Simple NN

young

2019 7 9

This is just for tasting how NN works. This is for test docs. This example is popular one but the point that i do analysis by myself is meaningful.

prepare dataset

```
library(keras)
data=dataset_mnist()

train_x=data$train$x
train_y=data$train$y
test_x=data$test$x
test_y=data$test$y
```

If you fail to load dataset with warning that contains 'allow_pickle=False', check .py file and change load(path) -> load(path, allow_pickle=True)

preprocessing for NN

```
train_x=array_reshape(train_x,c(60000,28*28))/255
test_x=array_reshape(test_x,c(10000,28*28))/255
```

set model & compile

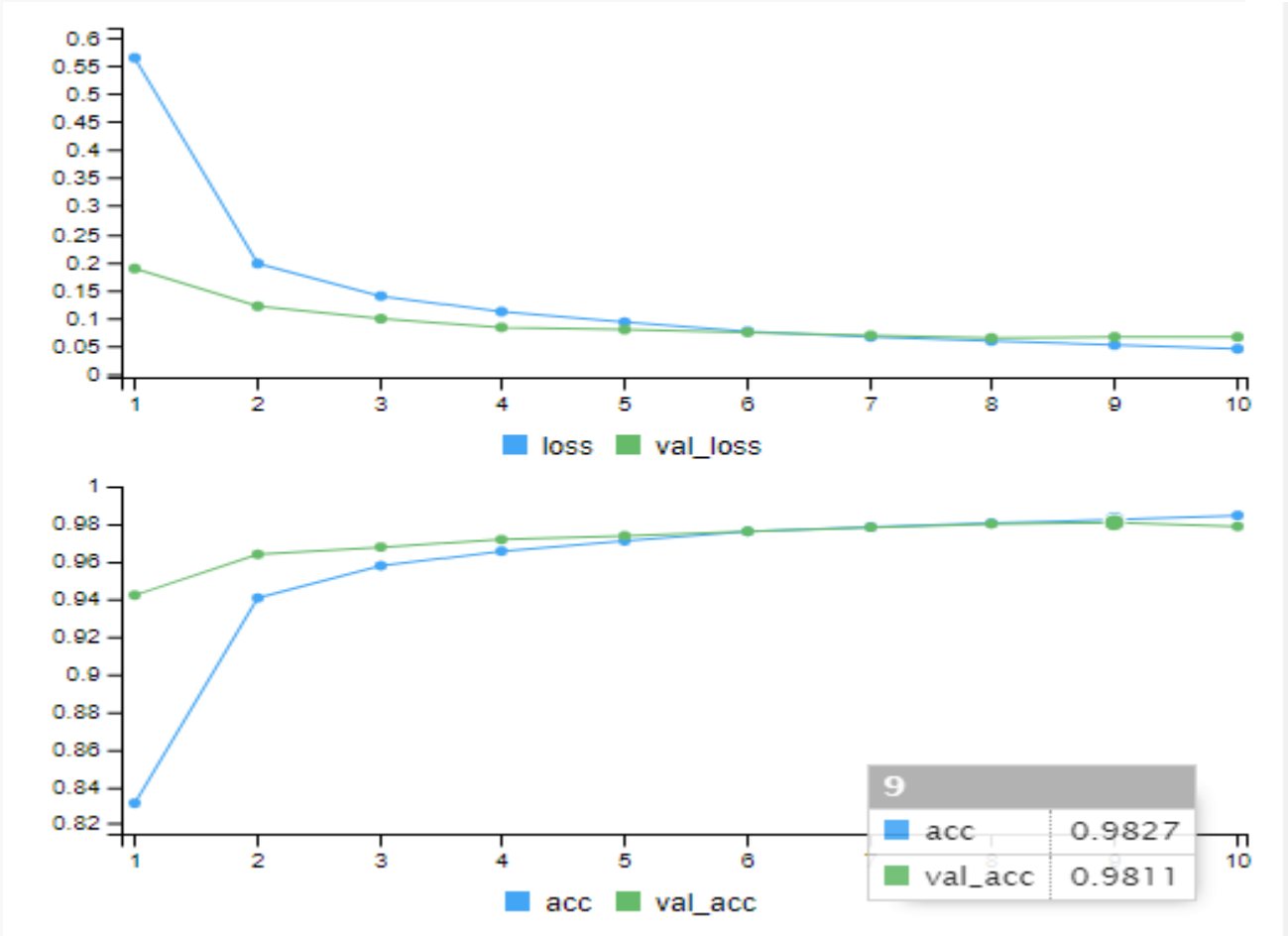
```
model=keras_model_sequential() %>%
  layer_dense(units=256,activation='relu',input_shape=c(28*28)) %>%
  layer_dropout(rate=0.2) %>%
  layer_dense(units=128,activation='relu') %>%
  layer_dropout(rate=0.2) %>%
  layer_dense(128,activation='relu') %>%
  layer_dense(units=10,activation='softmax')

model %>% compile(
  optimizer='adam',
  loss='sparse_categorical_crossentropy',
  metrics=c('accuracy')
)
```

check model

```
history=model %>% fit(train_x,train_y,
                      epoch=10,
                      batch_size=512,
                      validation_data=list(test_x,test_y))
```

```
Train on 60000 samples, validate on 10000 samples
Epoch 1/10
60000/60000 [==============================] - 2s 25us/sample - loss: 0.5626
- acc: 0.8310 - val_loss: 0.1885 - val_acc: 0.9423
Epoch 2/10
60000/60000 [==============================] - 1s 21us/sample - loss: 0.1974
- acc: 0.9408 - val_loss: 0.1216 - val_acc: 0.9641
Epoch 3/10
60000/60000 [==============================] - 1s 18us/sample - loss: 0.1396
- acc: 0.9580 - val_loss: 0.0994 - val_acc: 0.9680
Epoch 4/10
60000/60000 [==============================] - 1s 19us/sample - loss: 0.1120
- acc: 0.9658 - val_loss: 0.0841 - val_acc: 0.9720
```



With very simple NN, I can get 97% accuracy.