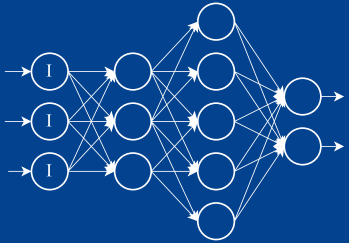
AARHUS UNIVERSITY
SCHOOL OF ENGINEERING

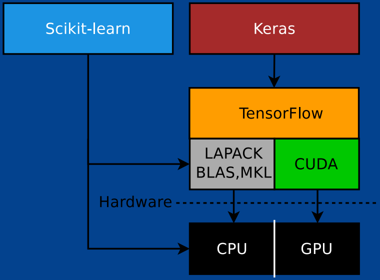
LESSON 7: Neural Networks II

CARSTEN EIE FRIGAARD

AUTUMN 2021

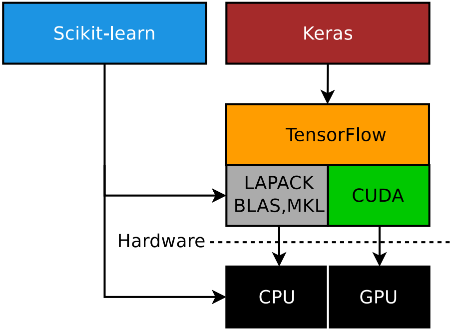


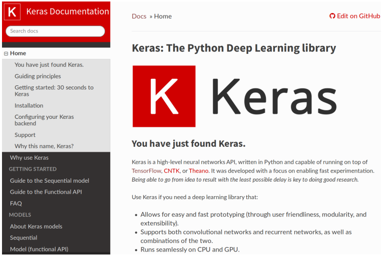
IMPLEMENTING MLPS IN KERAS



Keras and Tensorflow

Using the Keras API instead of Scikit-learn or TensorFlow



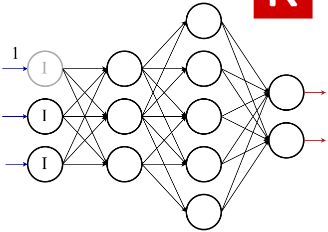


NOTE:

- documentation: `https://keras.io/`
- keras provides a `fit-predict`-interface,
- many similiarities to Scikit-learn,
- but also many differences!

Building Keras MLPs

Using the Keras `Sequential` class, programatical build up model:




```
# Build Keras model
model = Sequential()
model.add(Dense(input_dim=2, units=3, activation="tanh", ..)
model.add(Dense(units=5, activation="relu", ..)
model.add(Dense(units=2, activation="softmax"))
.
.
X_train, .. = train_test_split(X, y, .. )
.
.
y_train_categorical = to_categorical(y_train, num_classes=2)
y_test_categorical  = to_categorical(y_test,  num_classes=2)
.
.
history = model.fit(X_train, y_train_categorical, ..
.
.
score = model.evaluate(X_test, y_test_categorical)
```

Notes on Keras MLPs

Typical Keras MLP Supervised Classifier setup..

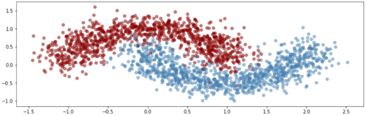
- loss function

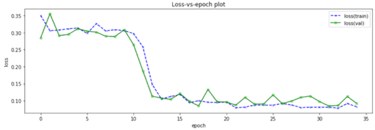
  `loss='categorical_crossentropy'`

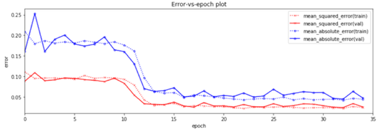
- metrics collected via history

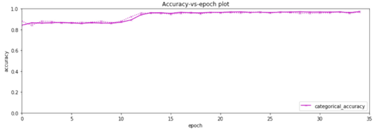
  ```
  metrics=[
    'categorical_accuracy',
    'mean_squared_error',
    'mean_absolute_error'])
  ```

- input lay.: categorical encoding,
- output lay.: softmax function,









And notice that Keras do *not* provide metrics like

`precision, recall, F1`

but instead

`categorical_accuracy, binary_accuracy`

Input Layer: Categorical Encoding

For MLP Classification

One-hot `to_categorical(·)` encoding in Keras:

- input layer: one-hot class encoding,
- output layer: one neuron per output class that fires,
  and use `softmax` for output neurons
- beware of misformated classes.

```
import numpy as np
from keras.utils.np_utils import to_categorical

y = np.array([1, 2, 0, 4, -1])
y_cat = to_categorical(y)

print(y_cat)

#[[0. 1. 0. 0. 0.]  => i=0, class 1
# [0. 0. 1. 0. 0.]  => i=1, class 2
# [1. 0. 0. 0. 0.]  => i=2, class 0
# [0. 0. 0. 0. 1.]  => i=3, class 4
# [0. 0. 0. 0. 1.]] => i=4, also class 4!
#                      NOTE: no class 3
```

[L07/Extra/keras_to_categorical.ipynb]

Output Layer: Softmax Function

For MLP Classification: Assing a Probability for each Class

Softmax (softargmax/normalized exponential) definition

$$\mathrm{softmax}(\mathbf{x})_i = \frac{e^{x_i}}{\sum_{i=1}^{n} e^{x_i}}$$

`softmax`: smooth approx. of `argmax` function.

`argmax`: the index-of-the-max-value for some data.

```python
# python demo of softmax/argmax
x = np.array([1, 2, -4, 5, 1])
i = np.argmax(x)

PrintMatrix(x,"x = ")
print(f"np.argmax(x) = {np.argmax(x)}")

def softmax(x):
    z = np.exp(x)
    s = np.sum(z)
    return z / s

PrintMatrix(softmax(x),"softmax(x) = ")
print(f"np.argmax(softmax(x)) = {np.argmax(softmax(x))}")
```

```
# output
x = [ 1  2 -4  5  1 ]
np.argmax(x) = 3
softmax(x) = [0.02 0.05 0.  0.92 0.02]
np.argmax(softmax(x)) = 3
```

[L07/Extra/softmax.ipynb]

KERAS INSTALL PARTY

Installing Keras

L07/Extras/installing_keras.ipyb

jupyter installing_keras (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

Run Markdown

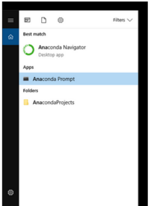
ITMAL Demo

Installing Keras

REVISIONS	
2018-0325	CEF, initial.
2020-0305	CEF, F20 ITMAL update.
2020-0306	CEF, investigated Anaconda 2019.10 on Windows and updated GPU server notes.
2021-1012	CEF, updated for ITMAL E21.

Install Keras via Anaconda Prompt

1: Launch the **anaconda prompt** console (CMD), via the Start menu



High-Performace-Computing (HPC)

Running on the ASE GPU cluster..

- Your group login: `itmal09-e21`
- Password (can be changed): `itmal09-e21_123`

Documentation (always-out-of-date):

- Brightspace | ITMAL | Kursusinfo.. | GPU Cluster
  [https://brightspace.au.dk/d2l/le/lessons/27524/topics/296678]

GPU CLUSTER CURRENTLY DOWN!

EXTRA SLIDES..

MLP Effect of Number of Hidden Layers

How Many Hidden Layers?

MNIST Search Quest Exercise:

- ITMAL Grp10:
  used a 20-50-70-100-70-50-20 layer `MLPClassifier`,
- found layers by trial-and-error,
- but what are the optimal hidden layer sizes and neurons per hidden layer?

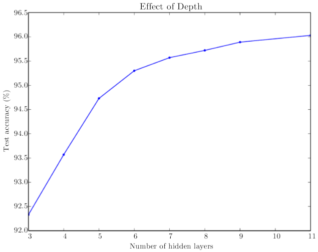


Figure 6.6: Empirical results showing that deeper networks generalize better when used to transcribe multi-digit numbers from photographs of addresses. Data from Goodfellow *et al.* (2014d). The test set accuracy consistently increases with increasing depth. See Fig. 6.7 for a control experiment demonstrating that other increases to the model size do not yield the same effect.

The ReLU Activation Function [DL]

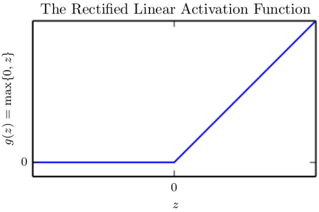


Figure 6.3: The rectified linear activation function. This activation function is the default activation function recommended for use with most feedforward neural networks. Applying this function to the output of a linear transformation yields a nonlinear transformation. However, the function remains very close to linear, in the sense that is a piecewise linear function with two linear pieces. Because rectified linear units are nearly linear, they preserve many of the properties that make linear models easy to optimize with gradient-based methods. They also preserve many of the properties that make linear models generalize well. A common principle throughout computer science is that we can build complicated systems from minimal components. Much as a Turing machine's memory needs only to be able to store 0 or 1 states, we can build a universal function approximator from rectified linear functions.

Effect of Depth [DL]

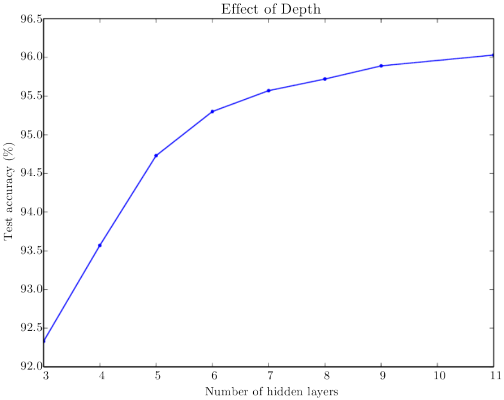


Figure 6.6: Empirical results showing that deeper networks generalize better when used to transcribe multi-digit numbers from photographs of addresses. Data from Goodfellow *et al.* (2014d). The test set accuracy consistently increases with increasing depth. See Fig. 6.7 for a control experiment demonstrating that other increases to the model size do not yield the same effect.

Effect of Number of Parameters [DL]

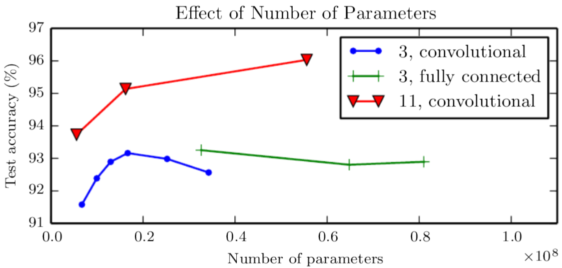


Figure 6.7: Deeper models tend to perform better. This is not merely because the model is larger. This experiment from Goodfellow *et al.* (2014d) shows that increasing the number of parameters in layers of convolutional networks without increasing their depth is not nearly as effective at increasing test set performance. The legend indicates the depth of network used to make each curve and whether the curve represents variation in the size of the convolutional or the fully connected layers. We observe that shallow models in this context overfit at around 20 million parameters while deep ones can benefit from having over 60 million. This suggests that using a deep model expresses a useful preference over the space of functions the model can learn. Specifically, it expresses a belief that the function should consist of many simpler functions composed together. This could result either in learning a representation that is composed in turn of simpler representations (e.g., corners defined in terms of edges) or in learning a program with sequentially dependent steps (e.g., first locate a set of objects, then segment them from each other, then recognize them).

Deep feedforward networks [DL]

Deep feedforward networks, also often called feedforward neural networks, or multi- layer perceptrons (MLPs), are the quintessential deep learning models.

[DL, p167]