```
install.packages("keras3")
install.packages("jpeg")
```

```
Installing package into ‘/usr/local/lib/R/site-library’
(as ‘lib’ is unspecified)

also installing the dependencies ‘RcppTOML’, ‘here’, ‘png’, ‘config’, ‘tfautograph’, ‘reticulate’, ‘tensorflow’, ‘tfruns


Installing package into ‘/usr/local/lib/R/site-library’
(as ‘lib’ is unspecified)
```

```
library(keras3)
library(jpeg)
```

```
#MNIST Dataset: 28 x 28 pixel grayscale digits (x) labelled (y)
mnist <- dataset_mnist()
```

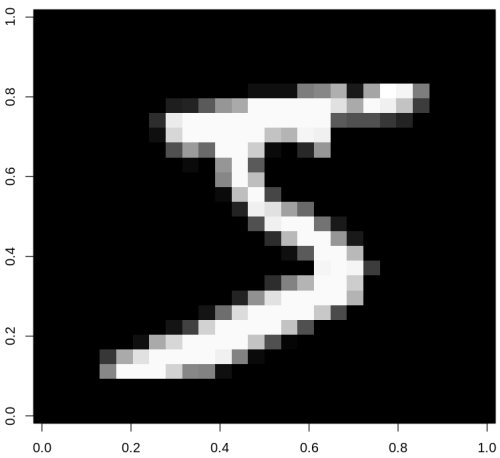
```
# show digits
# That how the pics look like:
for (i in 1:5) {
  rotate <- t(apply(mnist$train$x[i,,], 2, rev))
  image(rotate,col = grey(seq(0, 1, length = 256)))
  Sys.sleep(1)
}

# Training und Test-sets:
x_train <- mnist$train$x
y_train <- mnist$train$y

x_test <- mnist$test$x
y_test <- mnist$test$y

# reshaping x (28 x 28) into one dimensional vectors:
# matrix with number of samples x ?
dim(x_train) <- c(nrow(x_train), 784)
dim(x_test) <- c(nrow(x_test), 784)

# rescale from 0 - 255 to 0 - 1
x_train <- x_train / 255
x_test <- x_test / 255

# y data = 0 - 9
# we want a "one-hot encoded vector"
y_train <- to_categorical(y_train, 10)
y_test <- to_categorical(y_test, 10)
```

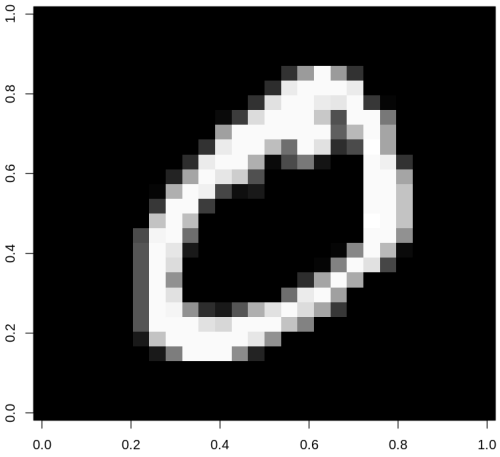
```
## The model
# sequential model... Layer after layer
# pipe operator add layer
model <- keras_model_sequential()
model %>%
  layer_dense(units = 256, activation = "relu", input_shape = c(784)) %>%
  layer_dropout(rate = 0.4) %>%
  layer_dense(units = 128, activation = "relu") %>%
  layer_dropout(rate = 0.3) %>%
  layer_dense(units = 128, activation = "relu") %>%
  layer_dropout(rate = 0.3) %>%
  layer_dense(units = 10, activation = "softmax")

summary(model)
# Dense: everything is connected
# units = dimensionality of the output space
# activation = relu, softmax, linear... etc...
# input shape..
# randomly setting a fraction rate of input units to 0
# at each update during training time, which helps prevent overfitting.
# softmax: logistic function
```

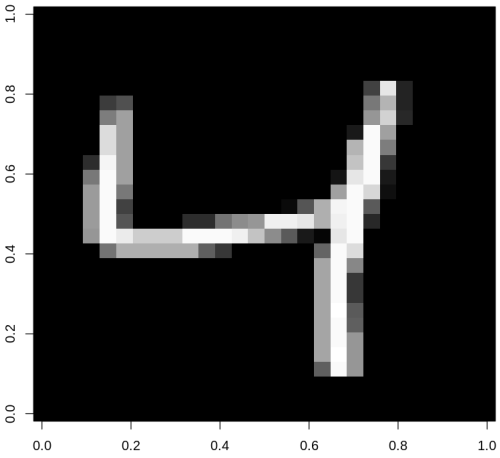
```
# loss: "categorical_crossentropy"
# optimizer: how the network is trained
```

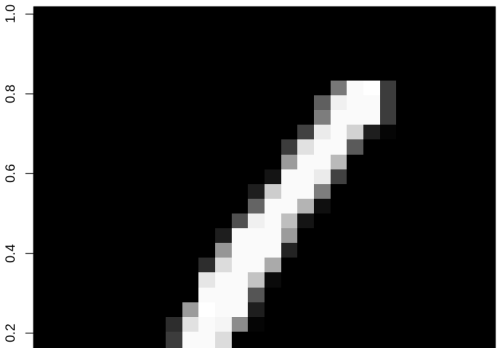
```
model %>% compile(
  optimizer = "rmsprop",             # network will update itself based on the training data & loss
  loss = "categorical_crossentropy", # measure mismatch between y_pred and y, calculated after each minibatch
```

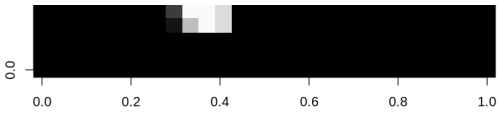
```
  metrics = c("accuracy")             # measure of performace - correctly classified images
)


history <- model %>% fit(
  x_train, y_train,
  epochs = 30, batch_size = 128,
  validation_split = 0.2
)

# training, validation, test
model %>% evaluate(x_test, y_test,verbose = 0)
```


0.0
0.2
0.4
0.6
0.8
1.0


0.0
0.2
0.4
0.6
0.8
1.0


0.0
0.2
0.4
0.6
0.8
1.0


0.2
0.4
0.6
0.8
1.0




```
Model: "sequential"
```

Layer (type)	Output Shape	Param #