With over 600 specialists and IT experts in your region.

- 14 Trivadis branches and more than 600 employees
- 200 Service Level Agreements
- Over 4,000 training participants
- Research and development budget: CHF 5.0 million
- Financially self-supporting and sustainably profitable
- Experience from more than 1,900 projects per year at over 800 customers
About me

My background
• from psychology/statistics via software development and database engineering to
data science/ML/DL

My passion
• machine learning and deep learning
• data science and (Bayesian) statistics
• explanation/understanding over prediction accuracy

Where to find me
• blog: http://recurrentnull.wordpress.com
• twitter: @zkajdan
Why deep learning?
Task: Separate the green triangles from the blue circles

Source: Goodfellow et al., Deep Learning, 2016
Features matter...

... so, can't we just hard code them?

Source: Parkhi et al., The Truth about Cats and Dogs
Example: Learning features for object classification

Source: Goodfellow et al., Deep Learning, 2016
Deep learning vs. machine learning

Pre-ML programs

in: data
in: rules
out: conclusion

Machine Learning (ML)

in: data
learn: mappings/functions
out: conclusion

Deep Learning (DL)

in: data
learn: features
learn: mappings/functions
out: conclusion
How does a deep neural network learn?
A deep neural network

Source: Stergiou & Siganos, Artificial Neurons
First, we need a loss function

… that is, a way to quantify our current error.

\[-\sum_j t_j \log(y_j)\]

\[\frac{1}{n} \sum_n (\hat{y} - y)^2\]

You said 98% probability it was a cat… but it’s a dog!

You said sales were gonna to up by 5%... but really they went down by 0.7%!
Second, we need to reduce that error

For $x < 0$, we have $f'(x) < 0$, so we can decrease $f$ by moving rightward. For $x > 0$, we have $f'(x) > 0$, so we can decrease $f$ by moving leftward.

Global minimum at $x = 0$. Since $f'(x) = 0$, gradient descent halts here.

$f(x) = \frac{1}{2} x^2$

$f'(x) = x$

Source: Goodfellow et al., Deep Learning, 2016
Third, we need to *propagate back* that error

... all through the network.

\[
\frac{\partial z}{\partial w} = \frac{\partial z}{\partial y} \frac{\partial y}{\partial x} \frac{\partial x}{\partial w} = f'(y)f'(x)f'(w) = f'(f(f(w)))f'(f(w))f'(w)
\]

Source: Goodfellow et al., Deep Learning, 2016
So how can we do that ourselves?
Deep Learning frameworks

- Caffe
- Keras
- PyTorch
- TensorFlow
- DL4J
“Being able to go from idea to result with the least possible delay is key to doing good research. “

- High level library running on top of TensorFlow, CNTK, or Theano
- Extensive documentation and lots of sample code
- Includes layers for all standard architectures (different kinds of CNNs and RNNs)

```python
model = Sequential()
model.add(Dense(units=64, activation='relu', input_dim=100))
model.add(Dense(units=10, activation='softmax'))
model.compile(loss='categorical_crossentropy', optimizer='sgd', metrics=['accuracy'])
model.fit(x_train, y_train, epochs=50)
```

```
classes = model.predict(x_test)
```
TensorFlow: Google Glamour included

- Probably the best-known deep learning framework outside the DL community
- Originally developed by the Google Brain team
- Super-fast release cycle
- Core framework not the easiest to use, but comes with several higher-level APIs on top

```
W = tf.Variable([.3], dtype=tf.float32)
b = tf.Variable([- .3], dtype=tf.float32)
x = tf.placeholder(tf.float32)
linear_model = W*x + b
y = tf.placeholder(tf.float32)
squared_deltas = tf.square(linear_model - y)
loss = tf.reduce_sum(squared_deltas)
init = tf.global_variables_initializer()
sess = tf.Session()
sess.run(init)
print(sess.run(loss, {x: [1, 2, 3, 4], y: [0, -1, -2, -3]}))
```
PyTorch: Dynamical neural networks with Python

- Torch (Lua) ported to Python
- Developed at and open-sourced by Facebook
- Fast on-GPU tensor computations (alternative to numpy)
- Dynamical computation graphs
- Rapidly gaining popularity

```python
model = Linear(input_dim, output_dim)
loss_fn = torch.nn.MSELoss()
optimizer = torch.optim.Adam(model.parameters(), lr=learning_rate)
for epoch in range(num_epochs):
    y_pred = model(X_train)
    loss_var = loss_fn(y_pred, y_train)
    optimizer.zero_grad()
    loss_var.backward()
    optimizer.step()
```
Case study: Distinguishing sports

- 6 classes, ~8000 images in training set, ~1800 in test set
- 4-block CNN in Keras, (raw) TensorFlow and PyTorch
Example CNN architecture (LeNet)
The Convolution Operation

Source: Goodfellow et al., Deep Learning, 2016
Demo time!

Notebooks:

- [https://github.com/skeydan/deeplearning_keras_tf_pytorch/blob/master/cnn_keras.ipynb](https://github.com/skeydan/deeplearning_keras_tf_pytorch/blob/master/cnn_keras.ipynb)
- [https://github.com/skeydan/deeplearning_keras_tf_pytorch/blob/master/cnn_tensorflow.ipynb](https://github.com/skeydan/deeplearning_keras_tf_pytorch/blob/master/cnn_tensorflow.ipynb)
Questions? Thank you!