Keras Library for Neural Networks and Deep Learning

Alex Klibisz

Oak Ridge National Lab (Intern)

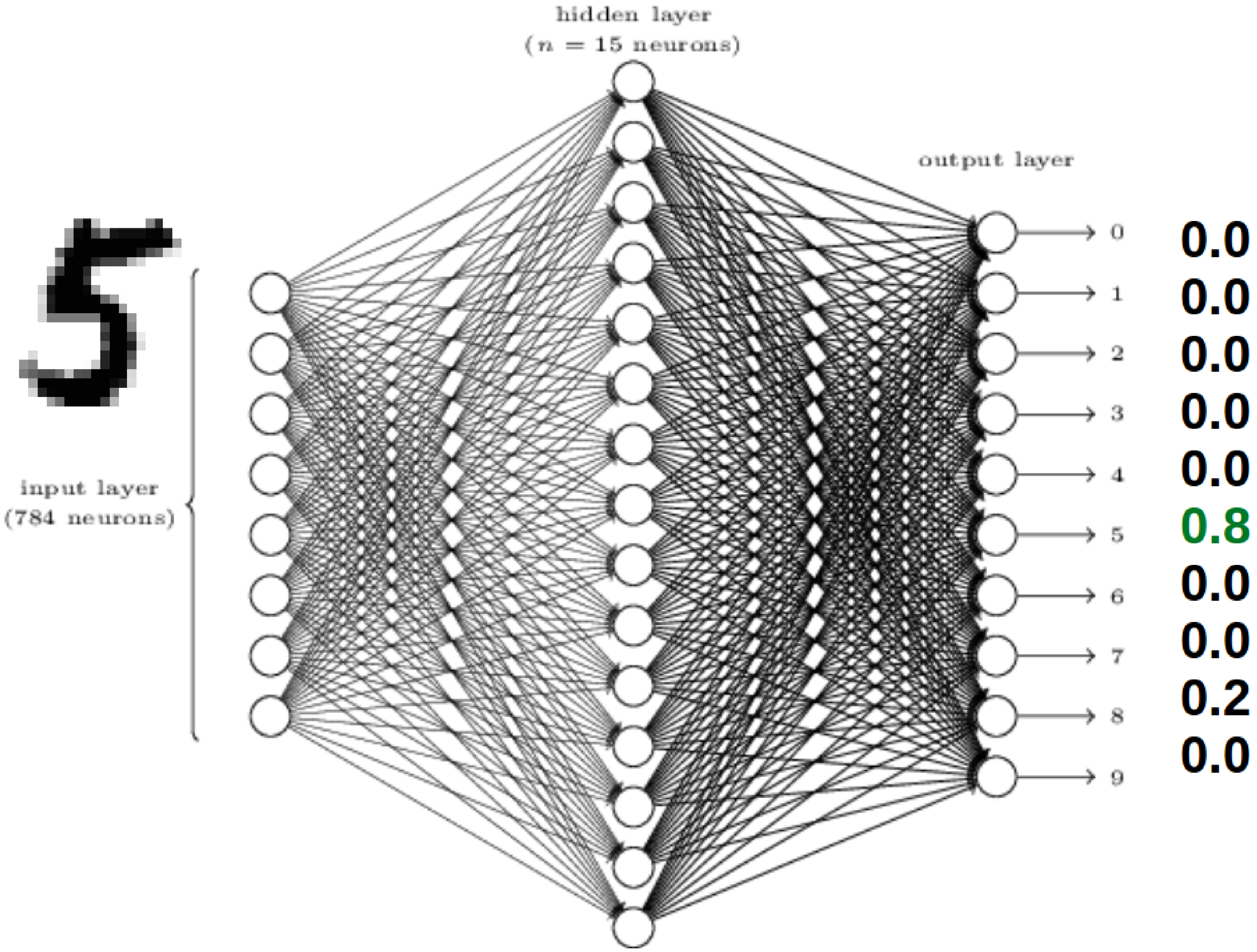
alex.klibisz.com, github.com/alexklibisz

Outline

- Neural networks introduction
- Keras Overview
    - Network Structure
    - Optimization
    - Training
    - Callbacks
    - Advanced Features
- Learning Resources
- Real Dataset Example (time-permitting)

Neural Networks Introduction

- Motivation: model high-dimensional data (images, video, audio, text) with minimal manual feature engineering.
- Layer-wise computation: input -> hidden -> output layers
- Training optimizes an objective function.
- Generally, more data helps.


hidden layer
(n = 15 neurons)
input layer
(784 neurons)
output layer
0
1
2
3
4
5
6
7
8
9
0.0
0.0
0.0
0.0
0.0
0.8
0.0
0.0
0.2
0.0

Common Terms

- Layer: take inputs, compute outputs, pass to next layer.
    - Convolutional: sliding filters amplify certain input features.
    - Max-pooling: take max of every h×w input window.
    - Dense: every node computes function over *all* inputs
    - Activation: apply non-linearity to inputs.
- Parameters: learnable weights in each layer.
- Cost: quantify error in predict vs. correct output.
- Training: update parameters to minimize cost.
- Batch: subset of training data used to update parameters.
- Epoch: approx. one pass through all training data.

```
while loss > 0:

  y_pred = network(X, weights)
  loss = (y_true - y_pred)^2
  weights = optimize(weights, loss)
```

Example: Image Classification

MNIST ConvNet Demo

Keras Overview

- Python
- Abstractions for layers, cost functions, optimization, etc.
- Similar level to Scikit-learn.
- Keras "front-end", Tensorflow/Theano/CNTK "back-end".

Why Keras?

- Python (R bindings)
- Clean abstractions → fast prototyping.
- GPU + CPU support.
- Thorough documentation, examples.
- Implement and train simple to state-of-the art networks.

Network Structure

- 28x28x1 image input → 10 class output.

```
model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3),
                 activation='relu',
                 input_shape=(28, 28, 1)))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(10, activation='softmax'))
```

Optimization

- Minimize $\mathrm{mean}((\mathrm{true} - \mathrm{predicted})^2)$
- Monitor accuracy during training.

```
model.compile(loss='mean_squared_error',
              optimizer=Adam(lr=0.001),
              metrics=['accuracy'])
```

Training

```
model.fit(x_train, y_train,
          batch_size=32,
          epochs=100,
          validation_data=(x_test, y_test))
```

```
60000/60000 [==============================] - 7s - loss:
Epoch 2/12
60000/60000 [==============================] - 5s - loss:
Epoch 3/12
55808/60000 [==========================>...] - ETA: 0s -
```

Callbacks

- Functions executed before/after training, epochs, batches.
- Saving metrics and weights, learning rate adjustment, + more

```
model.fit(x_train, y_train, ...,
          callbacks=[CSVLogger('metrics.csv')])
```

Advanced Features

- Functional API, Inception Example
- Multi-input/output models
- Training with generators

Learning Resources

Ordered easy to difficult

- Data Skeptic Mini Episodes
- ML Mastery E-books (Scikit-learn, Keras, XGBoost)
- PyImageSearch (image processing, deep learning)
- Neural Networks and Deep Learning (Nielsen)
- Stanford CS231n (try the assignments)
- Deep Learning Book (Advanced)

Good math review

- Mathematics for Political and Social Research

Time Left? - ISBI 2012 Segmentation Challenge

- Ground-truth Masks
- U-Net Architecture
- Keras U-Net Implemenatation
- Results

Thanks

- Website: alex.klibisz.com
- Github: https://github.com/alexklibisz