European Marketing Academy Conference
Hamburg, May 2019

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

What is deep learning, and why should I care?

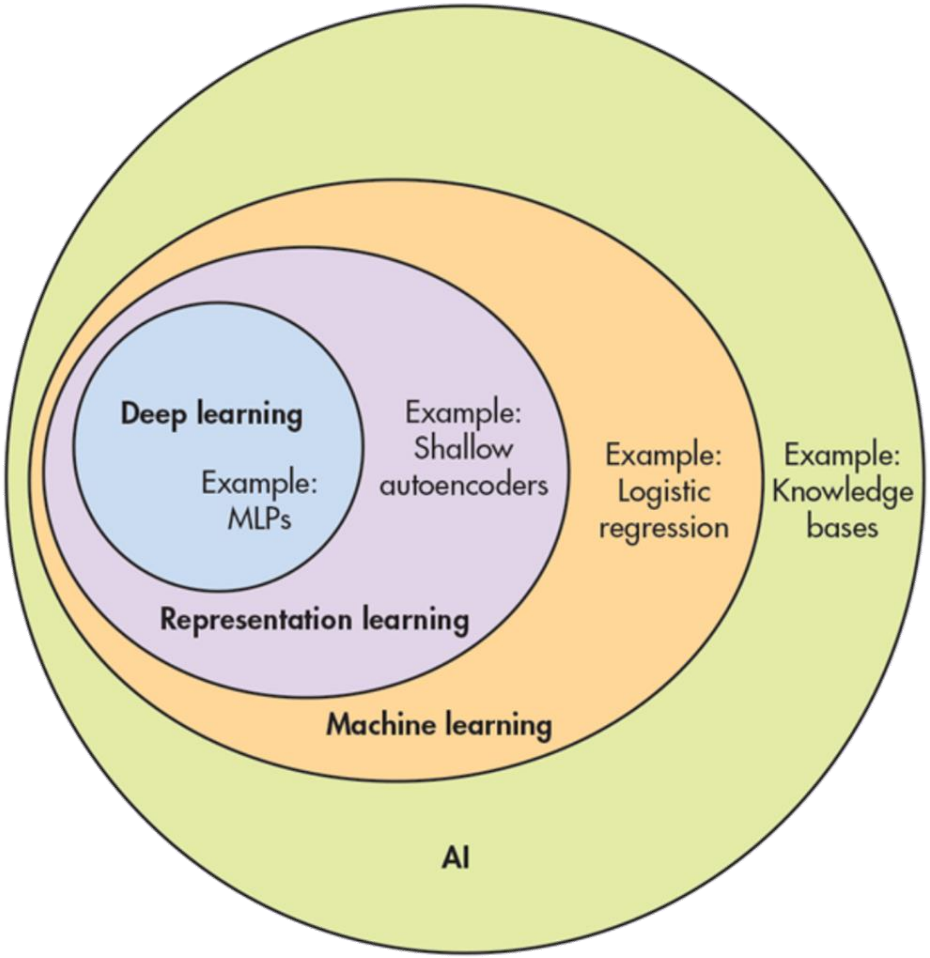
Stefan Mayer
School of Business and Economics, University of Tübingen

Deep Learning
is everywhere.

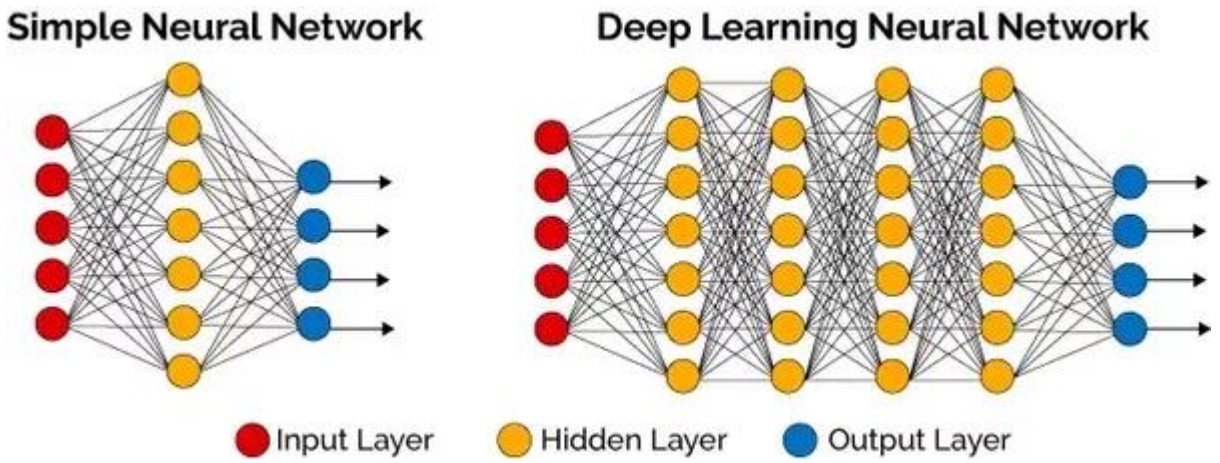
Outline

1. **Deep Learning**
   - → What is it (not), when is it useful
   - → Some examples from Marketing Research
2. **Deep Learning in R**
   - → Train model from scratch (brand logo classification)
   - → Transfer Learning
3. **Where to start**

1 | Deep Learning



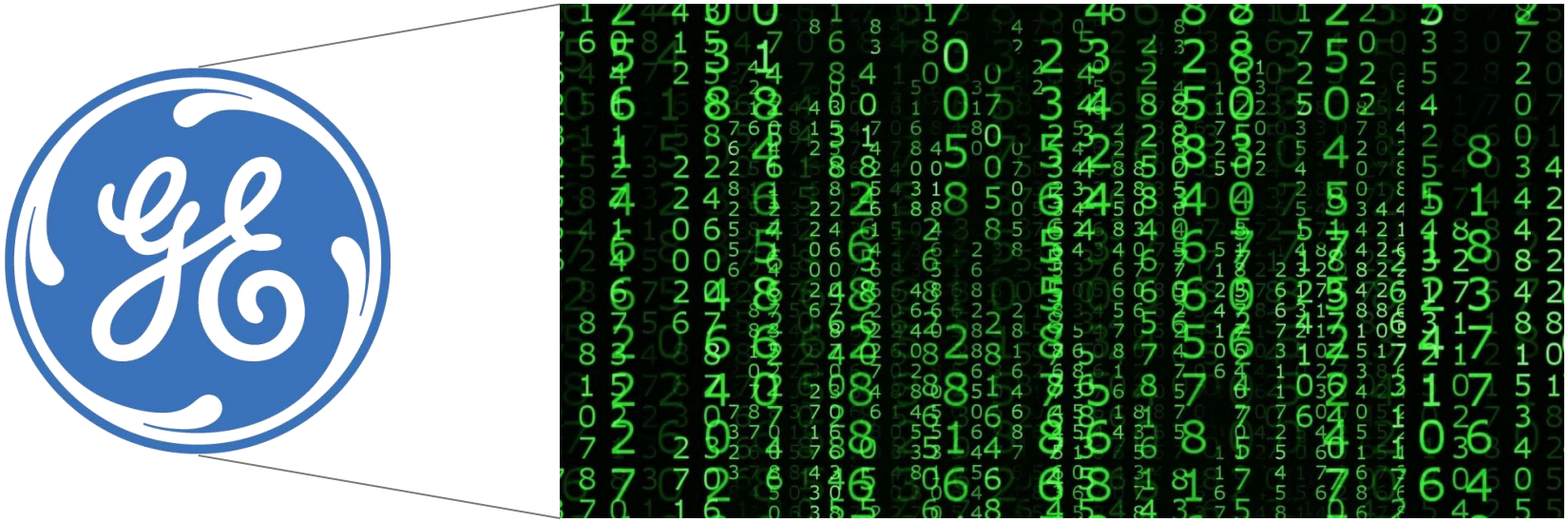
[Goodfellow, Bengio, & Courville (2016). Deep Learning. MIT Press.]


Simple Neural Network
Deep Learning Neural Network
Input Layer
Hidden Layer
Output Layer

Images / text consist of thousands of data points

- A single (small) 200 x 200 pixel colored image consists of 3 x 40k data points

  → How to describe the image then?



Traditional "Shallow" Architecture for Image Classification (i.e., traditional ML)

- Hand-crafted features (extraction can be automated)

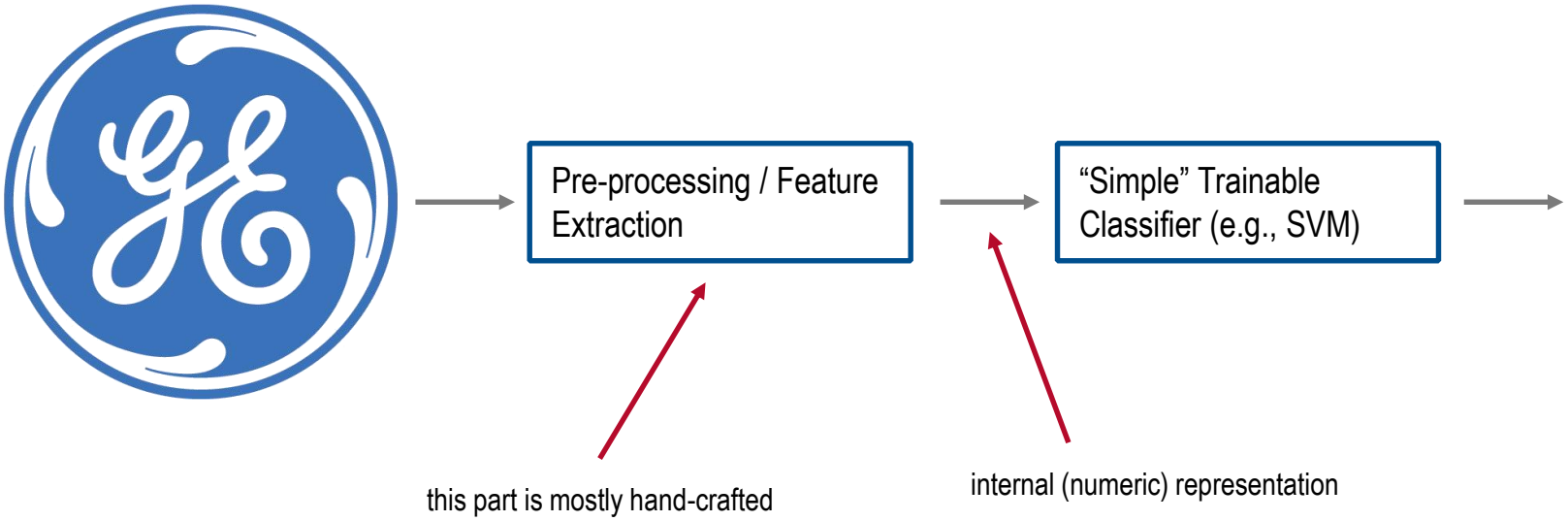
Color
- Primary color
- Color histogram

Shape
- Number of straight lines
- Percentage of parallel lines
- Histogram of line orientations
- Percentage of local corners
- Histogram of Oriented Gradients (HOG)

"Gestalt"
- Symmetry
- Complexity
- Contrast

Traditional "Shallow" Architecture for Image Classification (i.e., traditional ML)

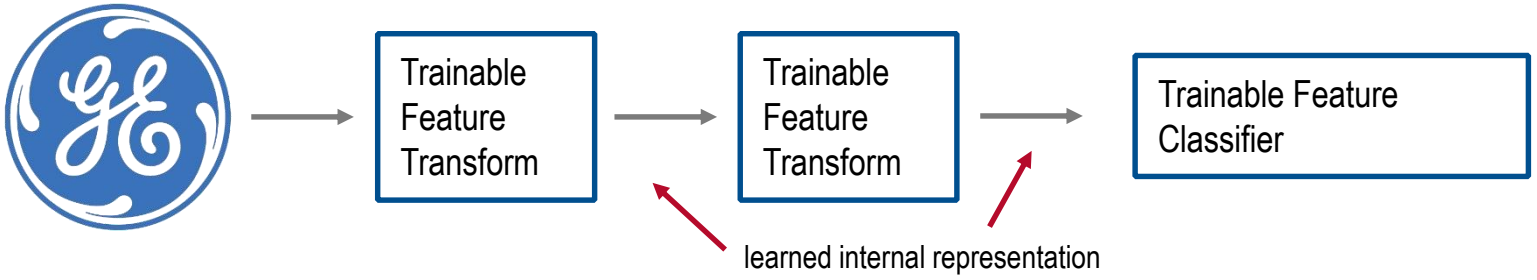
- Raw input is pre-processed through hand-crafted feature extractor
- The features are not learned



Deep Learning: Learning a hierarchy of internal representations

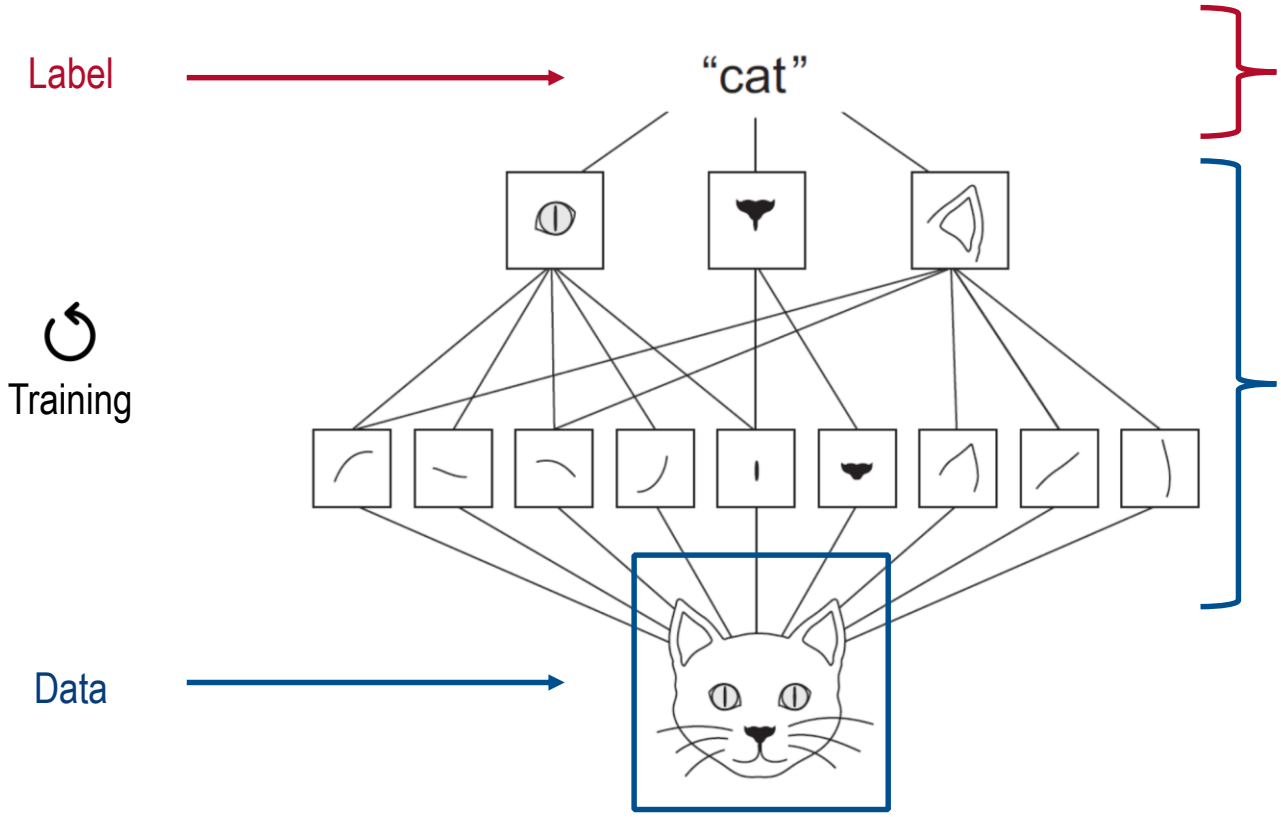
- Good internal representations are hierarchical (with increasing level of abstraction)
  - → Vision: Pixel → edge → shape → … → part → object → scenes
  - → Text: Character → word → word group → … → sentence → story
  - → Speech: Sample → spectral band → sound → … → phoneme → word → …

- The features are learned end-to-end (no hand-crafted features)



EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Spatial hierarchies of patterns

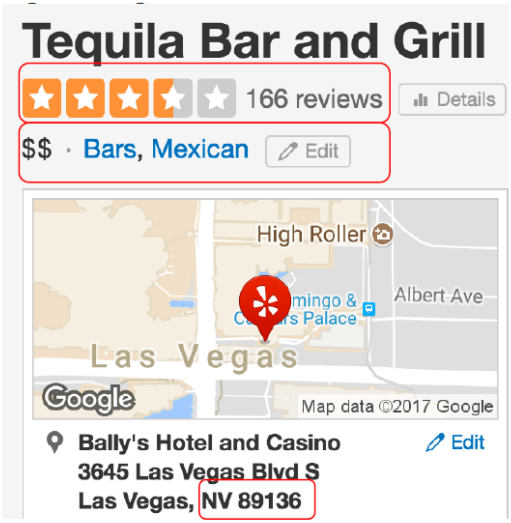


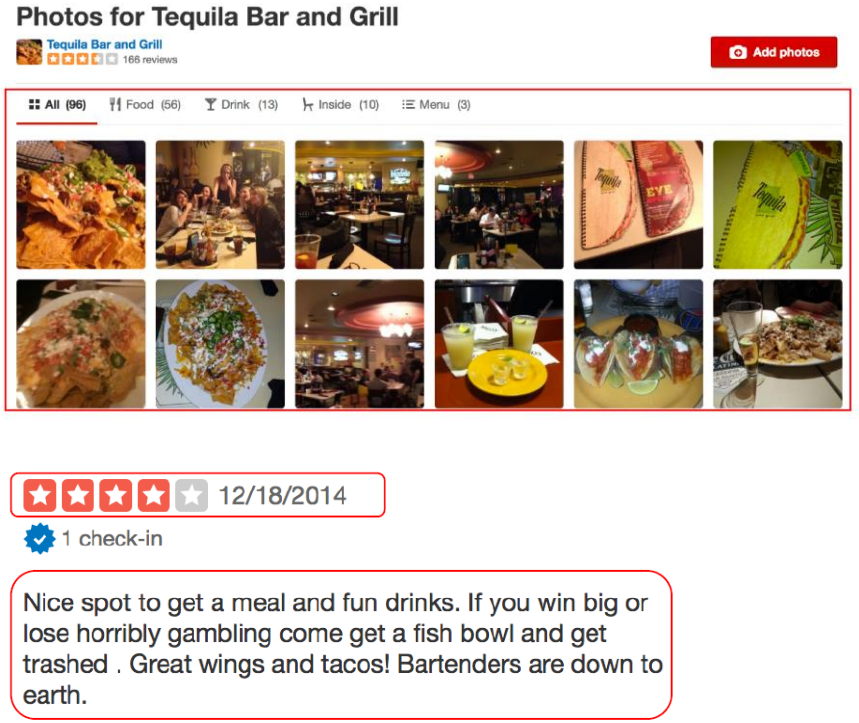
(cf. Chollet & Allaire 2018, p. 115)

some examples from marketing research.

Zhang & Luo (2018), Can User Generated Content Predict Restaurant Survival? Deep Learning of Yelp Photos and Reviews. SSRN.

Prediction of restaurant survival using Yelp reviews and photos





Liu, Dzyabura, & Mizik (2017), Visual Listening In: Extracting Brand Image Portrayed on Social Media. SSRN.

How are brands portrayed on social media?

- Use of DL to measure brand attributes from images (glamorous, rugged, healthy, fun)
- Comparison of firm's official Instagram account with consumers' social media posts and average consumer brand perceptions (national brand survey)

(a) #eddiebauer (b) #prada

Sample images from Instagram hashtagged with brands

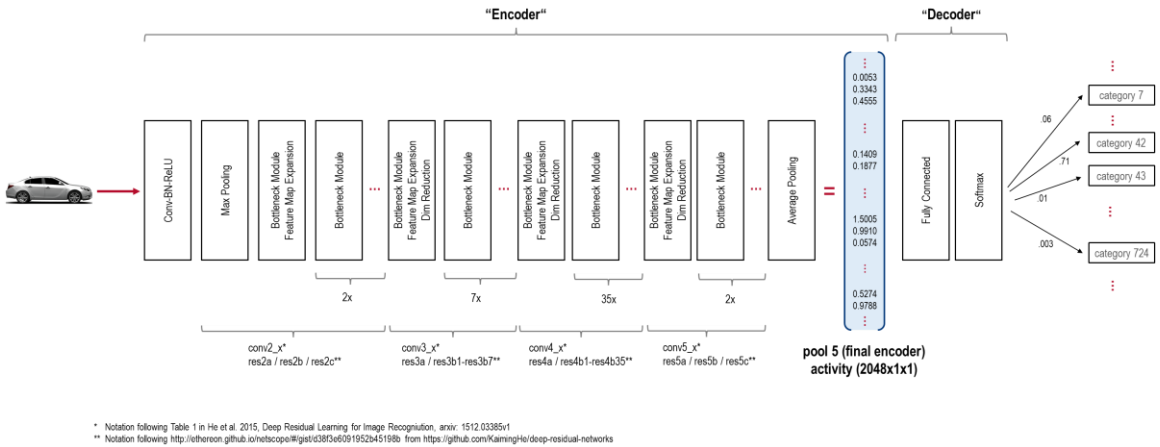
(a) glamorous (b) rugged (c) healthy (d) fun

(e) drab (f) gentle (g) unhealthy (h) dull

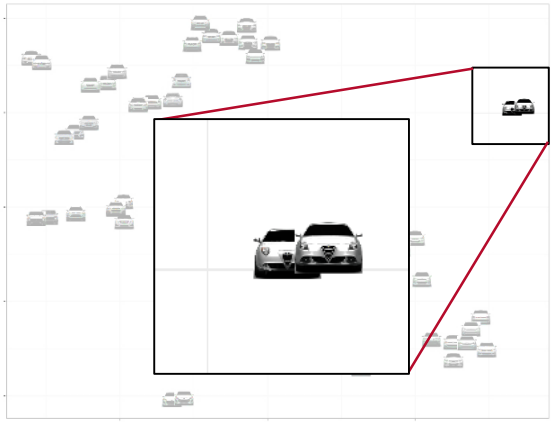
Figure 3 Sample images of different brand attributes and their antonyms in our data sets

Mayer, Landwehr, & Beck (2018), The Power of Deep Neural Networks: How Machine Learning can Advance the Forecasting of Product Success Based on Aesthetic Appearance. Working Paper.

How similar are car designs and how does this relate to sales?

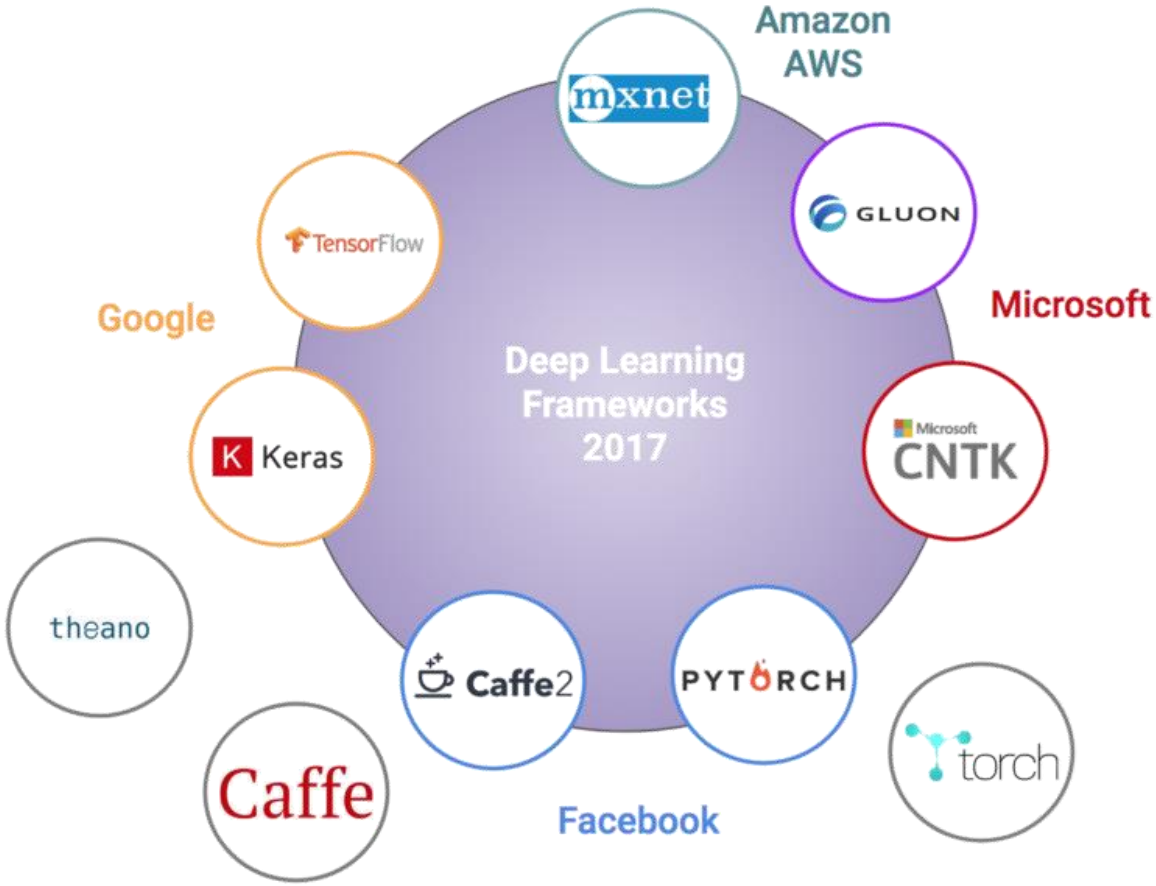


* Notation following Table 1 in He et al. 2015, Deep Residual Learning for Image Recognition, arxiv: 1512.03385v1
** Notation following http://ethereon.github.io/netscope/#/gist/d36f3e6091952b45198b from https://github.com/KaimingHe/deep-residual-networks



2 | Deep Learning in R

Keras supports TensorFlow, CNTK and Theano.



[https://towardsdatascience.com/battle-of-the-deep-learning-frameworks-part-i-cff0e3841750]

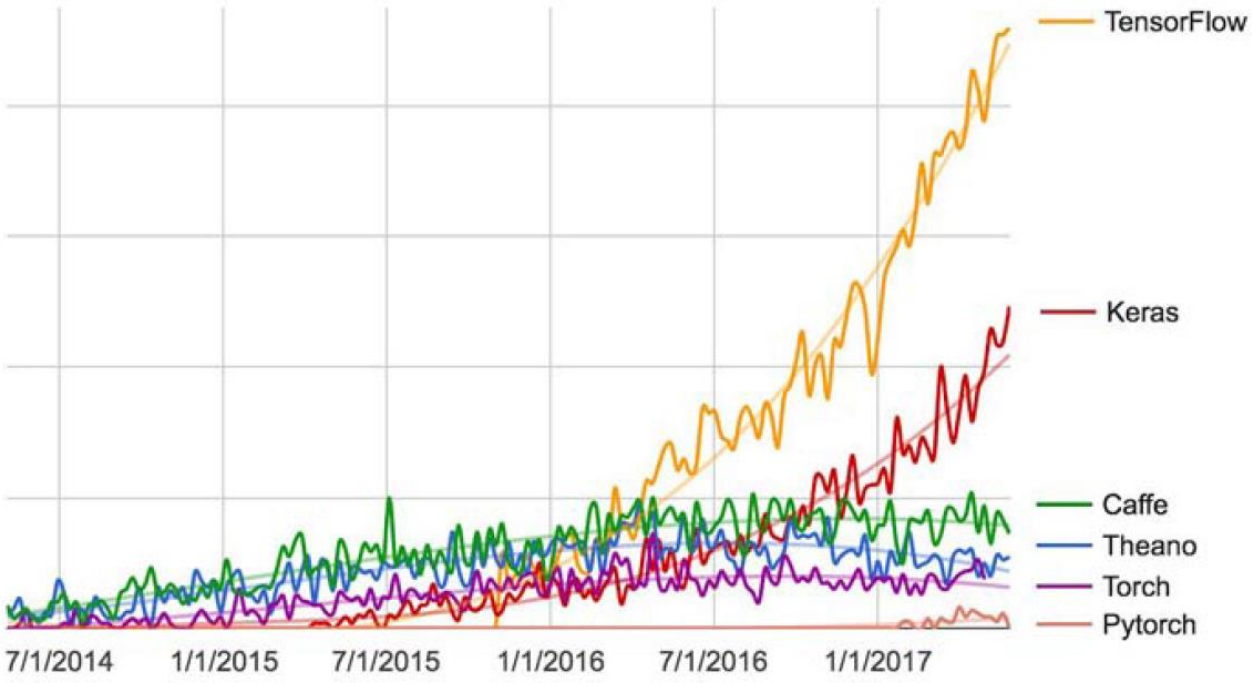


Figure 3.2 Google web search interest for different deep-learning frameworks over time

[Chollet & Allaire (2018). *Deep Learning with R*. Manning.]

Train a model from scratch: Brand logo classification (Pepsi vs. Coke)

- `brand_logos_binary_from_scratch.R`

Transfer learning (change a pre-trained network for your purpose)

- `brand_logos_binary_transfer_learning.R`
- `brand_logos_categorical_transfer_learning.R`
  (Pepsi vs. Coke vs. Heineken)

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Data based on Flickr Logos 27 dataset

(http://image.ntua.gr/iva/datasets/flickr_logos/)

Flickr Logos 27 dataset

The *Flickr Logos 27* dataset is an annotated logo dataset downloaded from Flickr and contains more than four thousand classes in total. It consists of three image collections/sets.

The *training set* contains 810 annotated images, corresponding to 27 logo classes/brands (30 images for each class). All images are annotated with bounding boxes of the logo instances in the image. We allow multiple logo instances per class image. The training set is randomly split in six subsets, each one containing five images per class.

The *distractor set* contains 4207 logo images/classes, that depict, in most cases, clean logos. All images come from the Flickr group Identity + Logo Design. Each one of the distractor set images defines its own logo class and we regard the whole image as bounding box.

Finally, the *query set* consists of 270 images. There are five images for each of the 27 annotated classes, summing up to 135 images that contain logos. Furthermore, the query set contains 135 Flickr images that do not depict any logo class, giving 270 test images in total.

Figure 1. Sample images from the 27 annotated classes.

Figure 2. Sample images from the distractor set.

Figure 3. Sample images from the queries set.

The 27 Annotated Logo Classes/Brands

The brands included in the dataset are: Adidas, Apple, BMW, Citroen, Coca Cola, DHL, Fedex, Ferrari, Ford, Google, Heineken, HP, McDonalds, Mini, Nbc, Nike, Pepsi, Porsche, Puma, Red Bull, Sprite, Starbucks, Intel, Texaco, Unisef, Vodafone and Yahoo.

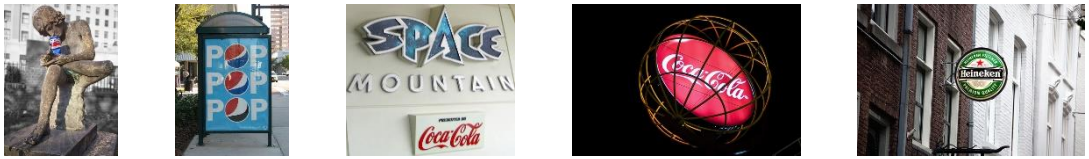
Data preprocessing [`flickr27_data_preprocessing.R`]

- selection of 3 brands: Coke, Pepsi, Heineken

- training image dataset: cropped based on logo bounding box

- sampling of 120 images per brand (training) and 40 images (validation) (70:30 split)

- original holdout test set (5 logo images per brand)



training a model from scratch.

Pepsi vs. Coke (binary image classification task)

4 essential steps for every deep learning task

- decide on batch size (and image size when working with images)
- define model (i.e., architecture of neural network)
- compile + train model
- evaluate the performance

`brand_logos_binary_from_scratch.R`

(1) Decide on batch size and image size

```
# set parameters: image width and height to use for the model
img_width <- 100
img_height <- 100

# batch size: how many samples (i.e., images) should be used in one training iteration
batch_size <- 10
```

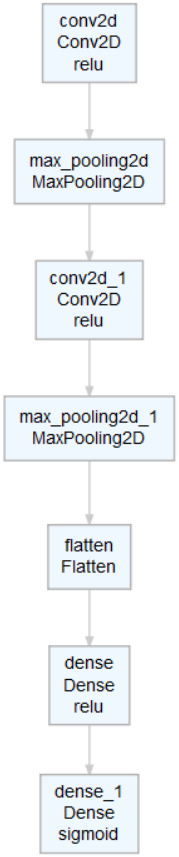
(2) Define model (i.e., architecture of neural network)

```
# define our neural network
model <- keras_model_sequential() %>%
  # first hidden (convolutional) layer
  layer_conv_2d(filters = 32, kernel_size = c(3, 3), activation = "relu",
    input_shape = c(img_width, img_height, 3)) %>%
  # max pooling layer_max_pooling_2d(pool_size = c(2, 2)) %>%
  # second hidden (convolutional) layer
  layer_conv_2d(filters = 64, kernel_size = c(3, 3), activation = "relu") %>%
  # max pooling layer_max_pooling_2d(pool_size = c(2, 2)) %>%
  # flatten max filtered outpot into (dense) feature vector
  layer_flatten() %>% layer_dense(units = 128, activation = "relu") %>%
  # output from dense layer are projected onto output layer
  layer_dense(units = 1, activation = "sigmoid")
```

first hidden layer

second hidden layer

units = 1 and sigmoid activation b/c of binary classification

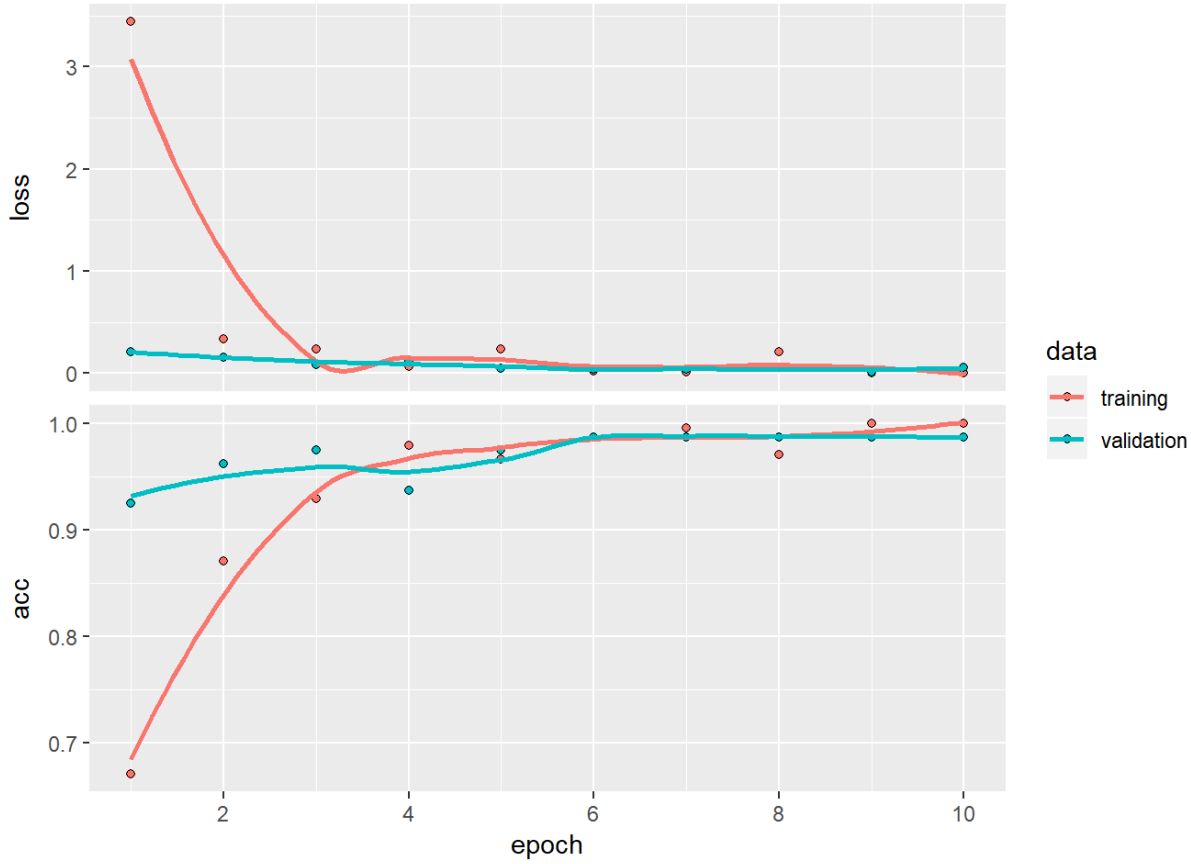


(3) Compile + train the model

```
# compile the model
#
# note that we use "binary_crossentropy" as loss function because of our binary
# classification task
model %>% compile(
  loss = "binary_crossentropy",
  optimizer = optimizer_rmsprop(lr = 1e-3, decay = 1e-6),
  metrics = "accuracy“
)

# train the model (this may take some time ...)
#
# note that the number of epochs defines how many iterations should be done.
hist <- model %>% fit_generator(
  train_generator,
  steps_per_epoch = as.integer(train_samples/batch_size),
  epochs = 10,
  validation_data = validation_generator,
  validation_steps = as.integer(validation_samples/batch_size)
)
```

binary crossentropy b/c of binary classification

(3) Compile + train the model: Visualization of training progress



(4) Evaluate the performance

P(Coca Cola) = 99.97%

P(Coca Cola) = 0%

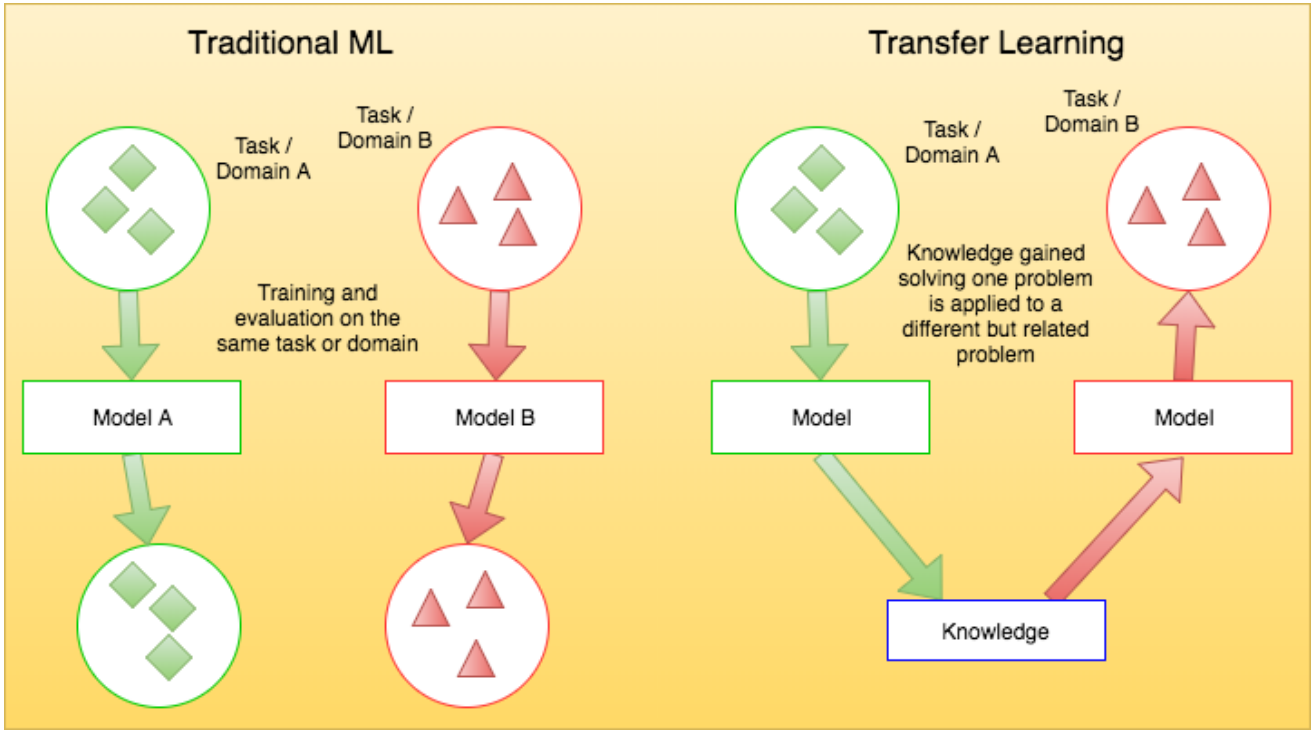
```
# test on 10 images from the test set
test_generator <- flow_images_from_directory(
  test_directory, generator = datagen_val,
  target_size = c(img_width, img_height),
  class_mode = "binary",
  batch_size = batch_size,
  classes = class_list, seed = myseed)

model %>% evaluate_generator(test_generator, steps = 5)
```

→ **Accuracy: 50%**

transfer learning.

Transfer Learning

- Resulting features from large-scale image recognition tasks generalize well to other datasets (Donahue et al 2014, Zeiler & Fergus 2013)



Transfer Learning

- Resulting features from large-scale image recognition tasks generalize well to other datasets (Donahue et al 2014, Zeiler & Fergus 2013)
- Fine tune pre-trained network on small(er) data set
  - → replace decoder-part (i.e., classifier) of network with your own (specific for your task)
  - → fine tune all layers (i.e., continue updating the weights via back propagation)
  - → fix some layers and fine tune only selected layers (usually higher layers)
- Use pre-rained CNN as fixed feature extractor for data and train a linear classifier like SVM using those features.
  - → Ideal if data set is very small (and, therefore, fine-tuning may result in overfitting)

Pepsi vs. Coke (binary image classification task) using transfer learning

- decide on the pre-trained net
  - → which architecture (e.g. VGG16, VGG19, Xception, ResNet50)
  - → which weights (e.g., imagenet)
- replace classifier with new one (step 2), everything else is the same

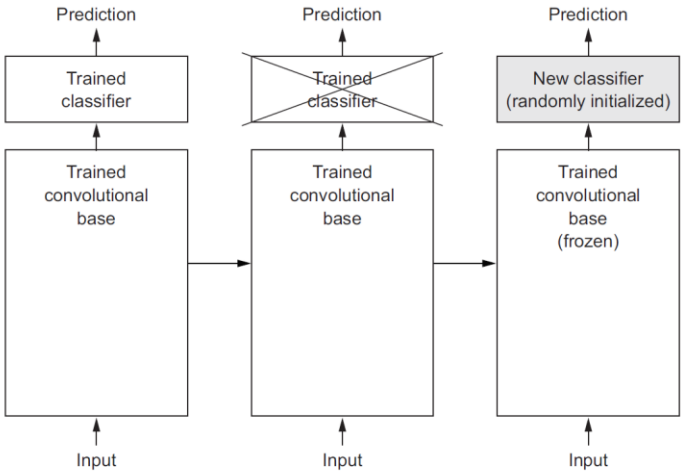


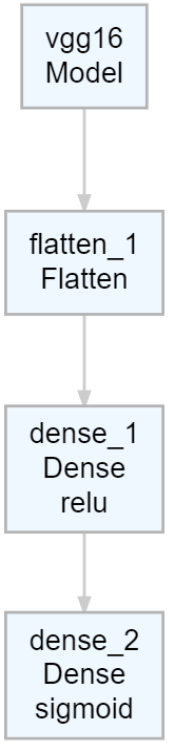
Figure 5.12 Swapping classifiers while keeping the same convolutional base

`brand_logos_binary_transfer_learning.R`

[Chollet & Allaire (2018). *Deep Learning with R*. Manning.]

(2) changed model architecture

```
# we use VGG16 as convolutional base with the imagenet weights
conv_base <- application_vgg16(
  weights = "imagenet",
  include_top = FALSE,
  input_shape = c(img_width, img_height, 3)
)

# add your custom layers
model <- keras_model_sequential() %>%
  conv_base %>%
  layer_flatten() %>%
  layer_dense(units = 128, activation = "relu") %>%
  layer_dense(units = 1, activation = "sigmoid")

# freeze weights of convolutional base
freeze_weights(conv_base)
```



Transfer learning: Performance

P(Coca Cola) = 66.64%

P(Coca Cola) = 0.001%

→ **however, test set accuracy: 80%**

more than two classes: multiclass image classification

Pepsi vs. Coke vs. Heineken (multiclass image classification task) using transfer learning

- adapt network architecture to predict 3 class probabilities
- change loss function to reflect multiclass classification
- everything else stays the same as in the previous example

`brand_logos_categorical_transfer_learning.R`

[Chollet & Allaire (2018). *Deep Learning with R*. Manning.]

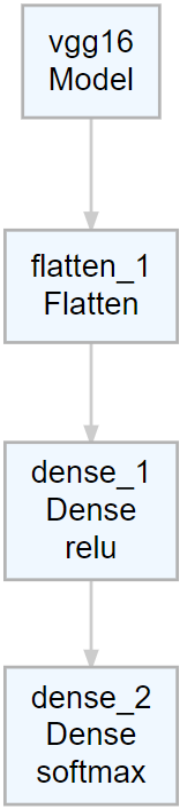
EBERHARD KARLS UNIVERSITÄT TÜBINGEN

(2) changed model definition, (3) changed loss function

```
# add your custom layers
model <- keras_model_sequential() %>%
  conv_base %>%
  layer_flatten() %>%
  layer_dense(units = 256, activation = "relu") %>%
  layer_dense(units = 3, activation = "softmax")

# compile the model # # note that we use “categorical_crossentropy" as loss
# function because of our multiclass classification task
model %>% compile(
  loss = "categorical_crossentropy",
  optimizer = optimizer_rmsprop(lr = 1e-5, decay = 1e-6),
  metrics = "accuracy“
)
```

multiclass classification: 3 units, softmax activation

multiclass classification: categorical crossentropy



Transfer learning (multiclass image classification): Performance

P(Coca Cola) = 46.75%

P(Pepsi) = 99.41%

P(Heineken) = 92.55%

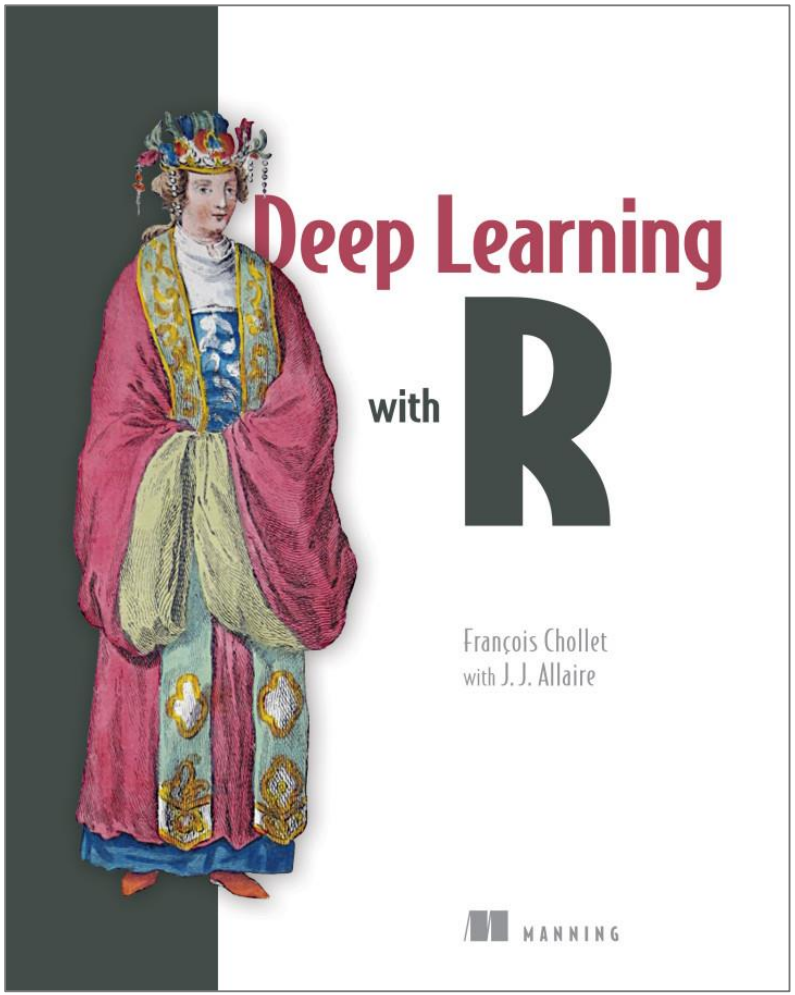
→ **test set accuracy: 80%**

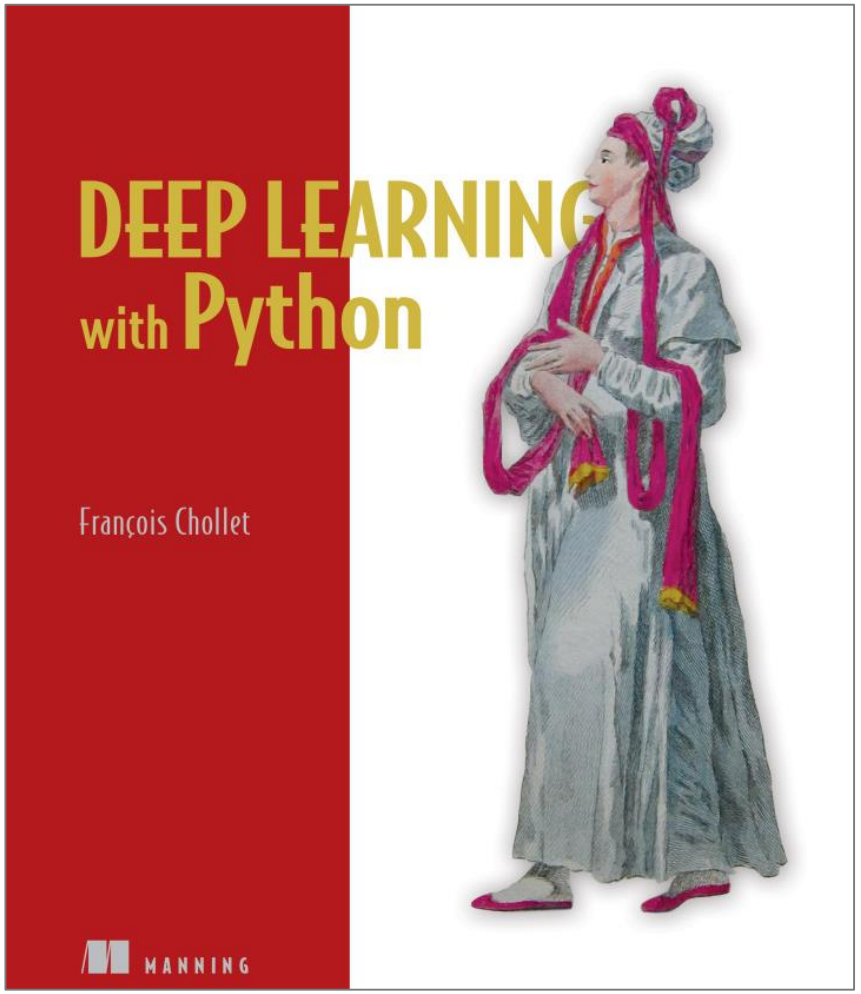
3 | Where to start.

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Keras & Tensorflow Tutorials

- https://tensorflow.rstudio.com/keras/#tutorials
  - → Basic image classification
  - → Text classification
  - → Basic regression (using Deep Learning)
  - → Overfitting and Underfitting
- https://blogs.rstudio.com/tensorflow/gallery.html
  - → Image classification on small datasets
  - → Deep learning for text classification
  - → Classifying physical activity from smartphone data


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


DEEP LEARNING
with Python
François Chollet
MANNING

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Thank you.

slides available at
github.com/stm/emac_2019_sig_quant

Stefan Mayer

School of Business and Economics
University of Tübingen

stefan.mayer@uni-tuebingen.de