some application specific tips

- GPUs are good at lots of simple operations that are highly parallel, no control flow/branching → all threads must be executing the same instruction.
- optimizing data structures to avoid cache misses, specialized numerical computation instructions etc go a long way.
- synchronous (with lock, mathematically equivalent to SGD) and asynchronous (lock free, gradient estimate not so good but many more updates) stochastic gradient descent with a parameter server helps often.
- model compression: train a super deep model on a ton of data and then iteratively remove computationally expensive layers that don't add too much value → experimental process.
- Train on high precision floating point arithmetic (32-bit) to get good gradients (with dropout, batch norm and noise to make it robust) and then deploy on 8-bit architectures to get more throughput → precision much less important during inference than during training.
- Speech recognition models often use Hidden Markov models + RNNs. The HMM helps model state a bit more explicitly. Maybe someday we will just replace the whole thing with a good RNN.
- Language models rely a lot on pre-trained "embeddings" which are pre-trained representations. Turns out its a lot easier to train specific tasks with this representation space as input instead of one hot word vectors that are really bad at conveying information → these are called neural language models.
- Neural machine translation is kind of like encoder decoder architectures we saw earlier.

More about applications in the notes on the Deep Learning for Vision and Deep Learning for NLP classes.