

# Introduction to Data Science and Machine Learning

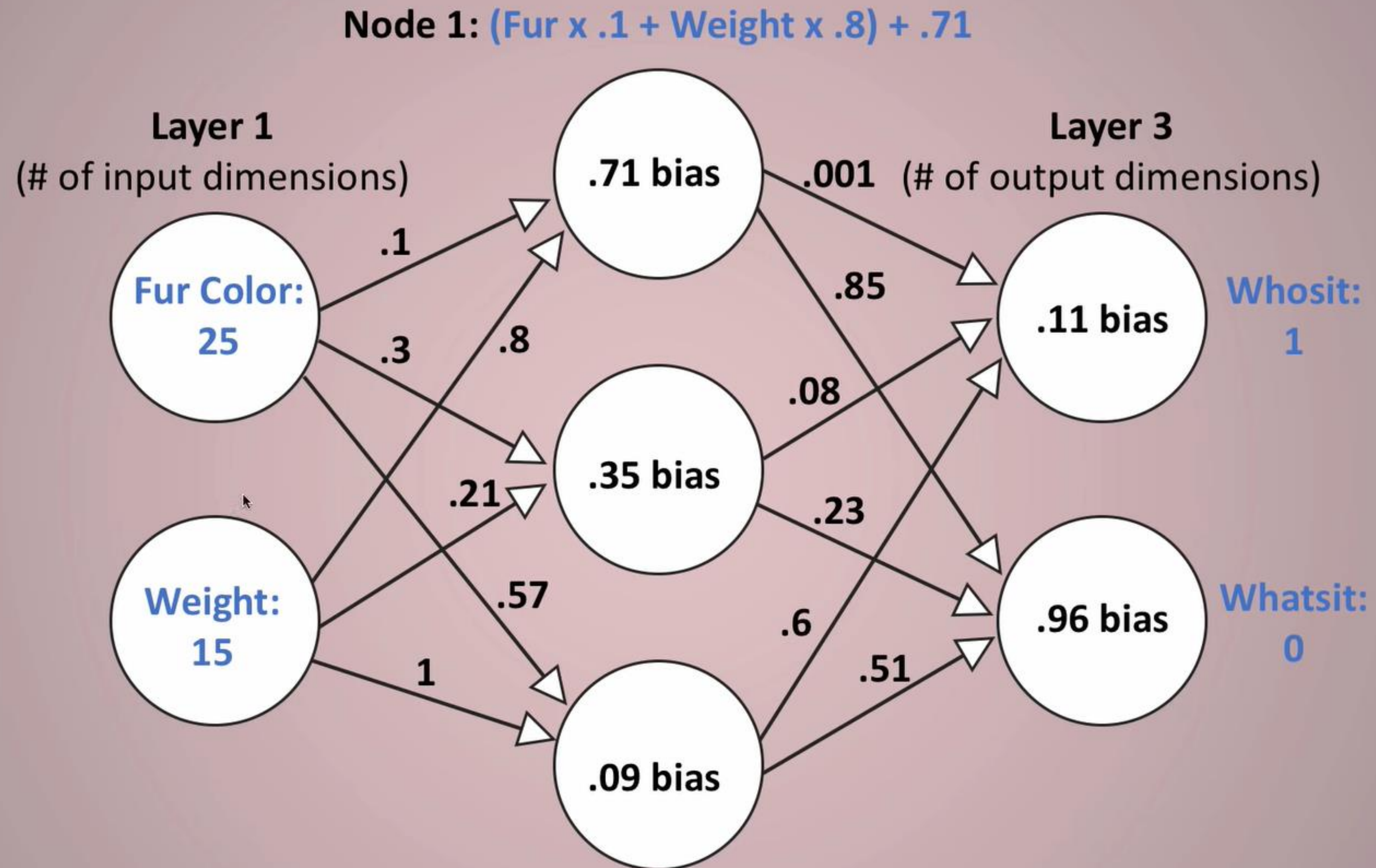
## NEURAL NETS

- **Quiz**
- **Frameworks for Implementing Neural Networks (NN)**
- **Implementing a NN with TensorFlow**
- **Additional Layer Types for NNs**

**QUIZ**







## Learning Rate:

How much should this step outcome affect our weights and biases?

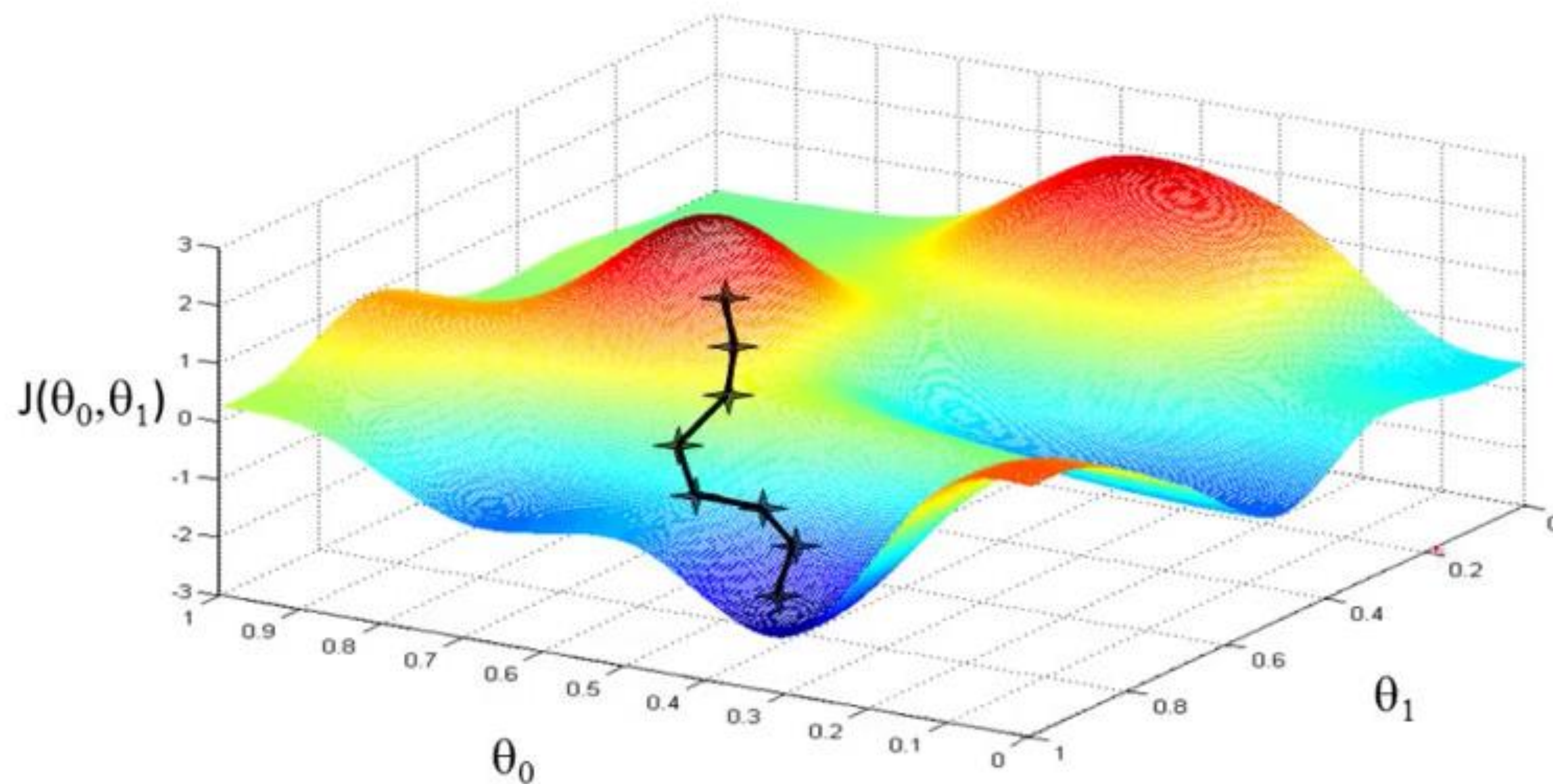
## Momentum:

How much should past outcomes affect our weights and biases?

$$\text{change} = (\text{learningRate} * \text{delta} * \text{value}) + (\text{momentum} * \text{pastChange})$$



# OPTIMIZER PARAMETERS



- **Step size for approaching the cost minimum ("Learning Rate")**
- **Resistance to direction changes ("Momentum")**

Quelle: <https://www.coursera.org/learn/machine-learning>

# PARAMETERS OF THE “ADAM” OPTIMIZER

- **Learning parameter for optimization:  
alpha (learning rate)**
- **Proportion of current gradient in calculating the next optimization step:**
  - **beta1 (decay rate for the direction) and**
  - **beta2 (decay rate for the magnitude of gradients)**

# **HYPERPARAMETERS IN NEURAL NETS**

- **Architecture Definition:**
  - **Number of hidden layers in the network**
  - **Types of hidden layers**
  - **Number of neurons per hidden layer**
  - **Selection of activation function**
- **Selection of cost function („Loss Function“)**
- **Selection of the optimizer**
- **Setting of the learning rate for the optimizer**



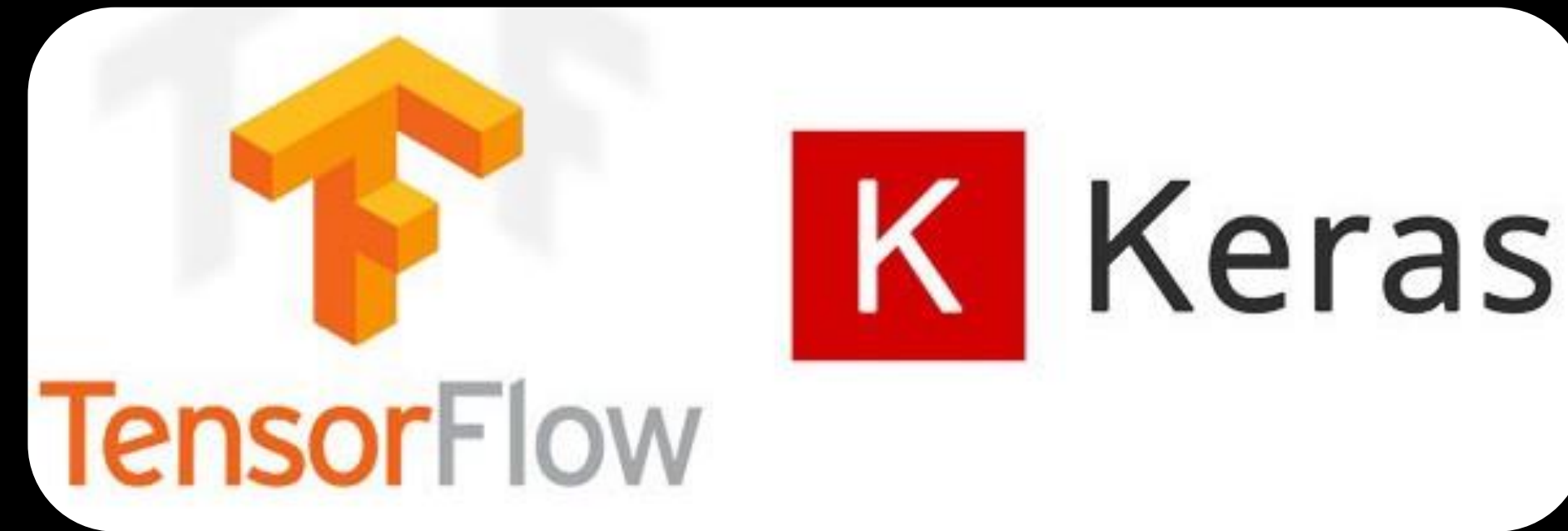
# **LIBRARIES FOR NEURAL NETS**



PYTORCH



Transformers



- **TensorFlow 0.1 (Nov 2015):** Released as open source software by Google; developed by the Google Brain Team for internal research and production)
- **TensorFlow 1.4 (Nov 2017):** Development of the Keras API as a high-level API for TensorFlow and other ML libraries, to increase user-friendliness for commonly used models.
- **TensorFlow 2.0 (Sep 2019):** Keras is integrated as a high-level API into TensorFlow.
- **TensorFlow 2.3 (October 2020):** Significant performance improvements, distributed training, quantized training, and improved mobile deployments.
- **Keras 3.0 (Dec 2023):** Major release that extends support for multiple backends, including TensorFlow, JAX, and PyTorch, making Keras a versatile framework for various deep learning needs.





# Transformers

- **Transformers 1.0 (December 2018):** Hugging Face releases the first version of the Transformer library as open source with implementations of popular transformer models like BERT, GPT, and others.
- **Transformers 2.0 (March 2020):** Support for fine-tuning, production deployment, and quantization. Higher performance and more intuitive APIs.
- **Transformers 3.0 (November 2020):** Support for more task areas like computer vision, audio, and reinforcement learning. Performance optimizations.
- **Transformers 4.0 (May 2022):** Improved performance and integration of additional new techniques for enhancing model optimization.

# **DATA PREPARATION**

**For every modeling, the data must have the following properties:**

- 1. There must be no missing values.**
- 2. All values must be numbers.**
- 3. Categorical variables are one-hot encoded.**

# EXAMPLE DATA PREPARATION FOR THE TRAINING OF A NEURAL NET

FileEditSelectionViewGoRunTerminalHelp

←→Untitled (Workspace)

🔍📄📄📄📄-🗑️✖

tokenizer.ipynbneural\_net\_data\_preparation.ipynb •neural\_net\_estimation.ipynb

🔍+ Code + Markdown | ▶ Run All ↺ Restart 🗑️ Clear All Outputs | 📄 Variables 📄 Outline ...

Python 3.8.0

18

Import Libraries and Data

▶

```
# Import necessary libraries
import pandas as pd
import numpy as np
import os

# Import Data
data = pd.read_csv("https://raw.githubusercontent.com/opencampus-sh/einfuehrung-in-data-science-und-ml/main/house_pricing_data/house_pricing_train.csv")
data.head() # Print first few rows to verify
```

[1] ✓ 5.5sPython

...

	id	date	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	...	sqft_above	sqft_basement	yr_built	yr_renovated	zipcode	lat	long	sqft_livi
0	6840701095	20150403T000000	3	1.00	1740	4400	1.5	0	0	3	...	1740	0	1924	0	98122	47.6059	-122.300	
1	1025049114	20140717T000000	3	2.25	1270	1566	2.0	0	0	3	...	1060	210	2014	0	98105	47.6647	-122.284	
2	4025300360	20150326T000000	3	2.00	1130	16875	1.0	0	0	4	...	1130	0	1947	0	98155	47.7489	-122.300	
3	5536500200	20140918T000000	5	3.50	3760	4857	2.0	0	3	3	...	2820	940	2004	0	98072	47.7398	-122.167	
4	1245003660	20150321T000000	3	2.00	1470	6000	1.0	0	0	3	...	1090	380	1950	1996	98033	47.6829	-122.202	

5 rows × 21 columns

...

Data Preparation

```
# Define categorical features
categorical_features = ['bathrooms', 'zipcode', 'condition']

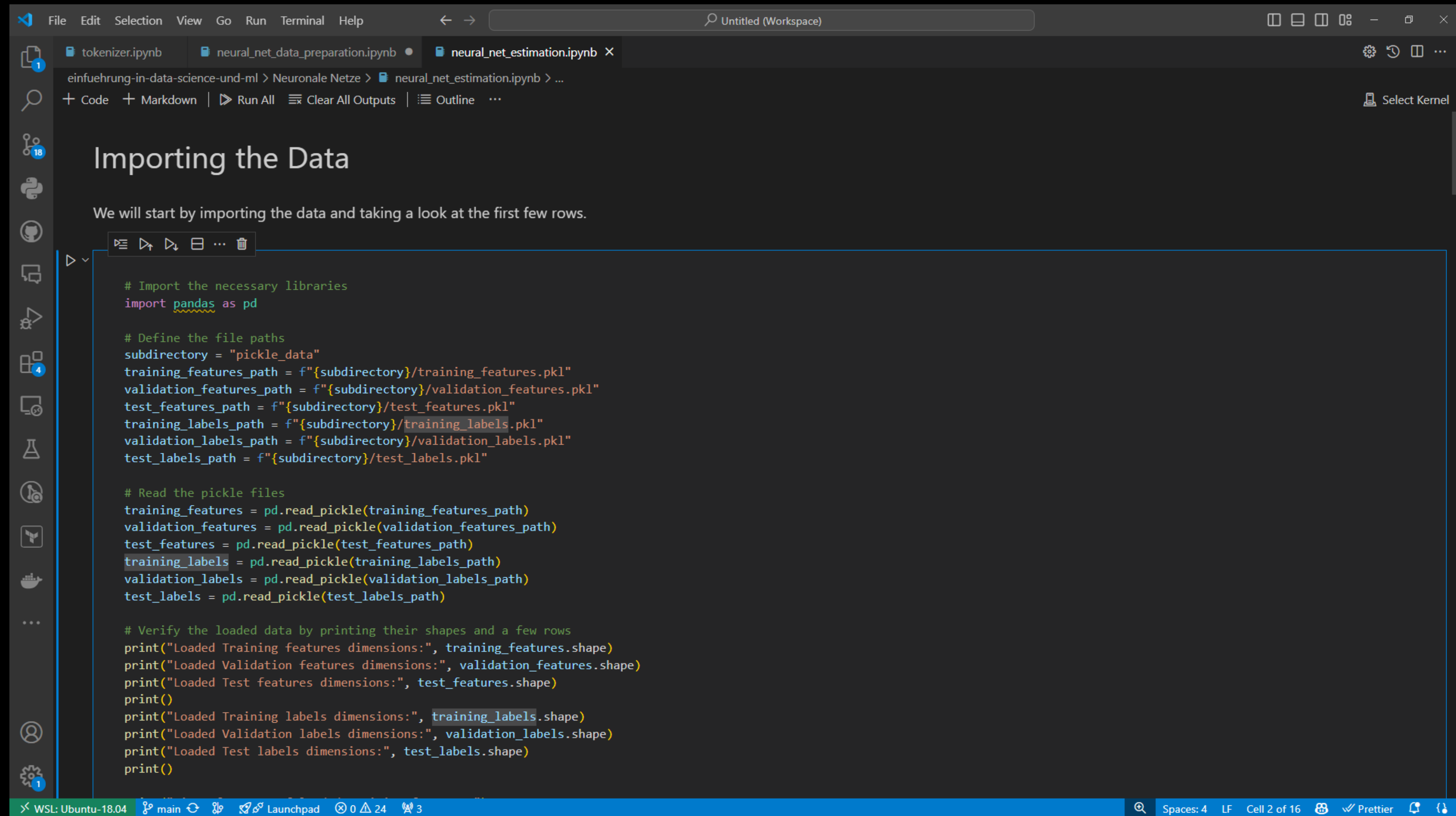
# Inspect data types and unique values for categorical columns
print(data[categorical_features].dtypes)
print("Unique Values:\n", data[categorical_features].apply(lambda x: x.unique()))
```

WSL: Ubuntu-18.04main ↺🔍Launchpad🔒 0 🔒 24🔒 3

🔍Spaces: 4LFCell 6 of 8🔒🔒



# EXAMPLE DEFINITION AND OPTIMIZATION OF A NEURAL NET



The screenshot shows a Jupyter Notebook interface with a dark theme. The top bar includes a menu (File, Edit, Selection, View, Go, Run, Terminal, Help) and a search bar. The notebook has three tabs: 'tokenizer.ipynb', 'neural\_net\_data\_preparation.ipynb', and 'neural\_net\_estimation.ipynb'. The active tab is 'neural\_net\_estimation.ipynb'. The notebook content shows a section titled 'Importing the Data' with a text block stating 'We will start by importing the data and taking a look at the first few rows.' Below this is a code cell containing Python code for importing libraries, defining file paths, reading pickle files, and printing dimensions and shapes of the loaded data.

```
# Import the necessary libraries
import pandas as pd

# Define the file paths
subdirectory = "pickle_data"
training_features_path = f"{subdirectory}/training_features.pkl"
validation_features_path = f"{subdirectory}/validation_features.pkl"
test_features_path = f"{subdirectory}/test_features.pkl"
training_labels_path = f"{subdirectory}/training_labels.pkl"
validation_labels_path = f"{subdirectory}/validation_labels.pkl"
test_labels_path = f"{subdirectory}/test_labels.pkl"

# Read the pickle files
training_features = pd.read_pickle(training_features_path)
validation_features = pd.read_pickle(validation_features_path)
test_features = pd.read_pickle(test_features_path)
training_labels = pd.read_pickle(training_labels_path)
validation_labels = pd.read_pickle(validation_labels_path)
test_labels = pd.read_pickle(test_labels_path)

# Verify the loaded data by printing their shapes and a few rows
print("Loaded Training features dimensions:", training_features.shape)
print("Loaded Validation features dimensions:", validation_features.shape)
print("Loaded Test features dimensions:", test_features.shape)
print()
print("Loaded Training labels dimensions:", training_labels.shape)
print("Loaded Validation labels dimensions:", validation_labels.shape)
print("Loaded Test labels dimensions:", test_labels.shape)
print()
```

The bottom status bar shows 'WSL: Ubuntu-18.04', 'main', 'Launchpad', '0 24', '3', 'Spaces: 4', 'LF', 'Cell 2 of 16', 'Prettier', and a user icon.

# BREAKOUT

- Load the example notebooks from below into your Codespace and run them once unchanged.
- Supplement your data preparation with the steps performed in [this example notebook](#):
  - 1) One-hot encoding of categorical variables
  - 2) Removing cases with missing values
  - 3) Export of training and validation data as pickle files
- Estimate a first neural network based on [this example notebook](#).

# BATCHES, STEPS AND EPOCHS

## *Batch*

- The entire set of training data is divided into separate subgroups of equal size.
- The standard batch size in TensorFlow is 32.

## *Step*

- A single iteration of gradient descent performed on one batch of data, during which all model weights are updated once.

## *Epoch*

- Optimization of the model using the complete training data:  
**Number of Steps × Batch Size = Training Sample Size**
- Depending on the model, very few epochs may suffice, or several hundred or thousand may be needed for optimization.



# NORMALIZATION

## Definition:

- **Subtracting the mean and dividing by the standard deviation.**

**Ensures all input features are on similar scales, which stabilizes training and speeds up convergence.**

# **BATCH NORMALIZATION**

- **Performing normalization at the batch level**

**Additional optimization parameters:**

- **Exactly identical means and standard deviations are not necessarily optimal for modeling purposes**
- Normalization parameters are incorporated as trainable parameters**

# LEARNING RESSOURCES

- Watch [this video](#) (7 minutes) to better understand the properties of dropout layers.
- Watch [this video](#) (5 minutes) to better understand the benefits of normalization.
- Complete the first chapter of [this course](#) on DataCamp to learn about identifying missing values.



# TASKS

- **Examine all your model variables for the existence of missing and implausible values.**
- **Train a first neural network for your dataset.  
(Delete all rows with missing values.)**