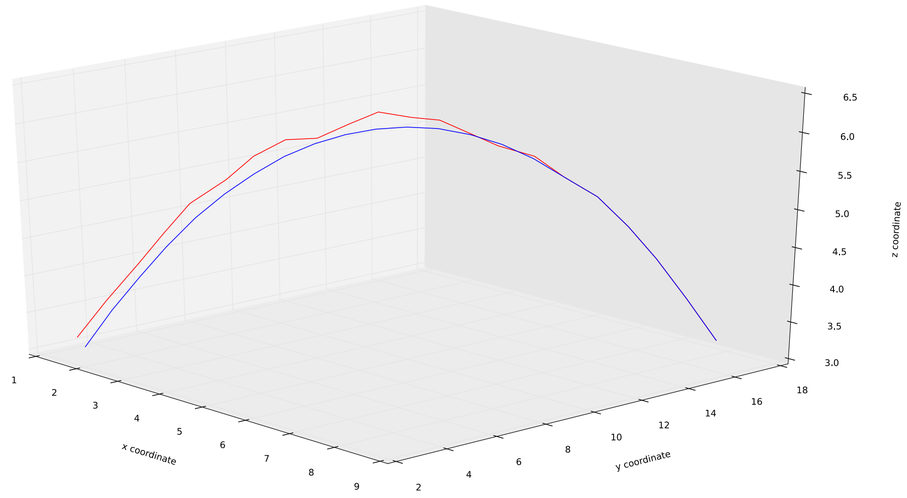
demo: handwriting



http://distill.pub/2016/handwriting/

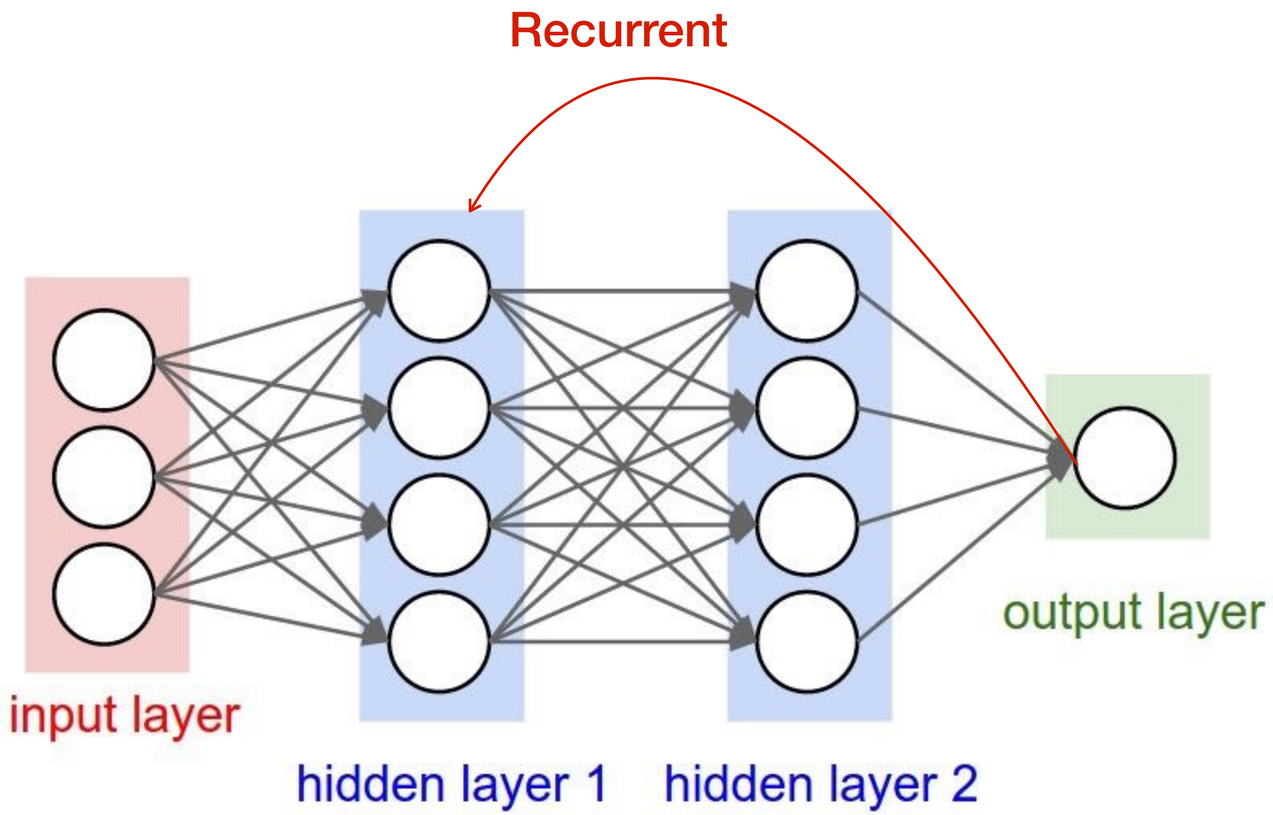
demo: basketball trajectories





http://tinyurl.com/traj-rnn

recurrent neural network

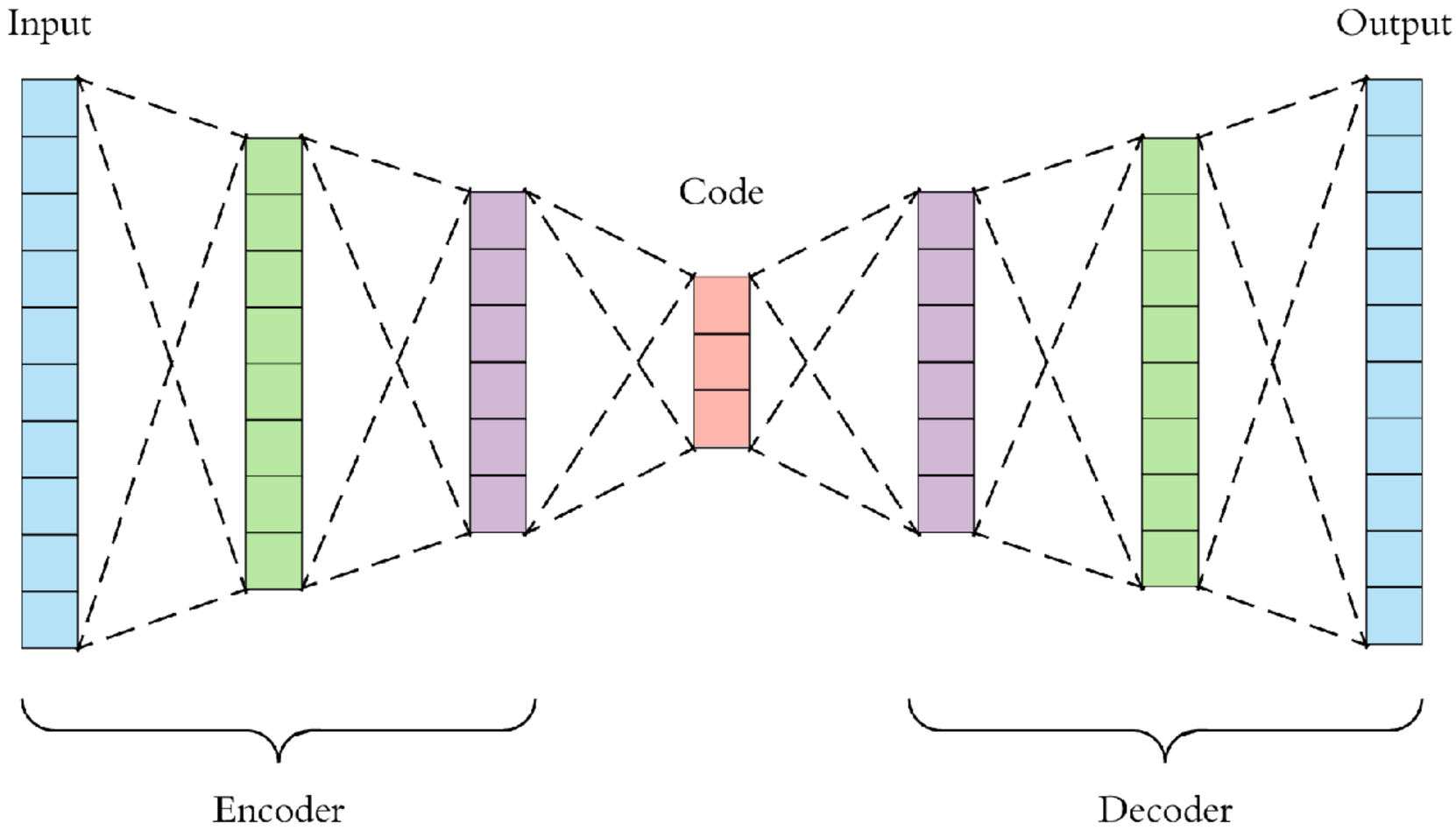


Recurrent neural network

```
model <- keras_model_sequential()
model %>%
  layer_embedding(input_dim = max_features, output_dim =
128, input_length = maxlen) %>%
  bidirectional(layer_lstm(units = 64)) %>%
  layer_dropout(rate = 0.5) %>%
  layer_dense(units = 1, activation = 'sigmoid')
```

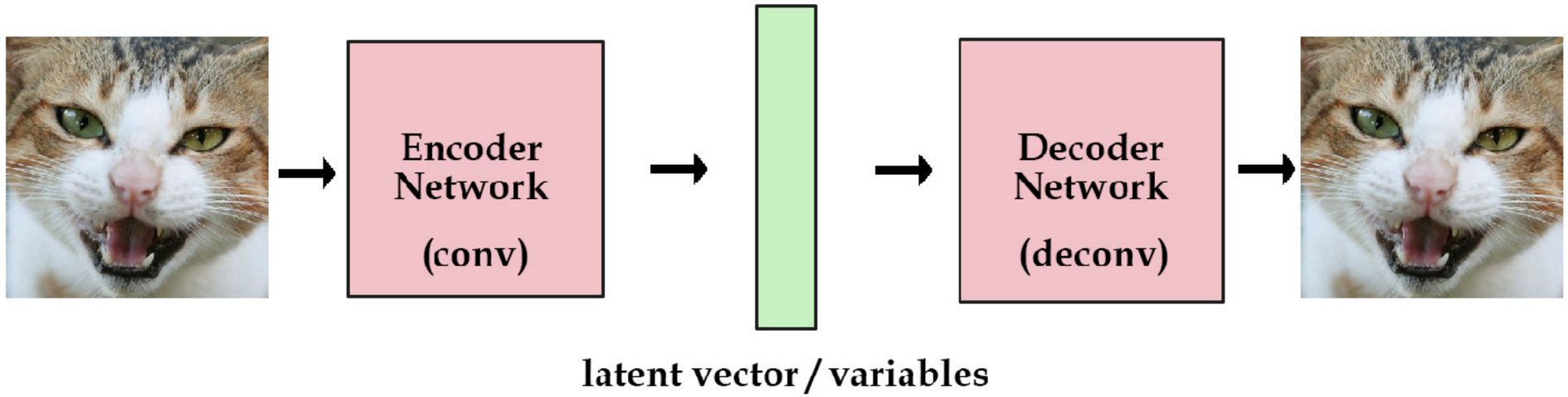
Generating new 

autoencoder



https://cdn-images-1.medium.com/max/1574/1*44eDEuZBEsmG_TCAKR13Kw@2x.png

autoencoder

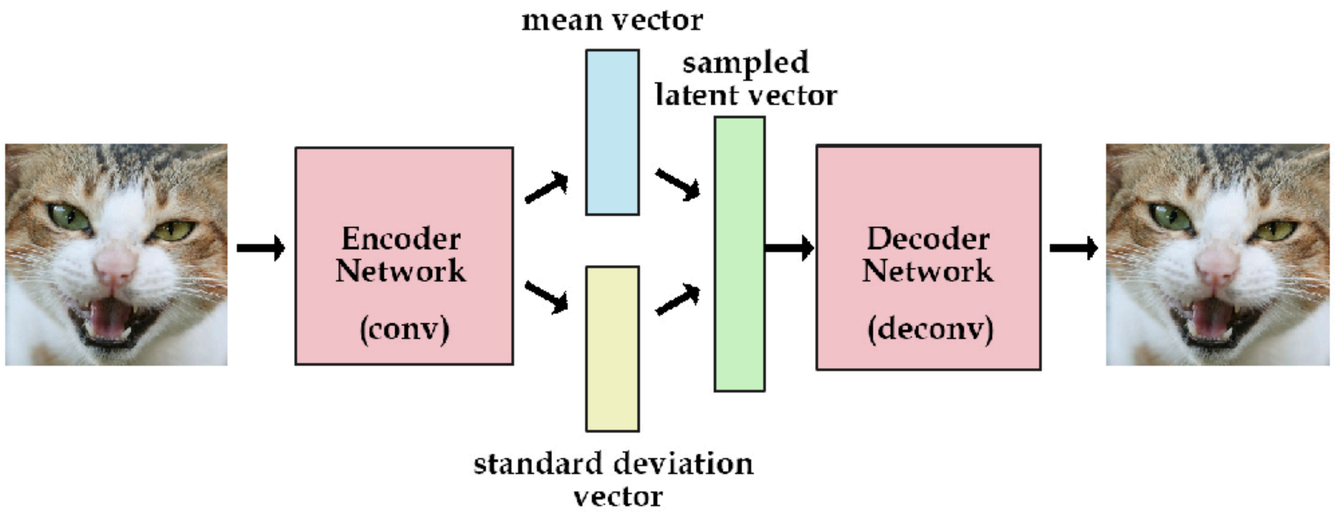


http://kvfrans.com/variational-autoencoders-explained/

Deep Autoencoder

```
input_img = Input(shape=(784,))
encoded = Dense(128, activation='relu')(input_img)
encoded = Dense(64, activation='relu')(encoded)
encoded = Dense(32, activation='relu')(encoded)

decoded = Dense(64, activation='relu')(encoded)
decoded = Dense(128, activation='relu')(decoded)
decoded = Dense(784, activation='sigmoid')(decoded)
```

variational autoencoder

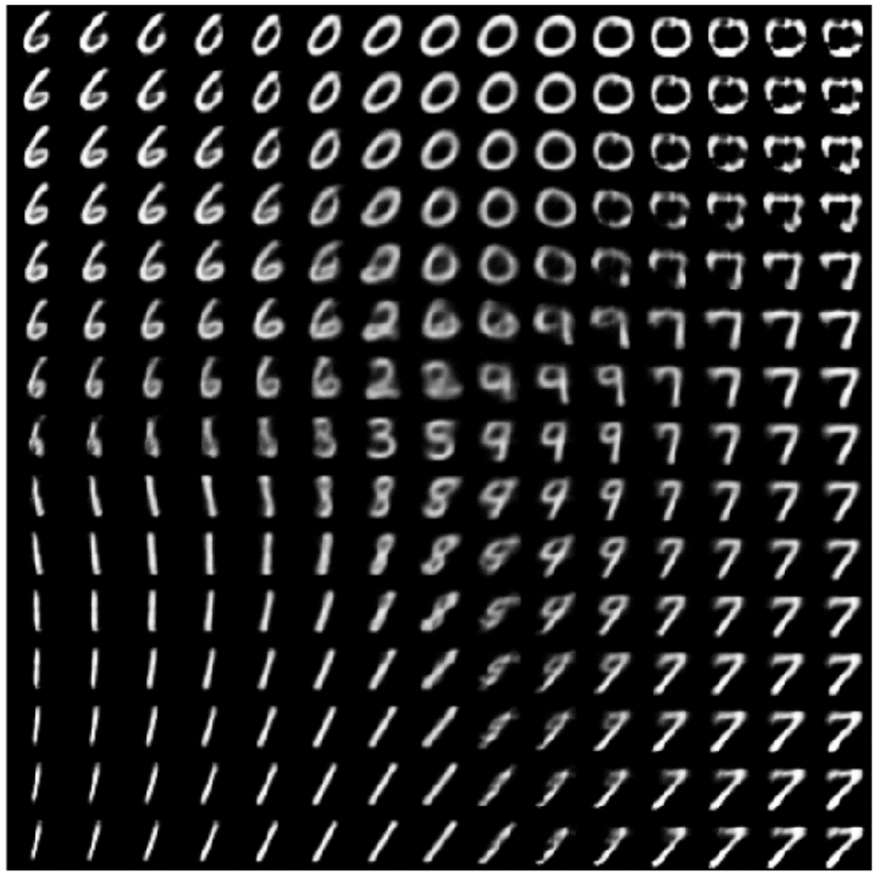


http://kvfrans.com/variational-autoencoders-explained/

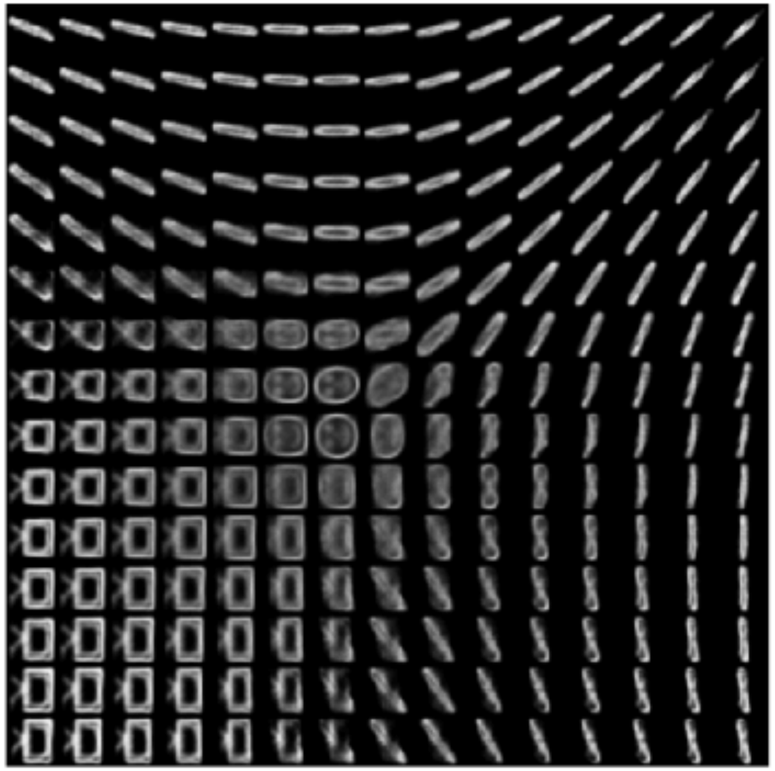
VAE loss

```
vae_loss <- function(x, x_decoded_mean){
  xent_loss <- (original_dim/1.0)*loss_binary_crossentropy(x, x_decoded_mean)
  kl_loss <- -0.5*K$mean(1 + z_log_var - K$square(z_mean) - K$exp(z_log_var),
axis = -1L)
  xent_loss + kl_loss
}
```

MNIST

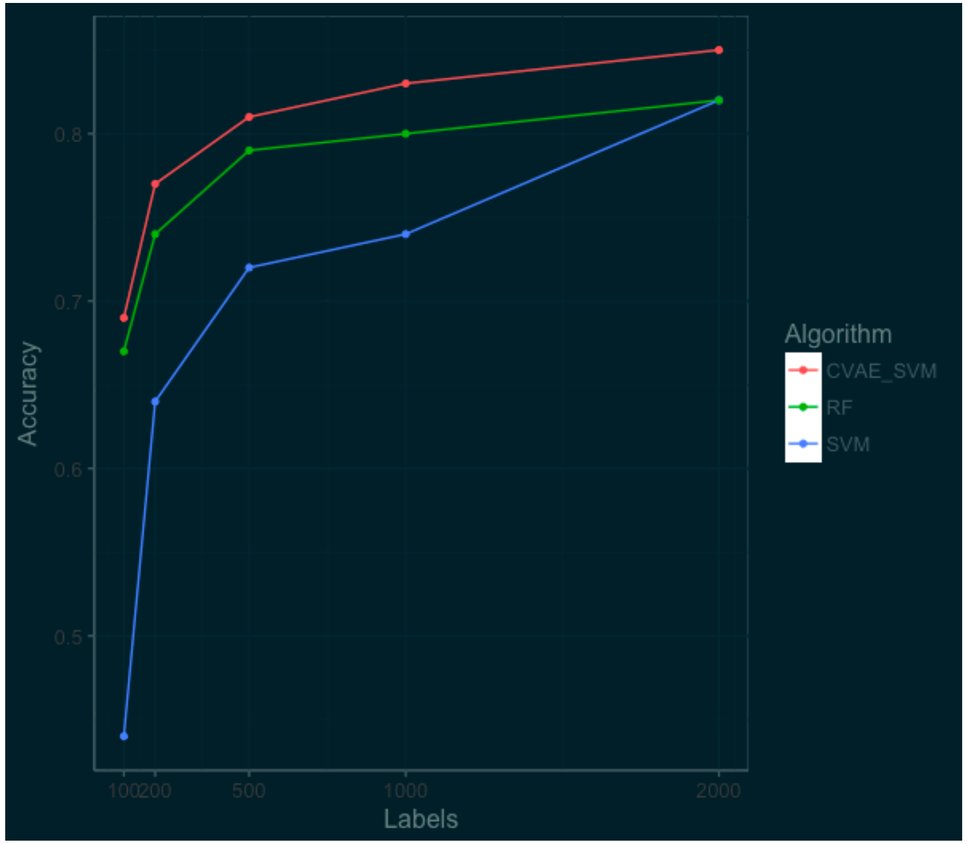


QuickDraw

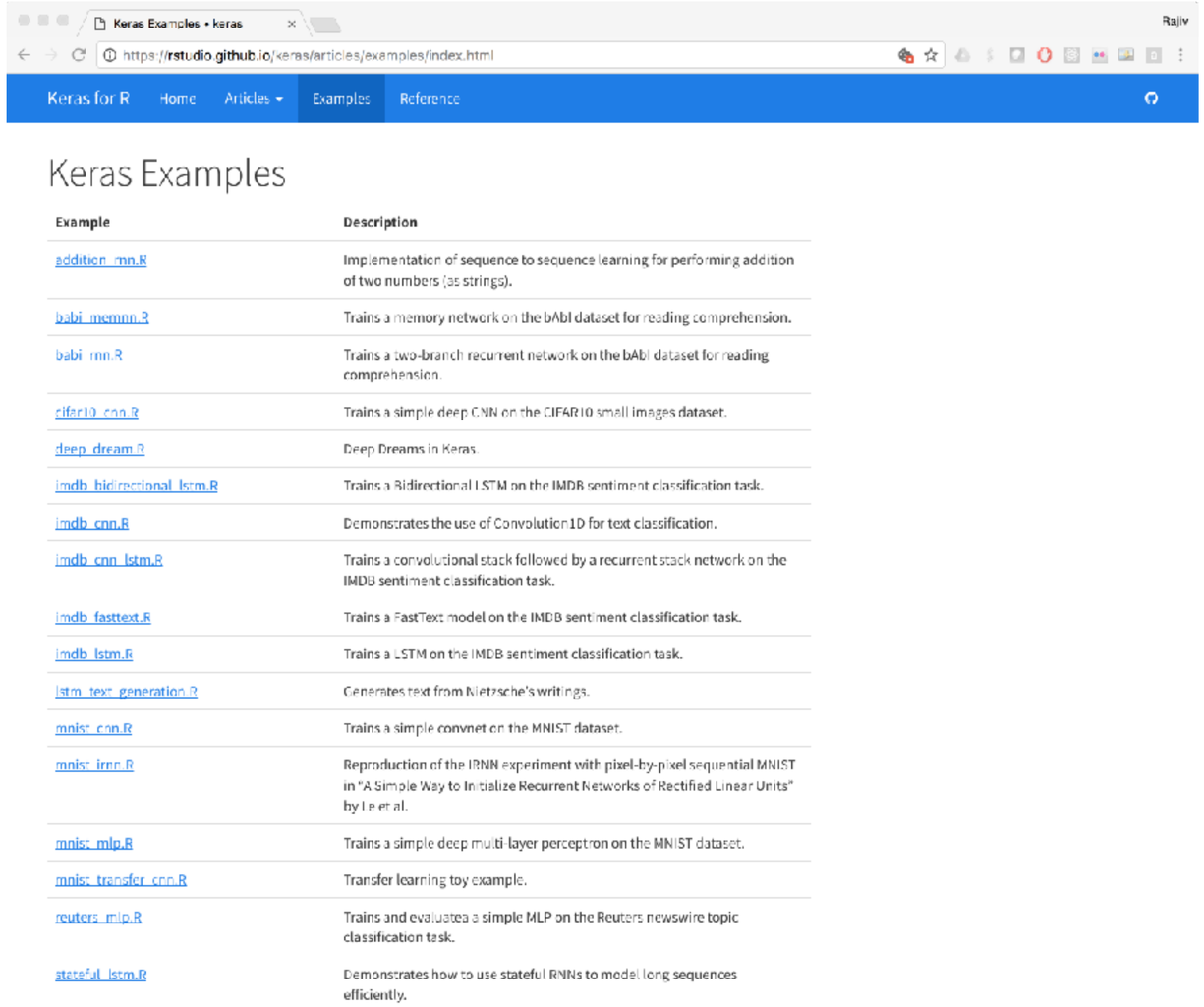


http://projects.rajivshah.com/blog/2017/07/14/QuickDraw/

Semi supervised learning



More examples . . .



whew . . .

The development of Keras/R

How to use Keras in R for:

- **convolutional neural networks**
- **recurrent neural networks**
- **variational autoencoders**

Deep Learning with Keras and R

Feb 15, 2018

RajivShah.com
rshah@pobox.com
 github.com/rajshah4/image_keras

 rajcs4