Training a Deep Learning model on a custom image data set

Julien Simon
Principal Evangelist, AI/ML, EMEA
@julsimon

Agenda

- “Amazon Rekognition is not enough”
- Preparing and storing the data set
- Picking a library
- Picking a model
- Optimizing training

Amazon Rekognition

Deep learning-based image recognition

Search, verify, and organize millions of images

TRY AMAZON REKOGNITION

”Amazon Rekognition is not enough”

Valid points

- We need more than labels
- We have domain specific images
- We can’t rely on the cloud

Not so valid points

- We need extra labels
- We want our own images
- We want privacy
- NIH syndrome

Data sets

Take a long hard look at your data set

- How many images?
  - < few 1000s: may not be enough for Deep Learning
  - < few 10,000s: may not be enough to train from scratch
- How many categories? Are they balanced?
  - (Number of images / number of categories) ratio: MNIST 7k, CIFAR-10 6k, ImageNet 1.2k
  - Each category should have about the same number of images
- Are images and categories close enough to an existing data set?
  - If yes, using a pre-trained network or fine-tuning may be good options.
- Are images diverse enough?
  - Multiple angles, multiple colors, multiple object sizes, etc. If not, data augmentation may be required.
- How large are images?
  - ImageNet models are typically trained on 224x224 images.
- How many labels per image?
  - MXNet support multi-label training https://github.com/miraclewkf/multilabel-MXNet

Some examples

- 10,000 images of pieces of furniture
- 10s of basic categories (chair, table, etc)
- Only one label

Gut feeling: fine-tune a pre-trained model (Imagenet?)

- 1 million images of pieces of furniture
- 100s of advanced categories (18th century desk, Pop Art couch, etc.)
- Only one label

Gut feeling: train from scratch on a predefined model

- 1 million images of pieces of furniture
- 100s of advanced categories (18th century desk, Pop Art couch, etc.)
- Multiple labels + object positions

Gut feeling: train from scratch on your own model

Preparing the data set

https://medium.com/@julsimon/imagenet-part-1-going-on-an-adventure-c0a62976dc72

- Deep Learning training sets are often very large, with a huge number of files
- How can we deploy them quickly, easily and reliably to instances?

- We strongly recommend packing the training set in a RecordIo file
  - https://mxnet.incubator.apache.org/architecture/note_data_loading.html
  - https://mxnet.incubator.apache.org/how_to/recordio.html
  - Only one file to move around!
  - Worth the effort: pack once, train many times

- In any case, you need to copy your data set to a central location

- Usual suspects: Amazon EBS, Amazon S3 and Amazon EFS

Storing data sets in Amazon S3

- MXNet has an S3 connector → USE_S3=1
  https://mxnet.incubator.apache.org/how_to/s3_integration.html

```
train_dataiter = mx.io.MNISTIter(
   image="s3://bucket-name/training-data/train-images-idx3-ubyte",
   label="s3://bucket-name/training-data/train-labels-idx1-ubyte", ...
```

- Best durability (11 9’s)
- Distributed training possible

- Caveats
  - Lower performance than EBS-optimized instances
  - Beware of hot spots if a lot of instances are running
    https://docs.aws.amazon.com/AmazonS3/latest/dev/request-rate-perf-considerations.html

Libraries

Picking a library

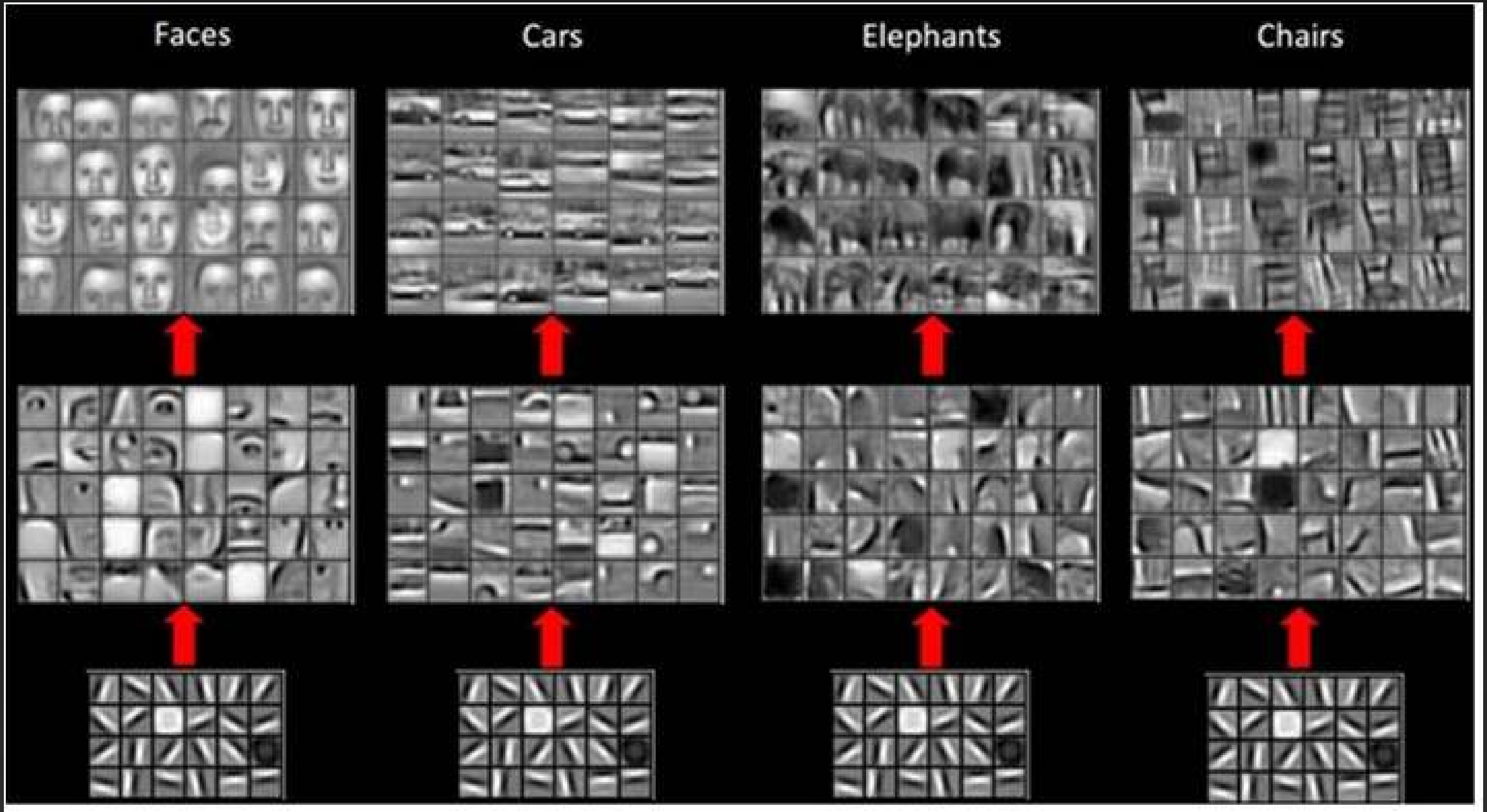
- We like MXNet, but we do what's best for the customer
  - Keep it as simple as possible
  - Work with tools the customer already knows
- Keras: super simple, great for small-scale data sets
  - One directory per category, < 100 lines of Python, 90%+ accuracy
  - blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html
  - Scales nicely with an MXNet backend, but you still need the Keras 1.2 fork (for now)
- Gluon: just as simple, more scalable… but not widespread
- MXNet: fastest, most scalable, not much harder ;)
- Deep Learning AMI, Deep Learning AMI, Deep Learning AMI

Models

Picking a model

- Using a pre-trained model
- Fine-tuning a pre-trained model
- Training an existing model from scratch
- Building your own model

The alchemy of Convolutional Neural Networks



Using a pre-trained model

- I would recommend trying this first
- Keras, MXNet and Gluon have model zoos
- Minimal code is required (demo in a minute)
- Get a performance baseline… and maybe a MVP
- Get a sense of how much progress could be made
- A model that works OK could be fine-tuned later on

Demo: using pre-trained models with MXNet

Fine-tuning a pre-trained model

- Extremely powerful technique:
  a lot of bang for your buck
- Customers do this (case study in a minute)
- The preferred option for smaller data sets
- Not a lot of code is required (demo in 2 minutes)

- Expedia have over 10M images from 300,000 hotels

- Using great images boosts conversion

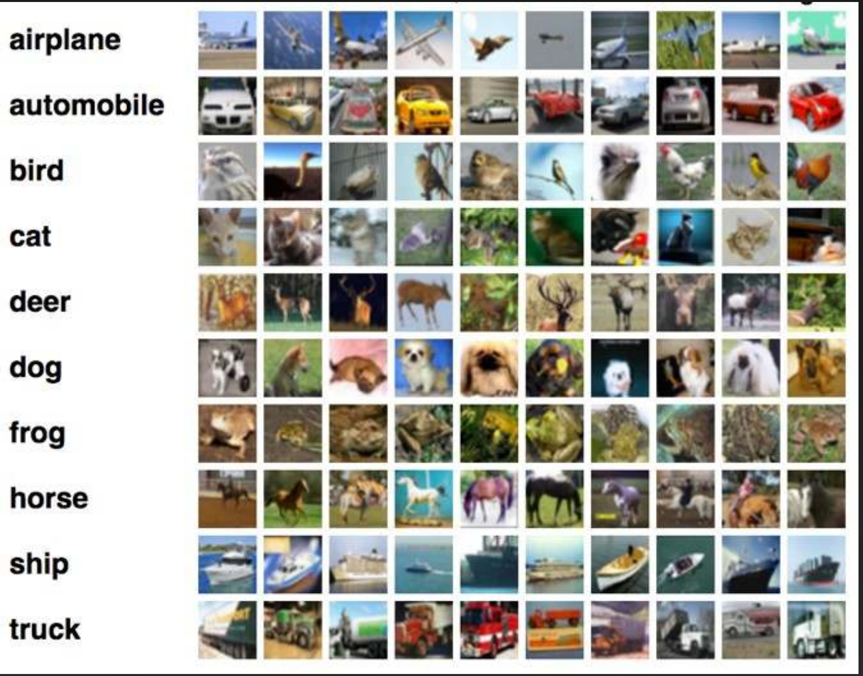
- Using Keras and AWS GPU instances, they fine-tuned a pre-trained Convolutional Neural Network using 100,000 images

- Hotel descriptions now automatically feature the best available images

https://news.developer.nvidia.com/expedia-ranking-hotel-images-with-deep-learning/

Demo: fine-tuning a pre-trained model with Keras

- CIFAR-10 data set
  - 60,000 images in 10 classes
  - 32x32 color images
- Initial training
  - Resnet-50 CNN
  - 200 epochs
  - 82.12% validation
- Cars vs. horses
  - 88.8% validation accuracy



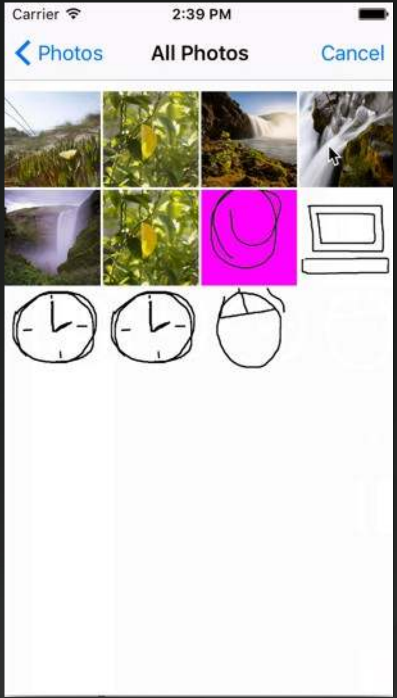
https://medium.com/@julsimon/keras-shoot-out-part-3-fine-tuning-7d1548c51a41

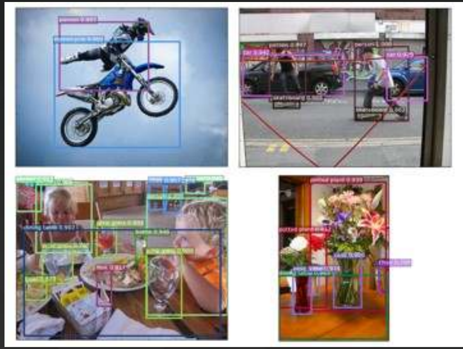
Training an existing model from scratch

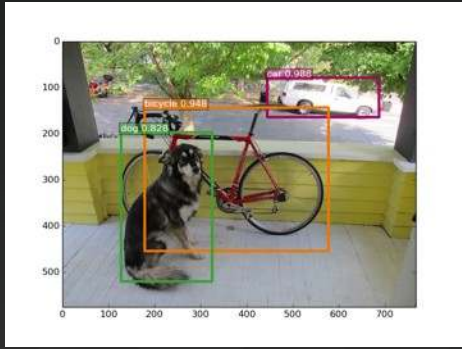
- Read research papers, explore GitHub
- Look for a model that performed well on similar problems
- Tweak it a bit if you know what you're doing
- Prepare your data (RecordIO, data augmentation, etc.)
- Use distributed training for larger data sets
  https://medium.com/@julsimon/training-mxnet-part-4-distributed-training-91def5ea3bb7
- Experiment with optimizers, initializers, hyper parameters
- Train 1,000 models, keep the best one

Open source projects based on MXNet

https://medium.com/@julsimon/10-deep-learning-projects-based-on-apache-mxnet-8231109f3f64







Building your own model

- Tweaking an existing network doesn't count ;)
- Designing a model is only for the 0.01% IMHO
- Deep Learning must truly be central to the business
- Beware of NIH! Remember people writing "their own version of Map Reduce" back in 2012? How did that end?

Last June, tuSimple drove an autonomous truck
for 200 miles from Yuma, AZ to San Diego, CA

https://www.oreilly.com/ideas/self-driving-trucks-enter-the-fast-lane-using-deep-learning

Maximizing GPU usage

- GPUs need a high-throughput, stable flow of training data to run at top speed
- Large datasets cannot fit in RAM
- Adding more GPUs requires more throughput
- How can we check that training is running at full speed?

- Keep track of performance indicators from previous trainings (images / sec, etc.)
- Look at performance indicators and benchmarks reported by others
- Use *nvidia-smi*
  - Look at power consumption, GPU utilization and GPU RAM
  - All these values should be maxed out and stable

Maximizing GPU usage: batch size

- Picking a batch size is a tradeoff between training speed and accuracy
  - Larger batch size is more computationally efficient
  - Smaller batch size helps find a better minimum

- Smaller data sets, few classes (MNIST, CIFAR)
  - Start with 32*GPU_COUNT
  - 1024 is probably the largest reasonable batch size

- Large data sets, lot of classes (ImageNet)
  - Use the largest possible batch size
  - Start at 32*GPU_COUNT and increase it until MXNet OOMs

Maximizing GPU usage: compute & I/O

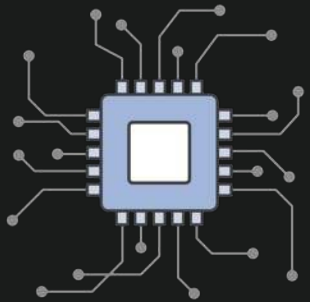
https://medium.com/@julsimon/imagenet-part-2-the-road-goes-ever-on-and-on-578f09a749f9

- Check power consumption and GPU usage after each modification
- If they're not maxed out, GPUs are probably stalling

- Can the Python process keep up? Loading images, pre-processing, etc.
    - Use *top* to check load and count threads
    - Use RecordIO and add more decoding threads

- Can the I/O layer keep up?
    - Use *iostat* to look at volume stats
    - Use faster storage: SSD or even a ramdisk!

What about CPU?



- Several libraries help speed up Deep Learning on CPUs
  - Fast implementation of math primitives
  - Dedicated instruction sets, e.g. Intel AVX or ARM NEON
  - Fast memory allocation

- Intel Math Kernel Library https://software.intel.com/en-us/mkl → USE_MKL = 1
- NNPACK https://github.com/Maratyszcza/NNPACK → USE_NNPACK = 1

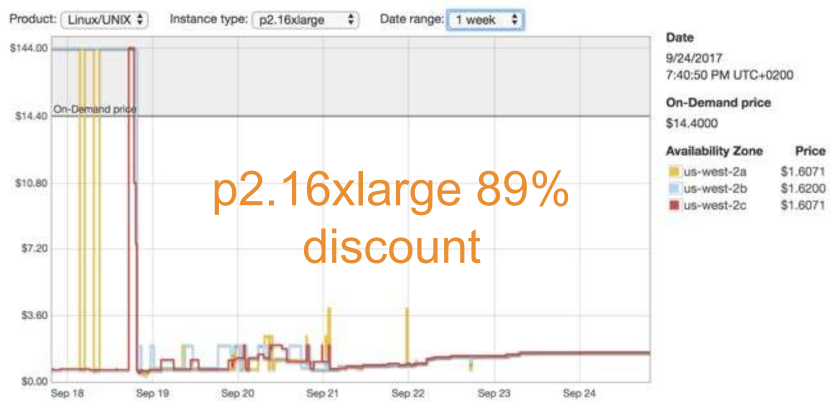
- Libjpeg-turbo https://www.libjpeg-turbo.org/ → USE_TURBO_JPEG = 1

- Jemalloc http://jemalloc.net/ → USE_JEMALLOC = 1
- Google Perf Tools https://github.com/gperftools → USE_GPERFTOOLS = 1

Optimizing cost

- Use Spot instances

https://aws.amazon.com/blogs/aws/natural-language-processing-at-clemson-university-1-1-million-vcpus-ec2-spot-instances/



- Sharing is caring: it's easy to share an instance for multiple jobs

```
mod = mx.mod.Module(lenet, context=(mx.gpu(7), mx.gpu(8),
mx.gpu(9)))
```

Conclusion

- Deep Learning is red hot
- We do what's best for the customer
- Sometimes, this means explaining to them that there is a better way
- "Remember your failure at Big Data"
- Fine-tuning is probably the best bet
- Keep it simple and iterate

aws

Thank you! See you in Vegas ☺

http://aws.amazon.com/evangelists/julien-simon

@julsimon