TensorFlow: Federated Learning for Image Classification

Andrea Hofer, Ignatz Ötzlinger, Sabine Hasenleithner,
Kim Schmider

image source: https://commons.wikimedia.org/wiki/File:Tensorflow_logo.svg

Introduction

- Introduction to Machine Learning and Deep Learning
- Image Recognition
- CIFAR-10 dataset
- Practical example
- Why Federated Learning?
- `FederatedAveraging` Algorithm

What is Machine Learning?

- Big Data: There is no data like more data

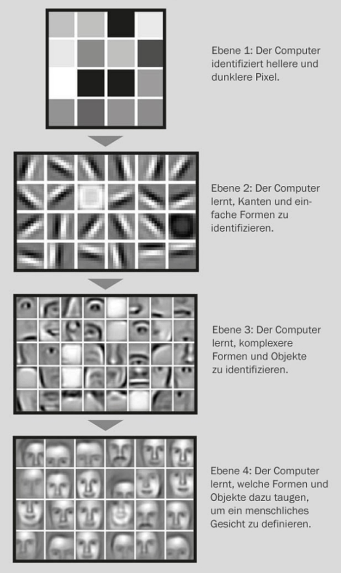
What is Machine Learning?

- Big Data: There is no data like more data
- Deep Learning: Error rate below 4%

What is Machine Learning?

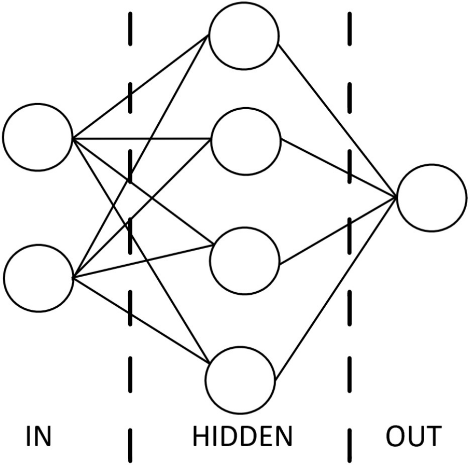
- Big Data: There is no data like more data
- Deep Learning: Error rate below 4%

source: https://www.computerwoche.de/a/was-sie-ueber-maschinelles-lernen-wissen-muessen,3329560



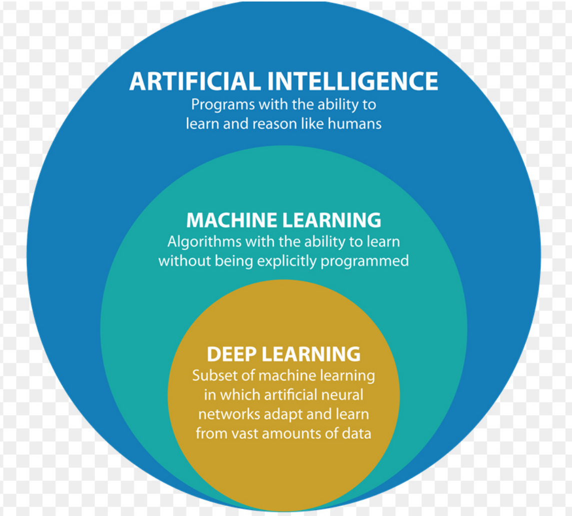
source:
`http://www.spektrum.de/news/maschinenlernen-deep-learning-macht-kuenstliche-intelligenz-praxistauglich/1220451`

Neural Network



source: own figure

Deep Learning



source: `https://de.cleanpng.com/png-vy5m68`

Image Recognition

- Classification

Image Recognition

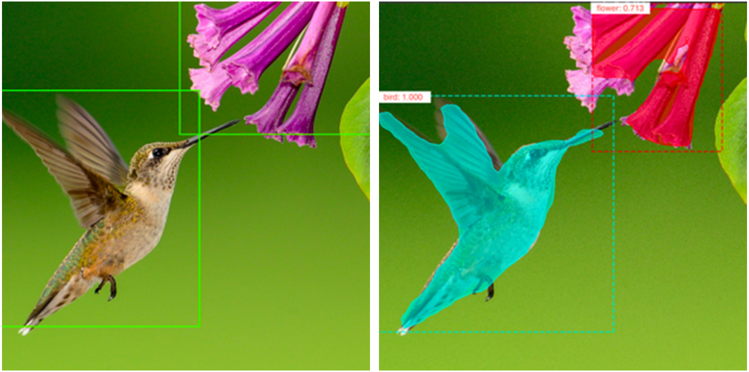
- Classification
- Object recognition

Image Recognition

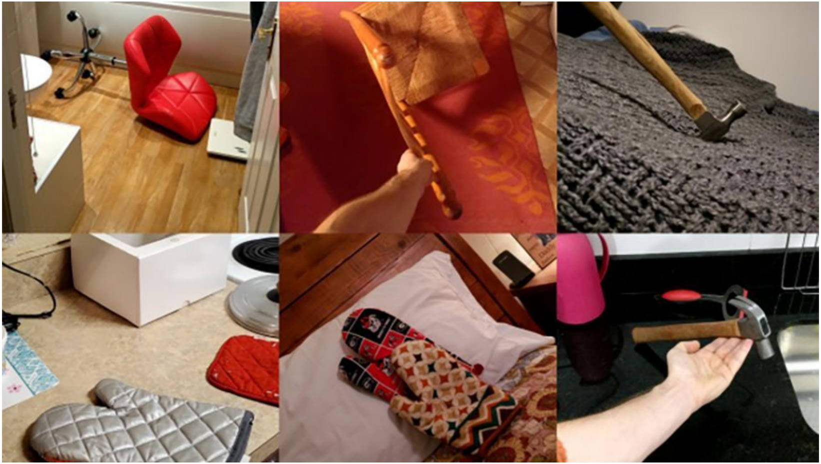
- Classification
- Object recognition
- Instance segmentation

Image Recognition

- Classification
- Object recognition
- Instance segmentation

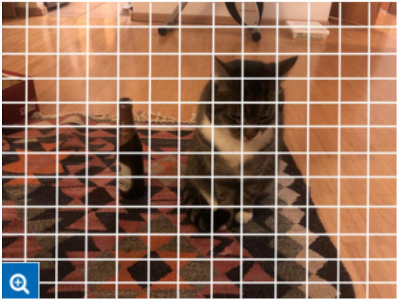
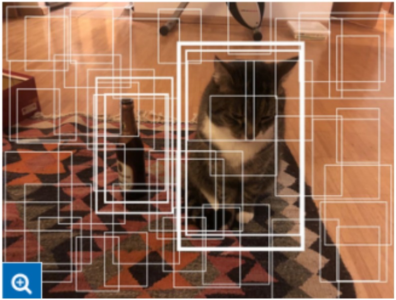


source: `https://www.burda.com/de/news/tech-einmaleins-bilderkennung-und-ki`



source:
https://www.spiegel.de/netzwelt/web/bilderkennung-an-diesen-fotos-scheitert-kuenstliche-intelligenz-a-1301234.html

YOLO Algorithm

source: https://www.sigs-datacom.de/trendletter/2018-10/4-wie-man-in-echtzeit-mehrere-objekte-mit-deep-learning-und-yolo-erkennen-und-klassifizieren-kann.html

CIFAR-10

- CIFAR-10 dataset (Canadian Institute For Advanced Research)

CIFAR-10

- CIFAR-10 dataset (Canadian Institute For Advanced Research)
- Collection of images

CIFAR-10

- CIFAR-10 dataset (Canadian Institute For Advanced Research)
- Collection of images
- Contains 60.000 images

CIFAR-10

- CIFAR-10 dataset (Canadian Institute For Advanced Research)
- Collection of images
- Contains 60.000 images
- 10 different classes:
  1. airplanes,
  2. cars,
  3. birds,
  4. cats,
  5. deer,
  6. dogs,
  7. frogs,
  8. horses,
  9. ships,
  10. trucks.

CIFAR-10

- CIFAR-10 dataset (Canadian Institute For Advanced Research)
- Collection of images
- Contains 60.000 images
- 10 different classes:
  1. airplanes,
  2. cars,
  3. birds,
  4. cats,
  5. deer,
  6. dogs,
  7. frogs,
  8. horses,
  9. ships,
  10. trucks.
- 6.000 images per class

CIFAR-10

- CIFAR-10 dataset (Canadian Institute For Advanced Research)
- Collection of images
- Contains 60.000 images
- 10 different classes:
  1. airplanes,
  2. cars,
  3. birds,
  4. cats,
  5. deer,
  6. dogs,
  7. frogs,
  8. horses,
  9. ships,
  10. trucks.
- 6.000 images per class
- Size 32 x 32

Image Classification

How does the image classification work?
`https://www.tensorflow.org/tutorials/images/cnn`
This tutorial demonstrates training a simple Convolutional Neural Network (CNN) to classify CIFAR images. (TensorFlow)

SGD – Stochastic gradient descent

- Iterative method for optimizing an objective function

SGD – Stochastic gradient descent

- Iterative method for optimizing an objective function
- Replaces the actual gradient (calculated from the entire data set) by an estimate thereof (calculated from a randomly selected subset of the data)

SGD – Stochastic gradient descent

- Iterative method for optimizing an objective function
- Replaces the actual gradient (calculated from the entire data set) by an estimate thereof (calculated from a randomly selected subset of the data)
- True gradient of $Q(w)$ is approximated by a gradient at a single example

SGD – Stochastic gradient descent

- Iterative method for optimizing an objective function
- Replaces the actual gradient (calculated from the entire data set) by an estimate thereof (calculated from a randomly selected subset of the data)
- True gradient of $Q(w)$ is approximated by a gradient at a single example
- $w := w - \eta \nabla Q_i(w)$

SGD – Stochastic gradient descent

- Iterative method for optimizing an objective function
- Replaces the actual gradient (calculated from the entire data set) by an estimate thereof (calculated from a randomly selected subset of the data)
- True gradient of $Q(w)$ is approximated by a gradient at a single example
- $w := w - \eta \nabla Q_i(w)$
- Algorithm sweeps through the training set and performs the above update for each training example. Several passes can be made over the training set until the algorithm converges. If this is done, the data can be shuffled for each pass to prevent cycles

SGD – Possible pseudocode

choose an initial vector of parameters w and learning rate η

SGD – Possible pseudocode

choose an initial vector of parameters w and learning rate η
repeat until an approximate minimum is obtained **do**

SGD – Possible pseudocode

choose an initial vector of parameters w and learning rate η
repeat until an approximate minimum is obtained **do**
    randomly shuffle examples in the training set

SGD – Possible pseudocode

choose an initial vector of parameters w and learning rate η
repeat until an approximate minimum is obtained **do**
    randomly shuffle examples in the training set
    for $i = 1, 2, \ldots, n$ **do**

SGD – Possible pseudocode

choose an initial vector of parameters w and learning rate η

repeat until an approximate minimum is obtained **do**

    randomly shuffle examples in the training set

    for $i = 1, 2, ..., n$ **do**

        $w := w - \eta \nabla Q_i(w)$

SGD – Possible pseudocode

choose an initial vector of parameters w and learning rate η
repeat until an approximate minimum is obtained **do**
    randomly shuffle examples in the training set
    for $i = 1, 2, ..., n$ **do**
        $w := w - \eta \nabla Q_i(w)$

source: Bottou, Léon (1998). "Online Algorithms and Stochastic Approximations". Online Learning and Neural Networks. Cambridge University Press. ISBN 978-0-521-65263-6.

Why Federated Learning?

Federated Learning

- Calculations on end device

Federated Learning

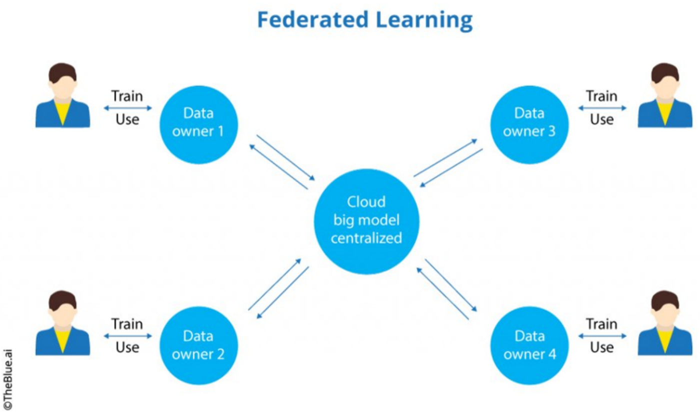
- Calculations on end device
- Results of calculations will be sent to global model

Federated Learning

- Calculations on end device
- Results of calculations will be sent to global model
- Data privacy

Federated Learning

- Calculations on end device
- Results of calculations will be sent to global model
- Data privacy



source: https://theblue.ai/blog-de/federated-learning-foederales-lernen

Federated Learning for image classification

- Can be simulated at TensorFlow: https://www.tensorflow.org/federated/tutorials/federated_learning_for_image_classification

Federated Learning for image classification

- Can be simulated at TensorFlow: `https://www.tensorflow.org/federated/tutorials/federated_learning_for_image_classification`
- this simulation example uses the NIST (National Institute of Standards and Technology) dataset

NIST Handprinted Forms and Characters Database

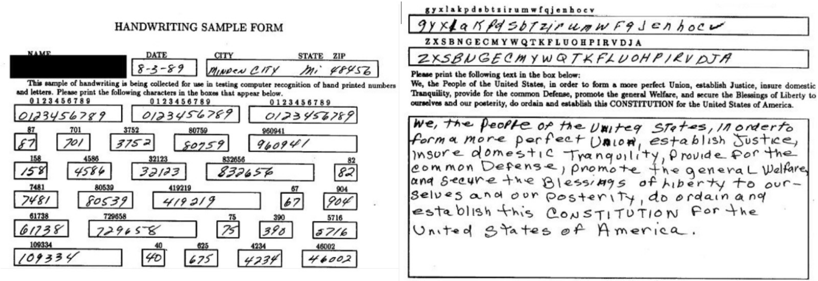
- 3600 writers

NIST Handprinted Forms and Characters Database

- 3600 writers
- 810,000 character images

NIST Handprinted Forms and Characters Database

- 3600 writers
- 810,000 character images



HANDWRITING SAMPLE FORM

NAME [redacted] DATE 8-3-89 CITY MINDEN CITY STATE MI ZIP 48456

This sample of handwriting is being collected for use in testing computer recognition of hand printed numbers and letters. Please print the following characters in the boxes that appear below.

0123456789 0123456789 0123456789

87 701 3752 80759 960941

158 4586 32123 832656 82

7481 80539 419219 67 904

61738 729658 75 390 5716

109334 40 625 4234 46002

gyxlakpdsbtzirumwfqjenhocv

ZXSBNGECMYWQTKFLUOHPIRVDJA

Please print the following text in the box below:

We, the People of the United States, in order to form a more perfect Union, establish Justice, insure domestic Tranquility, provide for the common Defense, promote the general Welfare, and secure the Blessings of Liberty to ourselves and our posterity, do ordain and establish this CONSTITUTION for the United States of America.

source: `https://www.nist.gov/srd/nist-special-database-19`

Short overview Simulation results

```
>>> state, metrics = iterative_process.next(state, federated_train_data)
>>> print('round  1, metrics={}'.format(metrics))
round  1, metrics=<sparse_categorical_accuracy=0.14228394627571106,loss=2.9735283851623535>
>>> for round_num in range(2, 11):
...   state, metrics = iterative_process.next(state, federated_train_data)
...   print('round {:2d}, metrics={}'.format(round_num, metrics))
...
round  2, metrics=<sparse_categorical_accuracy=0.18034979701042175,loss=2.6374611854553223>
round  3, metrics=<sparse_categorical_accuracy=0.21903292834758759,loss=2.547621965408325>
round  4, metrics=<sparse_categorical_accuracy=0.2531892955303192,loss=2.378369092941284>
round  5, metrics=<sparse_categorical_accuracy=0.32098764181137085,loss=2.047738552093506>
round  6, metrics=<sparse_categorical_accuracy=0.36450618505477905,loss=1.9402925968170166>
round  7, metrics=<sparse_categorical_accuracy=0.4194444417953491,loss=1.7680468559265137>
round  8, metrics=<sparse_categorical_accuracy=0.4552469253540039,loss=1.6488759517669678>
round  9, metrics=<sparse_categorical_accuracy=0.529629647731781,loss=1.4977922439575195>
round 10, metrics=<sparse_categorical_accuracy=0.5630658268928528,loss=1.3877880573272705>
```

source: own figure

Short overview Simulation results

```
>>> state, metrics = iterative_process.next(state, federated_train_data)
>>> print('round  1, metrics={}'.format(metrics))
round  1, metrics=<sparse_categorical_accuracy=0.14228394627571106,loss=2.9735283851623535>
>>> for round_num in range(2, 11):
...   state, metrics = iterative_process.next(state, federated_train_data)
...   print('round {:2d}, metrics={}'.format(round_num, metrics))
...
round  2, metrics=<sparse_categorical_accuracy=0.18034979701042175,loss=2.6374611854553223>
round  3, metrics=<sparse_categorical_accuracy=0.21903292834758759,loss=2.547621965408325>
round  4, metrics=<sparse_categorical_accuracy=0.2531892955303192,loss=2.378369092941284>
round  5, metrics=<sparse_categorical_accuracy=0.32098764181137085,loss=2.047738552093506>
round  6, metrics=<sparse_categorical_accuracy=0.36450618505477905,loss=1.9402925968170166>
round  7, metrics=<sparse_categorical_accuracy=0.4194444417953491,loss=1.7680468559265137>
round  8, metrics=<sparse_categorical_accuracy=0.4552469253540039,loss=1.6488759517669678>
round  9, metrics=<sparse_categorical_accuracy=0.529629647731781,loss=1.4977922439575195>
round 10, metrics=<sparse_categorical_accuracy=0.5630658268928528,loss=1.3877880573272705>
```

source: own figure

Training loss is decreasing after each round of federated training, indicating the model is converging

Short overview Simulation results

```
>>> state, metrics = iterative_process.next(state, federated_train_data)
>>> print('round  1, metrics={}'.format(metrics))
round  1, metrics=<sparse_categorical_accuracy=0.14228394627571106,loss=2.9735283851623535>
>>> for round_num in range(2, 11):
...   state, metrics = iterative_process.next(state, federated_train_data)
...   print('round {:2d}, metrics={}'.format(round_num, metrics))
...
round  2, metrics=<sparse_categorical_accuracy=0.18034979701042175,loss=2.6374611854553223>
round  3, metrics=<sparse_categorical_accuracy=0.21903292834758759,loss=2.547621965408325>
round  4, metrics=<sparse_categorical_accuracy=0.2531892955303192,loss=2.378369092941284>
round  5, metrics=<sparse_categorical_accuracy=0.32098764181137085,loss=2.047738552093506>
round  6, metrics=<sparse_categorical_accuracy=0.36450618505477905,loss=1.9402925968170166>
round  7, metrics=<sparse_categorical_accuracy=0.4194444417953491,loss=1.7680468559265137>
round  8, metrics=<sparse_categorical_accuracy=0.4552469253540039,loss=1.6488759517669678>
round  9, metrics=<sparse_categorical_accuracy=0.529629647731781,loss=1.4977922439575195>
round 10, metrics=<sparse_categorical_accuracy=0.5630658268928528,loss=1.3877880573272705>
```

source: own figure

- Training loss is decreasing after each round of federated training, indicating the model is converging
- The average metrics over all batches of data trained across all clients in the round

Example Uses of Federated Learning

- Medicine

Example Uses of Federated Learning

- Medicine
- Touch keyboard input prediction

Example Uses of Federated Learning

- Medicine
- Touch keyboard input prediction

- With learning based on user interactions labels are directly available

Practical Issues With Federated Learning

- Datasets are not representative of population

Practical Issues With Federated Learning

- Datasets are not representative of population
- Data differs from traditional centralized sources

Practical Issues With Federated Learning

- Datasets are not representative of population
- Data differs from traditional centralized sources
- Users' usage styles differ greatly

Practical Issues With Federated Learning

- Datasets are not representative of population
- Data differs from traditional centralized sources
- Users' usage styles differ greatly
- Many clients each with small datasets

Practical Issues With Federated Learning

- Datasets are not representative of population
- Data differs from traditional centralized sources
- Users' usage styles differ greatly
- Many clients each with small datasets
- Mobile devices are often offline or on a slow or expensive connection

Practical Issues With Federated Learning

- Datasets are not representative of population
- Data differs from traditional centralized sources
- Users' usage styles differ greatly
- Many clients each with small datasets
- Mobile devices are often offline or on a slow or expensive connection
- Time zone differences

Resource Utilization

- Data center: Computational costs dominate

Resource Utilization

- Data center: Computational costs dominate
- Federated Learning: Communication costs dominate, computation essentially free

Resource Utilization

- Data center: Computational costs dominate
- Federated Learning: Communication costs dominate, computation essentially free
- Number of updates is minimized:

Resource Utilization

- Data center: Computational costs dominate
- Federated Learning: Communication costs dominate, computation essentially free
- Number of updates is minimized:
  - by running on more clients in parallel (diminishing returns), or

Resource Utilization

- Data center: Computational costs dominate
- Federated Learning: Communication costs dominate, computation essentially free
- Number of updates is minimized:
  - by running on more clients in parallel (diminishing returns), or
  - by performing more computations between updates.

FederatedAveraging Algorithm

Server
Main() **do**
  initialize w_0
  for each round $t = 1, 2, \ldots$ **do**
    $S_t \leftarrow$ random fraction of clients
    for each client $k \in S_t$ **in parallel do**
      $(w_{t+1}^k, n_k) \leftarrow \text{ClientUpdate}(k, w_t)$

FederatedAveraging Algorithm

Server
Main() do
  initialize w_0
  for each round $t = 1, 2, \ldots$ **do**
    $S_t \leftarrow$ random fraction of clients
    for each client $k \in S_t$ **in parallel do**
      $(w_{t+1}^k, n_k) \leftarrow \text{ClientUpdate}(k, w_t)$

Client k
ClientUpdate(k, w) **do**
  for each epoch **do**
    $w \leftarrow w - \eta \nabla Q_i(w)$
  return (w, n_k) to server

FederatedAveraging Algorithm

Server

Main() **do**

  initialize w_0

  for each round $t = 1, 2, \ldots$ **do**

    $S_t \leftarrow$ random fraction of clients

    for each client $k \in S_t$ **in parallel do**

      $(w_{t+1}^k, n_k) \leftarrow \text{ClientUpdate}(k, w_t)$

    $n \leftarrow n_1 + n_2 + \ldots + n_k$

    $w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k$

Client k

ClientUpdate(k, w) **do**

  for each epoch **do**

    $w \leftarrow w - \eta \nabla Q_i(w)$

  return (w, n_k) to server

FederatedAveraging Algorithm

Server

Main() **do**

  initialize w_0

  for each round $t = 1, 2, \ldots$ **do**

    $S_t \leftarrow$ random fraction of clients

    for each client $k \in S_t$ **in parallel do**

      $(w_{t+1}^k, n_k) \leftarrow \text{ClientUpdate}(k, w_t)$

    $n \leftarrow n_1 + n_2 + \ldots + n_k$

    $w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k$

Client k

ClientUpdate(k, w) **do**

  for each epoch **do**

    $w \leftarrow w - \eta \nabla Q_i(w)$

  return (w, n_k) to server

source: H. Brendan McMahan, et. al. (2016). "Communication-Efficient Learning of Deep Networks from Decentralized Data". arXiv.org.

The End