Deep Learning as Software 2.0

Gil Arditi
Product Lead, Machine Learning

go.lyft.com/dl_software2_talk

“Machine Learning is the new electricity.
It will transform every industry in the next few years”

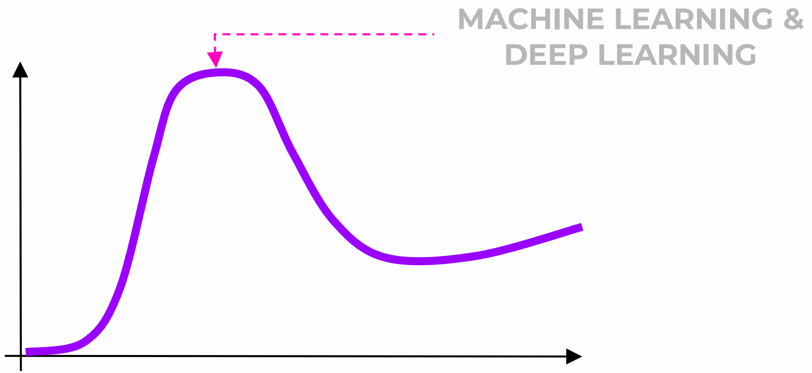
- Andrew Ng (3/2017)

Big bets by 1,000lb gorillas

*“Computing is evolving... We’re moving from mobile-first to **AI-first in all our products.”***

- Sundar Pichai (5/2017)

But might be over-hyped



Gartner 2017 emerging tech hype cycle

Deep Learning is Not the Only Kid in Town

Algorithm	Problem Type	Average predictive accuracy	Easy to explain algorithm to others?	Results interpretable by you?	Training speed	Prediction speed	Amount of parameter tuning needed (excluding feature selection)	Performs well with small number of observations?	Handles lots of irrelevant features well (separates signal from noise)?	Automatically learns feature interactions?	Gives calibrated probabilities of class membership?	Features might need scaling?
Neural Networks	**Regression & Classification**	**Higher**	**No**	**No**	**Slow**	**Fast**	**High**	**No**	**Yes**	**Yes**	**Possibly**	**Yes**
KNN	Regression & Classification	Lower	Yes	Yes	Fast	Depends on N	Minimal	No	No	No	Yes	Yes
Linear Regression	Regression	Lower	Yes	Yes	Fast	Fast	None (excluding regularization)	Yes	No	No	N/A	No (unless regularized)
Logistic Regression	Classification	Lower	Somewhat	Somewhat	Fast	Fast	None (excluding regularization)	Yes	No	No	Yes	No (unless regularized)
Naive Bayes	Classification	Lower	Somewhat	Somewhat	Fast (excluding feature extraction)	Fast	Some for feature extraction	Yes	Yes	No	No	No
Decision Trees	Regression & Classification	Lower	Somewhat	Somewhat	Fast	Fast	Some	No	Yes	Yes	Possibly	No
Random Forests	Regression & Classification	Higher	No	A little	Slow	Moderate	Some	No	Yes (unless noise ratio is very high)	Yes	Possibly	No
AdaBoost	Regression & Classification	Higher	No	A little	Slow	Fast	Some	No	Yes	Yes	Possibly	No

Computer Vision Before and After 2012

CLASSIFYING IMAGES

Feature Extraction and Engineering

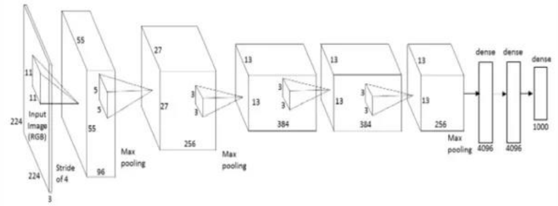
(lots of complex math)

Custom function on vector describing image stats

"Cat"

(1 of 1,000 possible classes)

Pre-2012



8-layer neural network (AlexNet)

Direct output from neural net

"Cat"

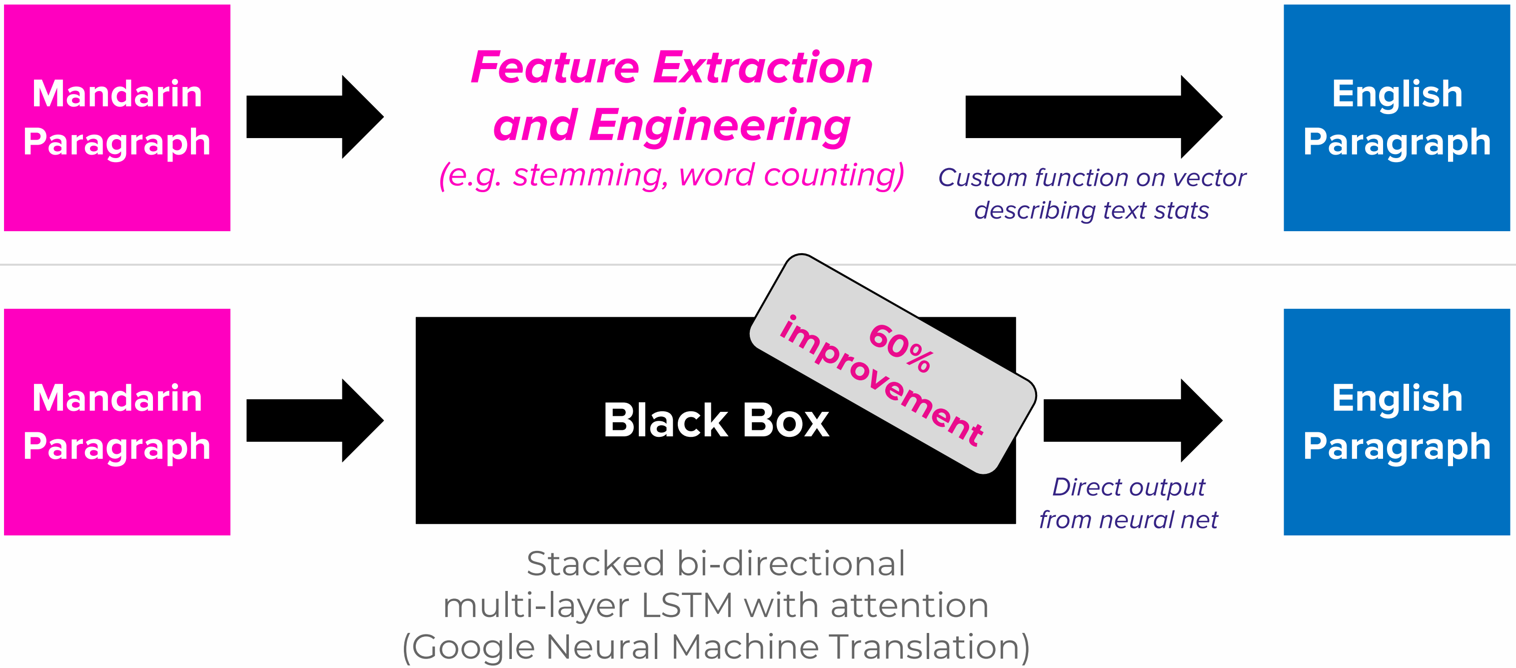
(1 of 1,000 possible classes)

2012+

For virtually every Computer Vision problem today the best performing model is based on Deep Learning

Machine Translation Before and After 2016

Translation between a language pair



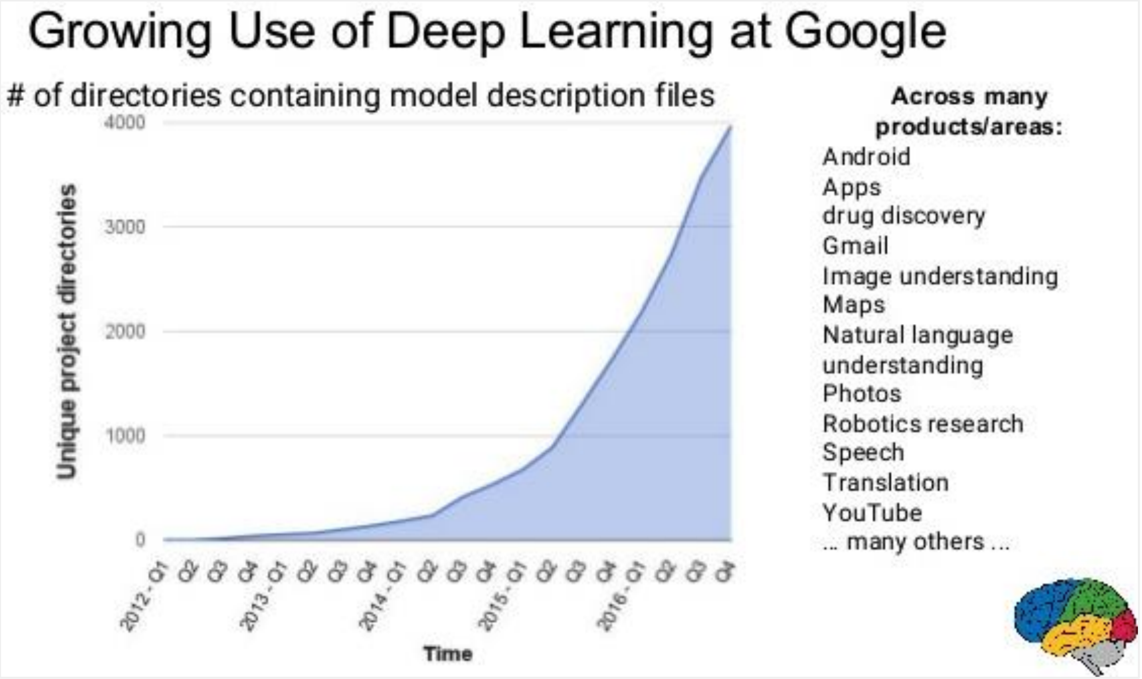
Similar results seen in speech recognition (human parity 8/17), text analysis and inference, and many others

The Promises of Deep Learning

1. **Automatic feature learning**
   No need to specify or engineer features

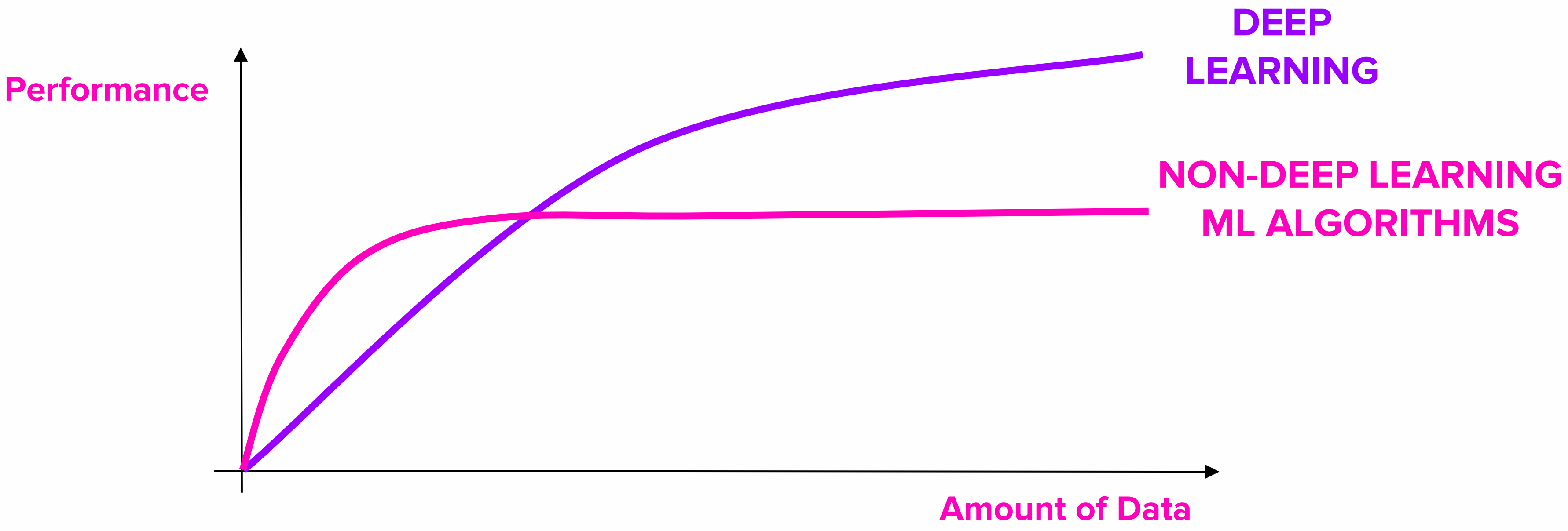
2. **End-to-end modeling**
   Getting the desired result directly.
   No need for intermediate steps

3. **Continued improvement**
   The more training data we have,
   the better the results



Source: Jeff Dean

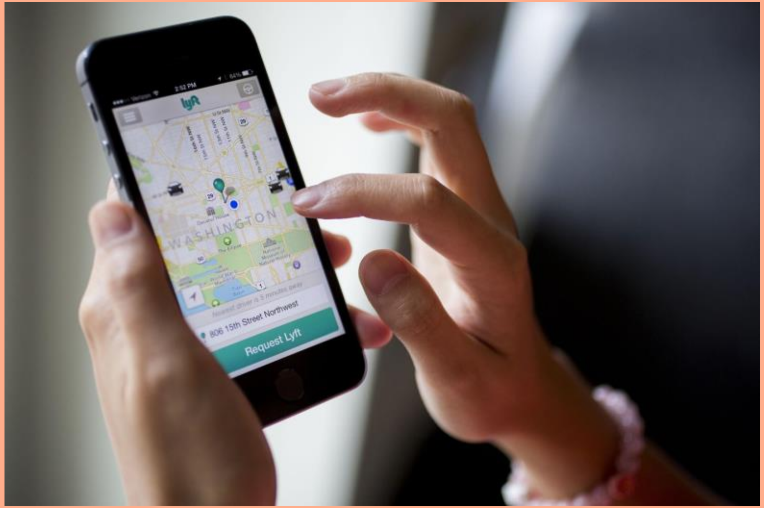
Deep Learning Handling Massive Data Really Well

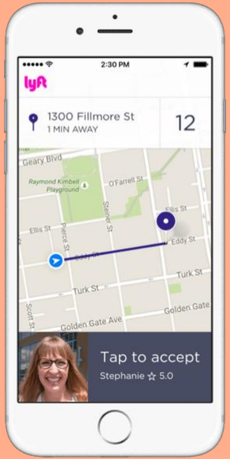


BUT:

Takes longer to train	Not always easy to architect solution	Can’t usually understand how it works

Lyft: Small Product Footprint





And Lots of Data

- Covering 95% of US population
- 1.4M drivers
- 10M+ weekly rides
- HD video (video + radar + Lidar)

Lyft Has Many Online and Offline Learning Problems

- At Lyft we have different classes of problems:
  - Online (real-time) vs. Offline (periodic)
  - Examples:
    - Online: pricing, ride dispatch policy, detect fraudsters
    - Offline: marketing spend planning, demand forecasting, Identify optimal pickup/dropoff spots

- We deal with them both offline and online
  - Marketing, growth, spam etc.

Example: Support Ticket Classification

Input: text coming in from passengers and drivers (typically complaints)

Output:

- What type of ticket is this (who and how should handle)
- Is this something that requires immediate attention (e.g. safety problem)

How we handle:

Using **Deep Learning** - train on manually classified tickets
(ticket text <-> safety issue or not)

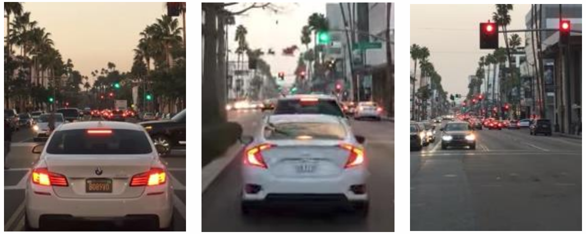
Deep Learning Becoming the Standard for Perception Problems AND BEYOND

Deep Learning-based model	Company
Alexa wakeup word pre-processing	**Amazon**
Text translation	**Google**
Image similarity searches	**Google**
Homepage “pin” ranking	**Pinterest**
News feed ranking	**Facebook**
Multiple rideshare-related and self-driving problems	**Lyft**

But Designing a Deep Learning Solution Can be Hard

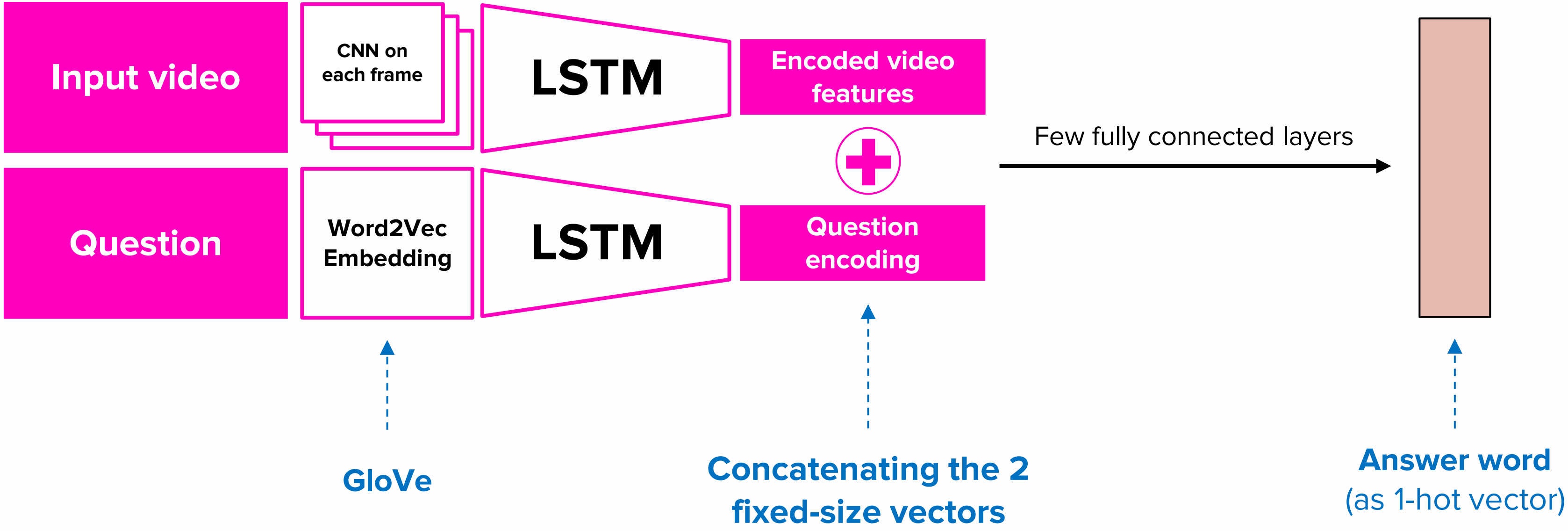
EXAMPLE: VIDEO Q&A

Input: Video + question about the video
Output: 1-word answer



Where is the car ahead turning?
Left

Are we moving now?
No



Keras Code for the Video Q&A Problem

```
video = tf.keras.layers.Input(shape=(None, 150, 150, 3))
cnn = tf.keras.applications.InceptionV3(weights='imagenet', include_top=False, pool='avg')
cnn.trainable = False

encoded_frames = tf.keras.layers.TimeDistributed(cnn)(video)
encoded_video = tf.keras.layers.LSTM(256)(encoded_frames)

question = tf.keras.layers.Input(shape=(100), dtype='int32')
x = tf.keras.layers.Embedding(10000, 256, mask_zero=True)(question)
encoded_question = tf.keras.layers.LSTM(128)(x)

x = tf.keras.layers.concat([encoded_video, encoded_question])
x = tf.keras.layers.Dense(128, activation=tf.nn.relu)(x)
outputs = tf.keras.layers.Dense(1000)(x)

model = tf.keras.models.Model([video, question], outputs)
model.compile(optimizer=tf.AdamOptimizer(), loss=tf.softmax_crossentropy_with_logits)
```

Software Engineering is Different in the Deep Learning Era

Classic Software Engineering

Subdomain expertise is critical

To solve computer vision, speech recognition, natural language processing problems we need:

1. Different types of experts
2. Processing input data differently
3. Using different development tools
4. Building different logic to predict the result

Deep Learning Development

Subdomain expertise much less important

To solve the problems on the left we need:

1. Common baseline architecture
2. Same building blocks (e.g. CNN, RNN)
3. Similar tools to develop and debug (e.g. TensorFlow and TensorBoard)
4. **Willingness to read academic papers and implement the ideas there**

Deep Learning Development Very Different vs. Classic Software

	Classic Software	Deep Learning
Architecture	Problem-dependent, using a **custom computational graph**	Problem-dependent, with **common baseline architectures for common problems**. Using building blocks like CNN and RNN layers.
Common Bottleneck		**GPU memory and speed, GPU connectivity**
Required compute resources	Varied and dynamic	**Static** - GPUs for training and typically for inferencing too.
Compute time	Varies based on software branching	**Fixed** - same neural network “flow” for all inputs
Memory Use	Involves dynamic allocation, caching, and tiers	**Static and uniform**

Think about Upleveling Your Software Engineering Teams to Know How to Use ML

Companies gain competitive advantage via ML

ML is critical for business success

ML becoming a must-have tool for **software engineers**

ML, and especially *Deep Learning* is general **software engineering**

Tiers of ML Knowledge Within The Org

ML Experts

Perform research on ML. Aware of state-of-the-art papers, come up with new algorithms, write and publish papers and present at conferences.

ML Practitioners

Can apply ML knowledge to solve problems in a specific domain. Know what tools are available, and understand how to use tools and algorithms.

ML Beginners

Have basic understanding of concepts and some limited experience in solving practical problems.

ML Development Involves a Lot of Non-Modeling Tasks

- Environment provisioning
- Data source linking & cleanup
- Feature extraction
- Machine resource management
- **ML Modeling**
- Distributed execution management
- Conversion to production
- Logging and analysis
- Production monitoring and re-training

Some Progress with Tools, but Still a Long Way to Go

Amazon SageMaker

Google Auto ML

TensorFlow™

Keras

mxnet

PYTORCH

H_2O.ai

A combination of “art” + engineering required to solve custom problems

Need a Set of ML Platform Tools to Simplify Development

Covering the entire model dev lifecycle:

FEATURE ENGINEERING	PROTOTYPING	MODEL TRAINING	PRODUCTIONIZATION
Select inputs for the model, clean up, and potentially compute periodically.	Get on-demand cloud dev env with flexible HW/SW setup to **try out different ideas quickly**.	Tune the prototyped models on **big data or big compute environment**.	Push trained models to prod and execute them to serve results in real-time or periodically.

Current Common ML Development Model

Data Scientist

- Pull input data, explore, and clean
- Create model prototype = “happy path”
- Analyze results

Might be resistant to adopting Deep Learning

Backend Software Engineer

- Turn into production-grade code
- Solve real-life big data and big-compute problems
- Deploy and monitor

Likes to learn more about ML but doesn’t know where to start

The Boosted Trees - Neural Networks Gap

Gradient Boosted Trees

- An ML algorithm where “shallow” decision trees are automatically optimized to predict a result.
- ~1 line of code and multiple popular implementations (xgboost, sklearn xgboost, LightGBM).
- Produces good results, especially for **small data**.

→

Little modeling work.
Data Scientist spends most time on
feature selection and engineering.

Neural Networks

- Unless solving well known problem (e.g. image classification), need to develop model from scratch.
- Model development can be long and frustrating.
- For **big data** and if modeling done right, produces better results than Boosted Trees.

→

Significant modeling work.
A whole different domain area
involving a lot of “art”.

Learning Curve Problem for Non-ML Engineers

Harder for many engineers to get started on Deep Learning because it's not "yet another software framework"

Requires willingness to:

- Wrangle data
- Read academic papers
- Apply "art"

Can be considered still a "Ph.D. domain"

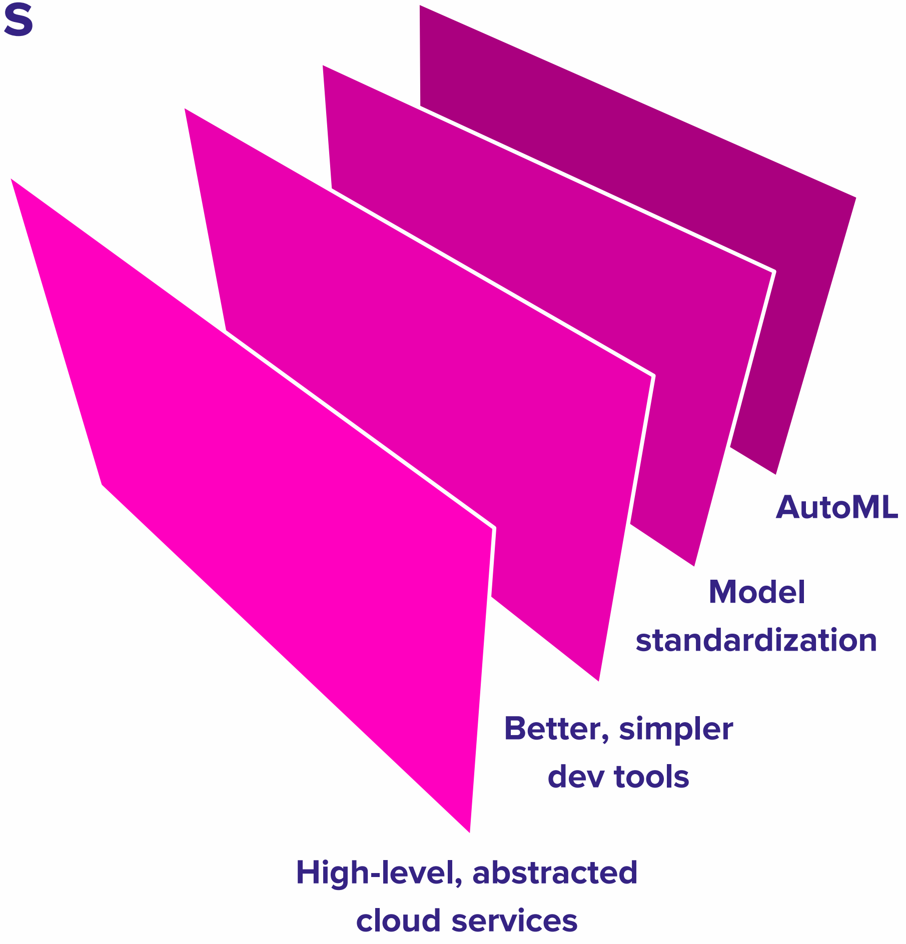
The ML Software Engineer

- Works on problems end-to-end
- Has Deep Learning focus (due to its complexity) but can use other ML tools well
- Handles all the “overhead” engineering work well
- Uses ML frameworks to their full potential
- Doesn’t need to be as strong in classical Data Science domains like Statistics and Operations Research

Might require some support for complex domains from ML researchers

Evolution in ML Modeling Tools

- Big cloud providers offering services for object classification, chatbots etc. **No coding needed**.
- Better and simpler tools constantly created for entire ML model dev lifecycle. **Big open source progress**.
- More standard “best weapon” models made available for common problems **with open code & published neural network weights**.
- Meta-Learning and Generic Algorithms could be tomorrow’s approach to automatically find the best neural net architecture from scratch.



Questions?

Lyft is hiring!

Contact me at gil.arditi@lyft.com

Additional Tutorials

Introduction to Deep Learning	go.lyft.com/intro_to_dl
Introduction to CNNs	go.lyft.com/intro_to_cnn
Introduction to RNNs	go.lyft.com/intro_to_rnn