Intel optimized TensorFlow

Дубна, Сентябрь 2018

Agenda

- Introduction to deep learning:
  - Models boost
  - Neural networks introduction
- Practice:
  - Non-optimized TensorFlow training
  - Tuning MKL parameters in optimized TensorFow
  - Performance comparison

Deep Learning Use Cases

Cloud Service Providers

Financial Services

Healthcare

Automotive

- Personal assistant
- Image & Video recognition/tagging
- Natural language processing
- Automatic Speech recognition
- Targeted Ads
- Fraud / face detection
- Gaming, check processing
- Computer server monitoring
- Financial forecasting and prediction
- Network intrusion detection
- Recommender Systems

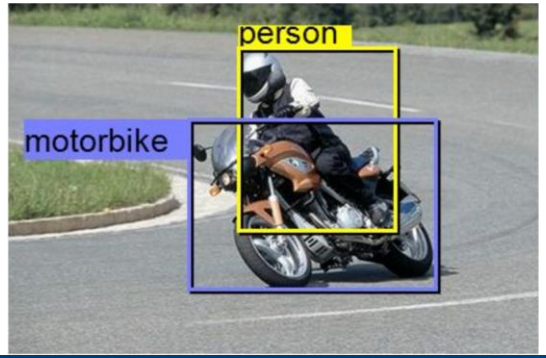
Neural networks for image recognition:

Classification:

Label the image

- Person
- Motorcyclist
- Bike

Object detection:



Semantic segmentation:

Neural networks for Natural Language Processing:

Machine translation:

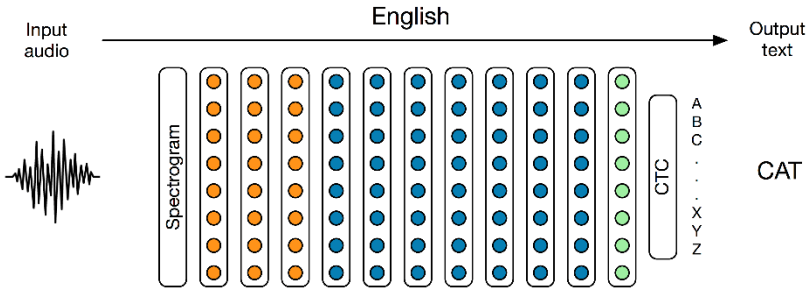
Google Translate

Break through language barriers

Personal Assistant:

Neural networks for audio:

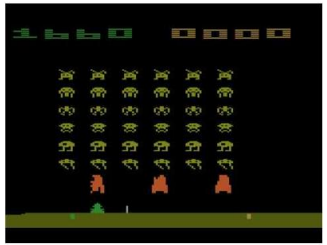
Speech recognition:



Music generation:

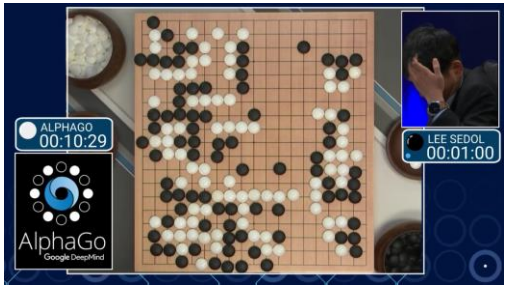
Neural networks for games:

Atari



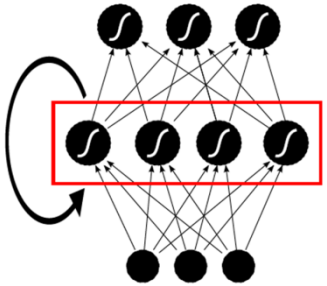
Breakout and Space Invaders, 2 of the 49 Atari games used in the paper

Alpha GO

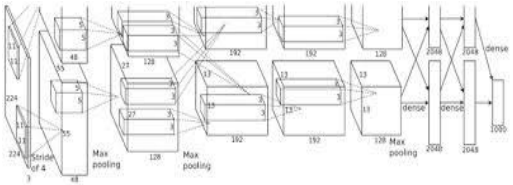


Dota 2

Diversity in Deep Networks



Recurrent NN



CNN - AlexNet

GoogLeNet

Variety in Network Topology

Recurrent NNs common for NLP/ASR, DAG for GoogLeNet, Networks with memory…

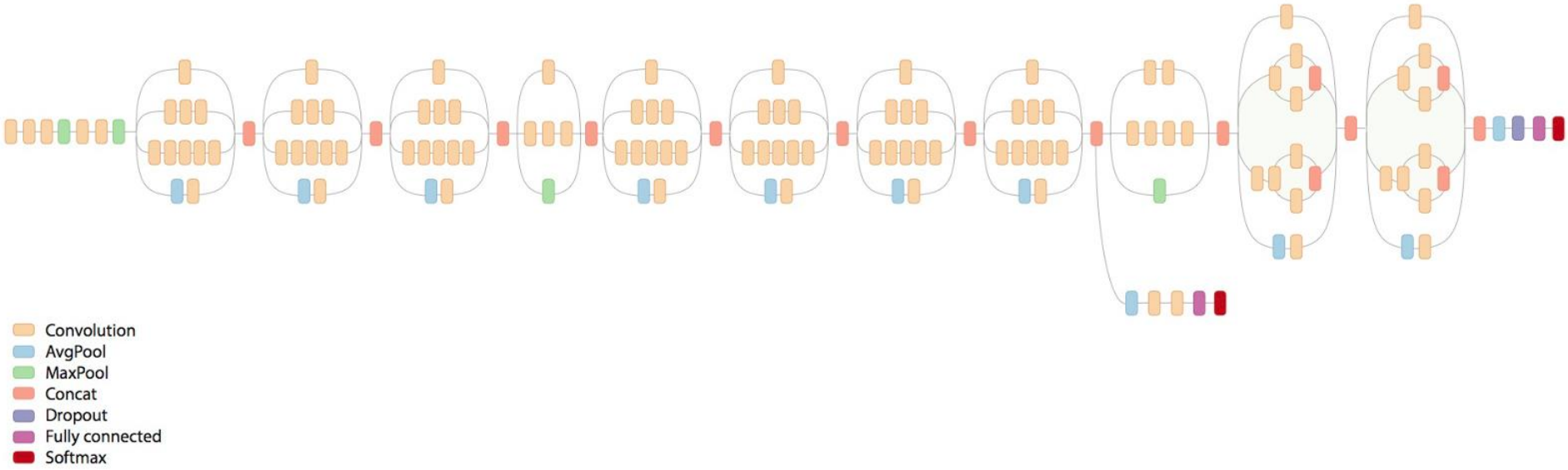
But there are a few well defined building blocks

Convolutions common for image recognition tasks

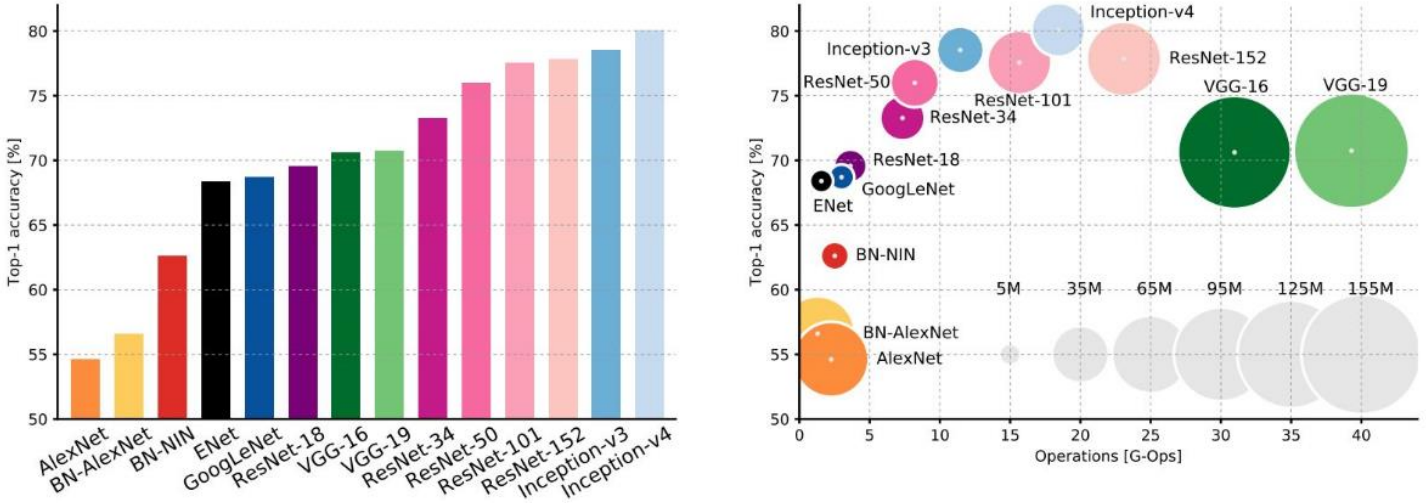
GEMMs for recurrent network layers—could be sparse

ReLU, tanh, softmax

Google Inception CNN



Comparing complexity...



An Analysis of Deep Neural Network Models for Practical Applications, 2017.

Настройка окружения

Installing Intel-optimized TensorFlow:

1. `conda create -n tf_intel -c intel python=3 pip numpy`
2. `source activate tf_intel`
3. `pip install https://anaconda.org/intel/tensorflow/1.6.0/download/tensorflow-1.6.0-cp36-cp36m-linux_x86_64.whl`
4. `pip install notebook matplotlib keras Pillow hdf5`

Installing non-optimized TensorFlow:

1. `conda create -n tf_simple python=3 tensorflow`

Convolution Neural Networks Layers:

Linear layer:

$$f(x) =< x, W >$$

Nonlinearities:

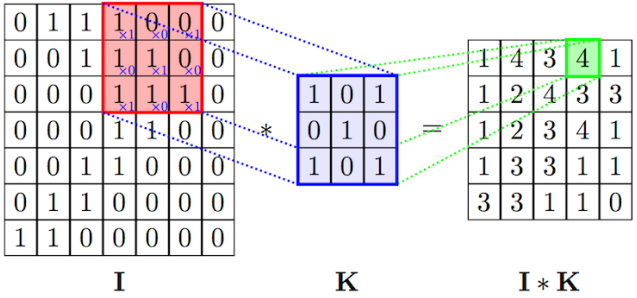
Sigmoid:

$$f(x) = \frac{1}{1 + e^{-x}}$$

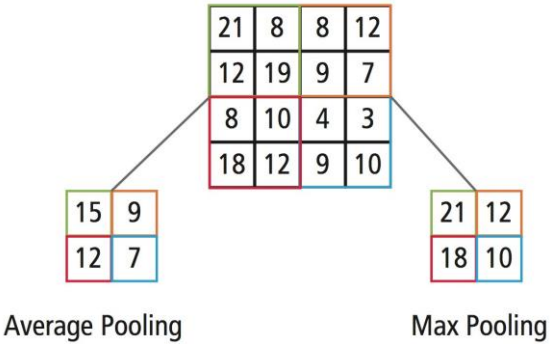
ReLU:

$$f(x) = max(x, 0)$$

Convolution:

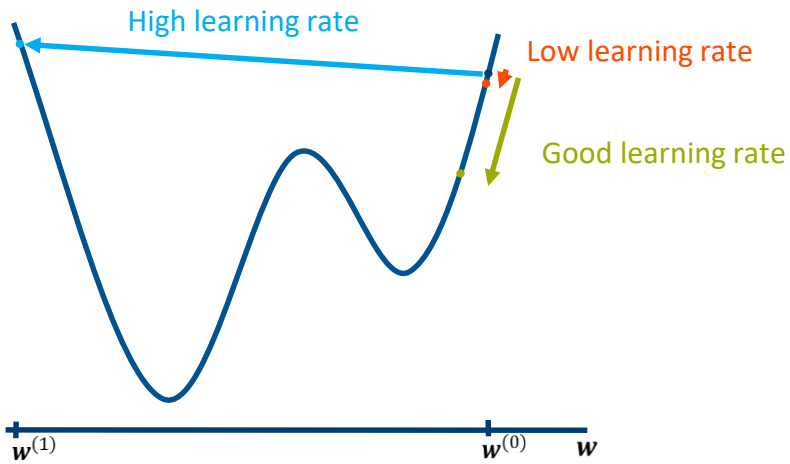


Pooling:

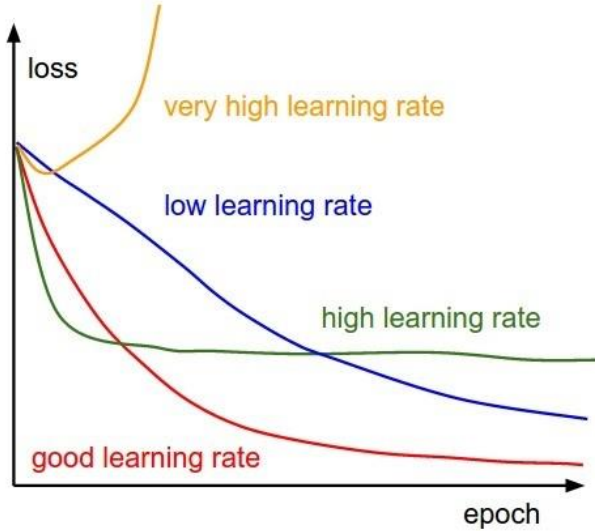


Choosing a learning rate:

Optimization task:



Loss function:



Model parameters:

- Network architecture
- Batch size
- Optimizer: SGD, Adam, Adadelta…
- Learning rate

Parameters TensorFlow:

- **`intra_op_parallelism_threads`** - Nodes that can use multiple threads to parallelize their execution will schedule the individual pieces into this pool
- **`inter_op_parallelism_threads -`** All ready nodes are scheduled in this pool

By default, both parameters is equal to the number of logical CPU cores.

MKL parameters:

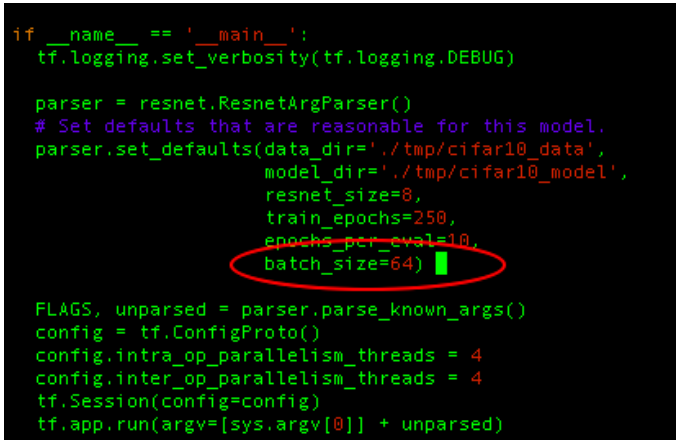
- **`KMP_SETTINGS`** – Enables (true) or disables (false) the printing of OpenMP* run-time library environment variables during program execution
- **`KMP_BLOCKTIME`** – Sets the time, in milliseconds, that a thread should wait, after completing the execution of a parallel region, before sleeping. Default time: 200
- **`KMP_AFFINITY`** – Enables the run-time library to bind threads to physical processing units
- **`OMP_NUM_THREADS`** – Specifies the number of threads to use intra_op_parallelism_thread

Important:

Use NCHW data format (channel-first) for better perfomance.

Changing batch_size:

1. Open cifar10_main.py file
2. At the end of the file change batch_size in parameters list:



```
if __name__ == '__main__':
  tf.logging.set_verbosity(tf.logging.DEBUG)

  parser = resnet.ResnetArgParser()
  # Set defaults that are reasonable for this model.
  parser.set_defaults(data_dir='./tmp/cifar10_data',
                      model_dir='./tmp/cifar10_model',
                      resnet_size=8,
                      train_epochs=250,
                      epochs_per_eval=10,
                      batch_size=64)

  FLAGS, unparsed = parser.parse_known_args()
  config = tf.ConfigProto()
  config.intra_op_parallelism_threads = 4
  config.inter_op_parallelism_threads = 4
  tf.Session(config=config)
  tf.app.run(argv=[sys.argv[0]] + unparsed)
```

Activity 1: non-optimised TensorFlow

1. `source activate tf_simple`
2. Launch training:

   `python cifar10_main.py`
3. Change batch_size to 64 and launch training again:

   `python cifar10_main.py`
4. Change batch_size to 256 64 and launch training again :

   `python cifar10_main.py`
5. `conda deactivate`

After 500 steps you can interrupt training (Ctrl+C)

Activity 2: Intel optimised TensorFlow

1. `source activate tf_intel`
2. Change KMP_BLOCKTIME и KMP_AFFINITY parameters in cifar10_main.py file:

```
os.environ["KMP_BLOCKTIME"] = str(0)
os.environ["KMP_AFFINITY"]= str("verbose,warnings,respect,granularity=fine,compact,1,0")
```

3. Launch training:

```
python cifar10_main.py
```

4. Change batch_size to 64 and launch training again:

```
python cifar10_main.py
```

5. Change batch_size to 256 and launch training again :

```
python cifar10_main.py
```

Results:

	batch_size=64	batch_size=128	batch_size=256
TensorFlow	58s	76s	102s
Intel optimized TensorFlow	11s	17s	26s

Intel optimizations provide high speedup during training.


intel
Software