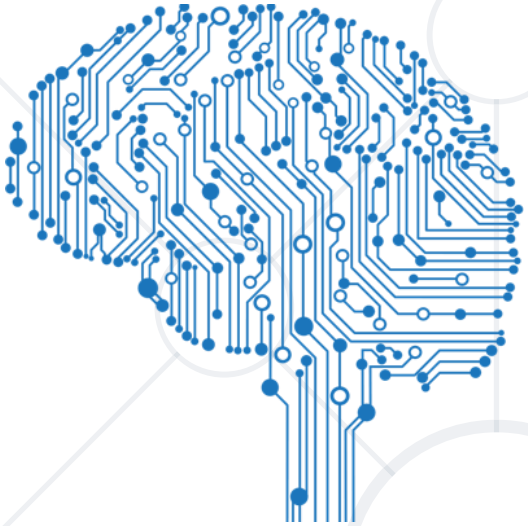
Introduction to Deep Learning

Neural networks, computation, optimization



SoftUni Team

Technical Trainers

Software University

SoftUni

Software University

https://softuni.bg

Have a Question?

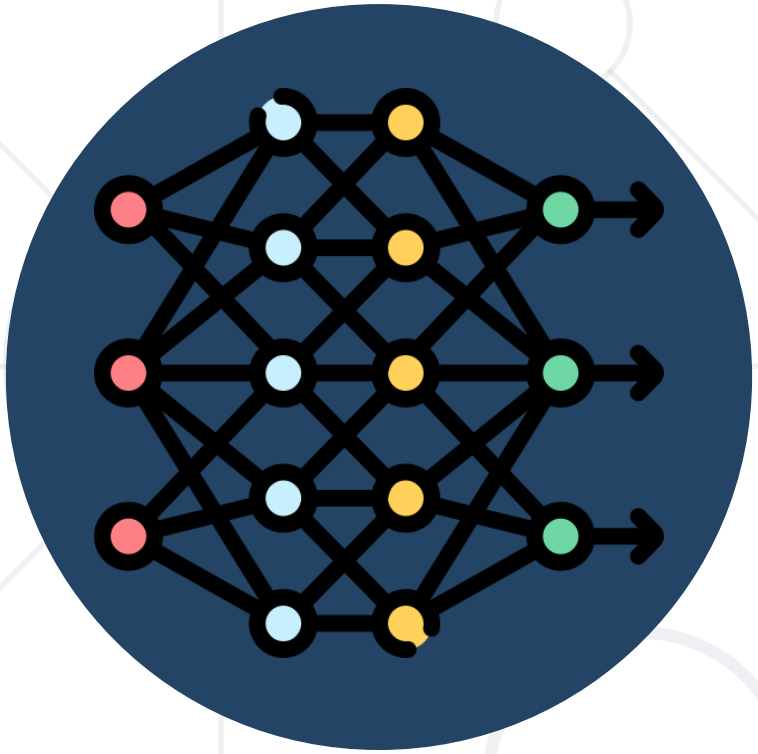
Software University

sli.do

#DeepLearning

Table of Contents

1. Computational graphs
2. Simple models with `tensorflow` and `pytorch`
   - Low-level API
3. Building neural networks
4. Training and evaluation
5. Regularization



Computational Graphs

Performing simple calculations…just harder

Installing tensorflow

Software University

- Use Anaconda
  - It's easier, and arguably much faster
  - Run as administrator
- If possible, try installing the GPU version
  - The package will try to install the GPU version if available
    - If you already have CUDA and / or cuDNN, check their compatibility
  - Otherwise, it'll fall back to the CPU (standard) version

```
conda install tensorflow
```

```
pip install tensorflow
```

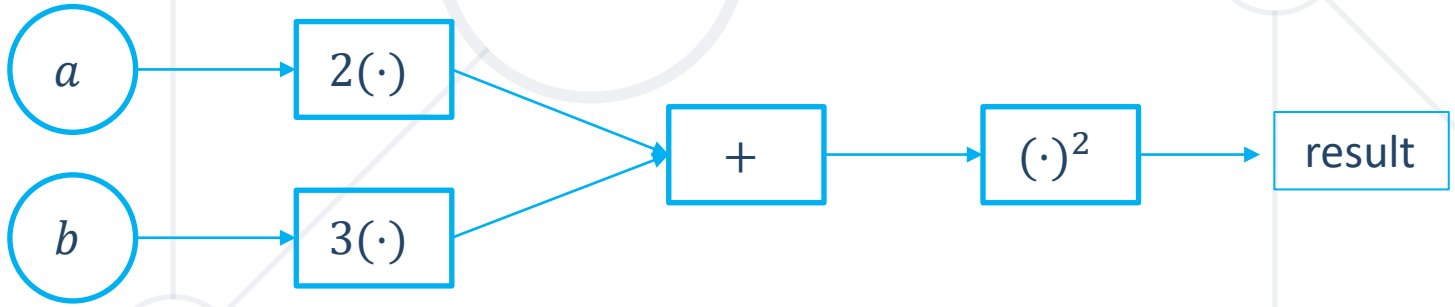
- There's no difference in the API
  - GPUs perform computations much faster
    - Sometimes $\sim 10^2 - 10^4$ times faster
- GPU plugins (in case you don't have an Nvidia device)

Installing pytorch

- You can use Anaconda or pip
- Get your configuration here
  - You'll need to install a bunch of additional libraries
    - If you already have CUDA, you'll need to check its compatibility
    - Sometimes, you'll need to install CUDA and cuDNN separately from Nvidia's archive
  - There are separate installers for the CPU and GPU versions

Computational Graphs

- "Flow of tensors (multidimensional matrices)"
- Computational graph (DAG)
  - A useful representation of computation sequences
  - Contains data and operations
  - Data "flows" through and gets transformed
- A simple example
  - $(2a + 3b)^2$



Working with Calculations

- Python functions work in vanilla Python, `numpy`, `tensorflow`, and `pytorch` ☺

```
a, b = 15, 20
(2 * a + 3 * b) ** 2
# 8100
```

```
a, b = np.array([15]), np.array([20])
(2 * a + 3 * b) ** 2
# array([8100])
```

```
a, b = tf.constant(15), tf.constant(20)
(2 * a + 3 * b) ** 2
# <tf.Tensor: shape=(), dtype=int32, numpy=8100>
```

```
a, b = torch.tensor(15), torch.tensor(20)
(2 * a + 3 * b) ** 2
# tensor(8100)
```

Linear Models

… from the viewpoint of deep learning

Logistic Regression with tensorflow

Software University

- For simplicity, we're using the Iris dataset

```
from sklearn.datasets import load_iris
iris = load_iris()
attributes, labels = iris.data, iris.target

n_features, classes = X.shape[1], len(set(y))
```

- Define the architecture

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Input, Dense
```

```
tf_model = Sequential([
    Input((n_features,)),
    Dense(n_classes, activation = "softmax")
])
tf_model.compile(
    loss = "sparse_categorical_crossentropy", metrics = ["accuracy"])
```

Logistic Regression with tensorflow

Software University

- Training

```
tf_model.compile(
    loss = "sparse_categorical_crossentropy",
    metrics = ["accuracy"]
)
```

```
tf_model.fit(X, y, epochs = 1000)
```

- Evaluation (accuracy)

```
tf_model.score(X, y)
```

Logistic Regression with pytorch

Software University

- We need to convert the data to tensors

```
X_pt = torch.FloatTensor(X)
y_pt = torch.LongTensor(y)
```

- PyTorch has an OOP-based API

```
class LogisticRegressionPT(torch.nn.Module):
    def __init__(self):
        super(LogisticRegressionPT, self).__init__()
        self.layer = torch.nn.Linear(n_features, n_classes)

    def forward(self, x):
        x = torch.nn.functional.softmax(self.layer(x))
        return x
```

```
learning_rate = 0.01
criterion = torch.nn.CrossEntropyLoss()
optimizer = torch.optim.Adam(pt_model.parameters(), lr = learning_rate)
```

Logistic Regression with pytorch

Software University

- Training

```
def train(model, optimizer, criterion, X, y, num_epochs, train_losses):
    for epoch in range(num_epochs):
        optimizer.zero_grad()
        output_train = model(X_train) # forward

        loss_train = criterion(output_train, y_train)
        loss_train.backward() # backward
        optimizer.step() # weight update

        train_losses[epoch] = loss_train.item()

        if (epoch + 1) % 50 == 0:
            print(f"Epoch {epoch+1}/{num_epochs}, Loss: {loss_train.item():.4f}")
```

```
num_epochs = 1000
train_losses = np.zeros(num_epochs)

train(pt_model, optimizer, criterion, X_pt, y_pt, num_epochs, train_losses)
```

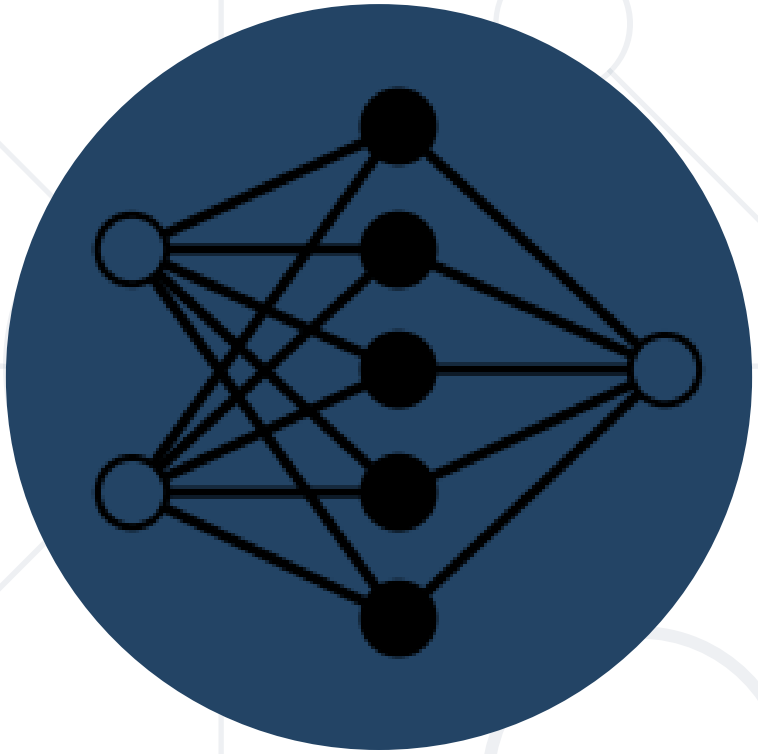
Logistic Regression with pytorch

Software University

- Yep! That's immensely tedious…
  - Use PyTorch Lightning to make it less so
- Evaluation

```
from torcheval.metrics.functional import multiclass_accuracy

predictions = torch.argmax(pt_model.forward(X_pt), dim = 1)
multiclass_accuracy(predictions, y_pt)
```

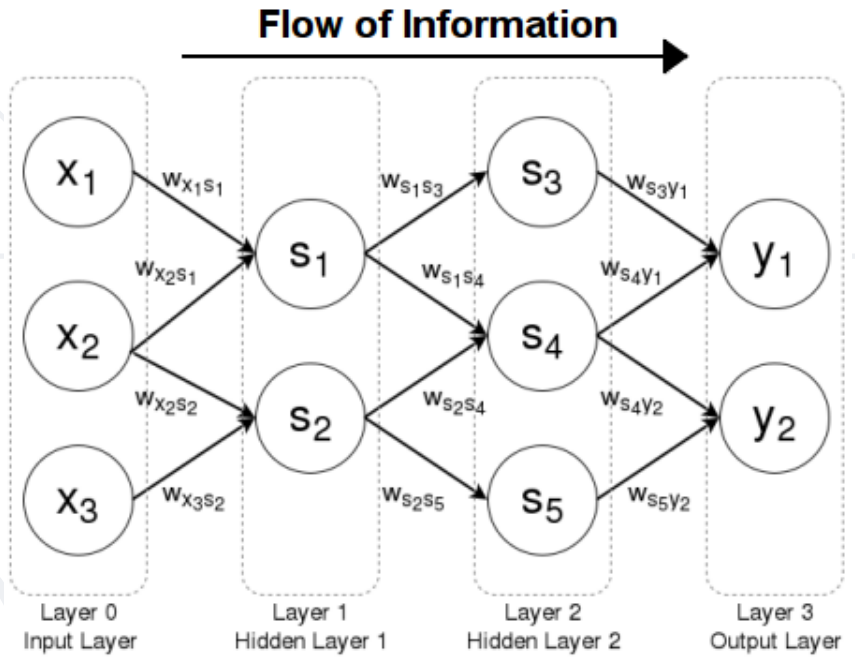


Neural Networks

Going deeper

Deep Feed-Forward Neural Network

- Many **perceptrons** arranged in **layers**
  - **Input** layer
  - **Hidden** layers
  - **Output** layer
- Computing output: forward propagation
- Training: backpropagation
- We can do this using the low-level API
  - If you want to implement this, don't forget
    - Bias term for each layer
    - Activation function
    - Random weight initialization (small numbers with $\mu = 0$)



Comparing Numbers

- Load and normalize the data

```
mnist = tf.keras.datasets.mnist

(x_train, y_train), (x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0
```

- Create and evaluate the model

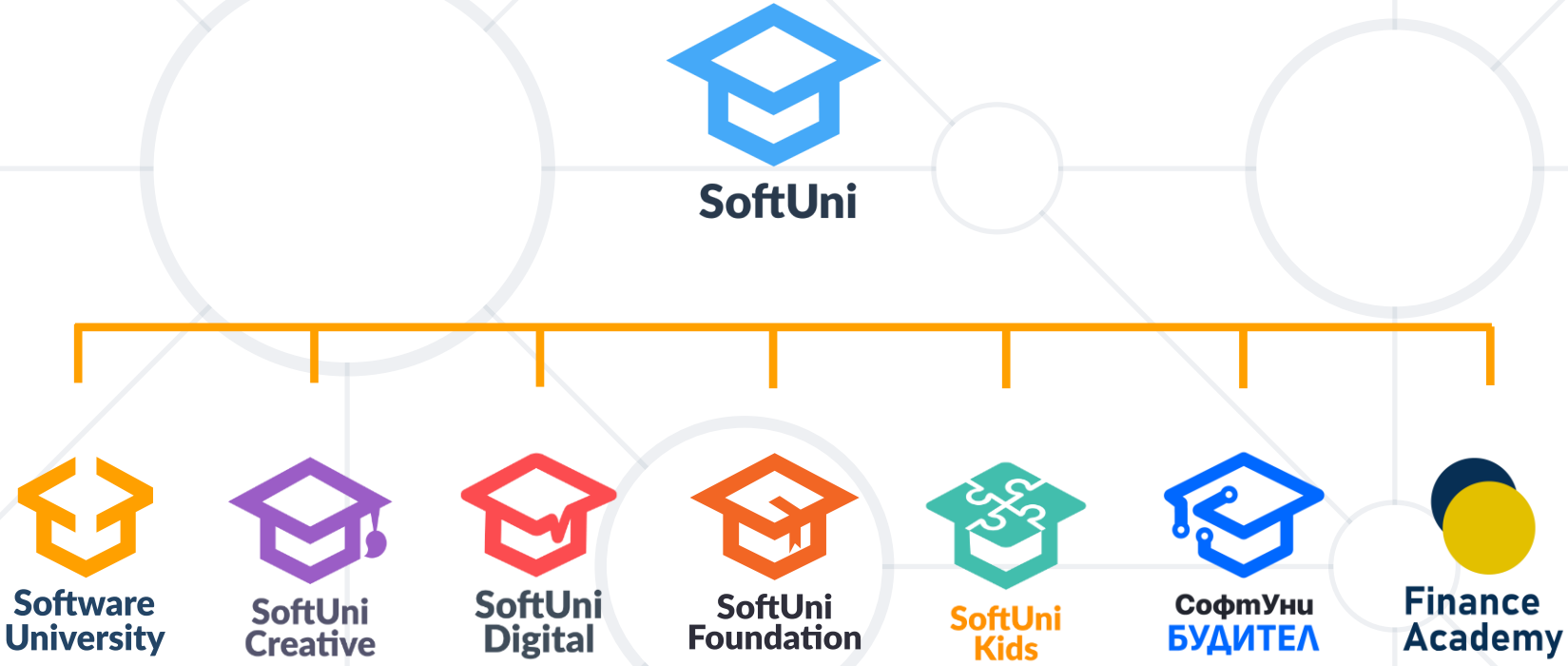
```
model = Sequential([
    Flatten(),
    Dense(512, activation = "relu"),
    Dropout(0.2),
    Dense(10, activation = "softmax")])
model.compile(
    optimizer = "adam",
    loss = "sparse_categorical_crossentropy",
    metrics = ["accuracy"])
```

```
model.fit(x_train, y_train, epochs = 5)
model.evaluate(x_test, y_test)
# or add validation_data = (x_test, y_test) to model.fit()
```

Summary

1. Computational graphs
2. Simple models with `tensorflow` and `pytorch`
   - Low-level API
3. Building neural networks
4. Training and evaluation
5. Regularization

Questions?



SoftUni Diamond Partners

Software University

SmartIT

Coca-Cola HBC Bulgaria

SUPER HOSTING .BG

createX

Postbank

top EMPLOYER България Bulgaria 2024 FOR A BETTER WORLD OF WORK

DRAFT KINGS

THE CROWN IS YOURS

VIVACOM

INDEAVR

Serving the high achievers

AMBITIONED

PHAR VISION

Trainings @ Software University (SoftUni)

- Software University – High-Quality Education, Profession and Job for Software Developers
  - softuni.bg, about.softuni.bg
- Software University Foundation
  - softuni.foundation
- Software University @ Facebook
  - facebook.com/SoftwareUniversity

License

- This course (slides, examples, demos, exercises, homework, documents, videos and other assets) is **copyrighted content**
- Unauthorized copy, reproduction or use is illegal
- © SoftUni – https://about.softuni.bg/
- © Software University – https://softuni.bg