

Neural Networks in R

Jacqueline Nolis
Head of Data Science
@skyetetra



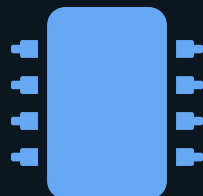


A flexible cloud platform for data science

(and free for 30 hours a month)



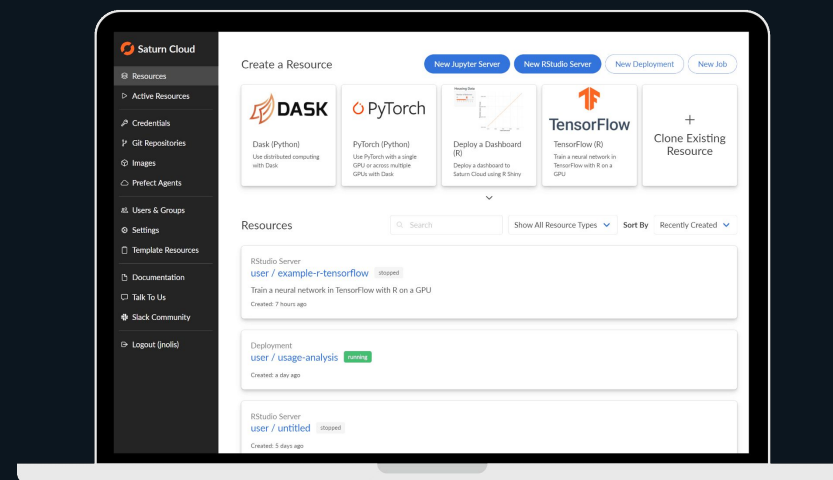
Switch from
developing on a
laptop to the
cloud



Use GPUs, up
to 4TB of RAM,
and over 100
core machines



Host
dashboards,
apis, and
scheduled jobs





Now has full R support!

- Edit code in RStudio
- Use R with GPUs, 4TB of RAM, 100 cores
- Host Shiny & Plumber apps, schedule R jobs
- Works with RStudio Connect and Package Manager

Neural Networks and R

(first: what are neural networks?)

What is a linear regression?

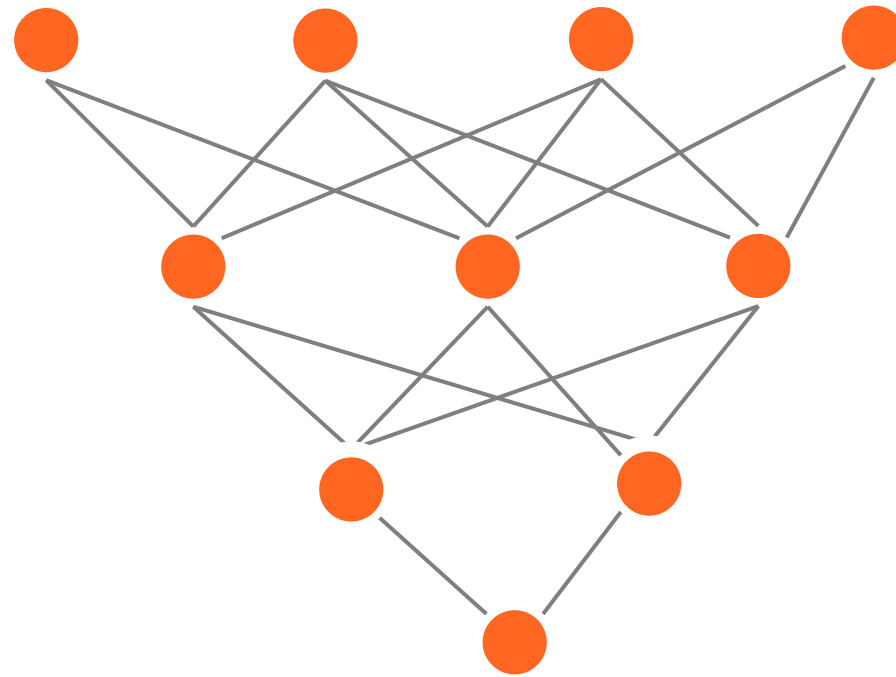
$$y = a_1x_1 + a_2x_2 + a_3x_3 + a_4x_4$$



Linear regression $y \sim X$

```
lm(y ~ x1 + x2 + x3 + x4, data)
```

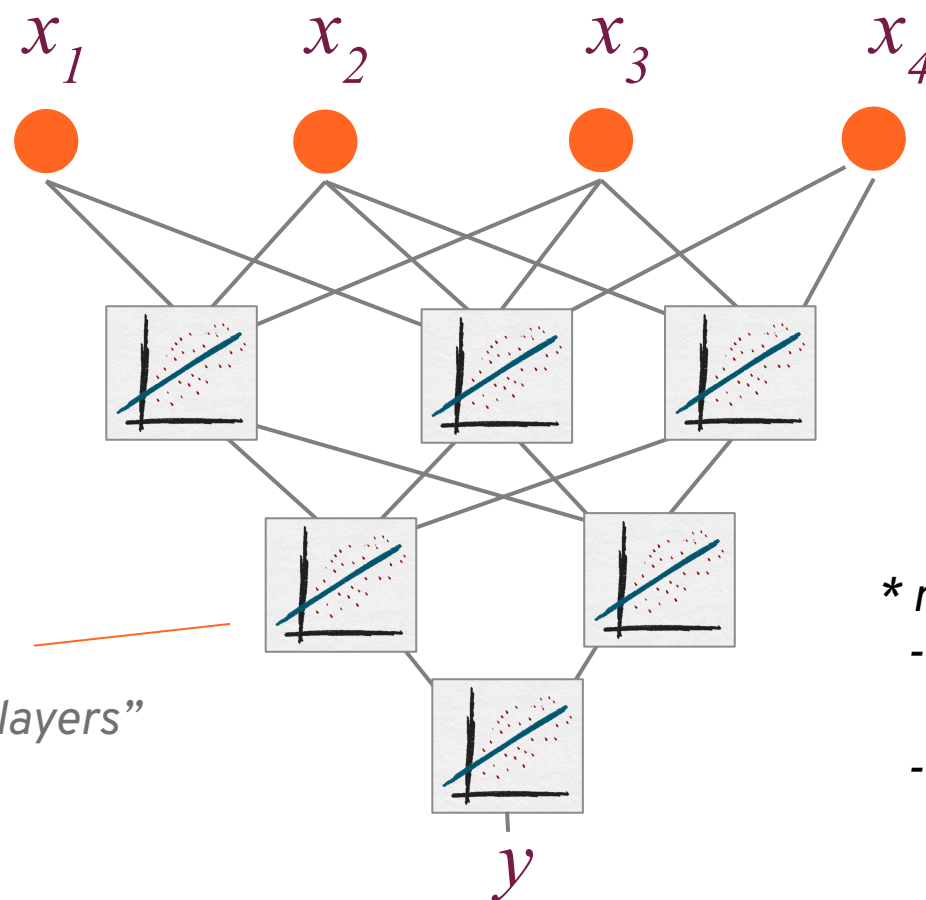
What are neural networks?



Neural network

What are neural networks?

Neural
networks are
powerful



The rows of linear
regressions called “layers”

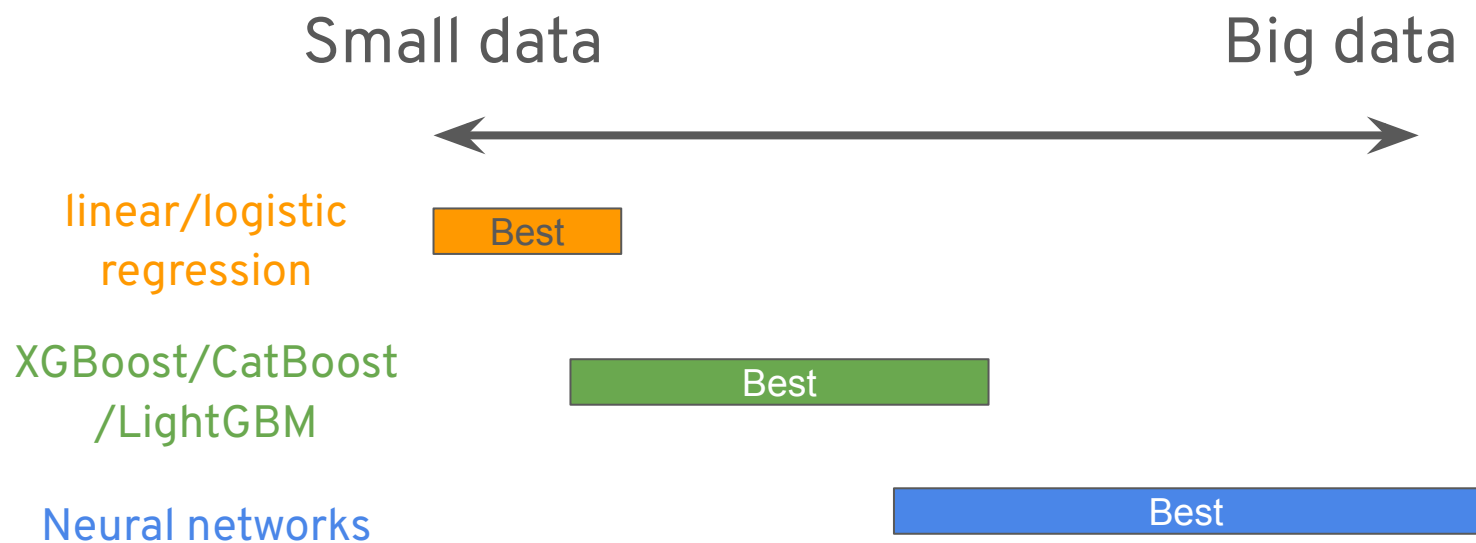
Basically* string a
bunch of linear
regressions together!

Still $y \sim X$

* notes:

- fitting the regressions gets complicated
- sometimes apply functions to output of each layer (called activation)

When are neural networks helpful?



- Can be extremely powerful with large data
- Not particularly effective with small amounts of data
- Some network layers are especially useful for different types of data (images, NLP)



Neural networks are FUNNY

(Great for generating data)

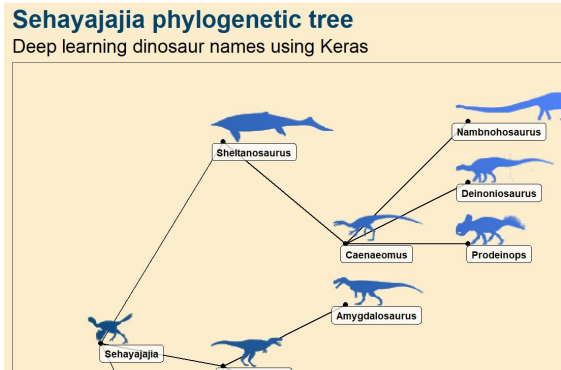


Ryan Timpe
@ryantimpe

Julia Silge
@julasilge

Namita
Nandakumar
@nnstats

(me)



Dinosaur species

the happiness of her sisters and had heard of the earnest of the servance of the consequence of the family of the very little to her friend to her family of the lady of the moment to the persuaded her to her sisters were allow to

Jane Austen texts

Aelora	Aelyx	Aerysa
Daeger	Daella	Daemion
Daenyra	Daeron	Dregon
Jaegor	Maegon	Maegys
Rhaegel	Rhaegon	Rhaenor
Vaella	Vaelon	Valera

House Targaryen names

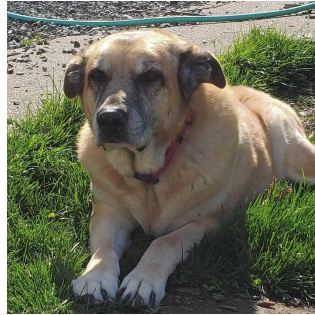


Offensive license plates

Training neural networks in R

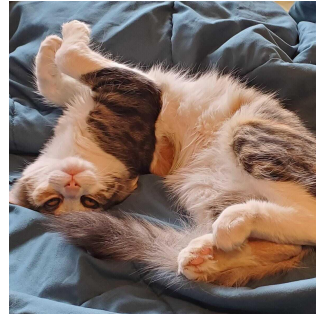
Project: let's make pet names!

Using data from the city of Seattle



~~Kaiser~~

Mosdin



~~Bliss~~

Graggie



~~Clark~~

Raggopies

Modern Neural Network Libraries



Deep learning is in R!

- **keras/tensorflow R libraries** (what we'll be using)
 - Simple tidy syntax for Keras
 - TensorFlow power when needed
 - “Secretly” runs Python and TensorFlow (with Keras) on the backend with reticulate package
- **torch R libraries**
 - C++ implementation (no Python needed)

(projects are lead by RStudio)



The data

- Years of pet license records
- Only care about the name
- Train a neural network to learn the patterns

	A	B	C	D	E	F	G
1	License Issue Date	License Number	Animal's Name	Species	Primary Breed	Secondary Breed	ZIP Code
2	May 30 2017	79347322	Mr Darcy	Cat	Domestic Shorthair		98103
3	December 17 201	20023870	Shasta	Dog	American Eskimo		98119
4	November 02 20	3104081	Kuya	Dog	Beagle		98144
5	November 29 20	1366685	Teddy	Cat	Domestic Shorthair	Mix	98122
6	November 04 20	1356097	Wally	Cat	Domestic Longhair	Mix	98115
7	October 02 2016	1213812	Kemp	Cat	Domestic Shorthair		98121
8	June 02 2016	1150096	Graham	Cat	Domestic Shorthair	Mix	98119
9	September 01 20	1000036	Mr Collins	Cat	Domestic Shorthair	Mix	98117
10	December 13 201	982441	Jingle	Cat	Domestic Shorthair		98116
11	August 23 2017	964907	Momo	Dog	Dachshund, Miniature Smooth Haired		98112
12	September 05 20	964906	Mabel	Dog	Spaniel	Poodle, Miniature	98105
13	July 27 2017	964903	Duncan	Dog	Dachshund, Miniature	Australian Cattle	98125
14	October 16 2017	964901	Amelie	Dog	Retriever, Labrador	Poodle, Standard	98112
15	September 20 20	964837	Ozzie	Cat	Domestic Shorthair		98107
16	January 26 2017	964836	Amelie	Cat	Domestic Medium Hair		98102-3269
17	January 25 2017	964834	Sundae	Cat	Domestic Shorthair		98199
18	January 25 2017	964833	Harvey	Cat	Domestic Shorthair		98199
19	December 08 201	964823	Maggie	Cat	Domestic Medium Hair		98115

Format the data

- Want to predict the next letter based on previous letters
- Training data for name “Spot”:

X_1	X_2	X_3	X_4	X_5	Y
(blank)	(blank)	(blank)	(blank)	(blank)	S
(blank)	(blank)	(blank)	(blank)	S	P
(blank)	(blank)	(blank)	S	P	O
(blank)	(blank)	S	P	O	T
(blank)	S	P	O	T	(stop)

Convert letters to numbers

- Create a dictionary (A=1,B=2,...)
- One hot encode each number (3 = [0,0,1,0,...,0])

__spo \square [0,0,19,16,15] \square [...,[0,0,...,1,...,0], ...]

Input X is a 3-dimensional binary matrix! Size: (num_rows, max_length, num_chars)

Target y is a 2-dimensional binary matrix! Size: (num_rows, num_chars)

Highlights from the R code

```
name %>%  
  str_c("+") %>%  
  str_split("") %>%  
  map( ~ purrr::accumulate(.x, c) )
```

Accumulate from purr is
an easy way to unroll the
names

```
text_matrix <-  
  subsequence_data %>%  
  map(~ char_lookup$id[match(.x, char_lookup$char)]) %>%  
  pad_sequences(maxlen = max_length+1) %>%  
  to_categorical(num_classes = num_chars)
```

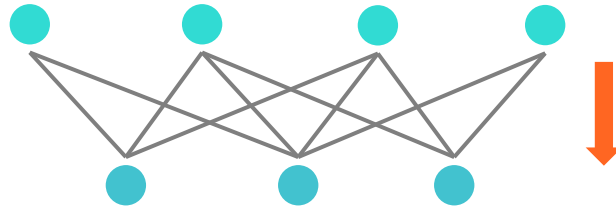
Keras has built in
functions for padding,
one-hot encoding, etc

Once we've formatted the data,
make the neural network!

Some network layers

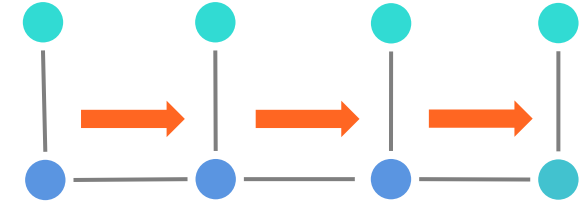
Dense

All inputs are fed into each regression



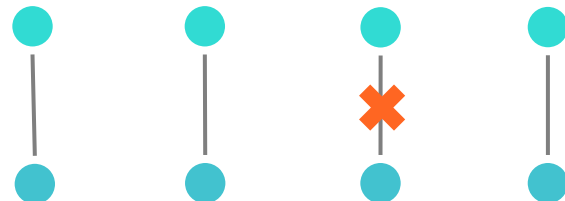
LSTM

Each input fed into one set of outputs, along with previous output



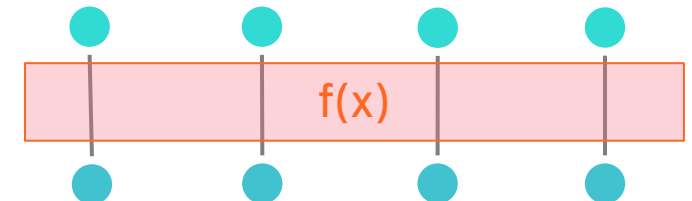
Dropout

Remove random values to avoid overfitting



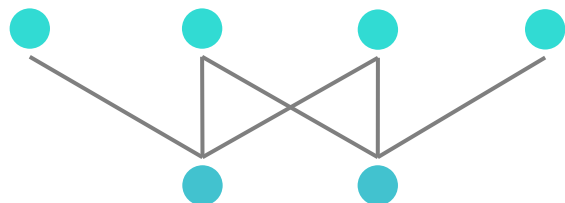
Activation

Apply a function across the values (sometimes built into layers)



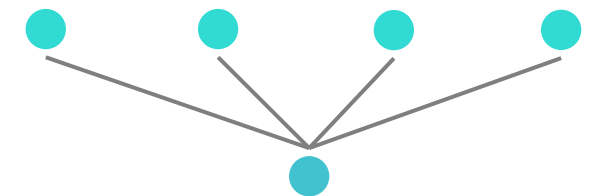
Convolution

Model adjacent points together



Global Pooling

Average values together



Our network

- Input data
- LSTM – figure out the patterns
- LSTM – figure out more patterns
- Dropout – don't overfit buddy 🙄
- Dense – to get one output for each letter
- Activation – make sure scores add up to one



The code

```
input <- layer_input(shape = c(max_length,num_characters))

output <-
  input %>%
  layer_lstm(units = 32, return_sequences = TRUE) %>%
  layer_lstm(units = 32, return_sequences = FALSE) %>%
  layer_dropout(rate = 0.2) %>%
  layer_dense(num_characters) %>%
  layer_activation("softmax")

model <- keras_model(inputs = input, outputs = output) %>%
  compile(
    loss = 'binary_crossentropy',
    optimizer = "adam"
  )
```

Set the input shape
based on data

Define the network
(each layer is a line of code!)

Choose how to optimize it
Choose loss based on output variable
Setting optimizer to “adam” usually fine

Train the neural network!

- Choices:

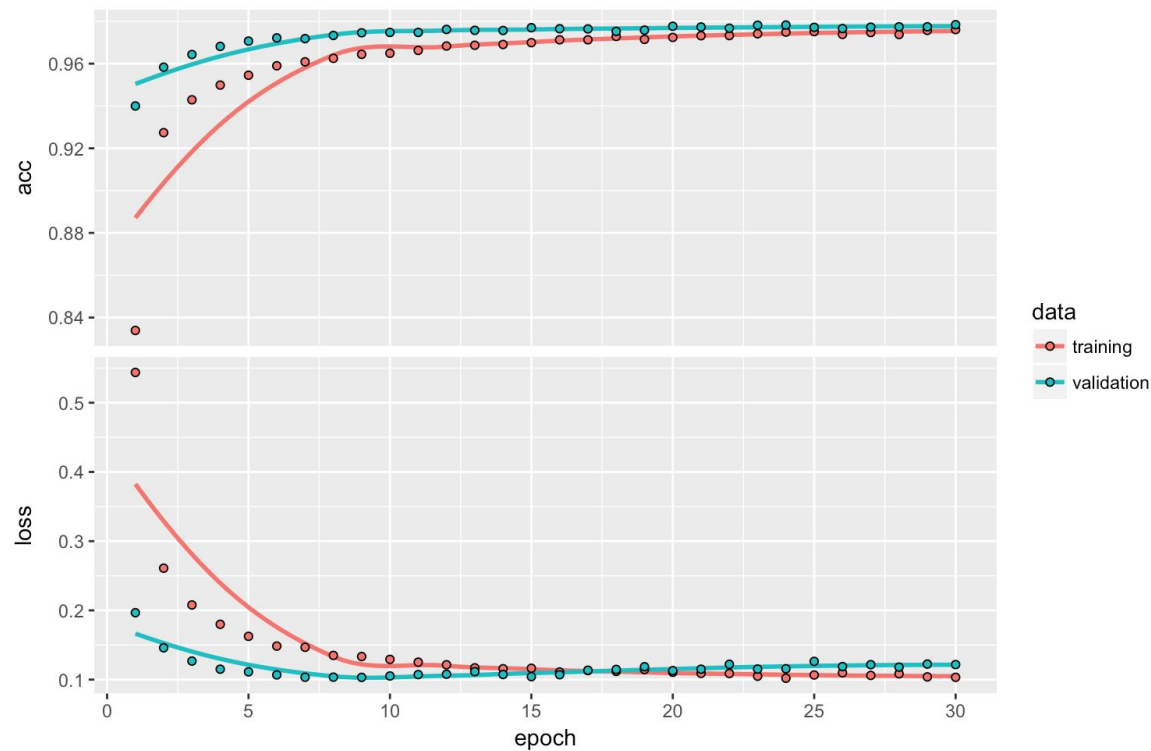
- Epochs - How many times to we want to fit data?
- Batch Size - How many rows of data do we want to fit at once?

- Answers:

- # epochs – Until it seems like it converges
- Batch size – (we'll come back to this one)

```
fit_results <-  
  model %>%  
  fit(  
    x_name,      Input X  
    y_name,      Target y  
    epochs = 25,  
    batch_size = 64  
  )
```

This might take a while...



You're done! Save your work!

```
save_model_hdf5(model, "model.h5")
```

Notes:

- You can't use standard RDS objects since the networks are python-y
- The network only works for predicting new inputs of the same size
- Loading can be fussy between different versions of TensorFlow or hardware

How to use the model?

- Like everything else in R, use predict: `predict(model,previous_letters_data)`
- For input letters, returns the probability of each letter being next
- Define a function that:
 - Starts with a blank string
 - Predicts the next letter & updates
 - Predicts the next letter & updates (Keep going!)
 - Stop character! Stop!!!



```
> generate_many_names(n=20, model, character_lookup, max_length)
[1] "Milo"      "Narma"    "Br"       "Madanda"  "Sweetie"  "Carbot"   "Sweester"
[8] "vippea"    "Daxs"     "Deone"    "Jake"     "Bokie"    "sajan"    "olivet"
[15] "Racho"     "Jinus"    "Akot"     "Leo"      "Sammy"    "Bingi"
```

Then it's fun time!

Hanta	Cheoper	Precy	Sowel	Northuba Hash	Skarma	Chica Buck	Kina	Pozella	Wharper
Dishon	Schunk	Booro	Eatha	Cobme Nandre	Inmles	Gared	Armie	Ruwiaodle	Sawie
Togar	Spok	Leaule	Alge	Shola Mitty	Shella	Freyna	Leesa	Mutch	Skio
Baice	Kentel	Rooce	Tiamet	Mog	Drolas	Toli	Kingy	Rarger	Mitcha
Aingo	Sabson	Feliga Sunpery	Losto	Sambie	Jori	Hark	Leeli	Sudley	Choose Vig
Dallie	Kand	Tetsie	Batley	Ruttie	Pomo	Devin	Inabelle	Spalat	Lura
Dia	Sorchie	Otan	Rocey	Siry	Pabk	Blaod	Ase	Kamon	Move
Phalmop	Choop	Ruccel	Kenasel	Chima	Makey	Linkie	Tugel	Koglot	Miron
Estoh	Mody	Channilo	Dernie	Brandel	Timuse	Pita Amming	Sammmy	Kittred	Mirhi
Poni	Them	Biaf	Tin	Cheet	Booby	Lule	Maycey	Isanna	Maggo
Cimisi	Aca	Timtle	Wintan	Arth	Leus	Minson	Boli	Castila	Proca

Live Code Time!

Switching to a GPU

Why do we care about GPUs?

- GPUs can do tiny parallel computations very efficiently
 - Neural network fittings are tiny computations
- Large datasets and large neural networks have huge speed boosts
 - Small neural networks or little data does not
- *On a GPU with a large batch size you can train much faster than a CPU*
 - *Multiple GPUs are even better*

Only useful if your problem is right for it!



Beware the flashy hype

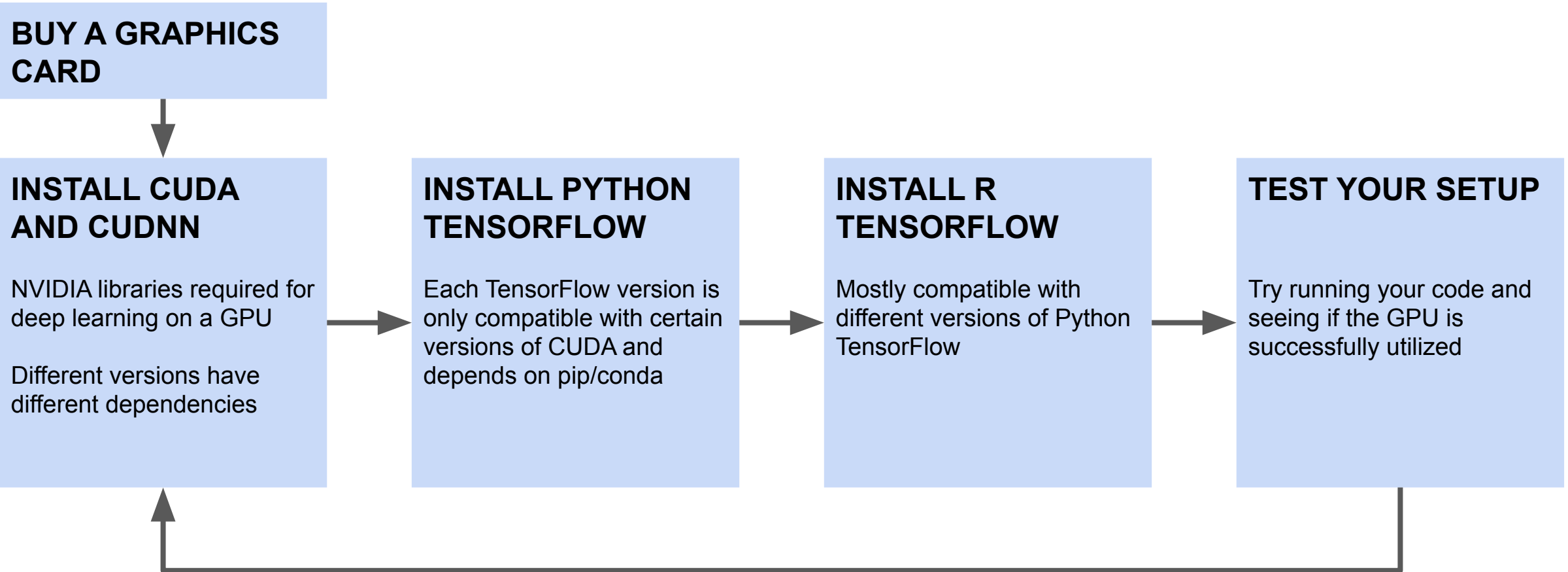
The good news

+0 -0

Switching to *running* on a GPU requires zero lines of R code change

The bad news

Using GPUs with TensorFlow in R requires a few repeated steps



The good news - utilize the help of others

rocker-project.org - R docker images for many situations (including GPU usage)!



saturn-rstudio-tensorflow - Saturn Cloud image based on [rocker/ml](https://rocker-project.org) with TensorFlow installed and working

Getting the benefits of GPUs

CPU

```
fit_results <-  
  model %>%  
  fit(  
    x_name,  
    y_name,  
    epochs = 25,  
    batch_size = 64  
  )
```

GPU

```
fit_results <-  
  model %>%  
  fit(  
    x_name,  
    y_name,  
    epochs = 500,  
    batch_size = 32768  
  )
```

- To get the GPU speed benefit you need a large batch size
- Large batch size = less learned per epoch (so need more)
- Only useful for large models!
- See saturncloud.io/blog/dask-with-gpus for more details

Let's try it

Wrapping it up

- Neural networks in R are not hard!
- Can be treated like a super-duper linear regression
- No R code change to use a GPU, but potentially lots of work to get packages set up
- Run this example on a GPU *right now*:
scld.io/ex/r-tensorflow
- Just see the code:
scld.io/docs/r-tensorflow



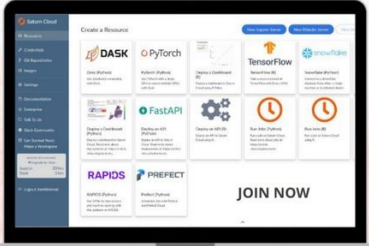
Thank you!

Jacqueline Nolis @skyetetra



Upcoming events

RStudio x Saturn Cloud Community Meetup



**Demo and
Q&A with
Jacqueline
Nolis**

January 19 @ 12 ET
add to calendar:
rstd.io/Jan19meetup

WEBINAR

Getting Started With Dask

Thursday, January 27 | 2 PM ET



Mitali Sanwal
Senior Data Scientist
SaturnCloud

REGISTER FOR FREE

