R + Tensorflow = Reproducibility, Transparency, & Trust

IBM

CODE

IBM Center for Open-Source Data & AI Technologies (http://codait.org)

Slides available at: http://bit.ly/rtensorflow-oscon18

gdequeiroz / mmmpork

Agenda

- TensorFlow and R
- Tools/Workflows
- Reproducibility
- Visualization

Speakers

GABRIELA DE QUEIROZ
Data & AI Developer Advocate, IBM CODAIT
gdq@ibm.com
gdequeiroz
https://k-roz.com/

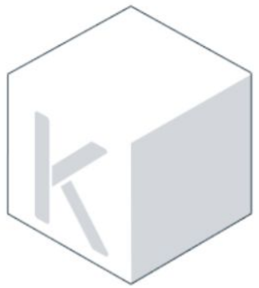
AUGUSTINA RAGWITZ
Computational Anthropologist, IBM CODAIT
Augustina.Ragwitz@ibm.com
mmmpork
http://rhappy.fun/

What is R?

- Free and Open Source Language and Environment
- Popular language for data scientists
- It has more extensions than any other data science software
- Primary tool for statistical research
- RStudio - an IDE with a lot of functionality
- Awesome Community (#rstats + R-Ladies + R Forwards)

Why TensorFlow + R?

TensorFlow APIs

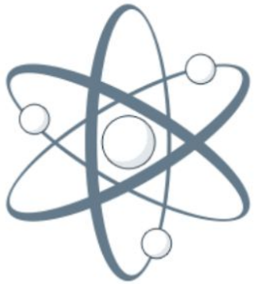


Keras API

The Keras API for TensorFlow provides a high-level interface for neural networks, with a focus on enabling fast experimentation.

Estimator API

The Estimator API for TensorFlow provides high-level implementations of common model types such as regressors and classifiers.



Core API

The Core TensorFlow API is a lower-level interface that provides full access to the TensorFlow computational graph.

Slides available at: http://bit.ly/rtensorflow-oscon18
gdequeiroz / mmmpork

Source/Credits: https://tensorflow.rstudio.com

Main R Packages + Supporting Tools

TensorFlow API

- keras
- tfestimators - Implementations of model types such as regressors and classifiers
- tensorflow - Low-level interface to the TensorFlow computational graph
- tfdatasets - Work with large datasets

Tools

- **tfruns** - Manage experiments (runs)
- **tfdeploy** - Share models across formats
- cloudml - Interface to Google Cloud ML

Tools/Workflows

tfruns

Track and Visualizing Training Runs

tfruns - Track and Visualizing Training Runs

- **Track** the hyperparameters, metrics, output, and source code of every training run.
- **Compare** hyperparameters and metrics across runs to find the best performing model.
- **Generate reports** to visualize individual training runs or comparisons between runs.

tfruns - Track and Visualizing Training Runs

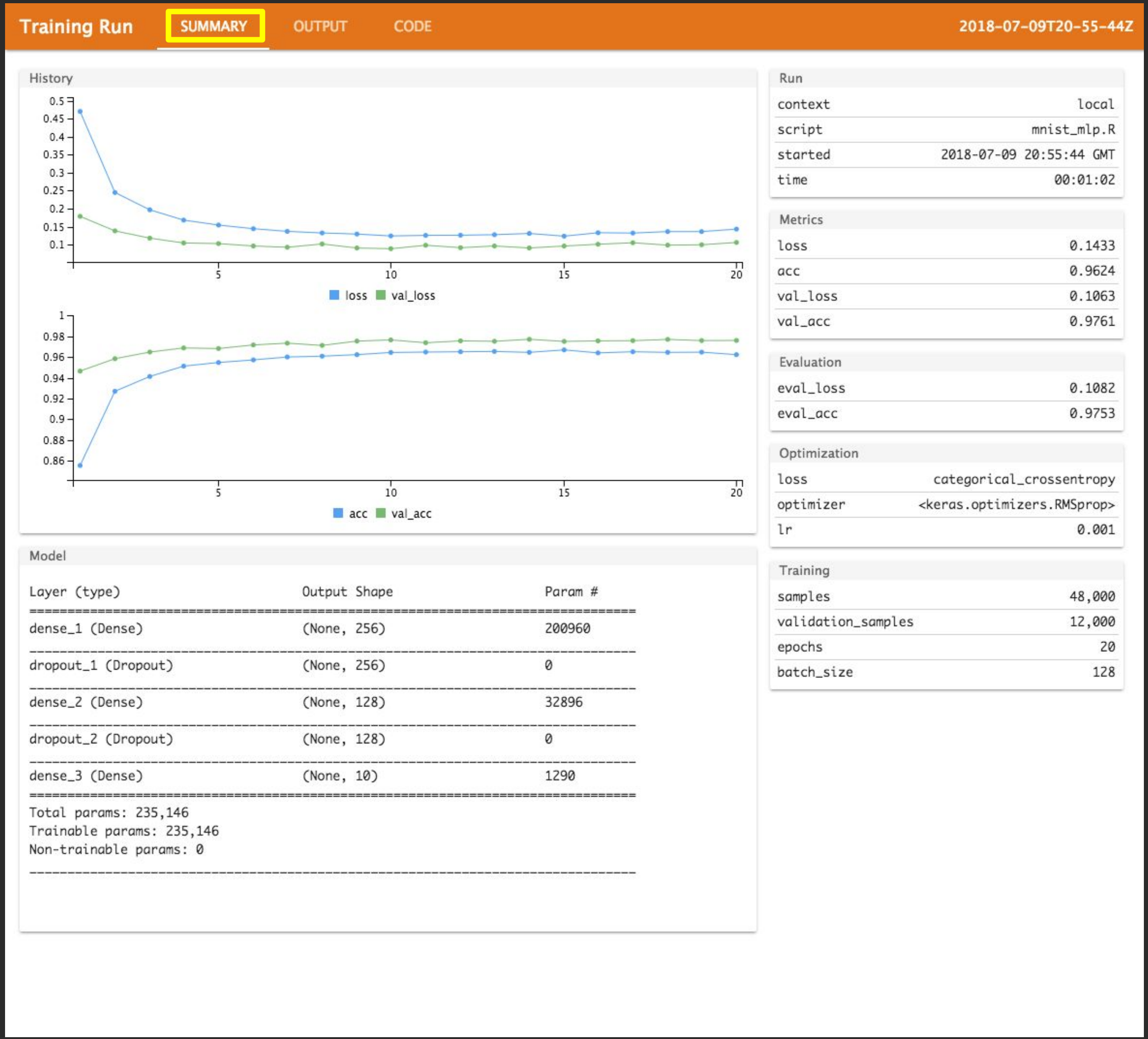
```r
# Define Model --------------------------------------------------------------

model <- keras_model_sequential()
model %>%
  layer_dense(units = 256, activation = 'relu', input_shape = c(784)) %>%
  layer_dropout(rate = 0.4) %>%
  layer_dense(units = 128, activation = 'relu') %>%
  layer_dropout(rate = 0.3) %>%
  layer_dense(units = 10, activation = 'softmax')


model %>% compile(
  loss = 'categorical_crossentropy',
  optimizer = optimizer_rmsprop(lr = 0.001),
  metrics = c('accuracy')
)

# Training & Evaluation ----------------------------------------------------

history <- model %>% fit(
  x_train, y_train,
  batch_size = batch_size,
  epochs = 20,
  verbose = 1,
  validation_split = 0.2
)

plot(history)

score <- model %>% evaluate(
  x_test, y_test,
  verbose = 0
)
```

```r
source("mnist_mlp.R")
```

OR

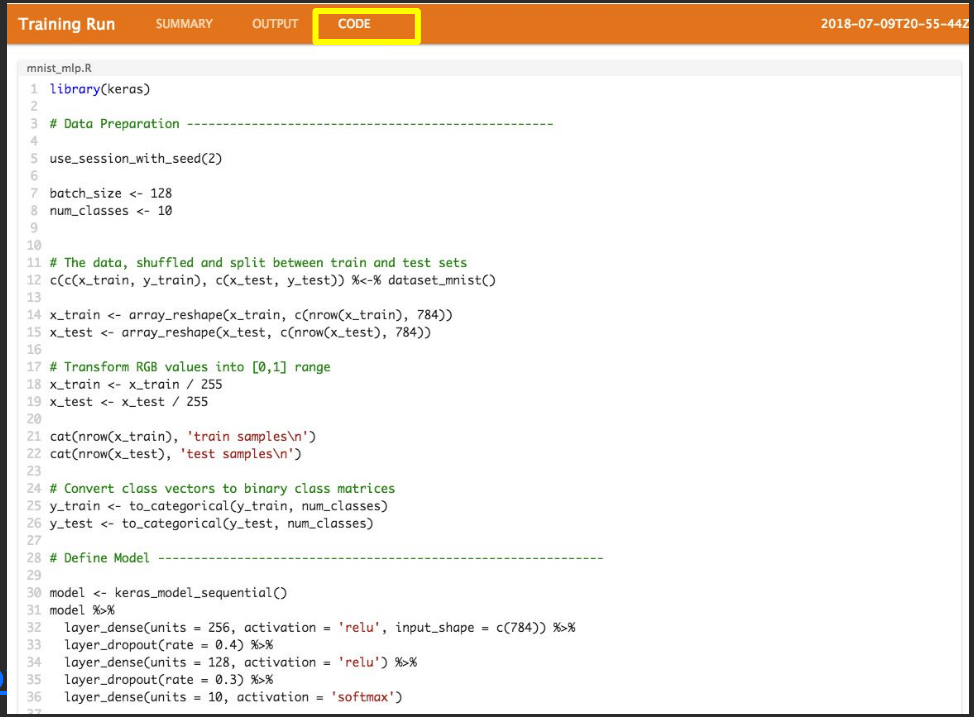
```r
library(tfruns)
tfruns::training_run("mnist_mlp.R")
```

tfruns



Training Run
SUMMARY
OUTPUT
CODE
2018-07-09T20-55-44Z
History
loss
val_loss
acc
val_acc
Run
context local
script mnist_mlp.R
started 2018-07-09 20:55:44 GMT
time 00:01:02
Metrics
loss 0.1433
acc 0.9624
val_loss 0.1063
val_acc 0.9761
Evaluation
eval_loss 0.1082
eval_acc 0.9753
Optimization
loss categorical_crossentropy
optimizer <keras.optimizers.RMSprop>
lr 0.001
Training
samples 48,000
validation_samples 12,000
epochs 20
batch_size 128
Model
Layer (type) Output Shape Param #
dense_1 (Dense) (None, 256) 200960
dropout_1 (Dropout) (None, 256) 0
dense_2 (Dense) (None, 128) 32896
dropout_2 (Dropout) (None, 128) 0
dense_3 (Dense) (None, 10) 1290
Total params: 235,146
Trainable params: 235,146
Non-trainable params: 0

Training Run
SUMMARY
OUTPUT
CODE
Plots
acc
loss
epoch
data
training
validation
Console
> library(keras)
> # Data Preparation ---
>
> use_session_with_seed(2)
> batch_size <- 128



Training Run
SUMMARY
OUTPUT
CODE
2018-07-09T20-55-44Z
mnist_mlp.R
library(keras)
# Data Preparation ---
use_session_with_seed(2)
batch_size <- 128
num_classes <- 10
# The data, shuffled and split between train and test sets
c(c(x_train, y_train), c(x_test, y_test)) %<-% dataset_mnist()
x_train <- array_reshape(x_train, c(nrow(x_train), 784))
x_test <- array_reshape(x_test, c(nrow(x_test), 784))
# Transform RGB values into [0,1] range
x_train <- x_train / 255
x_test <- x_test / 255
cat(nrow(x_train), 'train samples\n')
cat(nrow(x_test), 'test samples\n')
# Convert class vectors to binary class matrices
y_train <- to_categorical(y_train, num_classes)
y_test <- to_categorical(y_test, num_classes)
# Define Model --
model <- keras_model_sequential()
model %>%
layer_dense(units = 256, activation = 'relu', input_shape = c(784)) %>%
layer_dropout(rate = 0.4) %>%
layer_dense(units = 128, activation = 'relu') %>%
layer_dropout(rate = 0.3) %>%
layer_dense(units = 10, activation = 'softmax')

When running another model

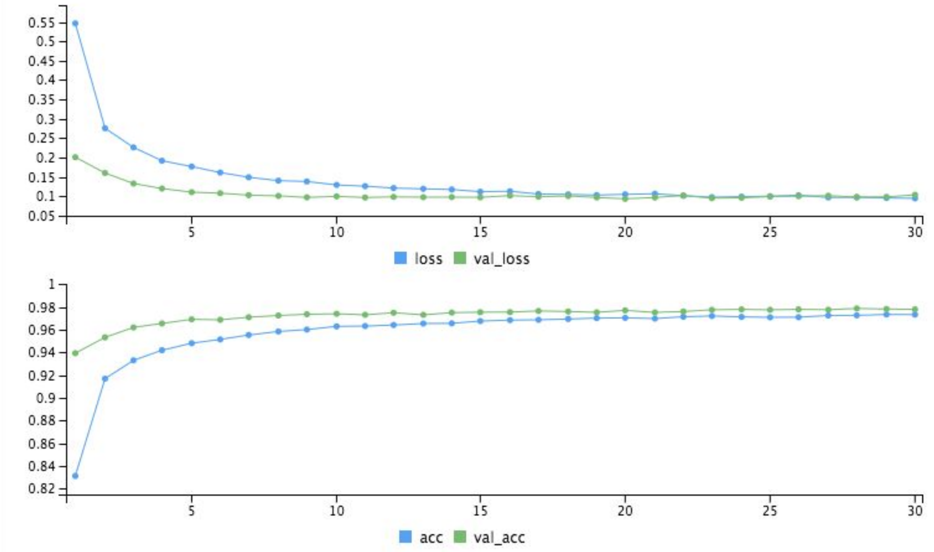
```r
# Define Model -------------------------------------------------------------

model <- keras_model_sequential()
model %>%
  layer_dense(units = 256, activation = 'relu', input_shape = c(784)) %>%
  layer_dropout(rate = 0.4) %>%
  layer_dense(units = 128, activation = 'relu') %>%
  layer_dropout(rate = 0.3) %>%
  layer_dense(units = 10, activation = 'softmax')


model %>% compile(
  loss = 'categorical_crossentropy',
  optimizer = optimizer_rmsprop(lr = 0.001),
  metrics = c('accuracy')
)

# Training & Evaluation ----------------------------------------------------

history <- model %>% fit(
  x_train, y_train,
  batch_size = batch_size,
  epochs = 20, 30
  verbose = 1,
  validation_split = 0.2
)
```

128

```r
library(tfruns)
tfruns::training_run("mnist_mlp.R")
```

Training Run SUMMARY OUTPUT CODE 2018-07-09T21-14-57Z

History



Model

```
Layer (type)                 Output Shape              Param #
================================================================
dense_1 (Dense)              (None, 128)               100480
________________________________________________________________
dropout_1 (Dropout)          (None, 128)               0
________________________________________________________________
dense_2 (Dense)              (None, 128)               16512
________________________________________________________________
dropout_2 (Dropout)          (None, 128)               0
________________________________________________________________
dense_3 (Dense)              (None, 10)                1290
================================================================
Total params: 118,282
Trainable params: 118,282
Non-trainable params: 0
________________________________________________________________
```

Run	
context	local
script	mnist_mlp.R
started	2018-07-09 21:14:57 GMT
time	00:01:01

Metrics	
loss	0.0942
acc	0.9735
val_loss	0.1034
val_acc	0.9781

Evaluation	
eval_loss	0.0986
eval_acc	0.9783

Optimization	
loss	categorical_crossentropy
optimizer	<keras.optimizers.RMSprop>
lr	0.001

Training	
samples	48,000
validation_samples	12,000
epochs	30
batch_size	128

When comparing runs (models)

```
tfruns::compare_runs()
```

Compare Runs 2018-07-09T20-55-44Z 2018-07-09T21-14-57Z

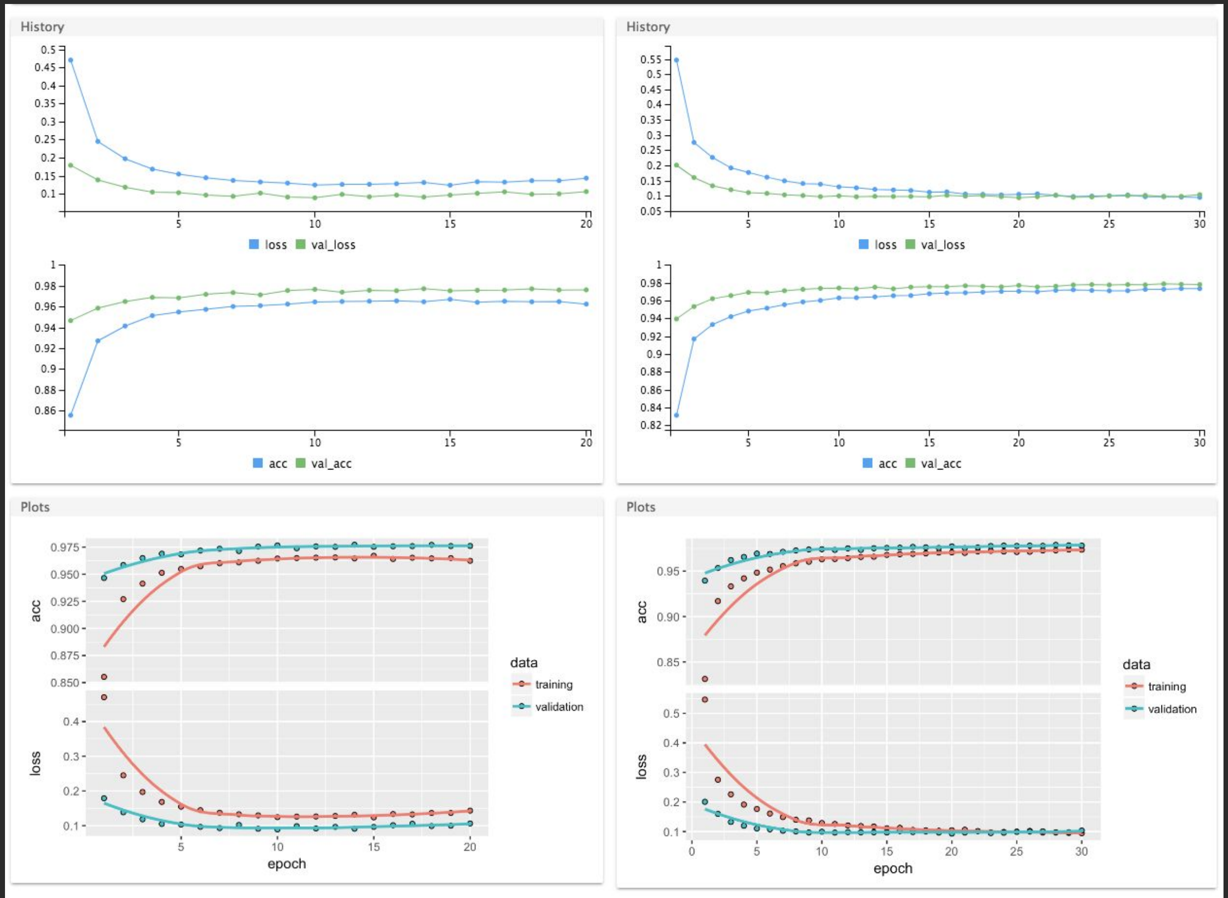
Run		Run	
context	local	context	local
script	mnist_mlp.R	script	mnist_mlp.R
started	2018-07-09 20:55:44 GMT	started	2018-07-09 21:14:57 GMT
time	00:01:02	time	00:01:01

Metrics		Metrics	
loss	0.1433	loss	0.0942
acc	0.9624	acc	0.9735
val_loss	0.1063	val_loss	0.1034
val_acc	0.9761	val_acc	0.9781

Evaluation		Evaluation	
eval_loss	0.1082	eval_loss	0.0986
eval_acc	0.9753	eval_acc	0.9783

mnist_mlp.R

```
@@ -28,9 +28,9 @@
  # Define Model --------------------------------------------------------------

  model <- keras_model_sequential()
  model %>%
-   layer_dense(units = 256, activation = 'relu', input_shape = c(784)) %>%
+   layer_dense(units = 128, activation = 'relu', input_shape = c(784)) %>%
    layer_dropout(rate = 0.4) %>%
    layer_dense(units = 128, activation = 'relu') %>%
    layer_dropout(rate = 0.3) %>%
    layer_dense(units = 10, activation = 'softmax')
@@ -46,9 +46,9 @@

  history <- model %>% fit(
    x_train, y_train,
    batch_size = batch_size,
-   epochs = 20,
+   epochs = 30,
    verbose = 1,
    validation_split = 0.2
  )
```



IBM CODE

Slides available at: http://bit.ly/rtensorflow-oscon18

gdequeiroz / mmmpork

Training Flags - tfruns::flags()

```r
# Define Model -------------------------------------------------------------
model <- keras_model_sequential()
model %>%
  layer_dense(units = FLAGS$dense_units1, activation = 'relu', input_shape = c(784)) %>%
  layer_dropout(rate = 0.4) %>%
  layer_dense(units = 128, activation = 'relu') %>%
  layer_dropout(rate = 0.3) %>%
  layer_dense(units = 10, activation = 'softmax')


model %>% compile(
  loss = 'categorical_crossentropy',
  optimizer = optimizer_rmsprop(lr = 0.001),
  metrics = c('accuracy')
)

# Training & Evaluation ----------------------------------------------------

history <- model %>% fit(
  x_train, y_train,
  batch_size = 128,
  epochs = FLAGS$epochs,
  verbose = 1,
  validation_split = 0.2
)
```

Tuning hyperparameters - tfruns::tuning_run()

```
runs <- tfruns::tuning_run("mnist_mlp_FLAGS_TUNING.R", flags = list(
  dense_units1 = c(256, 128),
  epochs = c(20, 30)
))
```

flags = list(256, 20);
flags = list(128, 20);
flags = list(256, 30);
flags = list(128, 30)

```
4 total combinations of flags (use sample parameter to run a random subset)
Proceed with tuning run? [Y/n]: Y
Training run 1/4 (flags = list(256, 20))
Using run directory runs/2018-07-16T22-09-32Z

> library(keras)

> # Data Preparation ---------------------------------------
>
```

Tuning hyperparameters - `tfruns::tuning_run()`

```
> runs[order(runs$eval_acc, decreasing = TRUE), 1:10]
Data frame: 4 x 10
```

	run_dir	eval_loss	eval_acc	metric_loss	metric_acc	metric_val_loss	metric_val_acc	flag_dense_units1	flag_epochs	samples
2	runs/2018-07-16T22-10-54Z	0.0964	0.9821	0.0526	0.9861	0.1107	0.9803	256	30	48000
4	runs/2018-07-16T22-09-32Z	0.0965	0.9798	0.0583	0.9843	0.1008	0.9791	256	20	48000
3	runs/2018-07-16T22-10-21Z	0.0998	0.9769	0.0974	0.9721	0.1066	0.9772	128	20	48000
1	runs/2018-07-16T22-11-53Z	0.1131	0.9766	0.0861	0.9763	0.1065	0.9785	128	30	48000

The best model is the model #2 with 256 dense units and 30 epochs

Reproducibility

tfdeploy
Sharing Models for Convenient Collaboration

- **Archive** Models for reproducible research
- **Export and Import** Models for later reuse
- **Deploy** Models as a Service

Archive Models for Reproducible Research

Save in HDF5 or human-readable formats YAML + JSON to use it in R

```
```{r}
write(model_to_yaml(model), "models/mnist.yaml")
```

```{r}
raw_model <- serialize_model(model)
write(raw_model, "models/mnist_raw.txt")
```

```{r, eval=FALSE}
save_model_hdf5(model, filepath = "models/mnist_hdf5.h5")
save_model_weights_hdf5(model, filepath="models/mnist_weights_hdf5.h5")
```
```

Load saved models for instant reuse

```
```{r}
model_dense <- load_model_hdf5("models/mnist_dense_hdf5.h5")
```
```

IBM CODE

Export Models

Use export_savedmodel() when you want to use it outside of R

```
```{r, eval=FALSE}
model %>%
  layer_dense(units = 256, activation = 'relu', input_shape = c(784),
              name = "image") %>%
  layer_dense(units = 128, activation = 'relu') %>%
  layer_dense(units = 10, activation = 'softmax',
              name = "prediction")
```

```{r}
export_savedmodel(model_dense, "models/savedmodel")
```

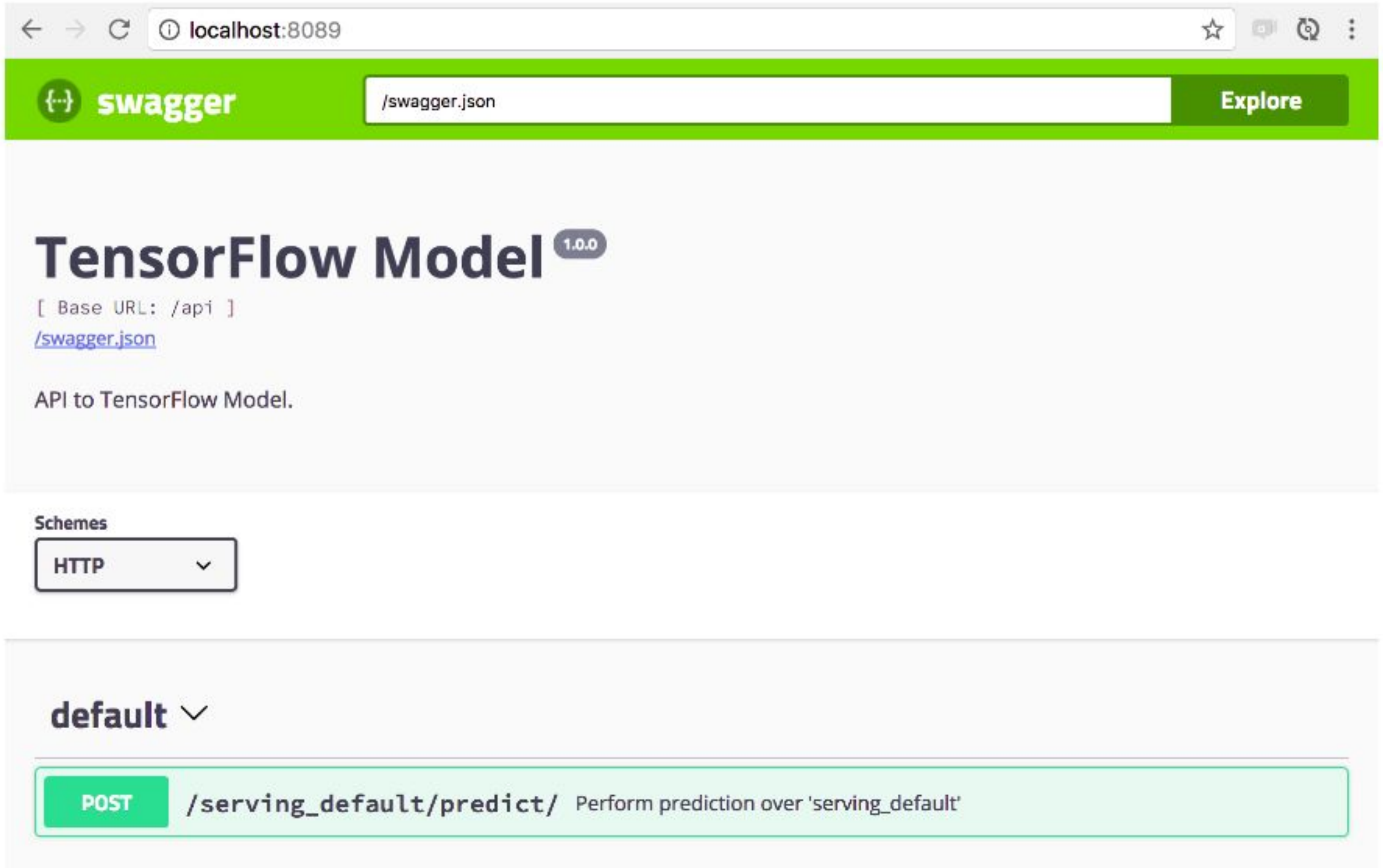
Keras learning phase set to 0 for export (restart R session before doing additional training)

```{r}
view_savedmodel("models/savedmodel")
```
```

IBM CODE

Deploy Models

```r
serve_savedmodel('savedmodel', browse = TRUE)
```



Transparency

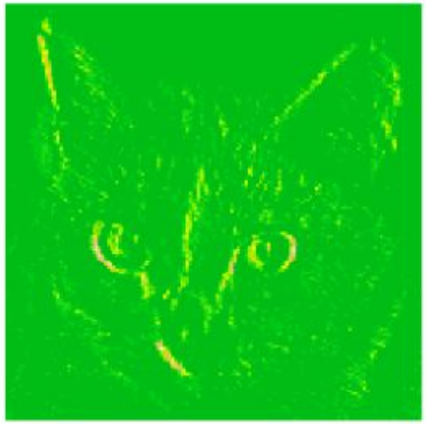
Explainable AI
Show Your Work for Transparency + Trust

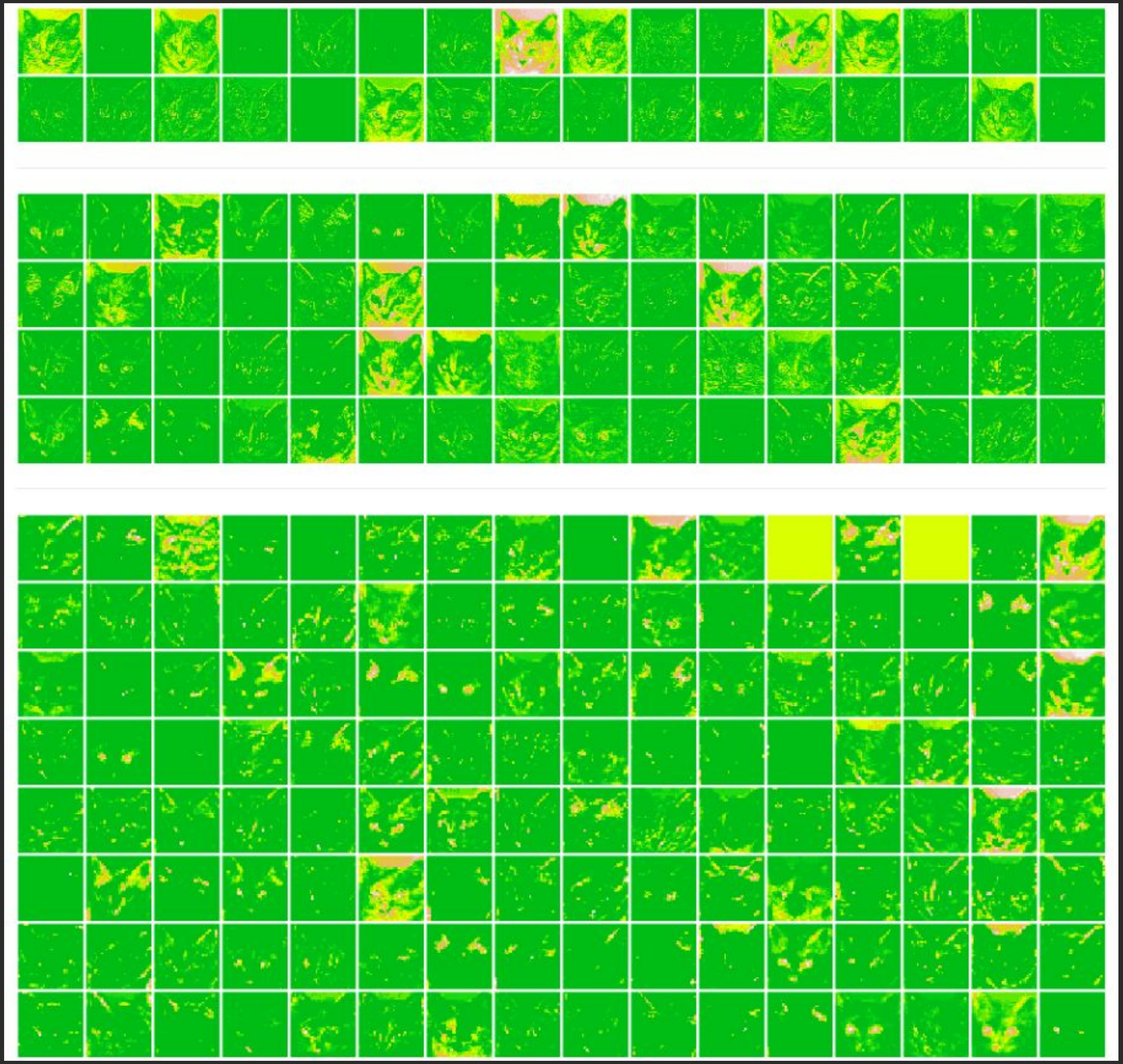
- **Visualize** model layers in Rmarkdown
- **Regression Analysis** with kerasformula::kms()
- **Introspect** blackbox models with LIME

Visualize Model Layers in Rmarkdown

```
plot(as.raster(img_tensor[1,,,]))
```

```
plot_channel(first_layer_activation[1,,,5])
```





IBM CODE

Slides available at: http://bit.ly/rtensorflow-oscon18

gdequeiroz / mmmpork

Instant Regression with kerasformula::kms()

```
popularity <- kms(pop_input, rstats[1:1000,])
predictions <- predict(popularity, rstats[1001:2000,])
predictions$accuracy
```

```
[1] 0.579
```

```
popularity$confusion
```

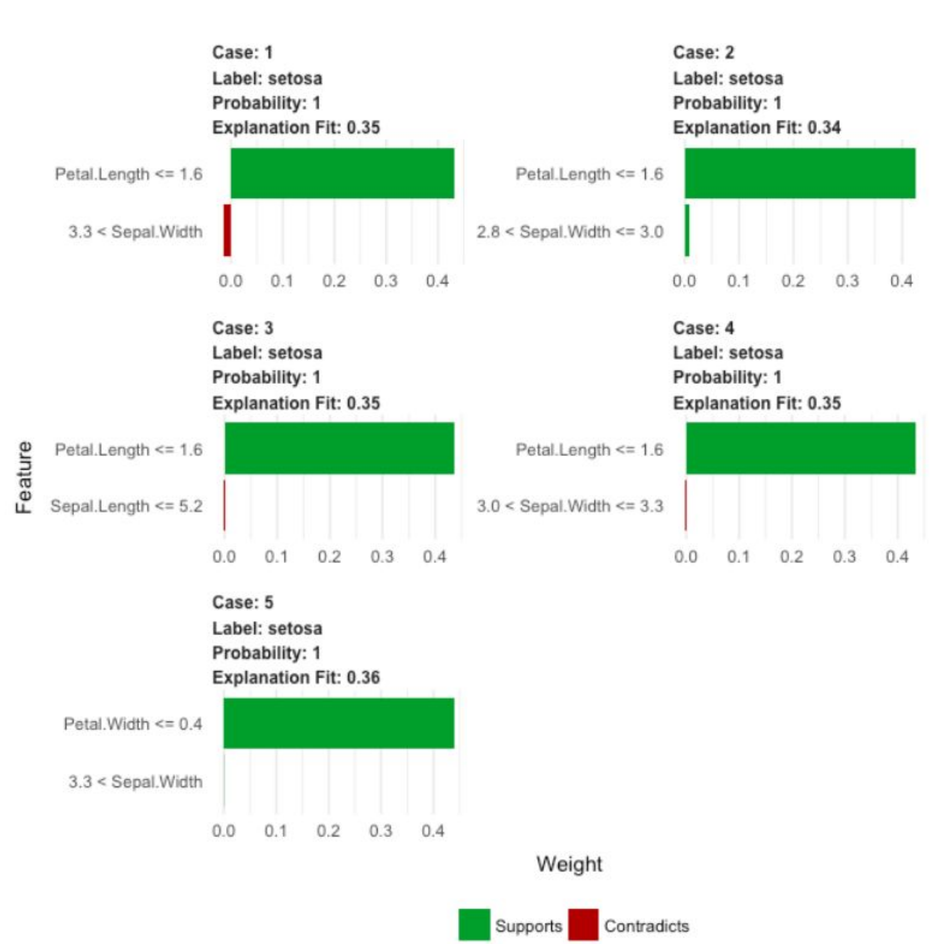
	(-1,0]	(0,1]	(1,10]	(10,100]	(100, 1000]	(1000
(-1,0]	37	12	28	2	0	0
(0,1]	14	19	72	1	0	0
(1,10]	6	11	187	30	0	0
(10,100]	1	3	54	68	0	0
(100, 1000]	0	0	4	10	0	0
(1000, 10000]	0	0	0	1	0	0

Introspect blackbox models with LIME

```r
# Create Random Forest model on iris data
model <- train(iris_train, iris_lab, method = 'rf')

# Create an explainer object
explainer <- lime(iris_train, model)

# Explain new observation
explanation <- explain(iris_test, explainer, n_labels = 1, n_features = 2)
```

```r
explanation <- .load_image_example()

plot_image_explanation(explanation)
```

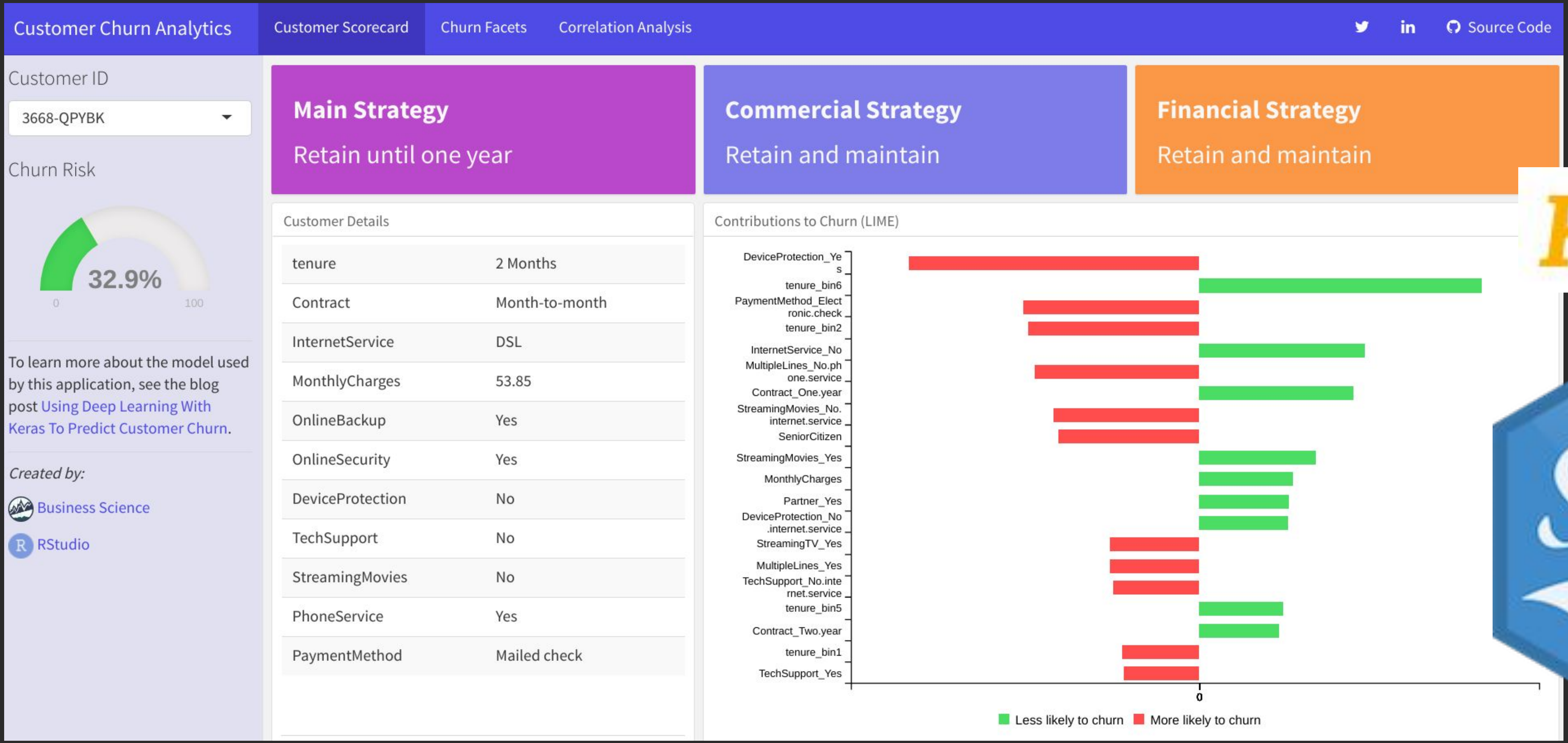

```r
plot_features(explanation)
```



IBM CODE

Slides available at: http://bit.ly/rtensorflow-oscon18
gdequeiroz / mmmpork

Put it in Action!



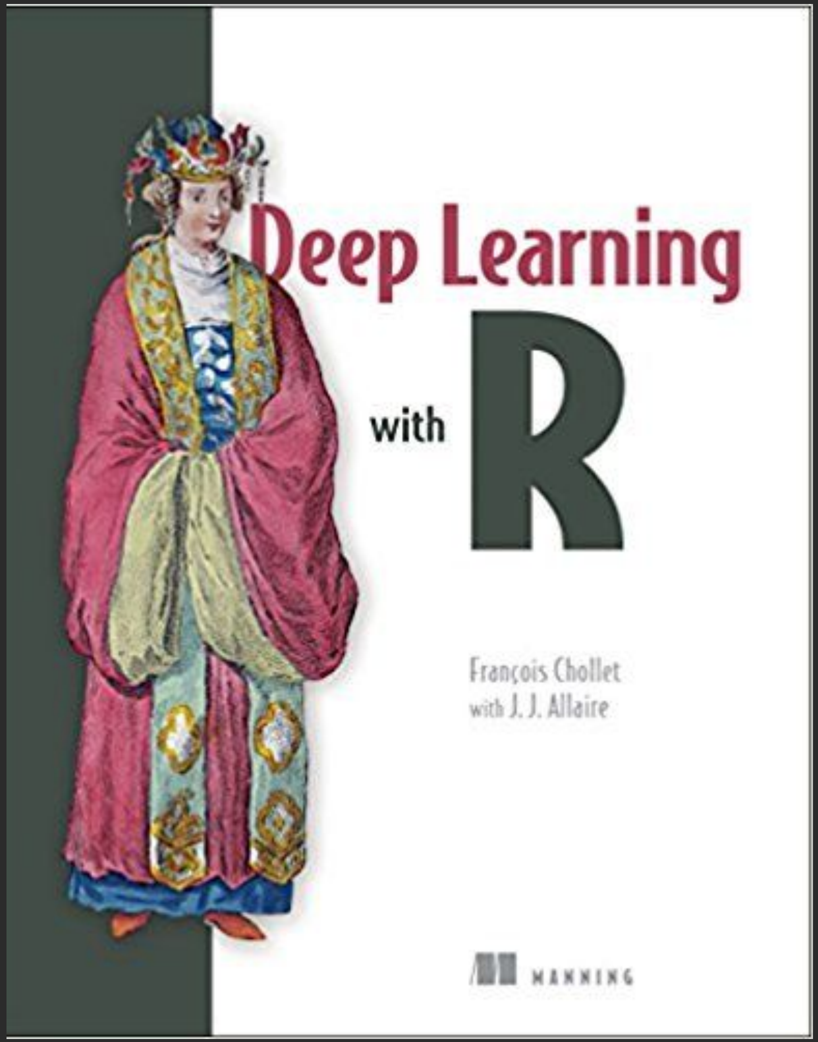
https://blogs.rstudio.com/tensorflow/posts/2018-01-11-keras-customer-churn/

See it live! https://jjallaire.shinyapps.io/keras-customer-churn/

Slides available at: http://bit.ly/rtensorflow-oscon18

gdequeiroz / mmmpork

Resources



- https://tensorflow.rstudio.com/
- https://keras.rstudio.com/

Slides available at: http://bit.ly/rtensorflow-oscon18
gdequeiroz / mmmpork

VISIT IBM BOOTH #401 AT 12:45-1:30 PM ON JULY 18 TO JOIN A WORKSHOP ABOUT THE CALL FOR CODE CHALLENGE.
LEARN HOW YOU CAN USE IBM WATSON TO ANSWER THE CALL.

FIND OUT HOW AT

developer.ibm.com/callforcode

CALL FOR CODE

IBM

Call for Code Founding Partner

Thank you!

codait.org

developer.ibm.com/code

http://github.com/codait

R Ladies

https://rladies.org/

@RLadiesGlobal

FfDL

MAX

Sign up for IBM Cloud and try Watson Studio!

https://ibm.biz/BdYRNi