Lecture 3A
Lightning Introduction to Keras/TF

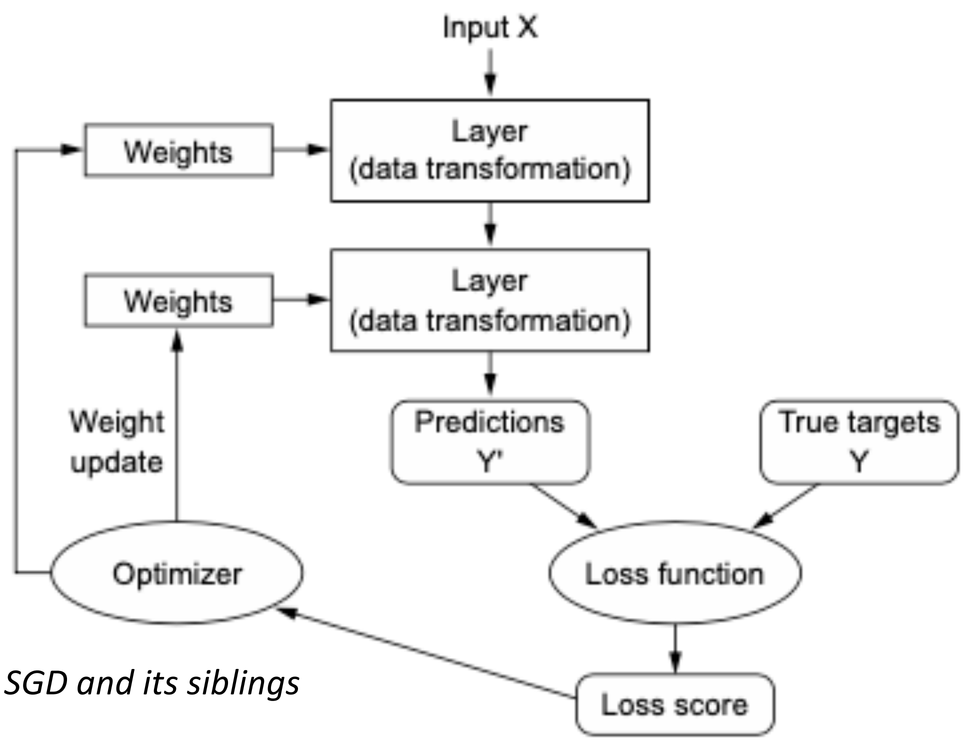
Training a DL Model for a Structured Data Problem

MIT MANAGEMENT SLOAN SCHOOL

15.S04: Hands-on Deep Learning
Spring 2024

Farias, Ramakrishnan

(Recap) Summary of overall training flow



SGD and its siblings

Figure 2.26 Relationship between the network, layers, loss function, and optimizer

Image: Page 61 of textbook (Deep Learning with Python, Francois Chollet, 2nd Edition)

(Recap) Gradient Descent vs Stochastic Gradient Descent

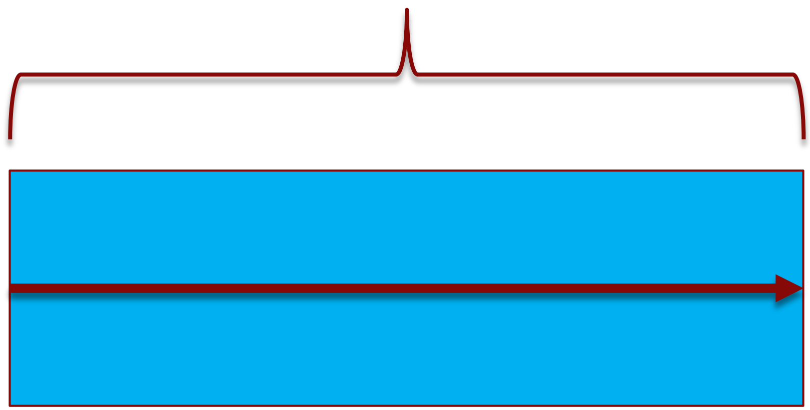
- At each iteration, use all data points to calculate the gradient of the loss function

- At each iteration, randomly choose just a few of the data points and use only these to compute the gradient of the loss function

Epochs and Batches

What is an epoch?

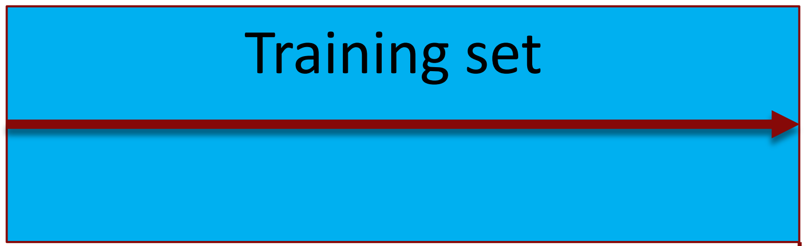
Training set



An epoch is one pass through the full training set.

But this plays out differently for Gradient Descent vs *Stochastic* Gradient Descent.

An epoch in Gradient Descent

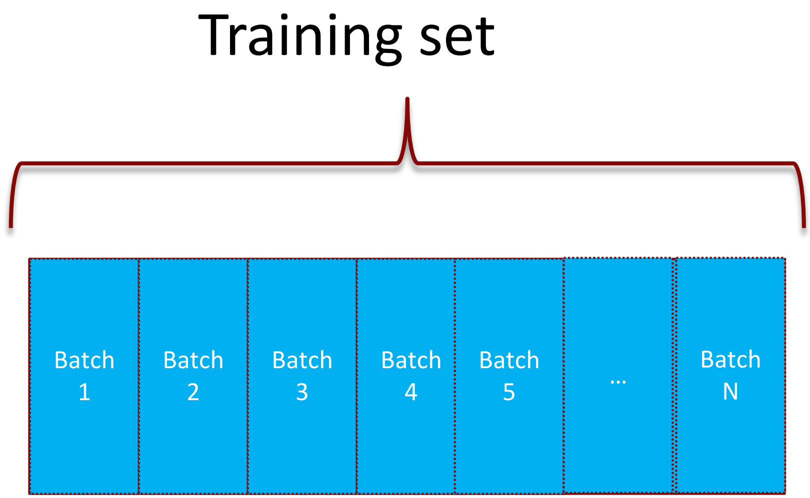


- We run every training sample through the network to get the predictions
- We calculate the gradient of the loss
- We update the parameters

$$w \leftarrow w - \alpha \frac{dLoss(w)}{dw}$$

This is done just once at the end of the epoch

An epoch in Stochastic Gradient Descent

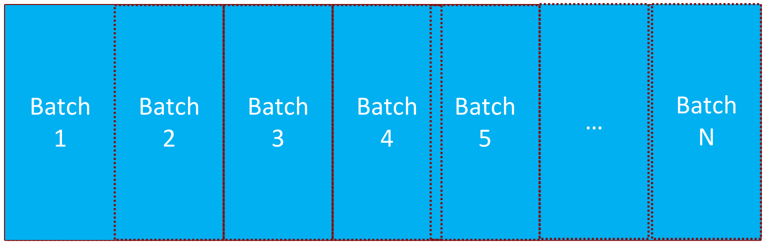


But when we do Stochastic Gradient Descent (SGD), we process the data in minibatches, one after the other*

*we will refer to minibatches as batches from now on for simplicity

An epoch in Stochastic Gradient Descent

Training set

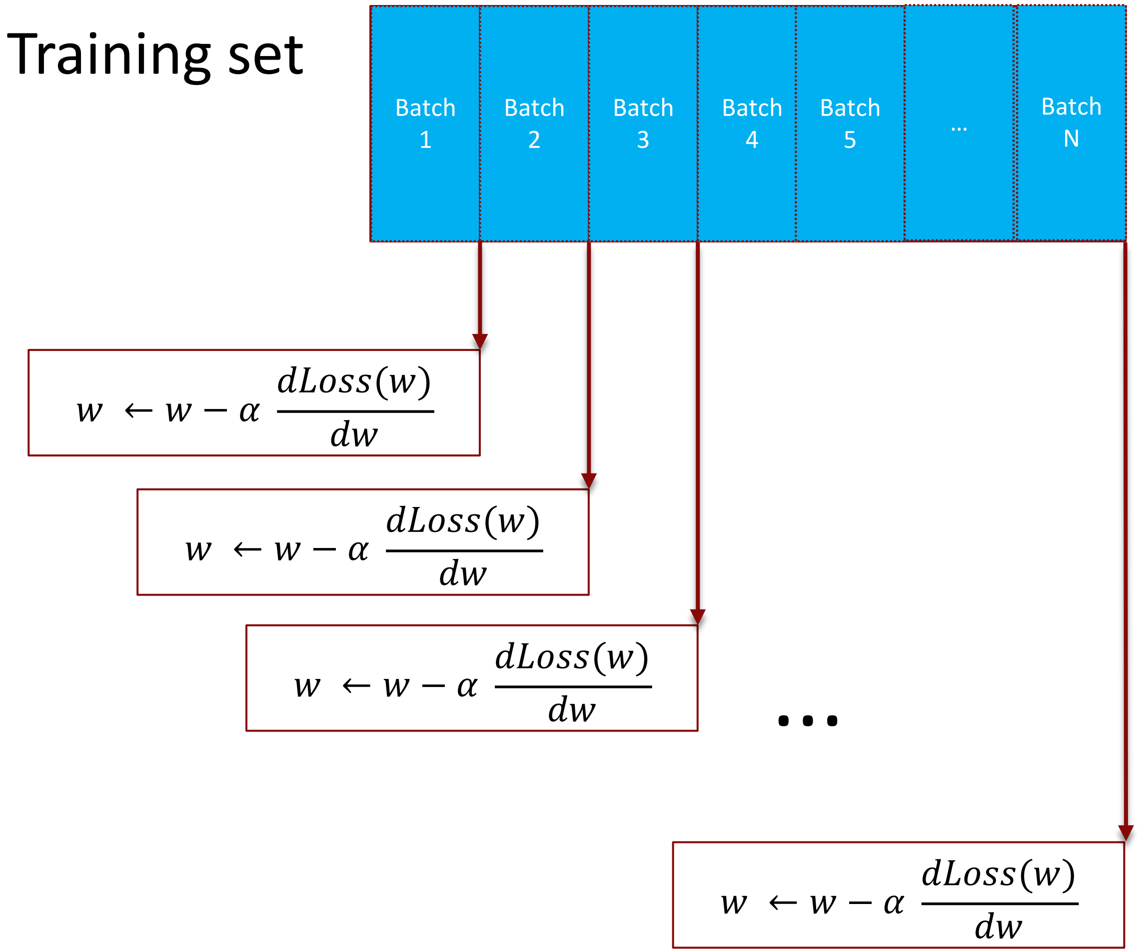


For each batch:

- *We run the training samples in that batch through the network to get predictions*
- *We calculate the gradient of the loss*
- *We update the parameters*

$$\boxed{w \leftarrow w - \alpha \frac{dLoss(w)}{dw}}$$

An epoch in Stochastic Gradient Descent

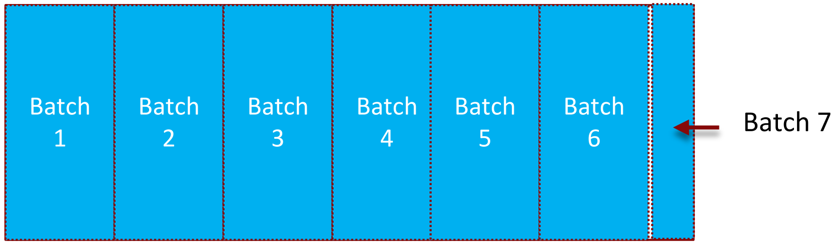


How many batches in an epoch when we do SGD?

of batches in one epoch = (Training set size / Batch size) rounded up

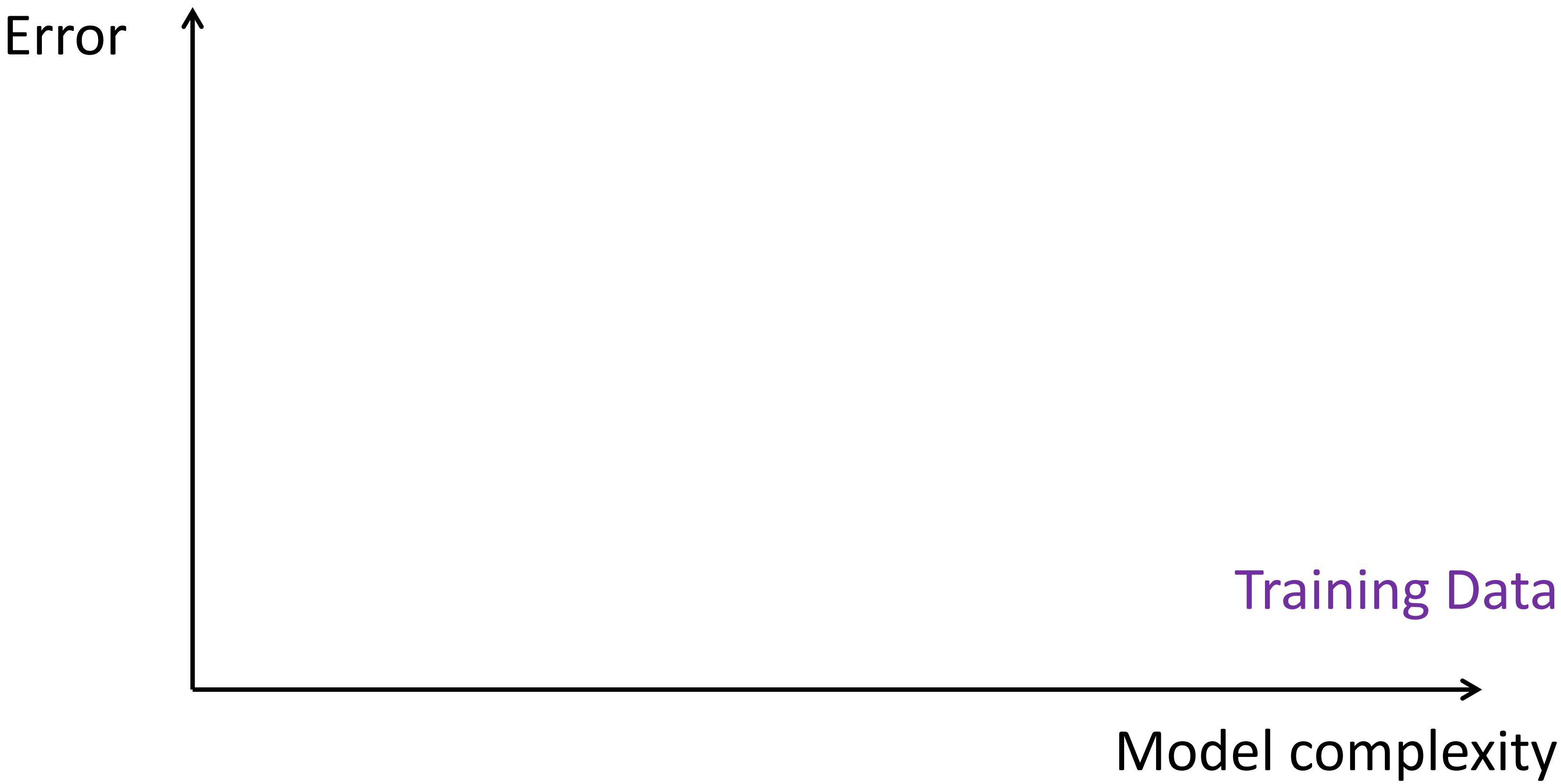
<u>For Neural Heart Disease Model</u>:
Training set size = 194
Batch size = 32
# of batches in one epoch = (194/32) rounded up = 7

The first 6 batches have 32 samples each, and the 7th batch has the last 2 samples.
32 * 6 + 2 = 194

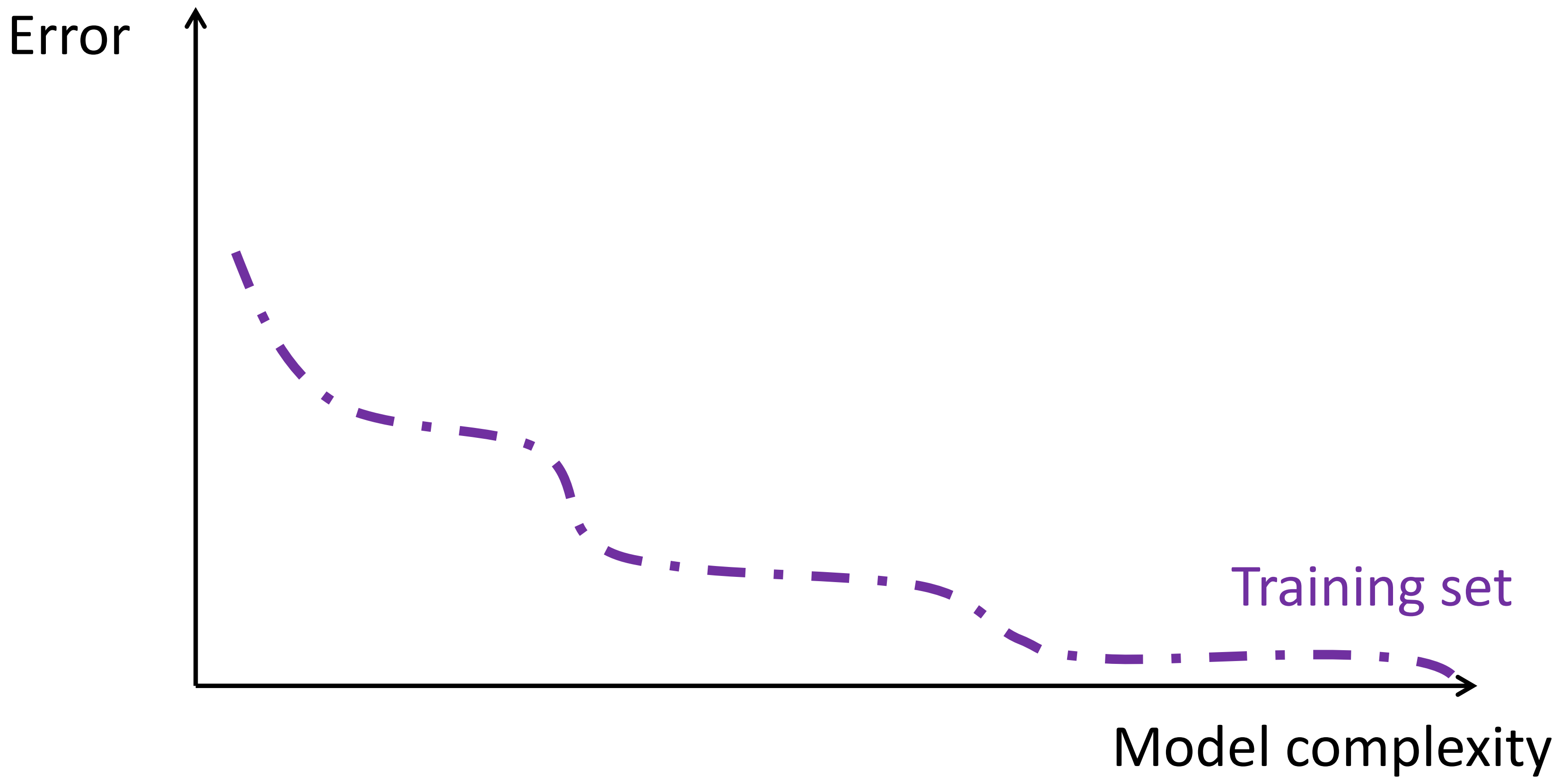


Overfitting and Regularization

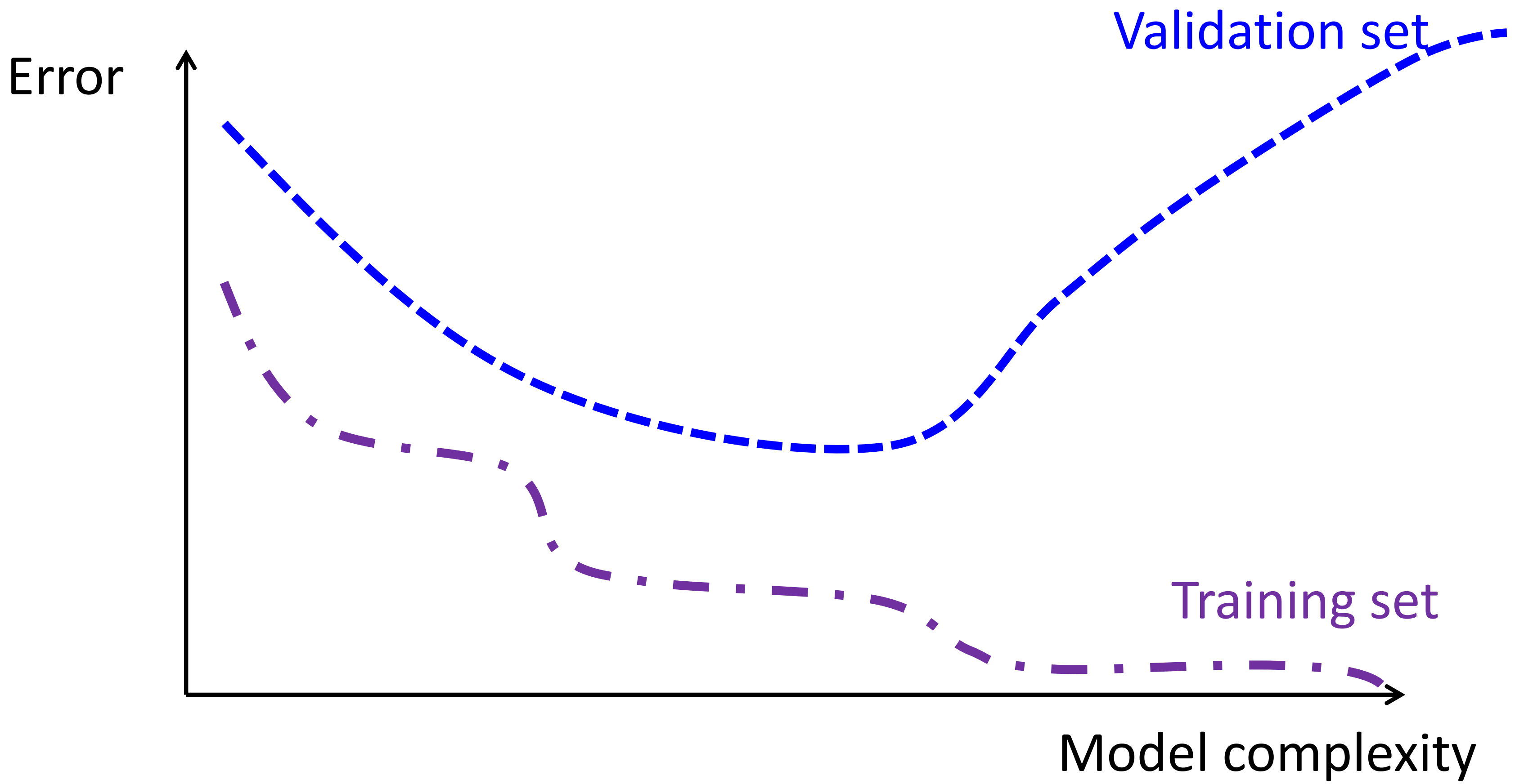
Recall Underfitting vs. Overfitting



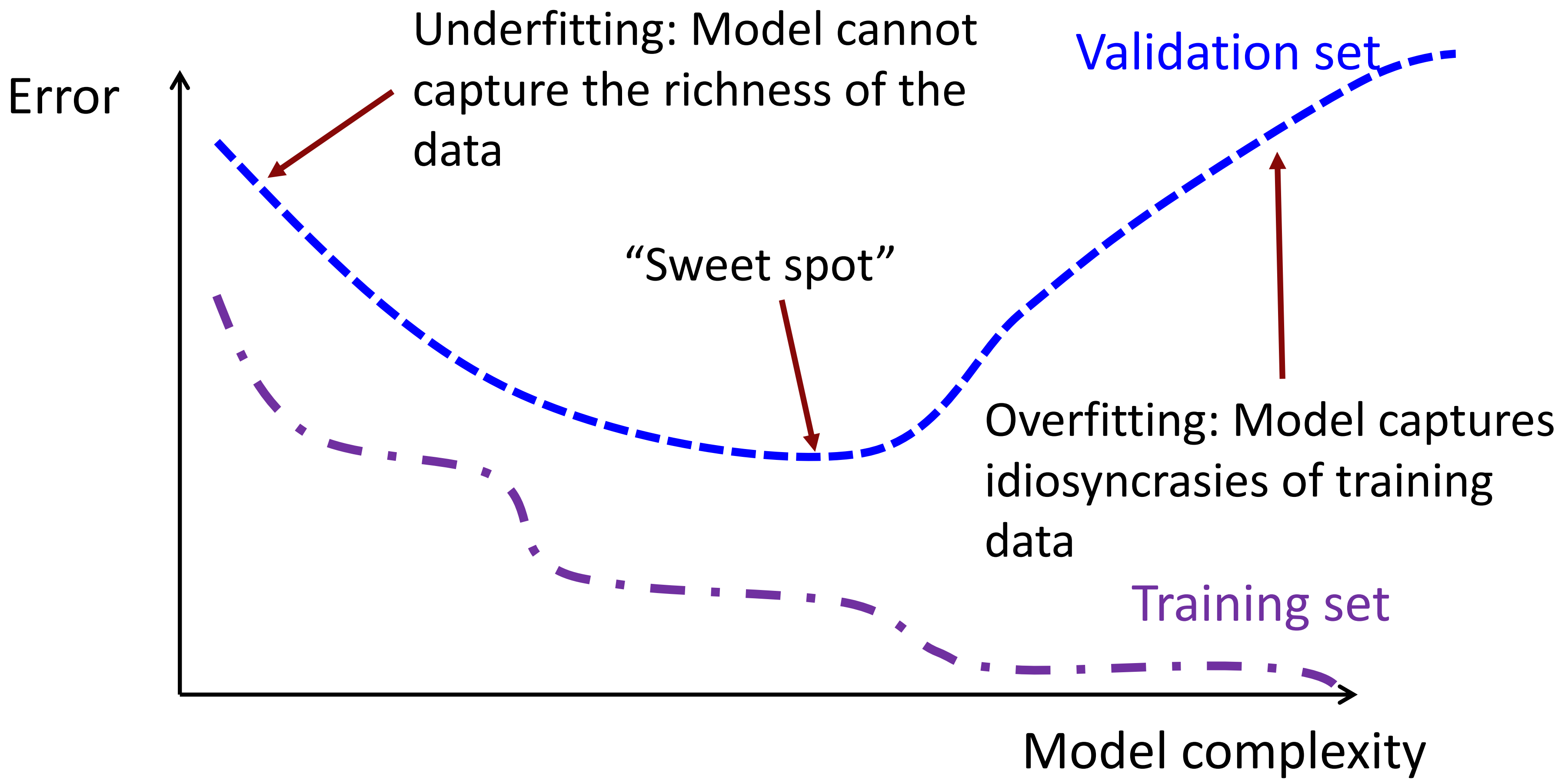
Recall Underfitting vs. Overfitting



Recall Underfitting vs. Overfitting



Recall Underfitting vs. Overfitting



Overfitting in Neural Networks

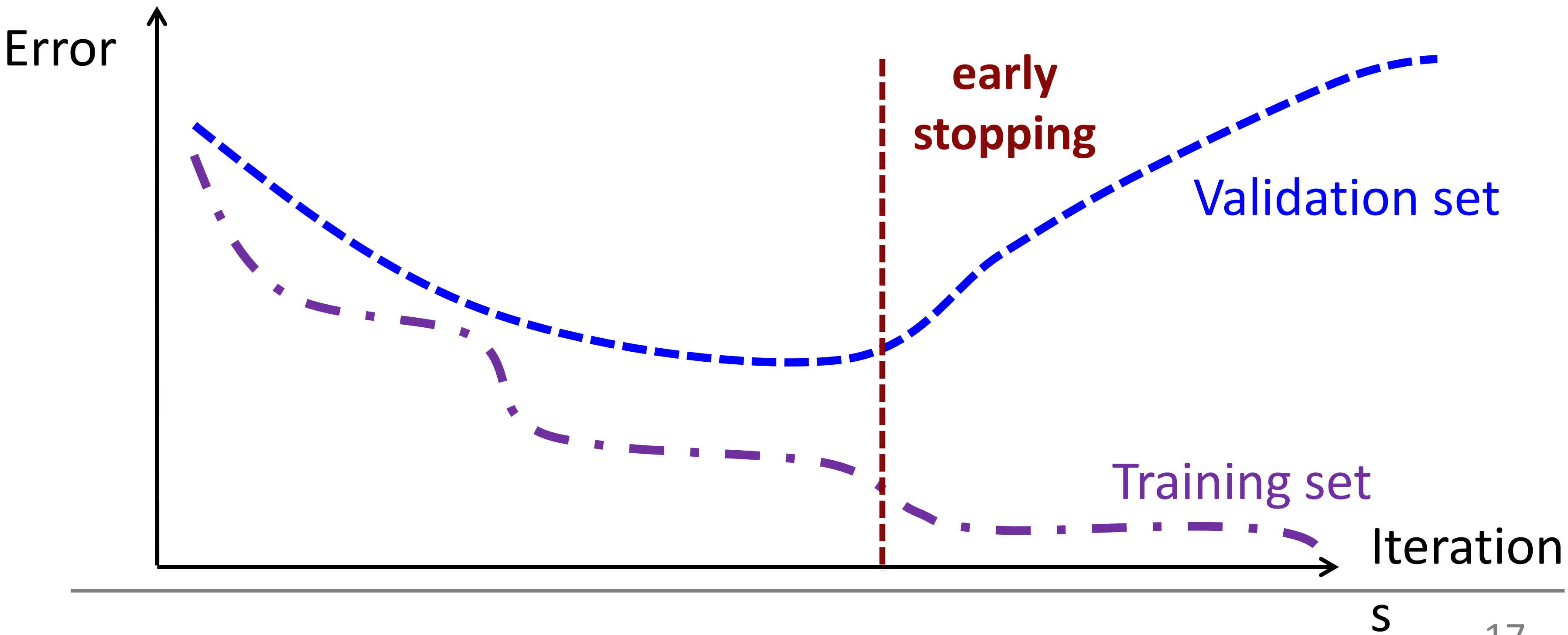
- To learn smart representations of complex, unstructured data, the NN needs to have large “capacity” i.e., many layers and many neurons in each layer

- But this raises the likelihood of overfitting so we need to add *regularization*

- Several regularization methods have been developed to address this problem

Regularization strategy: *Early Stopping*

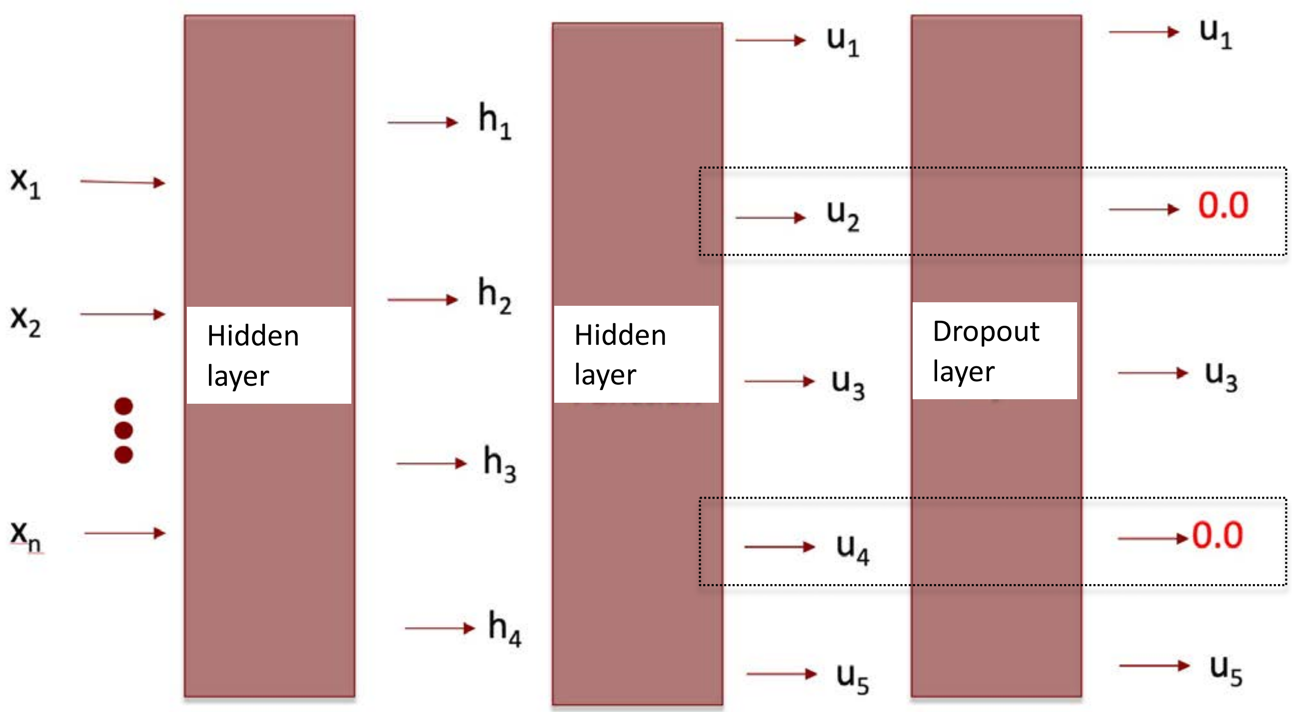
Stop the training early before the training loss is minimized by monitoring the loss on a validation dataset.



We will cover this in Lecture 4

Regularization strategy: *Dropout*

Randomly zero out the output from some of the nodes (typically 50% of the nodes) in a hidden layer (implemented as a “dropout layer” in Keras)



Summary: Creating and training a DNN from scratch

- We get the data ready

- We design i.e., “lay out” the network
  - Choose the number of hidden layers and the number of ‘neurons’ in each layer
  - Pick the right output layer based on the type of the output (more on this shortly)

- We pick
  - An appropriate loss function based on the type of the output (more on this shortly)
  - An optimizer from the many SGD flavors that are available and a “good” learning rate

- We decide on a regularization strategy

- We set things up in Keras/Tensorflow and start training!

Lightning Intro to Tensorflow/Keras

What's a Tensor?

What’s a Tensor?

Tensor of rank 0 (Scalar)

42

What's a Tensor?

Tensor of rank 0 (Scalar)

42

Tensor of rank 1 (aka Vector)

(42, 23.4, 11.2)

What’s a Tensor?

Tensor of rank 0 (Scalar)

42

Tensor of rank 1 (aka Vector)

(42, 23.4, 11.2)

Tensor of rank 2 (aka Matrix)

	[,1]	[,2]	[,3]	[,4]	[,5]	[,6]	[,7]	[,8]	[,9]	[,10]	[,11]	[,12]	[,13]	[,14]	[,15]	[,16]	[,17]	[,18]	[,19]	[,20]	[,21]	[,22]	[,23]	[,24]	[,25]	[,26]	[,27]	[,28]
[1,]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
[2,]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
[3,]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
[4,]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
[5,]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
[6,]	0	0	0	0	0	0	0	0	0	0	0	0	3	18	18	18	126	136	175	26	166	255	247	127	0	0	0	0
[7,]	0	0	0	0	0	0	0	0	30	36	94	154	170	253	253	253	253	253	225	172	253	242	195	64	0	0	0	0
[8,]	0	0	0	0	0	0	0	49	238	253	253	253	253	253	253	253	253	251	93	82	82	56	39	0	0	0	0	0
[9,]	0	0	0	0	0	0	0	18	219	253	253	253	253	253	198	182	247	241	0	0	0	0	0	0	0	0	0	0
[10,]	0	0	0	0	0	0	0	0	80	156	107	253	253	205	11	0	43	154	0	0	0	0	0	0	0	0	0	0
[11,]	0	0	0	0	0	0	0	0	0	14	1	154	253	90	0	0	0	0	0	0	0	0	0	0	0	0	0	0
[12,]	0	0	0	0	0	0	0	0	0	0	0	139	253	190	2	0	0	0	0	0	0	0	0	0	0	0	0	0
[13,]	0	0	0	0	0	0	0	0	0	0	0	11	190	253	70	0	0	0	0	0	0	0	0	0	0	0	0	0
[14,]	0	0	0	0	0	0	0	0	0	0	0	0	35	241	225	160	108	1	0	0	0	0	0	0	0	0	0	0
[15,]	0	0	0	0	0	0	0	0	0	0	0	0	0	81	240	253	253	119	25	0	0	0	0	0	0	0	0	0
[16,]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	45	186	253	253	150	27	0	0	0	0	0	0	0	0
[17,]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	16	93	252	253	187	0	0	0	0	0	0	0	0
[18,]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	249	253	249	64	0	0	0	0	0	0	0
[19,]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	46	130	183	253	253	207	2	0	0	0	0	0	0	0
[20,]	0	0	0	0	0	0	0	0	0	0	0	0	39	148	229	253	253	253	250	182	0	0	0	0	0	0	0	0
[21,]	0	0	0	0	0	0	0	0	0	0	24	114	221	253	253	253	253	201	78	0	0	0	0	0	0	0	0	0
[22,]	0	0	0	0	0	0	0	0	23	66	213	253	253	253	253	198	81	2	0	0	0	0	0	0	0	0	0	0
[23,]	0	0	0	0	0	0	18	171	219	253	253	253	253	195	80	9	0	0	0	0	0	0	0	0	0	0	0	0
[24,]	0	0	0	0	55	172	226	253	253	253	253	244	133	11	0	0	0	0	0	0	0	0	0	0	0	0	0	0
[25,]	0	0	0	0	136	253	253	253	212	135	132	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
[26,]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
[27,]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
[28,]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Image credit: fast.ai

What’s a Tensor?

Tensor of rank 0 (Scalar)

42

Tensor of rank 1 (aka Vector)

(42, 23.4, 11.2)

Tensor of rank 2 (aka Matrix)

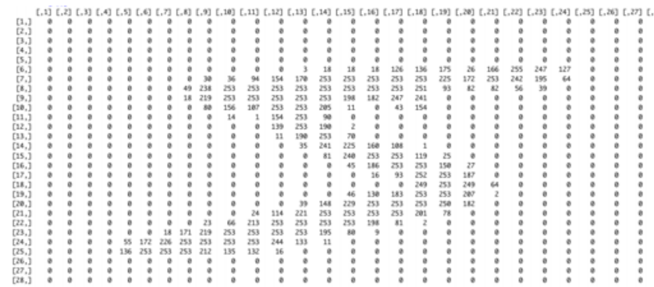
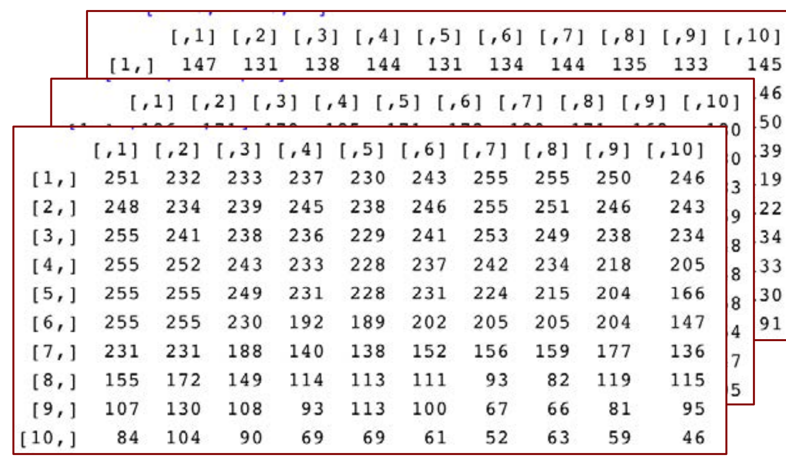


Image credit: fast.ai

Tensor of rank 3 (aka “cube”)



	[,1]	[,2]	[,3]	[,4]	[,5]	[,6]	[,7]	[,8]	[,9]	[,10]
[1,]	251	232	233	237	230	243	255	255	250	246
[2,]	248	234	239	245	238	246	255	251	246	243
[3,]	255	241	238	236	229	241	253	249	238	234
[4,]	255	252	243	233	228	237	242	234	218	205
[5,]	255	255	249	231	228	231	224	215	204	166
[6,]	255	255	230	192	189	202	205	205	204	147
[7,]	231	231	188	140	138	152	156	159	177	136
[8,]	155	172	149	114	113	111	93	82	119	115
[9,]	107	130	108	93	113	100	67	66	81	95
[10,]	84	104	90	69	69	61	52	63	59	46

Can you give an example of a rank-4 tensor?

What’s a Tensor?

See Chapter 2.2 of text for more detail

Tensorflow

Tensorflow (TF) is a library that provides

- <u>Automatic</u> calculation of the gradient of (complicated) loss functions

$$\nabla Loss(w) = \left[\frac{\partial Loss}{\partial w_1}, \frac{\partial Loss}{\partial w_2}, \ldots, \frac{\partial Loss}{\partial wn}\right]$$

Tensorflow

Tensorflow (TF) is a library that provides

- Automatic calculation of the gradient of (complicated) loss functions
- Library of state-of-the-art optimizers

Tensorflow

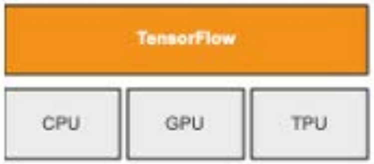
Tensorflow (TF) is a library that provides

- Automatic calculation of the gradient of (complicated) loss functions
- Library of state-of-the-art optimizers
- Automatic distribution of computational load across servers

Tensorflow

Tensorflow (TF) is a library that provides

- Automatic calculation of the gradient of (complicated) loss functions
- Library of state-of-the-art optimizers
- Automatic distribution of computational load across servers
- Automatic adaptation of code to work on parallel hardware (GPUs and TPUs)



Keras “sits on top of” Tensorflow …

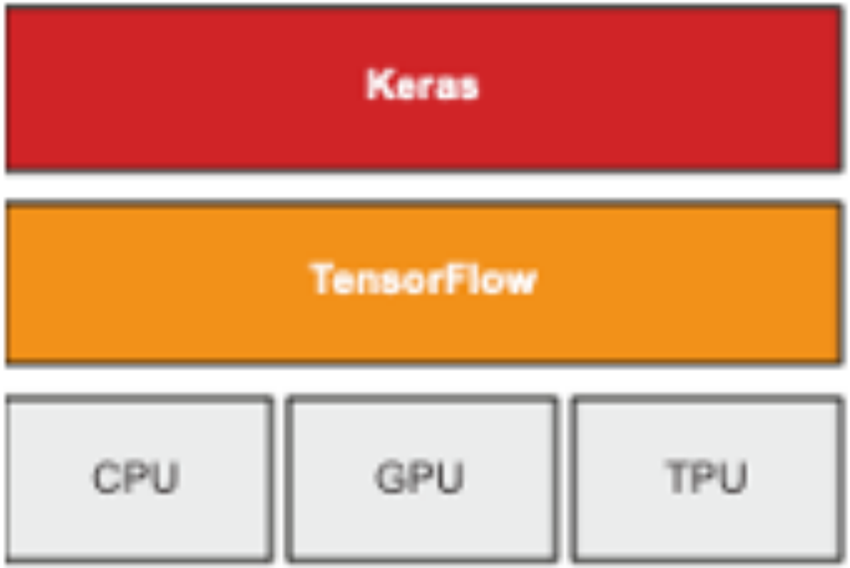


Image: Page 70 of textbook

… and provides “convenience” features

- Pre-defined layers
- Incredibly flexible ways to specify network architectures
- Easy ways to preprocess data
- Easy ways to train models and report metrics
- Pre-trained models you can download and customize

Keras APIs

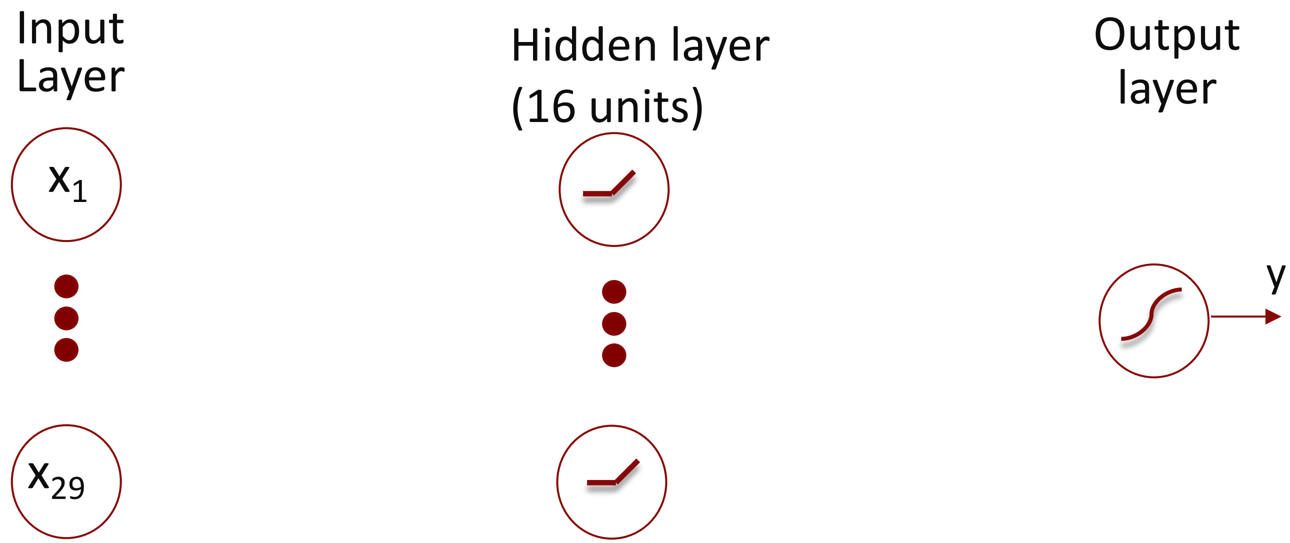
- There are three broad ways to build DL models with Keras
  - Sequential
  - Functional API
  - Subclassing

- We will almost exclusively use the Functional API. The model we built for heart disease prediction is an example.

- Please read 7.2.2 of the textbook to understand in detail how the Keras Functional API works

Check out the wealth of introductory and advanced material, with accompanying colabs, at tensorflow.org and keras.io

Let’s revisit the Neural Model for Heart Disease Prediction we designed previously




```
input = keras.Input(shape=29)
h = keras.layers.Dense(16, activation=“relu”)(input)
output = keras.layers.Dense(1, activation="sigmoid”)(h)
model = keras.Model(input, output)
```

Let’s train this model!

Training Checklist

- We get the data ready (will cover in the colab)
- We design i.e., "lay out" the network 1 hidden layer with 16 ReLU neurons
  - Choose ***the number of hidden layers*** and ***the number of 'neurons' in each layer***
  - Pick the ***right output layer*** based on the type of the output Sigmoid
- We pick
  - An appropriate ***loss function*** based on the type of the output ________
  - An ***optimizer from the many SGD flavors*** that are available
- We decide on a ***regularization strategy***
- We set things up in Keras/Tensorflow and start training!

Training Checklist

- We get the data ready (will cover in the colab)
- We design i.e., “lay out” the network 1 hidden layer with 16 ReLU neurons
  - Choose ***the number of hidden layers*** and ***the number of ‘neurons’ in each layer***
  - Pick the ***right output layer*** based on the type of the output Sigmoid
- We pick
  - An appropriate ***loss function*** based on the type of the output binary crossentropy
  - An ***optimizer from the many SGD flavors*** that are available
- We decide on a ***regularization strategy***
- We set things up in Keras/Tensorflow and start training!

Training Checklist

- We get the data ready (will cover in the colab)
- We design i.e., “lay out” the network 1 hidden layer with 16 ReLU neurons
  - Choose ***the number of hidden layers*** and ***the number of ‘neurons’ in each layer***
  - Pick the ***right output layer*** based on the type of the output Sigmoid
- We pick
  - An appropriate ***loss function*** based on the type of the output binary crossentropy
  - An ***optimizer from the many SGD flavors*** that are available“adam"
- We decide on a ***regularization strategy*** Early stopping
- We set things up in Keras/Tensorflow and start training!

Colab

Predicting Heart Disease

Before we start coding …

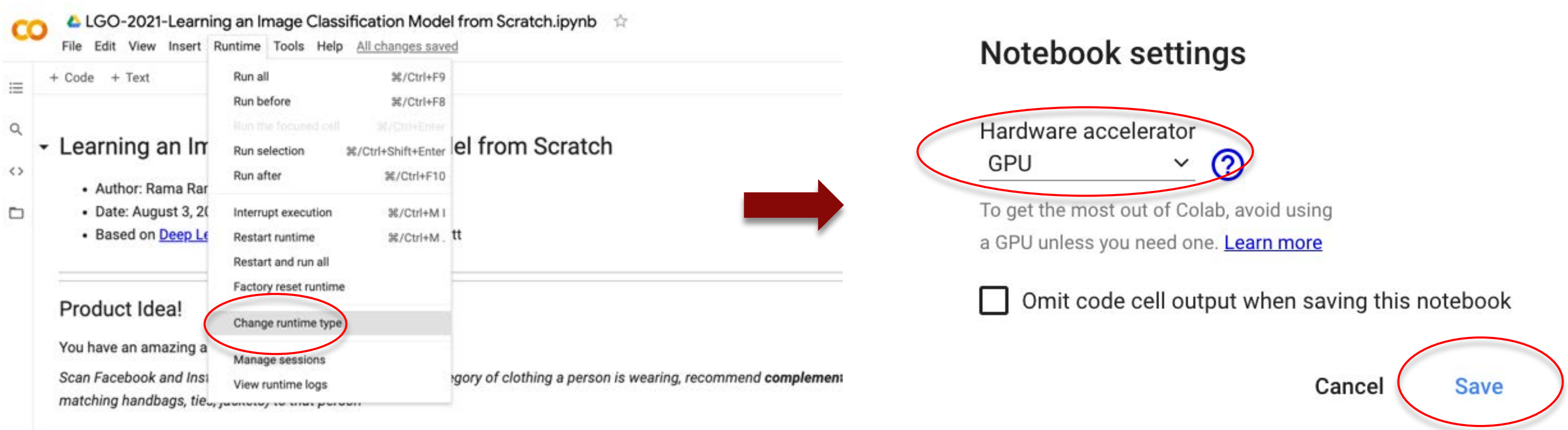
- Don't worry if you don't understand every detail of what we will do in class.

- But go through the Colab notebooks carefully later, play around with the code and make sure you understand every line

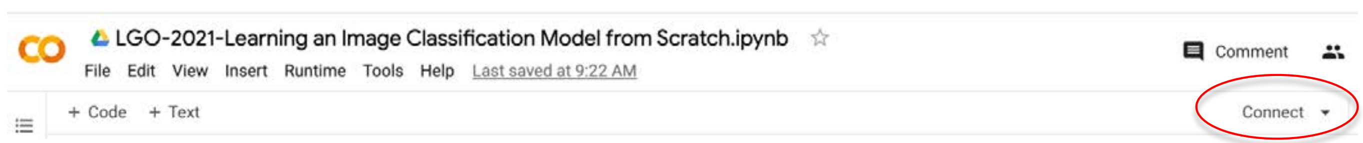
Colab General Instructions

Step 1 Make your own copy of the notebook

Step 2 Request a GPU for your notebook*



Step 3 Start your notebook



You need to do steps 1 and 2 just the first time you use a notebook. From the second time onwards, jump to Step 3.