Deep Learning in 30 Minuten

© 2023 USEDATA – Mag. Helmut Grillenberger

Statistiker - KI-Experte - R/Python-Spezialist

© 2023 USEDATA – Mag. Helmut Grillenberger

MNIST

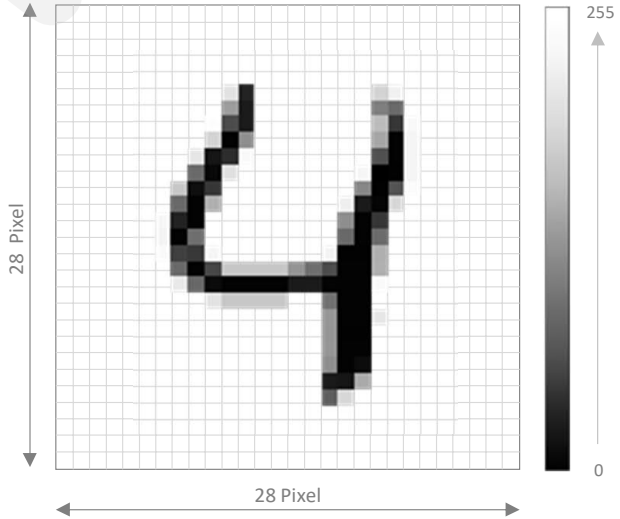
MNIST-Datenbestand

Modified National Institute of Standards and Technology

70.000 Graustufenbilder 28 x 28 Pixel

60.000
Trainingsdaten

10.000
Testdaten

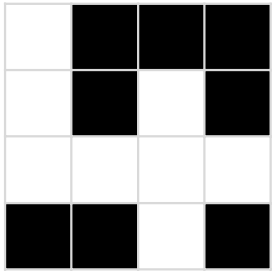


USEDATA

© 2023 USEDATA – Mag. Helmut Grillenberger

Vektorisierung eines Bildes

4 x 4 x 1
Schwarzweiß-Bild
nur für Demo



Daten-Vektorisierung

16 Pixel

Andere Beispiele:

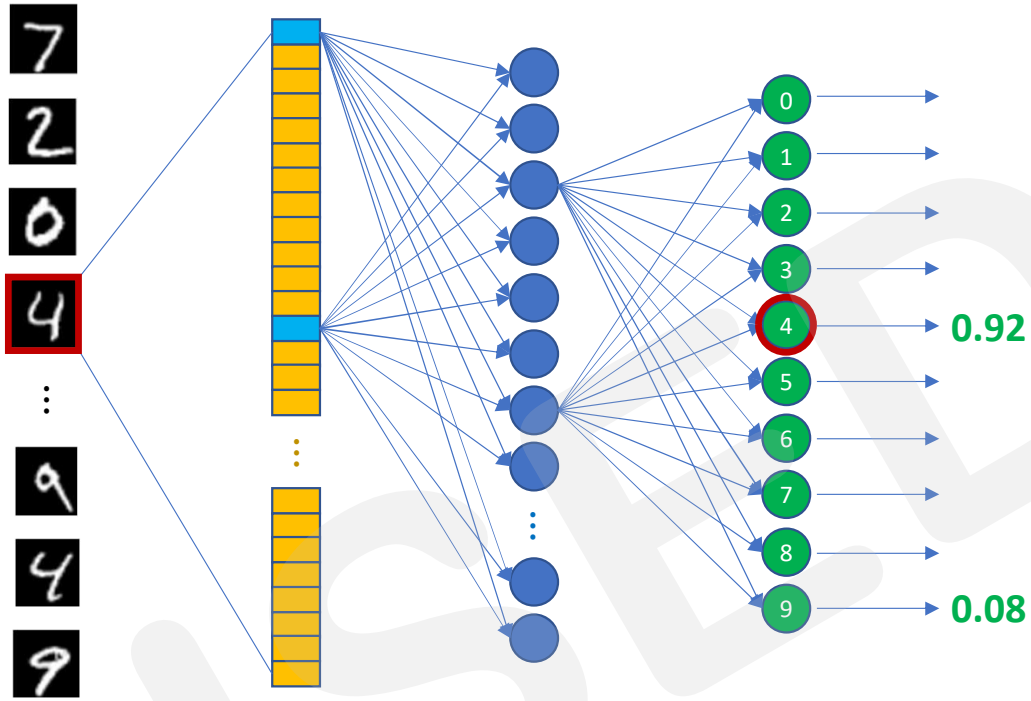
28 x 28 x 1 Graustufenbild: 28 * 28 * 1 = 784 Pixel

...

28 x 28 x 3 Farbbild (RGB): 28 * 28 * 3 = 2.352 Pixel

© 2023 USEDATA – Mag. Helmut Grillenberger

vollständig verbundenes neuronales Netzwerk

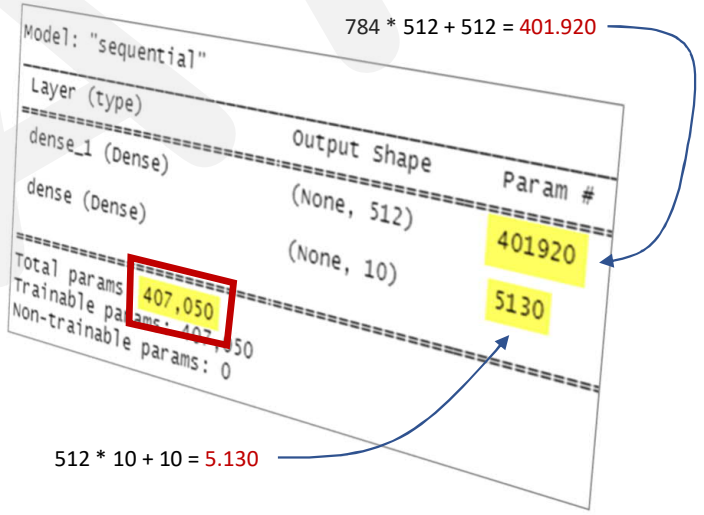


Graustufenbilder
60.000 x 28 x 28 x 1

input-layer
784 Input-Neuronen
Tensor

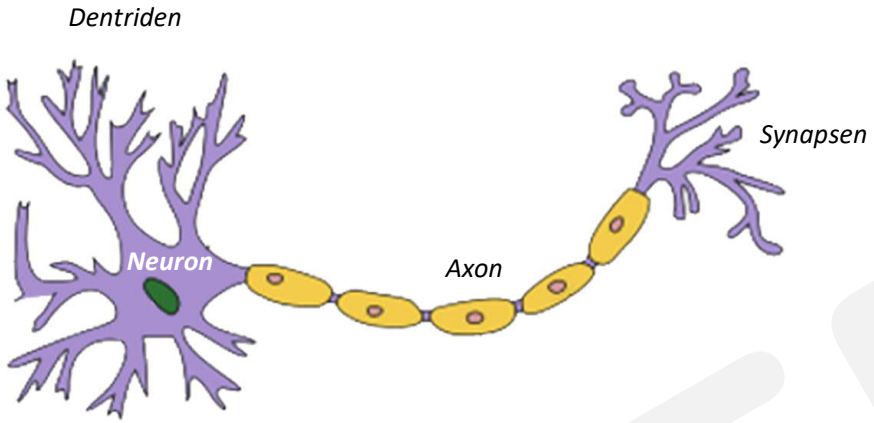
hidden-layer
512 Neuronen
Activation = “relu”

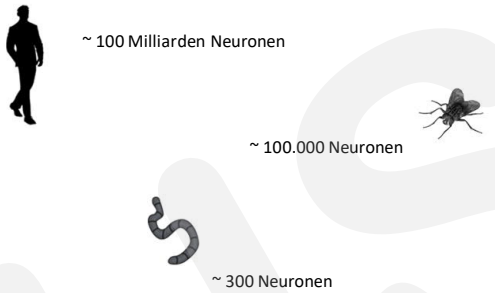
output-layer
10 Output-Neuronen
Activation = “softmax”

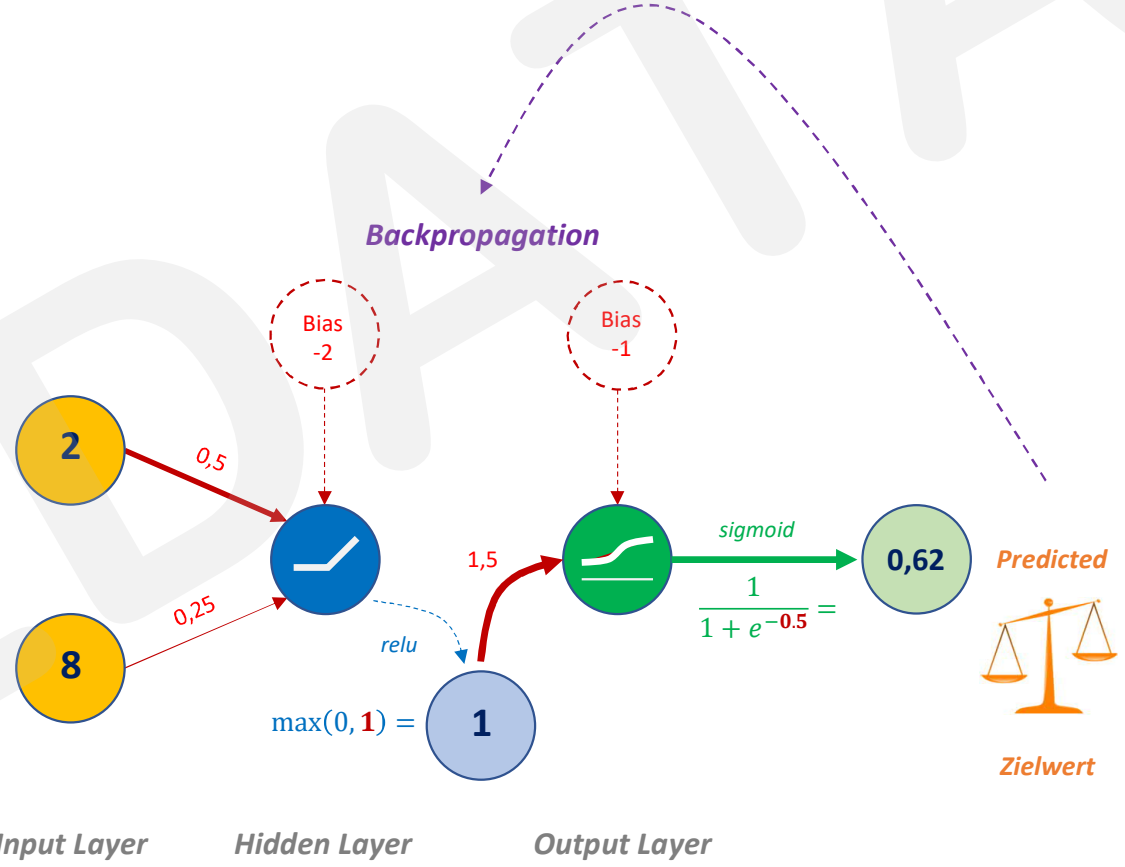


© 2023 USEDATA – Mag. Helmut Grillenberger

wie ticken neuronale Netze

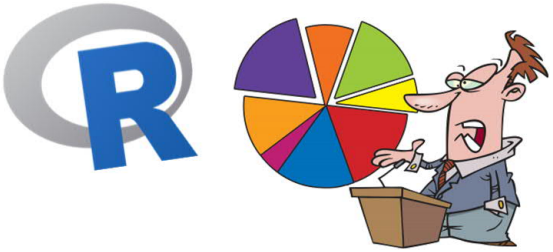




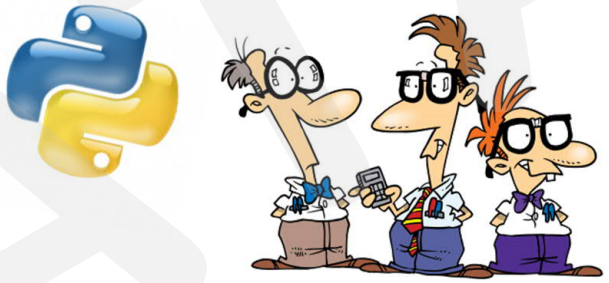


USEDATA

Sprachen zur Umsetzung von KI-Projekten



R wurde mit Blick auf den ***Statistiker*** entwickelt und ist die Sprache vieler ***Fachbereiche***.



Phyton wurde mit Blick auf den ***Programmierer*** entwickelt und ist für sehr allgemeine Aufgaben einsetzbar

beide Produkte sind …

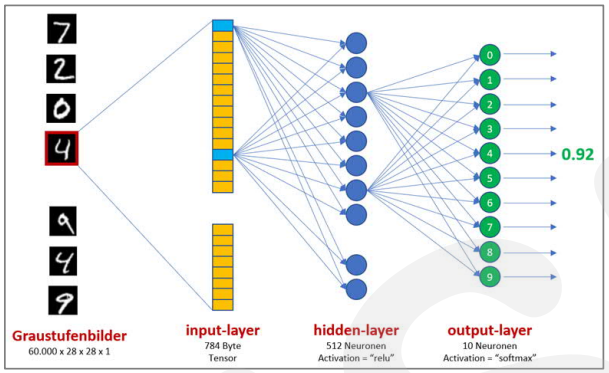
… die erste Wahl für den Einsatz ***moderner KI-Techniken***

… ***kostenlos*** *und* ***kommerziell frei*** *nutzbar*

USEDATA

© 2023 USEDATA – Mag. Helmut Grillenberger

erster Schritt: Modell definieren




```r
# Netz Architektur ------------------------------------------------------------

model <- keras_model_sequential() %>%
  layer_dense(units = 512, activation = "relu", input_shape = c(28 * 28)) %>%
  layer_dense(units = 10, activation = "softmax")
```

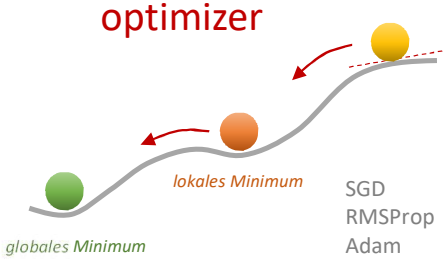
```python
# Netz Architektur ------------------------------------------------------------

model = models.Sequential()
model.add(layers.Dense(512, activation="relu", input_shape=(28 * 28,)))
model.add(layers.Dense(10, activation="softmax"))
```

USEDATA

© 2023 USEDATA – Mag. Helmut Grillenberger

zweiter Schritt: Hyperparameter konfigurieren

optimizer



SGD
RMSProp
Adam

loss

mse
binary_crossentropy
categorical_crossentropy

metrics

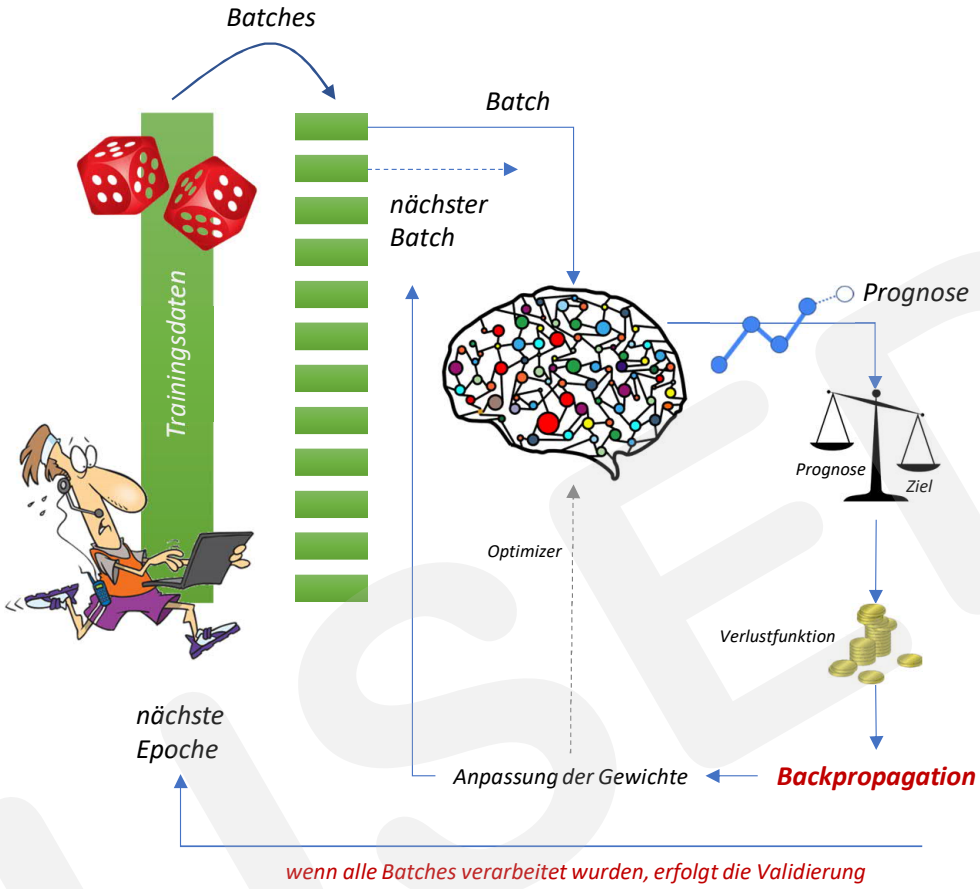
accuracy

```r
# Netz Kompilierung ----------------------------------------------------------

model %>% compile(
  optimizer = "rmsprop",
  loss = "categorical_crossentropy",
  metrics = c("accuracy")
)
```

```python
# Netz Kompilierung ----------------------------------------------------------

model.compile(
    optimizer="rmsprop",
    loss="categorical_crossentropy",
    metrics=["accuracy"])
```

© 2023 USEDATA – Mag. Helmut Grillenberger

dritter Schritt: Modell trainieren

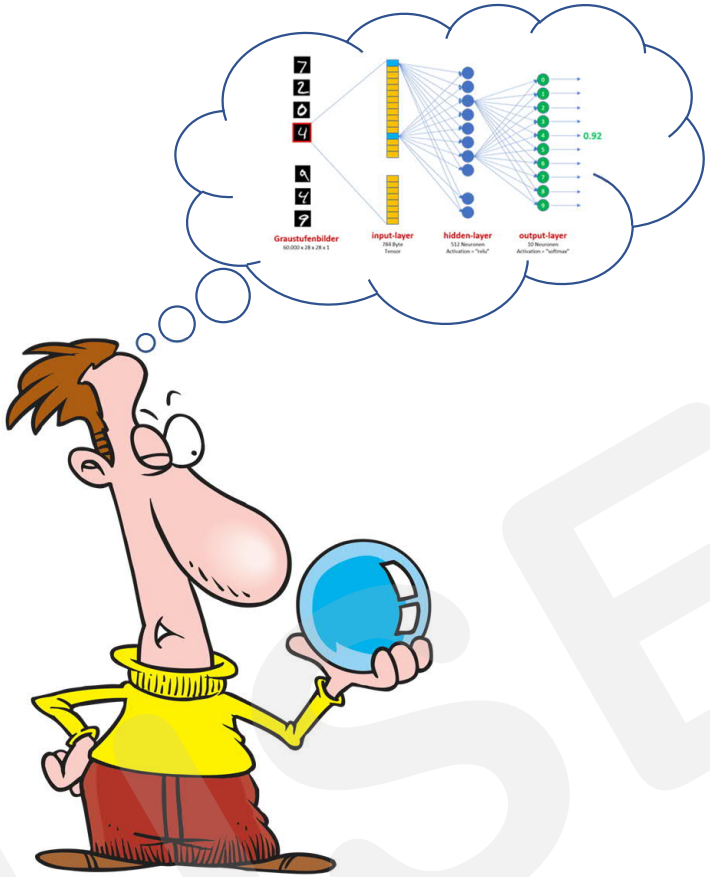



```r
# Netz trainieren ----------------
model %>% fit(
  train_images, train_labels,
  epochs = 5, batch_size = 128
)
```

```python
# Netz trainieren ----------------
model.fit(
    train_images, train_labels,
    epochs=5, batch_size=128)
```

© 2023 USEDATA – Mag. Helmut Grillenberger

vierter Schritt: Modell testen

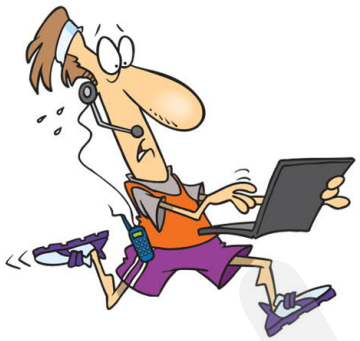


```r
# Netz testen ----------------------------------------------

results <- model %>% evaluate(test_images, test_labels)
```

```python
# Netz testen ----------------------------------------------

test_loss, test_acc = model.evaluate(test_images, test_labels)
```

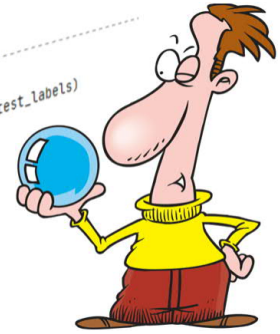
© 2023 USEDATA – Mag. Helmut Grillenberger

vier Anweisungen genügen



```r
# Netz Architektur ----------------------------------------------------

model <- keras_model_sequential() %>%
  layer_dense(units = 512, activation = "relu", input_shape = c(28 * 28)) %>%
  layer_dense(units = 10, activation = "softmax")

# Netz Kompilierung ---------------------------------------------------

model %>% compile(
  optimizer = "rmsprop",
  loss = "categorical_crossentropy",
  metrics = c("accuracy")
)

# Netz trainieren -----------------------------------------------------

model %>% fit(
  train_images, train_labels,
  epochs = 5, batch_size = 128
)

# Netz testen ---------------------------------------------------------

results <- model %>% evaluate(test_images, test_labels)
```

```python
# Netz Architektur ----------------------------------------------------

model = models.Sequential()
model.add(layers.Dense(512, activation="relu", input_shape=(28 * 28,)))
model.add(layers.Dense(10, activation="softmax"))

# Netz Kompilierung ---------------------------------------------------

model.compile(
    optimizer="rmsprop",
    loss="categorical_crossentropy",
    metrics=["accuracy"])

# Netz trainieren -----------------------------------------------------

model.fit(
    train_images, train_labels,
    epochs=5, batch_size=128)

# Netz testen ---------------------------------------------------------

test_loss, test_acc = model.evaluate(test_images, test_labels)
```



USEDATA

Hands on

MNIST,
das **Hello World** Beispiel
in der Welt des Deep Learning!

USEDATA

© 2023 USEDATA – Mag. Helmut Grillenberger

Danke für Ihr Interesse

Ihr Helmut Grillenberger

Festnetz: ***+43 6274 20804***
Mobile: ***+43 664 4212247***
E-Mail: *helmut.grillenberger@usedata.com*
URL: *https://www.**usedata.com***

Inhalt © 2023 USEDATA - Mag. Helmut Grillenberger
Cartoons wurden über clipart.com lizensiert