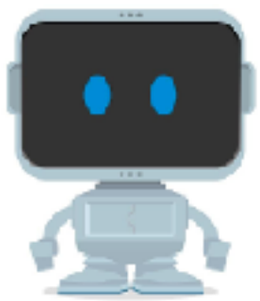
Keras and R: examples

July 6, 2017

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

rajcs4



goals

Using Python with R

Why Keras is important

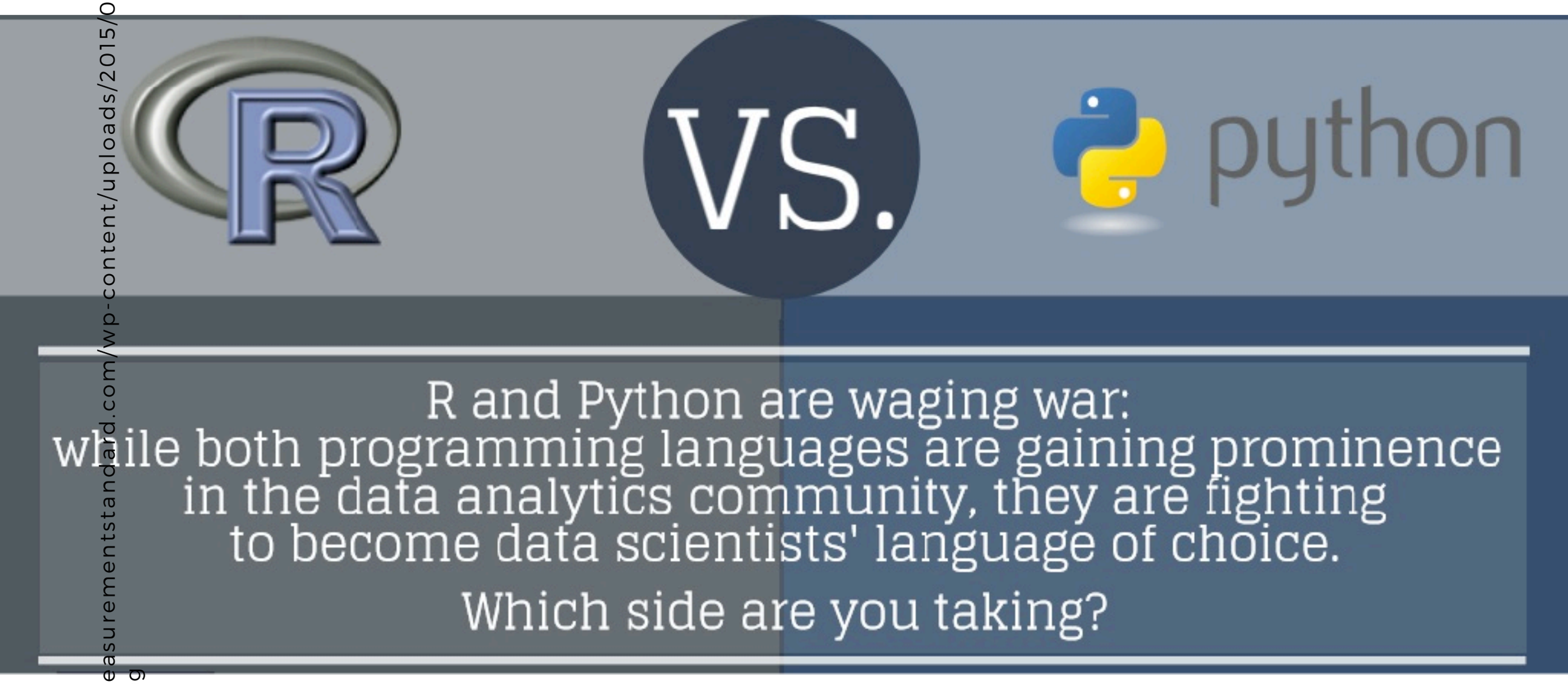
Examples:

Telling 🐱 from 🐶

😻 . . . 😽 . . . 🐱🐱🐱

Generating new 🐱

Python & R wars



http://www.themeasurementstandard.com/wp-content/uploads/2015/05/R-vs-Python-thumb.jpg

#4 And The Winner is...

It's a tie!
It's up to you, the data scientist,
to pick the language that best fits your needs.
The following questions can guide you in your decision.

1. What problems do you want to solve?

2. What are the net costs for learning a language?*

   * it will cost time to learn a new system that is better aligned for the problem you want to solve, but staying with the system you know may not be made for that kind of problem.

3. What are the commonly used tool(s) in your field?

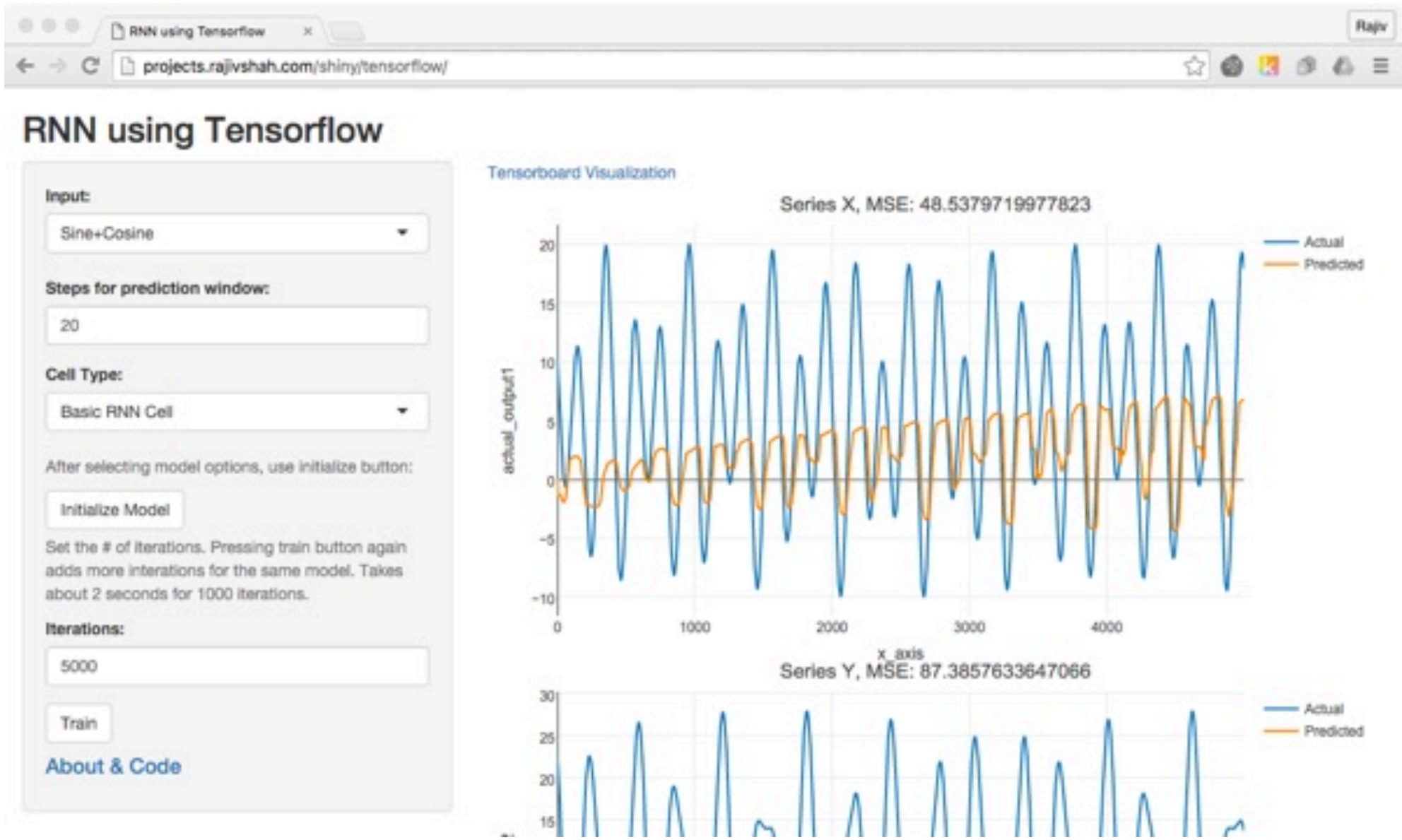
4. What are the other available tools in your field and how do these relate to the commonly used tool(s)?

https://www.datacamp.com/community/tutorials/r-or-python-for-data-analysis#gs.aTluUvY

Don’t be a tool hater
Room for both

Python & R interfaces

rPython (c. 2010)



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

rPython (c. 2010)

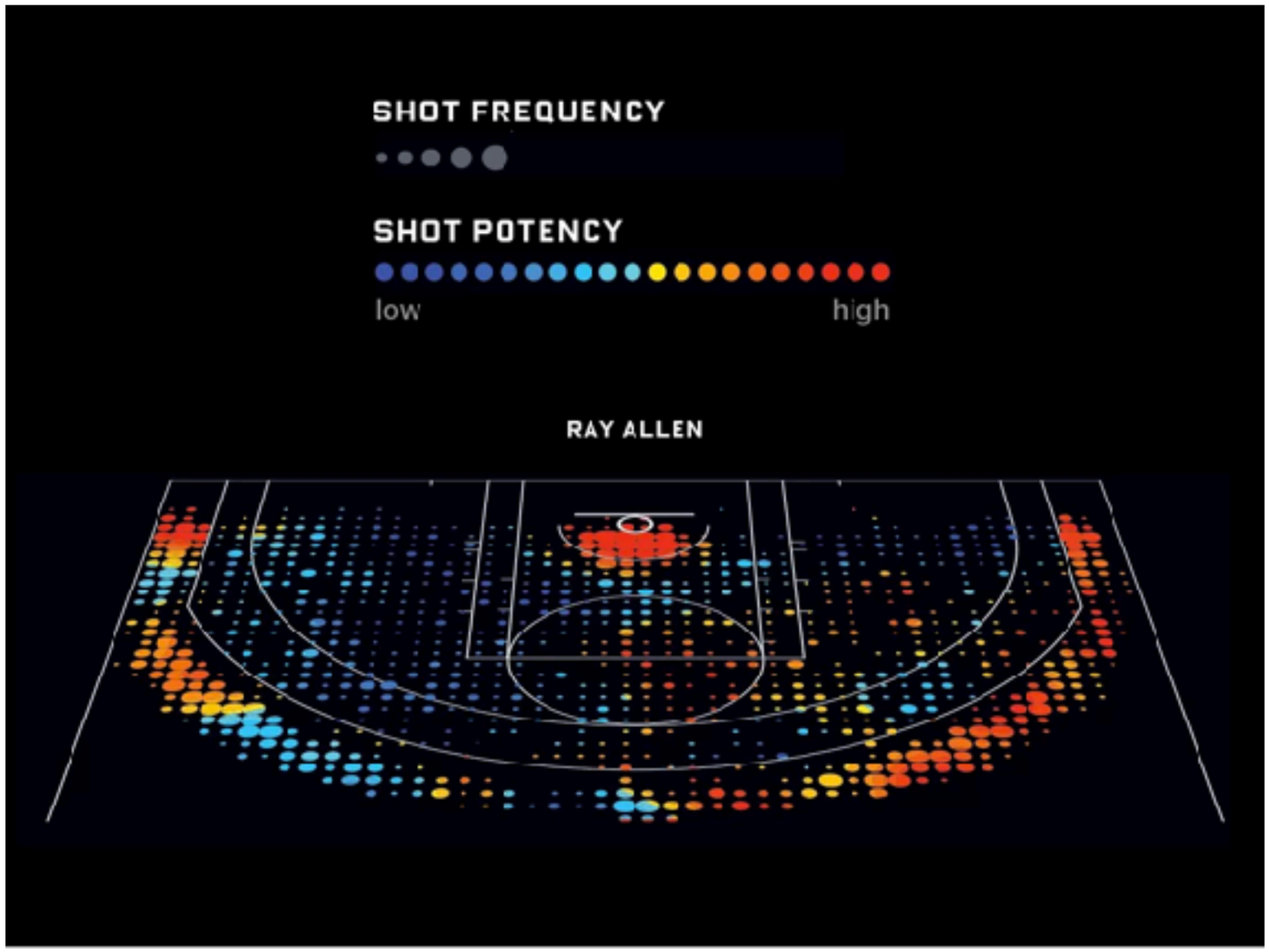
```
iw <- as.numeric(input$input_wave)-1
python.assign( "input_wave", iw )
python.assign( "prlag", input$prlag )
python.exec("
            lag = prlag
            def get_sample():
            global angle1, angle2
            angle1 += 2*pi/float(frequency1)
            angle2 += 2*pi/float(frequency2)
            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())
```

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

Reticulate (Feb. 2017)



Reticulate (Feb. 2017)

```
library(reticulate)

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

Keras

Keras for R (June 2017)

High Level Neural Network API

Compatible Deep Learning Backends:

Theano

Core ML

Tensorflow

CNTK

~PyTorch

faq

Works with GPU

Approximately the same speed as python

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

Actively being developed

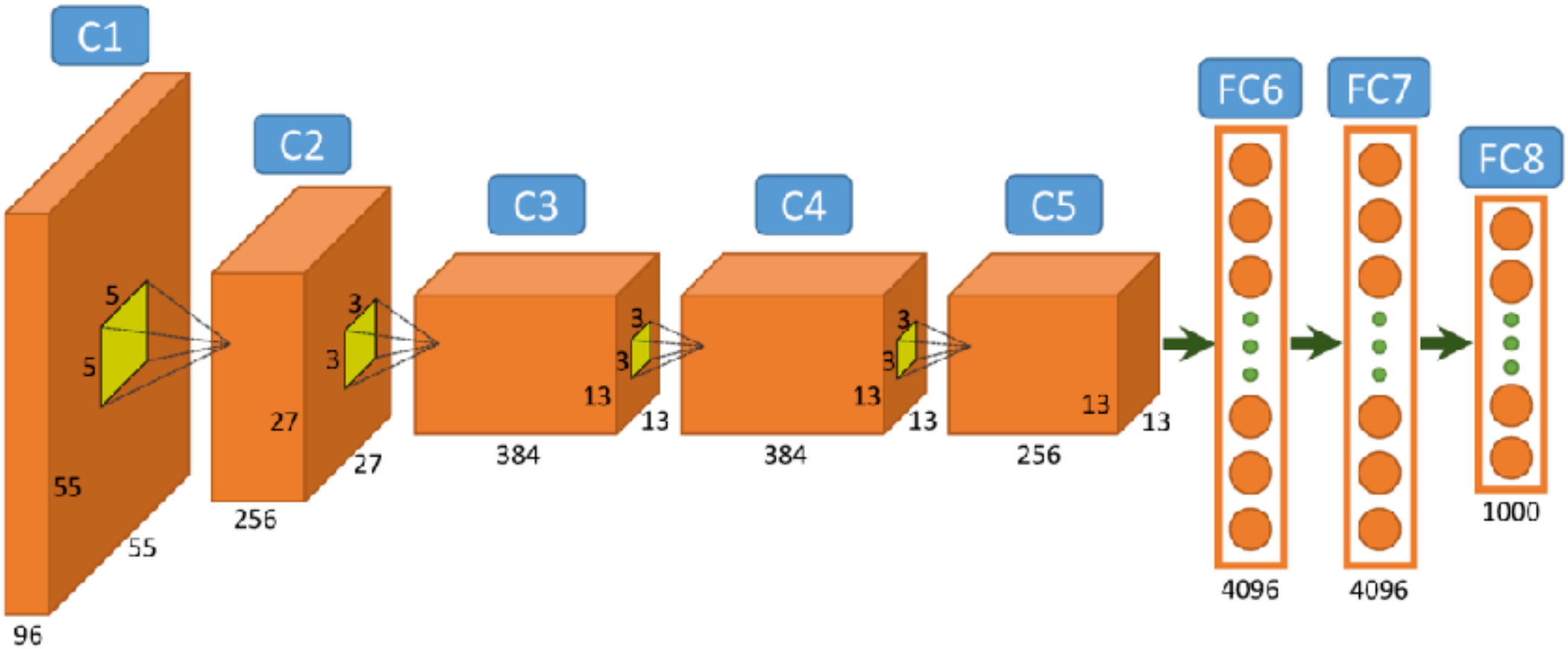
Awesome work by J.J. Allaire

Examples

Telling 🐱 from 🐶

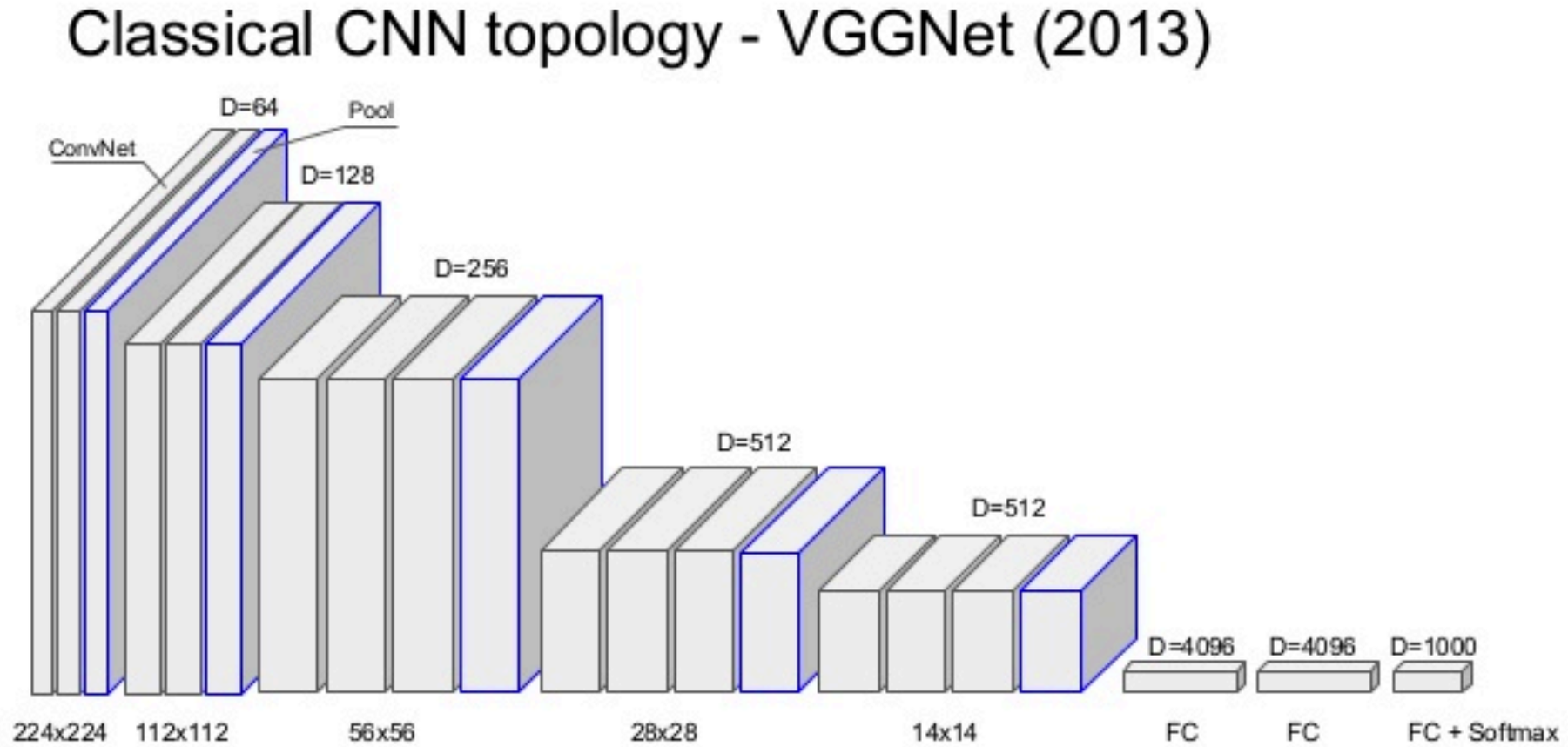
https://github.com/rajshah4/image_keras

Alexnet architecture



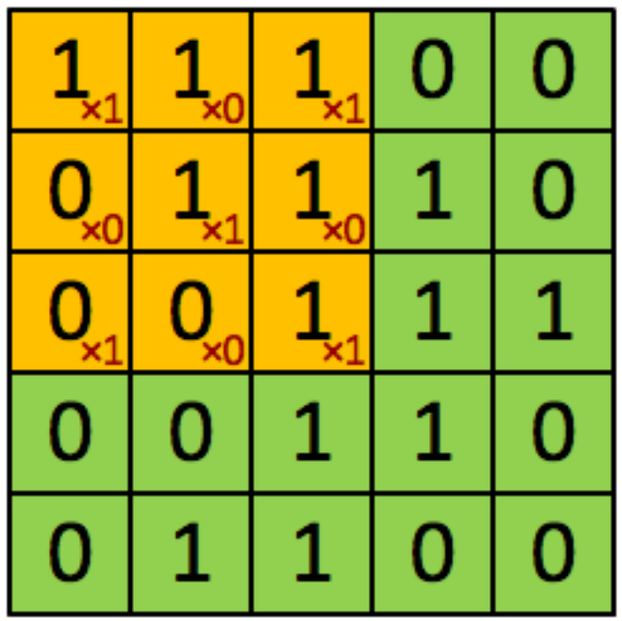
https://www.saagie.com/blog/object-detection-part1

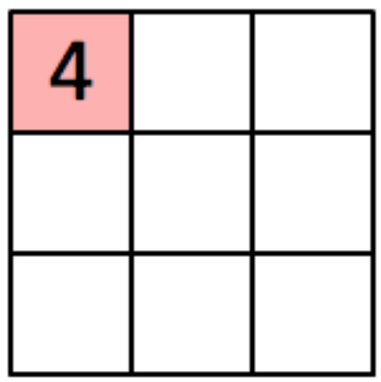
VGG architecture



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

understanding convolution





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")
```

techniques:

Augment data

Use a pretrained convolutional neural network

Use transfer learning (fine tuning a pretrained network)

Not Hotdog

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)

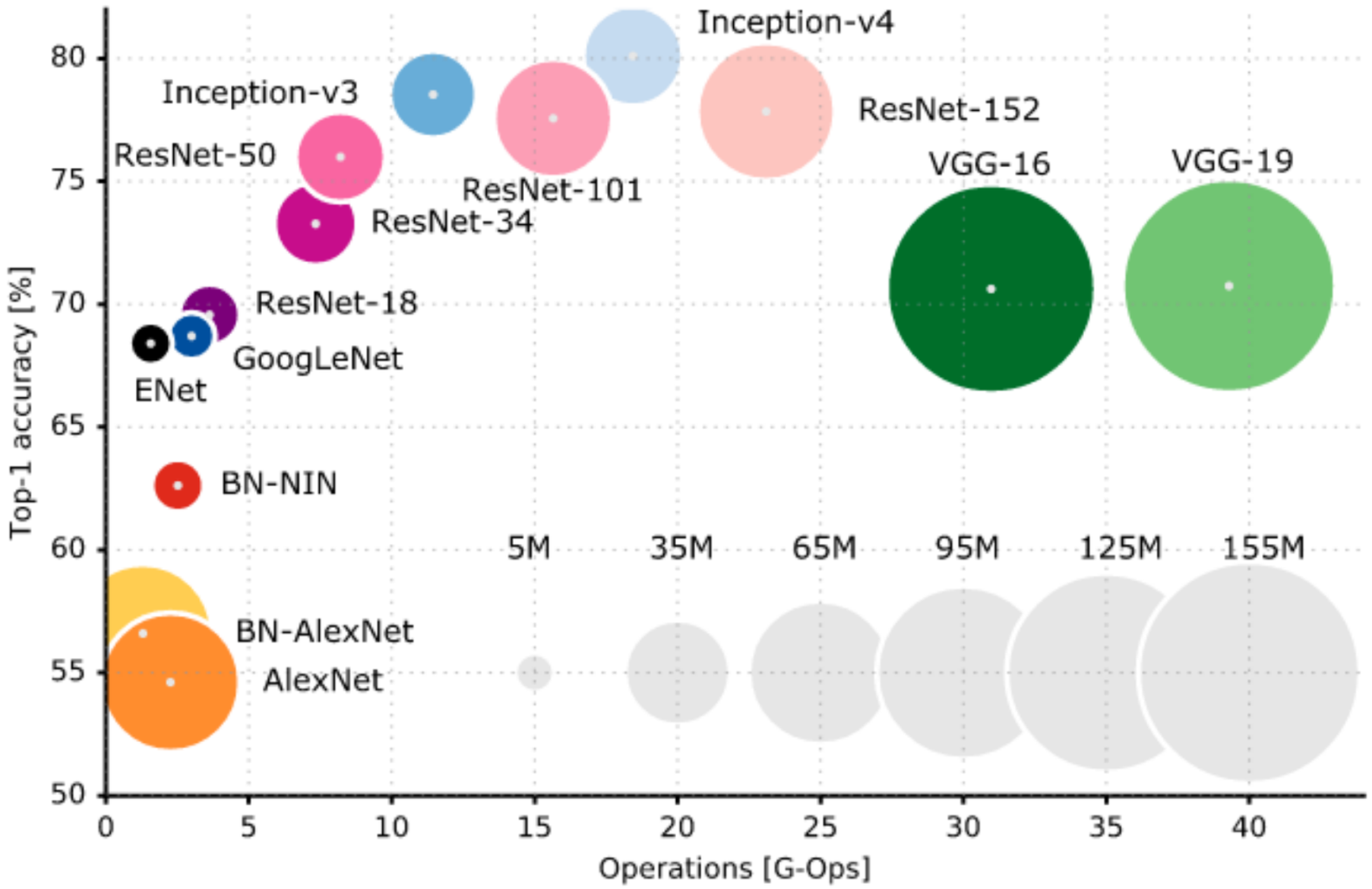
To load 🖼️ from a folder:

```
train_generator <- flow_images_from_directory(train_directory,
                    generator = image_data_generator(),
                    target_size = c(img_width, img_height),
                    color_mode = "rgb",
                    class_mode = "binary",
                    batch_size = batch_size,
                    shuffle = TRUE,
                    seed = 123)
```

To augment data:

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

pre-trained networks



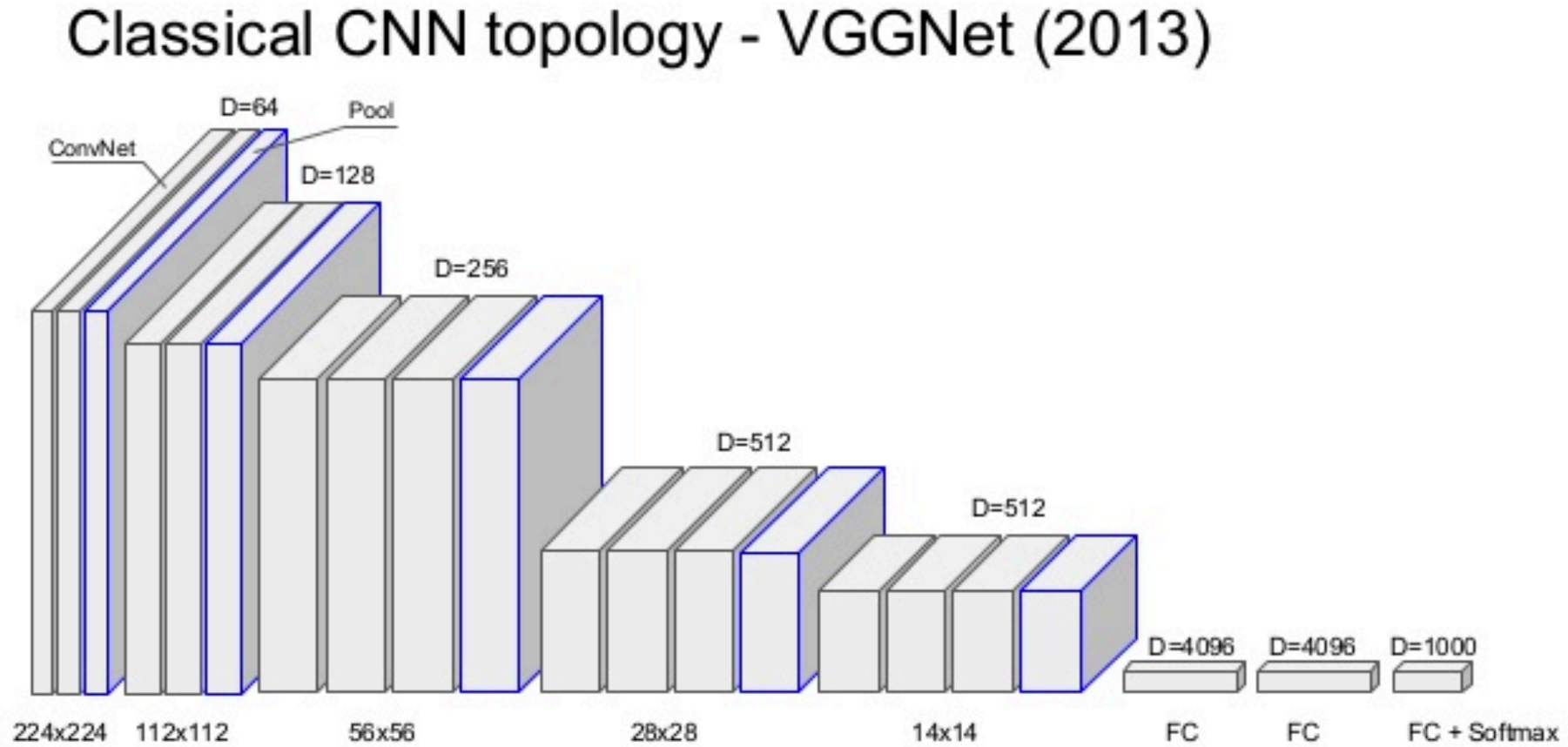
Load a pretrained network:

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

Save model weights:

```
save_model_weights_hdf5(model_ft, 'finetuning_30epochs_vggR.h5', overwrite = TRUE)
```

VGG architecture

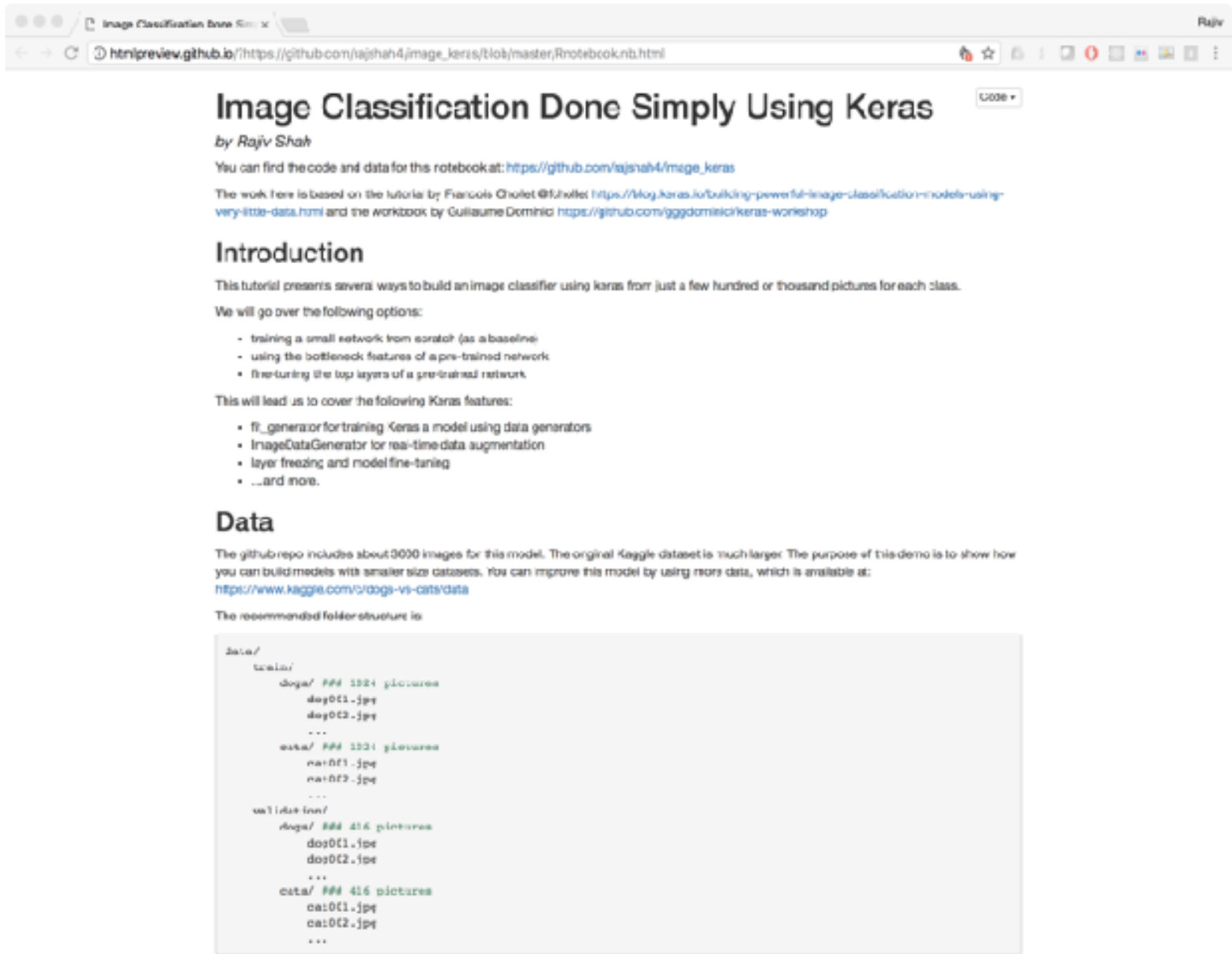


https://www.saagie.com/blog/object-detection-part1

Train only last few layers:

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

Complete Notebook



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 Dominic https://github.com/gggdominic/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

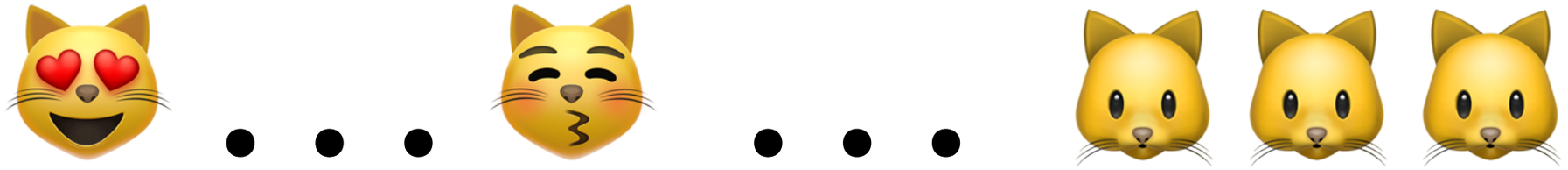
The github repo includes about 3000 images for this model. The original Kaggle dataset is much larger. The purpose of this demo is to show how you can build models with smaller size datasets. You can improve this model by using more data, which is available at: https://www.kaggle.com/c/dogs-vs-cats/data

The recommended folder structure is:

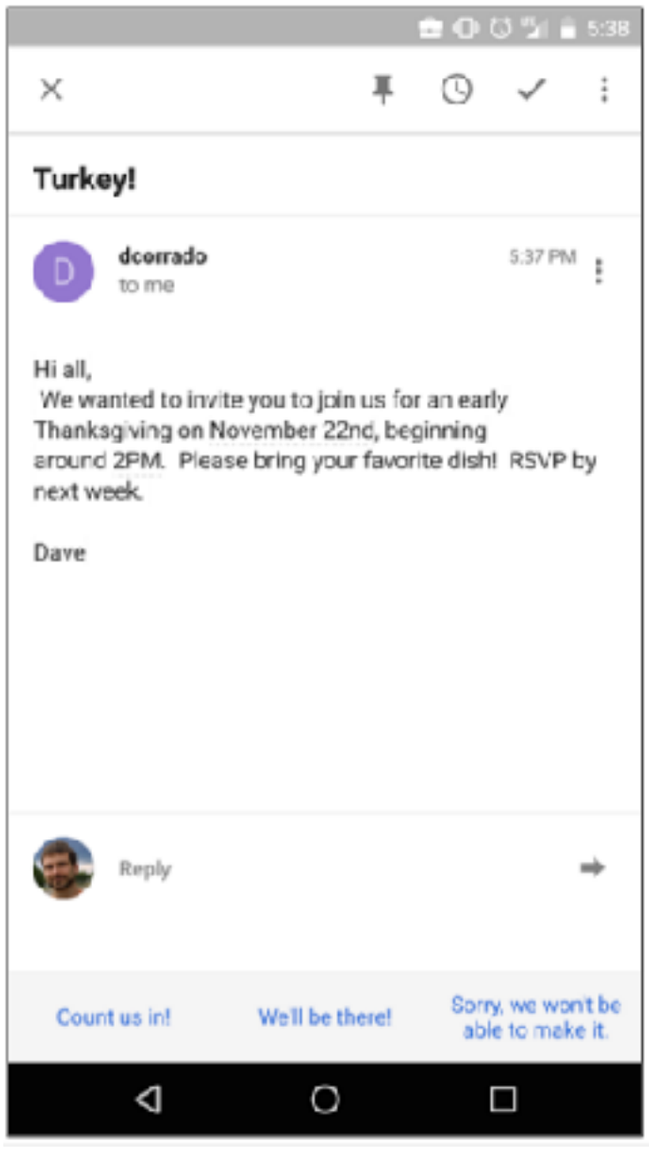
```
data/
    train/
        dogs/ ### 1024 pictures
            dog001.jpg
            dog002.jpg
            ...
        cats/ ### 1024 pictures
            cat001.jpg
            cat002.jpg
            ...
    validation/
        dogs/ ### 416 pictures
            dog001.jpg
            dog002.jpg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

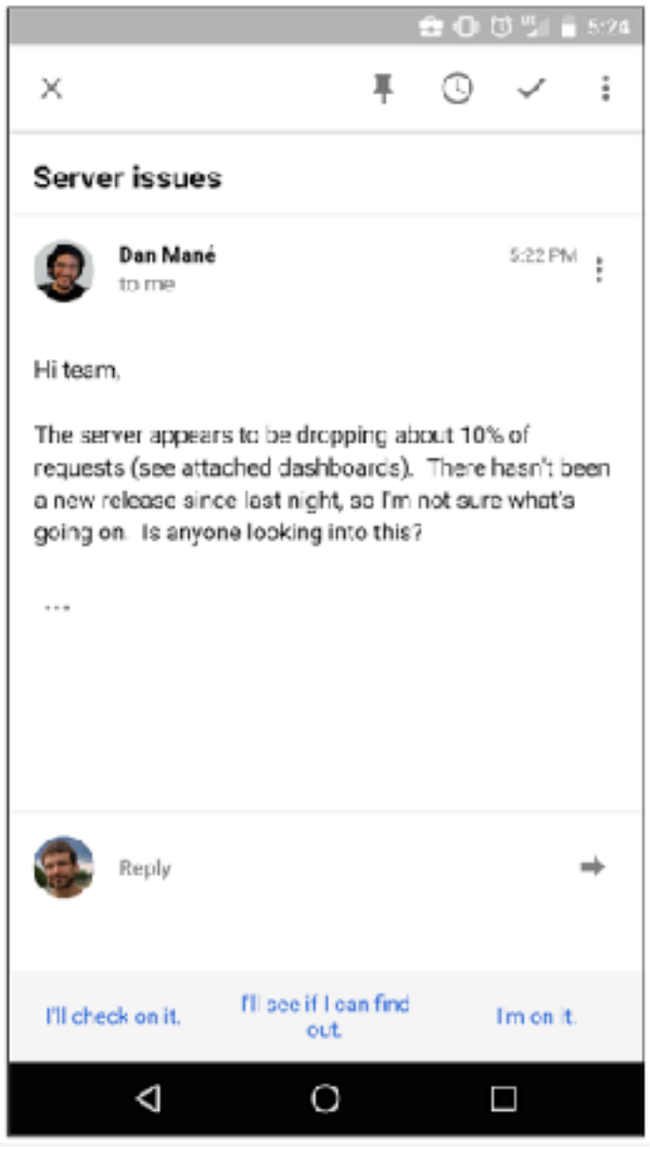
Loading Tensorflow and Keras

https://github.com/rajshah4/image_keras



smart reply

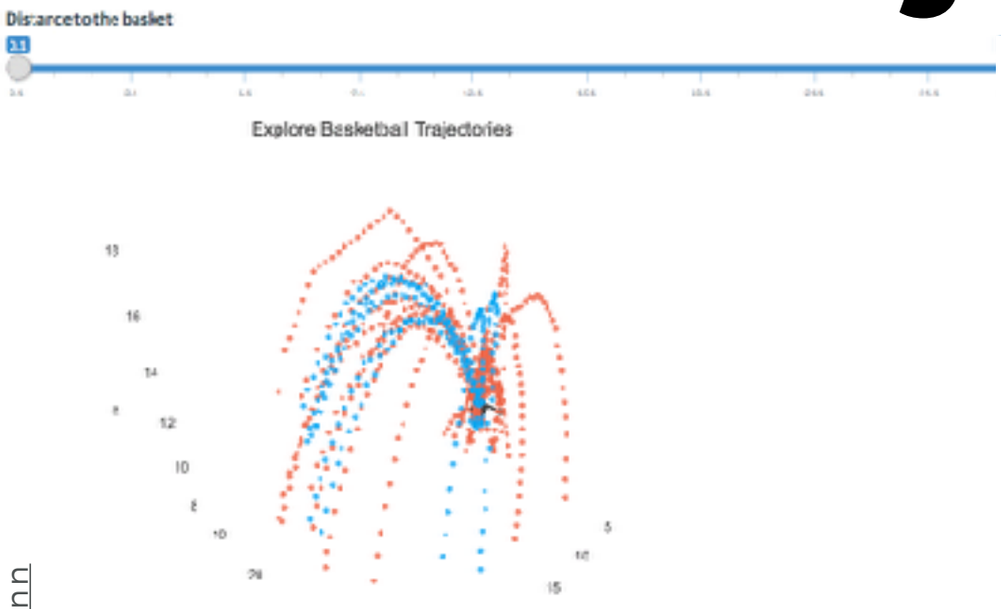


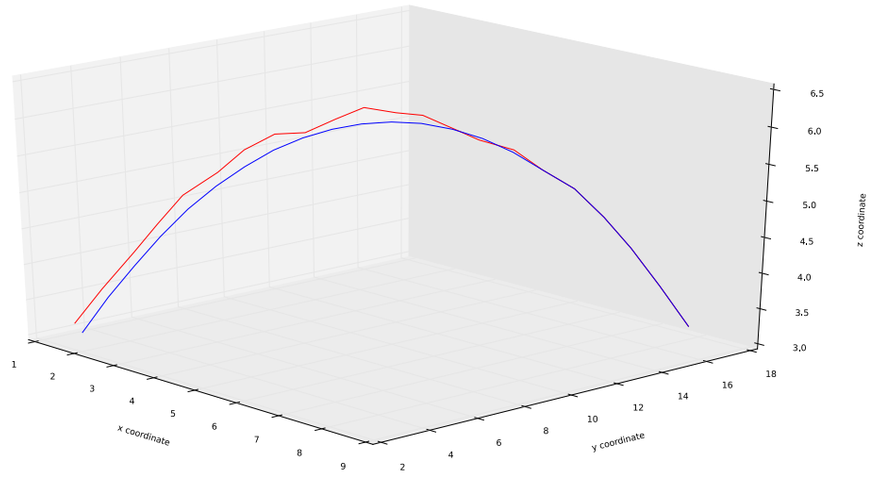


demo: handwriting

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

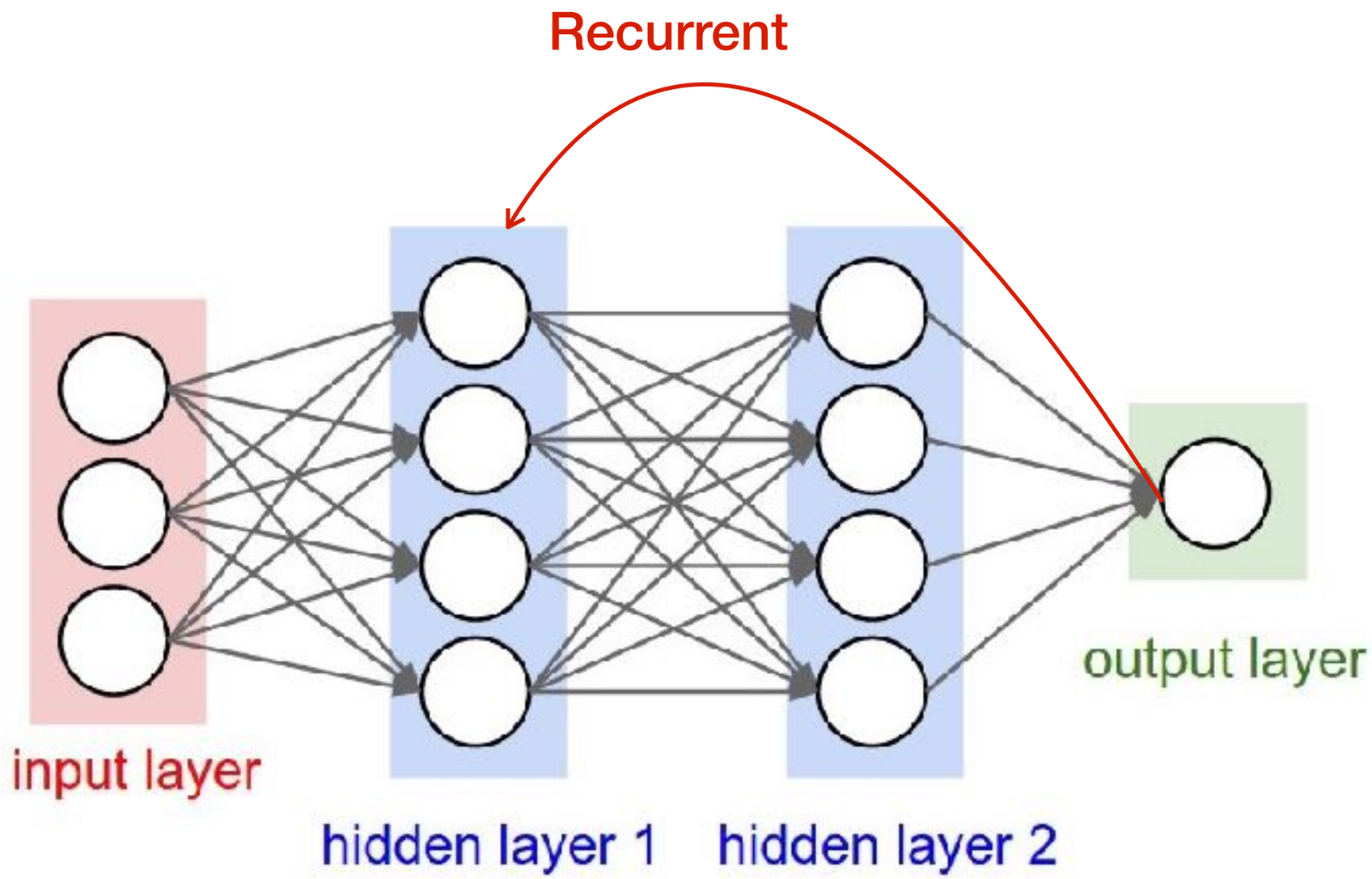
demo: basketball trajectories





http://tinyurl.com/traj-rnn

recurrent neural network

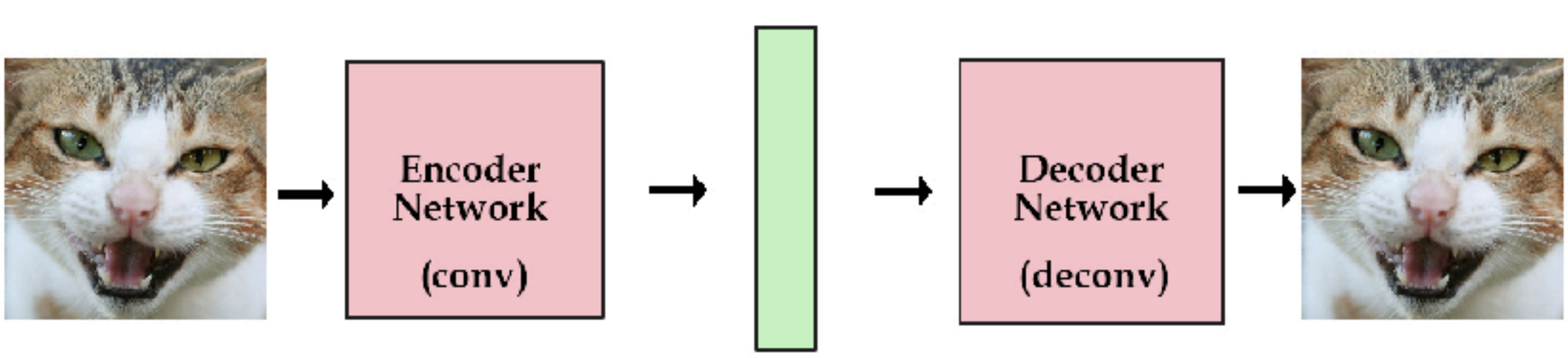


Recurrent neural network

```
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

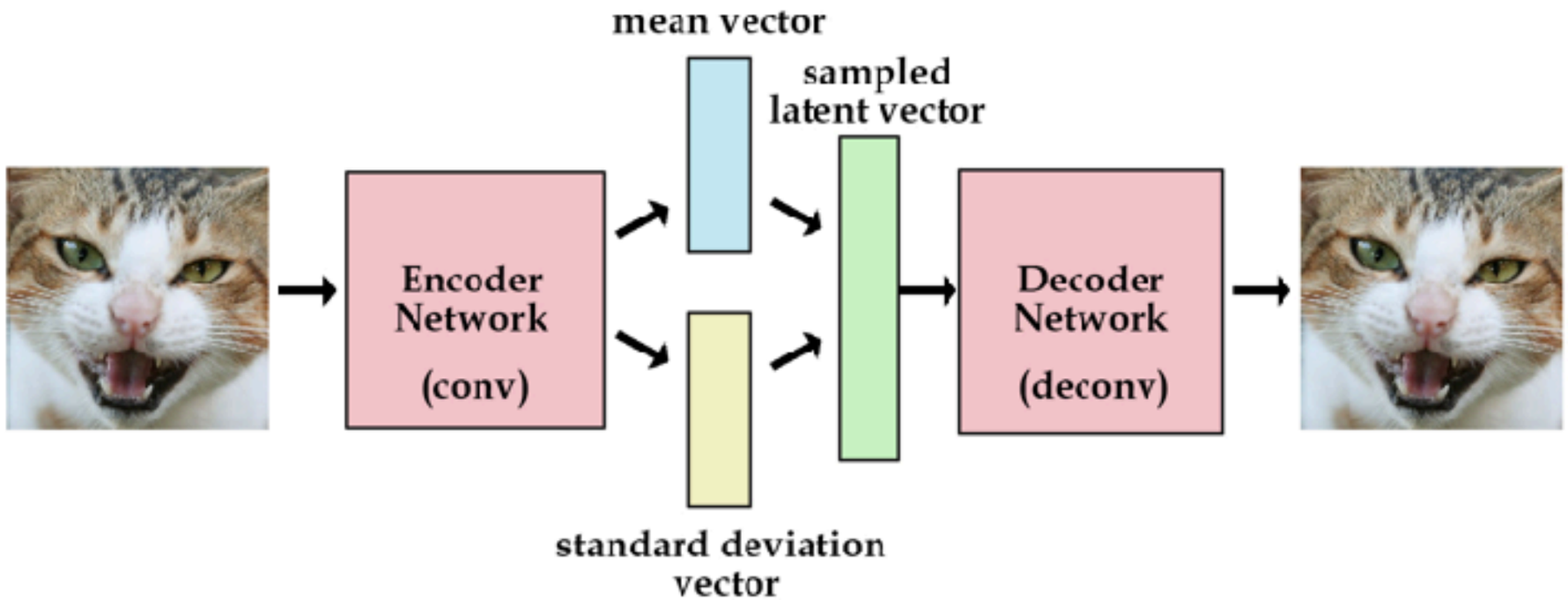


http://kvfrans.com/variational-autoencoders-explained/

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

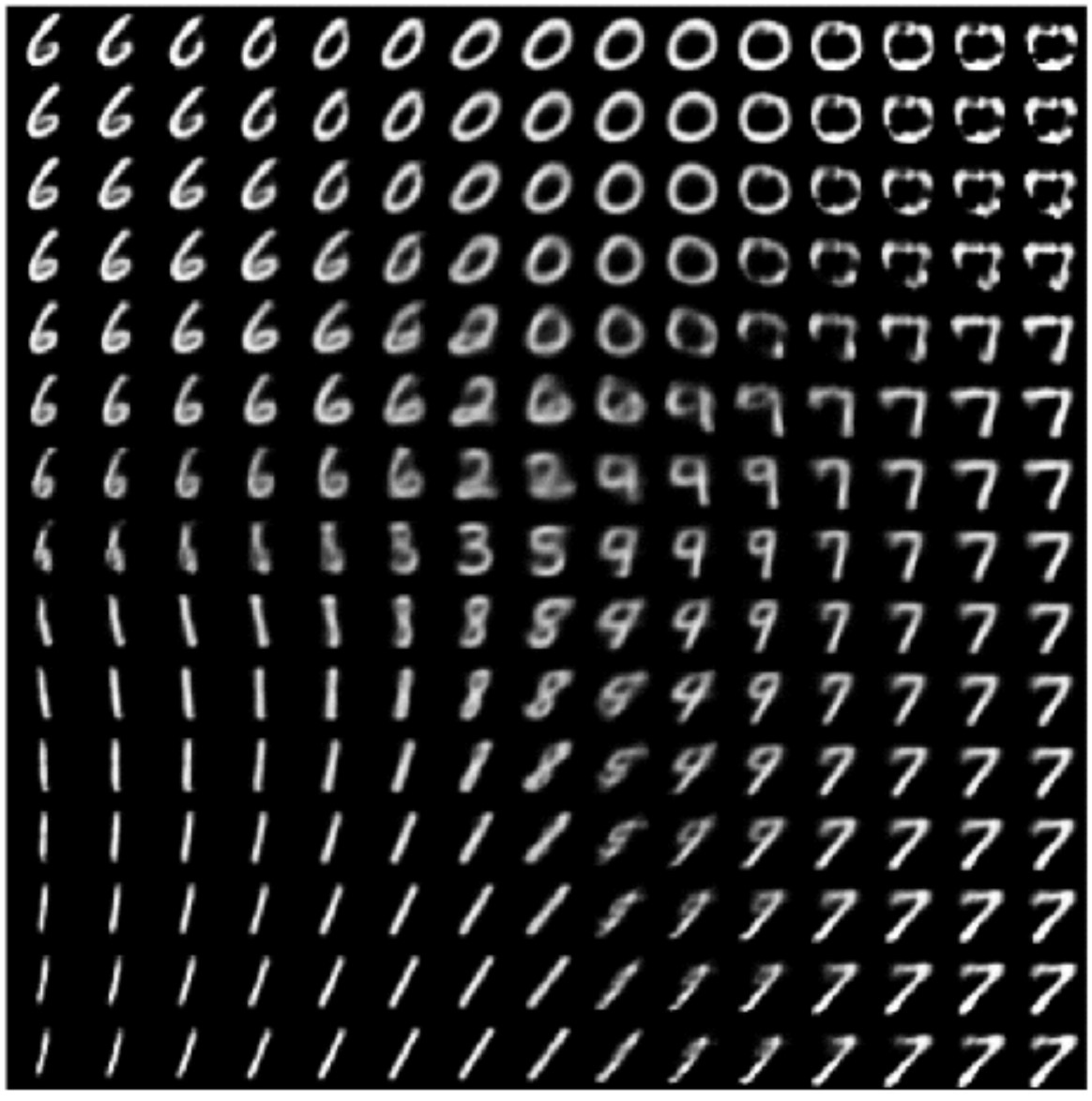


http://kvfrans.com/variational-autoencoders-explained/

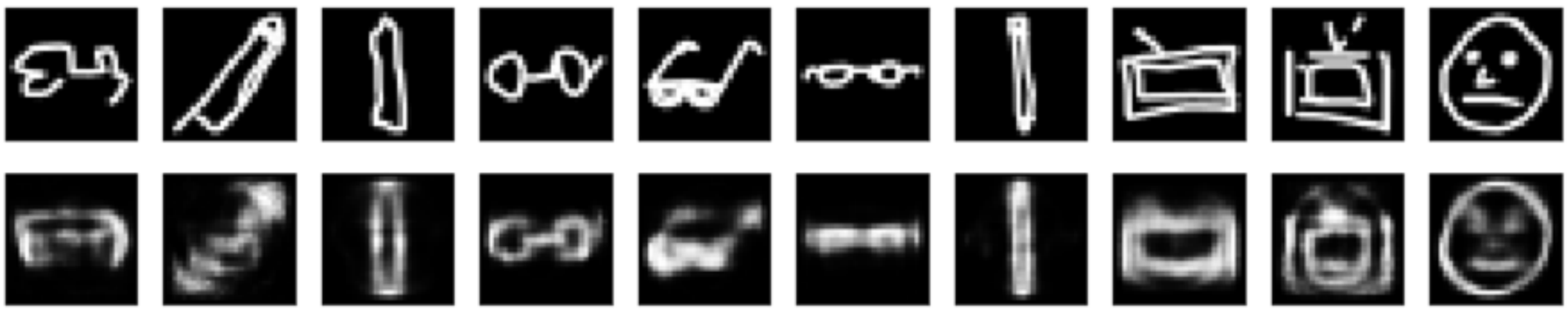
VAE loss

```
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
}
```

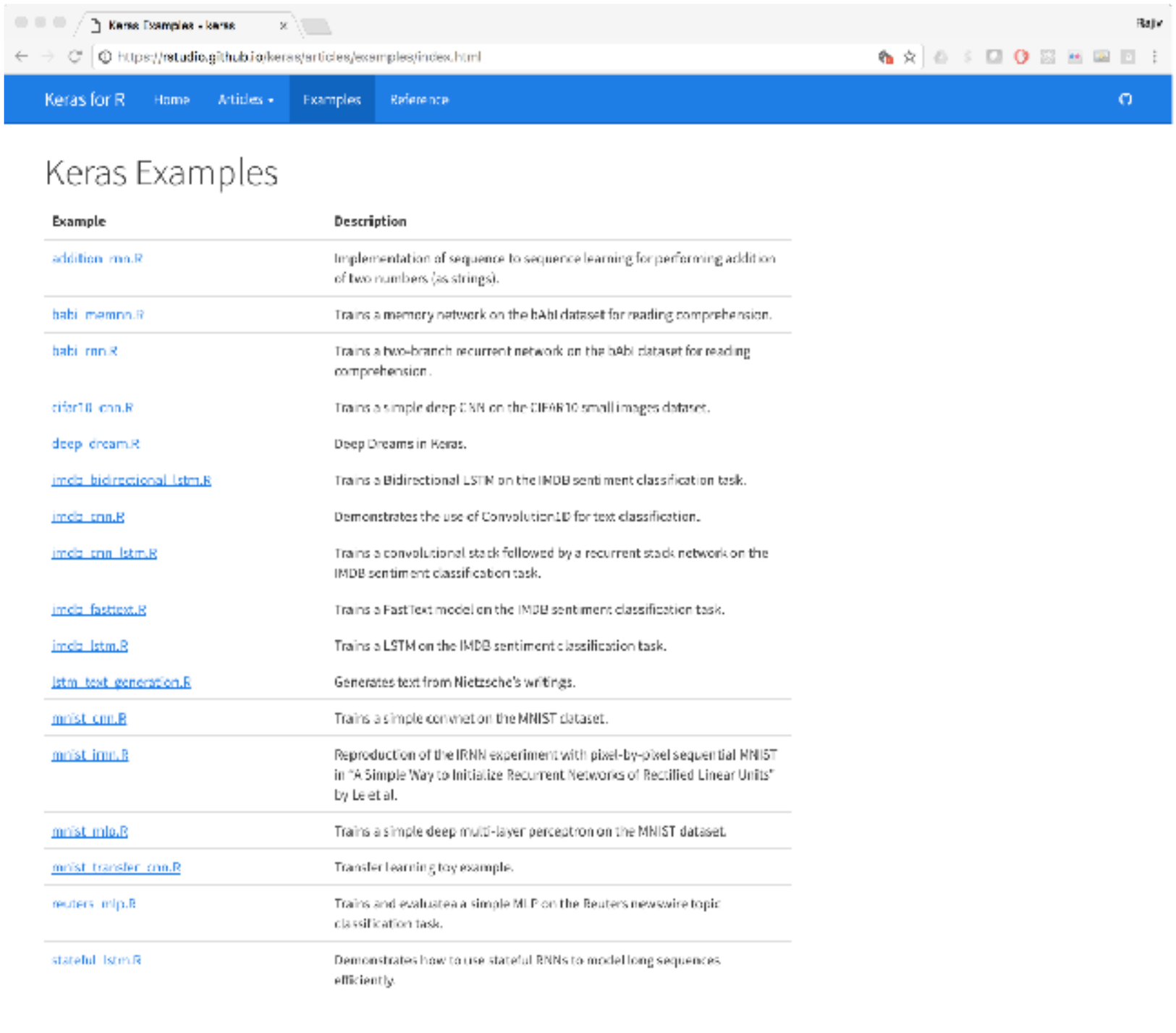
MNIST



QuickDraw



More examples . . .



whew . . .

No more python versus R

You can use python packages in R

How to use Keras in R for:

convolutional neural networks
recurrent neural networks
variational auto encoders

Keras and R: examples

July 6, 2017

RajivShah.com
rshah@pobox.com
CHICAGOAI.SLACK.COM
github.com/rajshah4/image_keras

rajcs4

DataRobot