## Deep learning applications

<table>
<thead>
<tr>
<th>Vision</th>
<th>Speech</th>
<th>Text</th>
<th>Other</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Search &amp; information extraction</td>
<td>• Interactive voice response (IVR) systems</td>
<td>• Search and ranking</td>
<td>• Recommendation engines</td>
</tr>
<tr>
<td>• Security/Video surveillance</td>
<td>• Voice interfaces (Mobile, Cars, Gaming, Home)</td>
<td>• Sentiment analysis</td>
<td>• Advertising</td>
</tr>
<tr>
<td>• Self-driving cars</td>
<td>• Security (speaker identification)</td>
<td>• Machine translation</td>
<td>• Fraud detection</td>
</tr>
<tr>
<td>• Medical imaging</td>
<td>• Health care</td>
<td>• Question answering</td>
<td>• AI challenges</td>
</tr>
<tr>
<td>• Robotics</td>
<td>• Simultaneous interpretation</td>
<td></td>
<td>• Drug discovery</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Sensor data analysis</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Diagnostic support</td>
</tr>
</tbody>
</table>
Deep learning ecosystem

**Software**
- Caffe
- Keras
- theano
- torch
- KALDI
- Caffe2
- CNTK
- mxnet
- TensorFlow
- Chainer

**Hardware**
- NVIDIA
- Intel
- ARM
- Google
- XILINX
- Movidius
- AMD
- Qualcomm
How to pick the right hardware/software stack?

Does one size fit all?
Applications break down

- **Images**
  - Tissue classification in medical images
- **Video**
  - Video surveillance
- **Speech**
  - Speech recognition
- **Text**
  - Sentiment analysis
- **Sensor**
  - Predictive maintenance
- **Other**
  - Fraud detection

### Detection
Look for a known object/pattern

### Generation
Generate content

### Classification
Assign a label from a predefined set of labels

### Anomaly detection
Look for abnormal, unknown patterns
Types of artificial neural networks
Topology to fit data characteristics

Images:
Convolutional (CNN)

Speech, time series, sequences:
Fully Connected (FC), Recurrent (RNN)
One size does NOT fit all

Application

Data type

Data size

Model (topology of artificial neural network):
- How many layers
- How many neurons per layer
- Connections between neurons (types of layers)
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Model size (# params)</th>
<th>Model size (MB)</th>
<th>GFLOPs (forward pass)</th>
</tr>
</thead>
<tbody>
<tr>
<td>AlexNet</td>
<td>CNN</td>
<td>60,965,224</td>
<td>233 MB</td>
<td>0.7</td>
</tr>
<tr>
<td>GoogleNet</td>
<td>CNN</td>
<td>6,998,552</td>
<td>27 MB</td>
<td>1.6</td>
</tr>
<tr>
<td>VGG-16</td>
<td>CNN</td>
<td>138,357,544</td>
<td>528 MB</td>
<td>15.5</td>
</tr>
<tr>
<td>VGG-19</td>
<td>CNN</td>
<td>143,667,240</td>
<td>548 MB</td>
<td>19.6</td>
</tr>
<tr>
<td>ResNet50</td>
<td>CNN</td>
<td>25,610,269</td>
<td>98 MB</td>
<td>3.9</td>
</tr>
<tr>
<td>ResNet101</td>
<td>CNN</td>
<td>44,654,608</td>
<td>170 MB</td>
<td>7.6</td>
</tr>
<tr>
<td>ResNet152</td>
<td>CNN</td>
<td>60,344,387</td>
<td>230 MB</td>
<td>11.3</td>
</tr>
<tr>
<td>Eng Acoustic Model</td>
<td>RNN</td>
<td>34,678,784</td>
<td>132 MB</td>
<td>0.035</td>
</tr>
<tr>
<td>TextCNN</td>
<td>CNN</td>
<td>151,690</td>
<td>0.6 MB</td>
<td>0.009</td>
</tr>
</tbody>
</table>
## Popular models

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Model size (# params)</th>
<th>Model size (MB)</th>
<th>GFLOPs (forward pass)</th>
</tr>
</thead>
<tbody>
<tr>
<td>AlexNet</td>
<td>CNN</td>
<td>60,965,224</td>
<td>233 MB</td>
<td>0.7</td>
</tr>
<tr>
<td>GoogleNet</td>
<td>CNN</td>
<td>6,998,552</td>
<td>27 MB</td>
<td>1.6</td>
</tr>
<tr>
<td>VGG-16</td>
<td>CNN</td>
<td>138,357,544</td>
<td>528 MB</td>
<td>15.5</td>
</tr>
<tr>
<td>VGG-19</td>
<td>CNN</td>
<td>143,667,240</td>
<td>548 MB</td>
<td>19.6</td>
</tr>
<tr>
<td>ResNet50</td>
<td>CNN</td>
<td>25,610,269</td>
<td>98 MB</td>
<td>3.9</td>
</tr>
<tr>
<td>ResNet101</td>
<td>CNN</td>
<td>44,654,608</td>
<td>170 MB</td>
<td>7.6</td>
</tr>
<tr>
<td>ResNet152</td>
<td>CNN</td>
<td>60,344,387</td>
<td>230 MB</td>
<td>11.3</td>
</tr>
<tr>
<td>Eng Acoustic Model</td>
<td>RNN</td>
<td>34,678,784</td>
<td>132 MB</td>
<td>0.035</td>
</tr>
<tr>
<td>TextCNN</td>
<td>CNN</td>
<td>151,690</td>
<td>0.6 MB</td>
<td>0.009</td>
</tr>
</tbody>
</table>
Compute requirements

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Model size (# params)</th>
<th>Model size (MB)</th>
<th>GFLOPs (forward pass)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResNet152</td>
<td>CNN</td>
<td>60,344,387</td>
<td>230 MB</td>
<td>11.3</td>
</tr>
</tbody>
</table>

Training data: 14M images (ImageNet)
FLOPs per epoch: $3 \times 11.3 \times 10^9 \times 14 \times 10^6 \approx 5 \times 10^{17}$
1 epoch per hour: ~140 TFLOPS

Today’s hardware:
- Google TPU2: 180 TFLOPS Tensor ops
- NVIDIA Tesla V100: 15 TFLOPS SP (30 TFLOPS FP16, 120 TFLOPS Tensor ops), 12 GB memory
- NVIDIA Tesla P100: 10.6 TFLOPS SP, 16 GB memory
- NVIDIA Tesla K40: 4.29 TFLOPS SP, 12 GB memory
- NVIDIA Tesla K80: 5.6 TFLOPS SP (8.74 TFLOPS SP with GPU boost), 24 GB memory
- INTEL Xeon Phi: 2.4 TFLOPS SP
Model parallelism

– Can be achieved with scalable distributed matrix operations
– Requires a certain compute/bandwidth ratio

Let’s assume:

\[ n \text{ – input size = batch size = output size} \]
\[ \gamma \text{ – compute power of the device (FLOPS)} \]
\[ \beta \text{ – bandwidth (memory or interconnect)} \]
\[ p^2 \text{ – number of compute devices} \]

\[ T_{\text{compute}} = \frac{2n^3}{p^2\gamma} \quad T_{\text{data read}} = \frac{2n^2}{p\beta} \]

\[ \beta \geq \frac{4p\gamma}{n} \text{ for FP32} \]

“SUMMA: Scalable Universal Matrix Multiplication Algorithm”, R.A. van de Geijn, J. Watts
Model parallelism

– Can be achieved with scalable distributed matrix operations
– Requires a certain compute/bandwidth ratio

Let’s assume:

\( n \) – input size = batch size = output size  
\( \gamma \) – compute power of the device (FLOPS)  
\( \beta \) – bandwidth (memory or interconnect)  
\( p^2 \) – number of compute devices

\[
T_{\text{compute}} = \frac{2n^3}{p^2\gamma} \quad T_{\text{data\_read}} = \frac{2n^2}{p\beta}
\]

\( \beta \geq \frac{4p\gamma}{n} \) for FP32

\[
n = 2000, \quad \gamma = 15 \text{ TFLOPS}
\]

\[
p = 10, \quad \beta \geq 300 \text{ GB/s}
\]

\[
p = 1, \quad \beta \geq 30 \text{ GB/s}
\]
Data parallelism

\[ T_{\text{compute}}(p, c, \gamma) = c / (p\gamma) \]
\[ T_{\text{communicate}}(p, w, \beta) = 2w\log(p) / \beta \]

- \( p \) – number of workers (nodes),
- \( \gamma \) – the computational power of the node,
- \( c \) – the computational complexity of the model,
- \( \beta \) – bandwidth,
- \( w \) – the size of the weights in bits.
Data parallelism

\[ T_{\text{compute}}(p, c, \gamma) = \frac{c}{p \gamma} \]
\[ T_{\text{communicate}}(p, w, \beta) = \frac{2w \log(p)}{\beta} \]

- \( p \) – number of workers (nodes),
- \( \gamma \) – the computational power of the node,
- \( c \) – the computational complexity of the model,
- \( \beta \) – bandwidth,
- \( w \) – the size of the weights in bits.

NVIDIA K40 (~4 TFLOPS), PCIe v3 (~16 GB/s)
Data parallelism

\[ T_{\text{compute}}(p, c, \gamma) = \frac{c}{(p\gamma)} \]
\[ T_{\text{communicate}}(p, w, \beta) = \frac{2w\log(p)}{\beta} \]

- \( p \) – number of workers (nodes),
- \( \gamma \) – the computational power of the node,
- \( c \) – the computational complexity of the model,
- \( \beta \) – bandwidth,
- \( w \) – the size of the weights in bits.

NVIDIA K40 (~4 TFLOPS), Infiniband (~56 Gb/s)
Deep Learning Cookbook helps to pick the right HW/SW stack

- **Benchmarking suite**
  - Benchmarking scripts
  - Set of benchmarks (for core operations and reference models)

- **Performance measurements** for a subset of applications, models and HW/SW stacks
  - 11 models
  - 8 frameworks
  - 6 hardware systems

- **Analytical performance and scalability models**
  - Performance prediction for arbitrary models
  - Scalability prediction

- Reference solutions, white papers
Selected scalability results
HPE Apollo 6500 (8 x NVIDIA P100)

AlexNet Weak Scaling
Batch size: 64
Batch size: 128

DeepMNIST Weak Scaling
Batch size: 32
Batch size: 64
Batch size: 128

EngAcousticModel Weak Scaling
Batch size: 32
Batch size: 64
Batch size: 128

GoogleNet Weak Scaling
Batch size: 32
Batch size: 64
Batch size: 128

VGG16 Weak Scaling
Batch size: 16
Batch size: 32
Batch size: 64

VGG19 Weak Scaling
Batch size: 16
Batch size: 32
Batch size: 64
Selected observations and tips

– Larger models are easier to scale (such as ResNet and VGG)
  – A single GPU can hold only small batches (the rest of memory is occupied by a model)
– Fast interconnect is more important for less compute-intensive models (FCC)
– A rule of thumb: 1 or 2 CPU cores per GPU
– PCIe topology of the system is important
Further into the future: neuromorphic research projects

**Neuromorphic Computing** – the integration of algorithms, architectures, and technologies, informed by neuroscience, to create new computational approaches.

– Memristor Dot-Product Engine (DPE) – successfully demonstrated
  – Memristor crossbar analog vector-matrix multiplication accelerator

– Hopfield Network (electronic and photonic) – in progress

\[ I_j^0 = \sum_i G_{ij} \cdot V_i^1 \]
Thank you

Natalia Vassilieva
nvassilieva@hpe.com