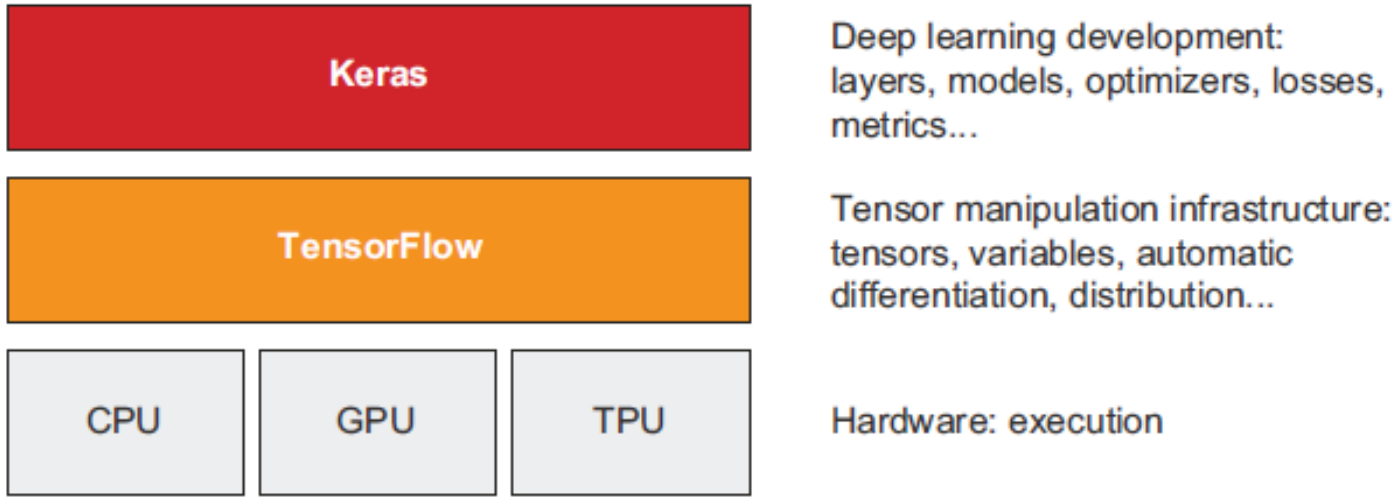
Machine Learning Live
Session #3

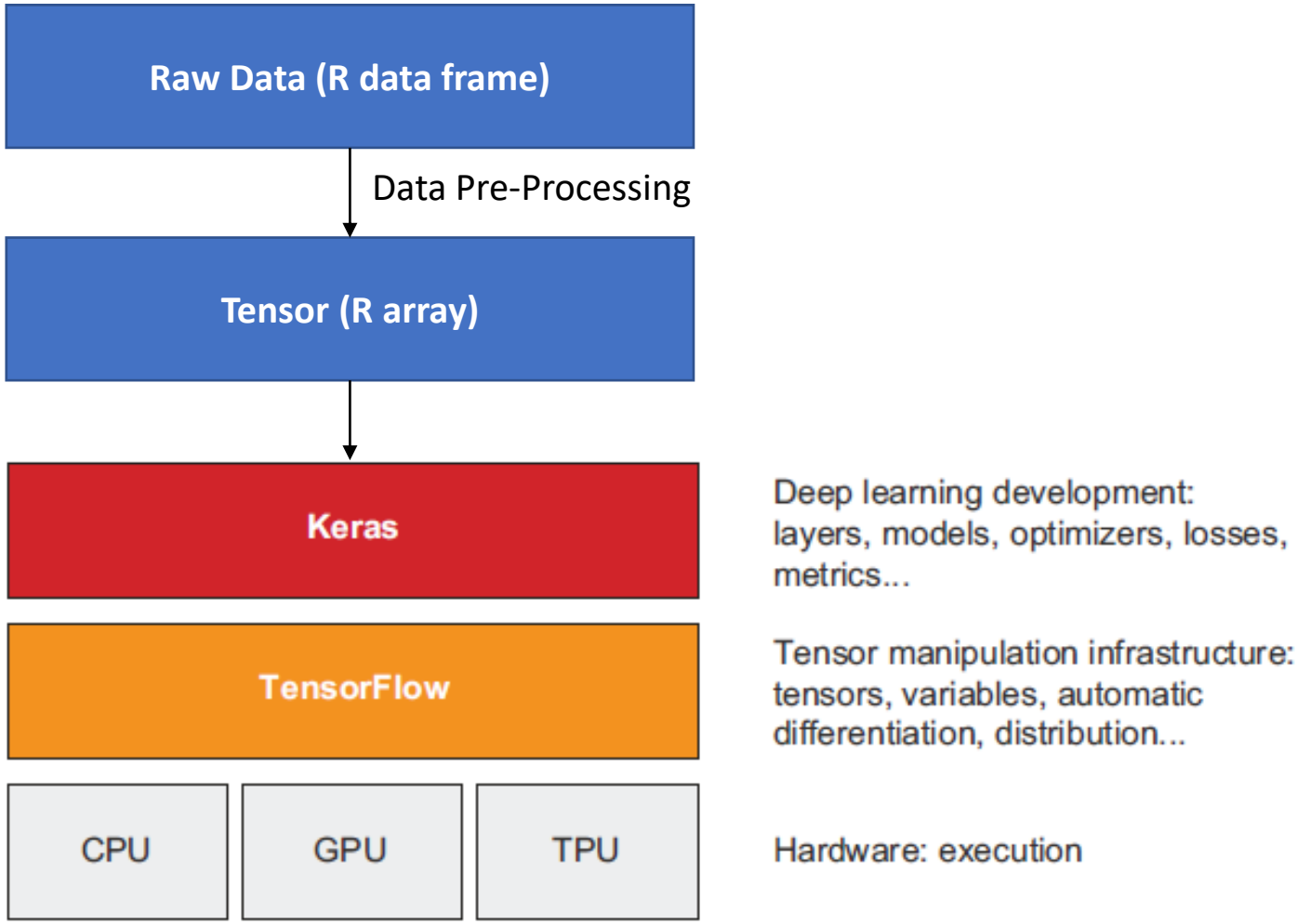
Keras and TensorFlow

- Keras is the high-level API for building neural network models
  - In this course, we focus on densely connected feed-forward neural networks (i.e., we mainly use Keras layers called layer_dense)
- TensorFlow provides the infrastructure for tensor operations, optimization, and interfacing with computer hardware
- At the lowest level is the CPU (central processing unit), GPU (graphics processing unit), and TPU (tensor processing unit) that execute the tasks



Keras and TensorFlow

- We start with a raw dataset that needs to be pre-processed before it can be used with functions in Keras.



Data Pre-Processing

- Categorical variables need to be encoded as numerical: use one-hot encoding
  - Break out each level into its own binary indicator (0/1) feature

Raw dataset →

customer_age (numerical)	customer_state (categorical)
25	Massachusetts
32	New Hampshire
73	Colorado
56	New Hampshire
43	Massachusetts
21	Massachusetts
67	Colorado

Transformed dataset →

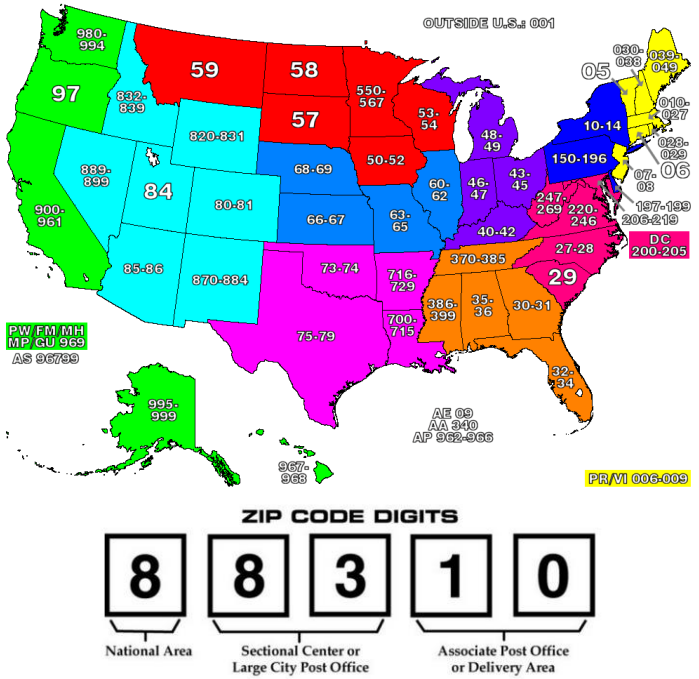
customer_age (numerical)	customer_state_ma	customer_state_nh	customer_state_co
25	1	0	0
32	0	1	0
73	0	0	1
56	0	1	0
43	1	0	0
21	1	0	0
67	0	0	1

- Causes high-dimensionality when there are many levels
- Be careful! Some machine learning models require full-rank, i.e., features that are not redundant
  - If there are l levels, only $l - 1$ binary indicator features are needed
  - In above example, can remove co_ind, since we know the level is “Colorado” when all other indicators are zero

Data Pre-Processing

Categorical variables with many levels

- Combine levels before one-hot encoding
  - Requires subject matter expertise and domain knowledge
  - E.g., zip codes, ICD9 or ICD10 codes (medical diagnoses)



Raw dataset →

customer_age (numerical)	customer_state (categorical)	customer_zip (categorical)
25	Massachusetts	01843
32	New Hampshire	03580
73	Colorado	80150
56	New Hampshire	03103
43	Massachusetts	02783
21	Massachusetts	01854
67	Colorado	80534

Transformed dataset →

customer_age (numerical)	customer_state (categorical)	customer_reduced_zip (categorical)
25	Massachusetts	018
32	New Hampshire	035
73	Colorado	801
56	New Hampshire	031
43	Massachusetts	027
21	Massachusetts	018
67	Colorado	805

Zip Codes 101. https://smartystreets.com/docs/zip-codes-101

Data Pre-Processing

- Scale variables to have a mean of 0 and standard deviation of 1.
- Important for:
  - Algorithms that use distance-based metrics (e.g., k-means clustering or k-nearest neighbor)
  - Algorithms that use gradient-descent optimization (e.g., neural networks)
  - Consistency for comparing models
  - Regularization

customer_age	customer_state_ma	customer_state_nh	customer_state_co
25	1	0	0
32	0	1	0
73	0	0	1
56	0	1	0
43	1	0	0
21	1	0	0
67	0	0	1

Before Scaling

customer_age	customer_state_ma	customer_state_nh	customer_state_co
-0.986	1.07	-0.59	-0.59
-0.646	-0.8	1.46	-0.59
1.348	-0.8	-0.59	1.46
0.521	-0.8	1.46	-0.59
-0.111	1.07	-0.59	-0.59
-1.181	1.07	-0.59	-0.59
1.056	-0.8	-0.59	1.46

After Scaling

Tensors

2.2.8 Real-world examples of data tensors

Let's make data tensors more concrete with a few examples similar to what you'll encounter later. The data you'll manipulate will almost always fall into one of the following categories:

- *Vector data*—Rank 2 tensors of shape `(samples, features)`, where each sample is a vector of numerical attributes ("features")
- *Times-series data or sequence data*—Rank 3 tensors of shape `(samples, timesteps, features)`, where each sample is a sequence (of length `timesteps`) of feature vectors
- *Images*—Rank 4 tensors of shape `(samples, height, width, channels)`, where each sample is a 2D grid of pixels, and each pixel is represented by a vector of values ("channels")
- *Video*—Rank 5 tensors of shape `(samples, frames, height, width, channels)`, where each sample is a sequence (of length `frames`) of images

customer_age	customer_state_ma	customer_state_nh	customer_state_co
-0.986	1.07	-0.59	-0.59
-0.646	-0.8	1.46	-0.59
1.348	-0.8	-0.59	1.46
0.521	-0.8	1.46	-0.59
-0.111	1.07	-0.59	-0.59
-1.181	1.07	-0.59	-0.59
1.056	-0.8	-0.59	1.46