

**E C M L P K D D 2 0 2 0 H a n d s o n T u t o r i a l**

# Put Deep Learning to Work

Accelerate Deep Learning through AWS SageMaker and ML Services

## **Tim O'Brien (AWS)**

Sr. Solutions Architect AI/ML  
[tpobrien@amazon.com](mailto:tpobrien@amazon.com)

## **Rachel Hu (AWS)**

Applied Scientist  
[rlhu@amazon.com](mailto:rlhu@amazon.com)

## **Wen-ming Ye (AWS)**

Sr. Solutions Architect AI/ML  
[wye@amazon.com](mailto:wye@amazon.com)



# Objectives

---

- Understand AI/ML industry trends
- Overview on Machine Learning Tools at AWS
- Introduce to BERT deep dive
- Get Hands on with AWS
  - Elastic Container Service + NGC (In SageMaker)
  - Build, Train (multi-GPU), and Deploy (GPU host)

# Agenda <https://github.com/goldmermaid/ECMLPKDD2020>

- Welcome and Logistics

---

- Amazon AI and SageMaker Overview

---

- BERT and Transformer

---

- Lab: Training/FineTuning Q&A model

---

(10 min break during model training)

---

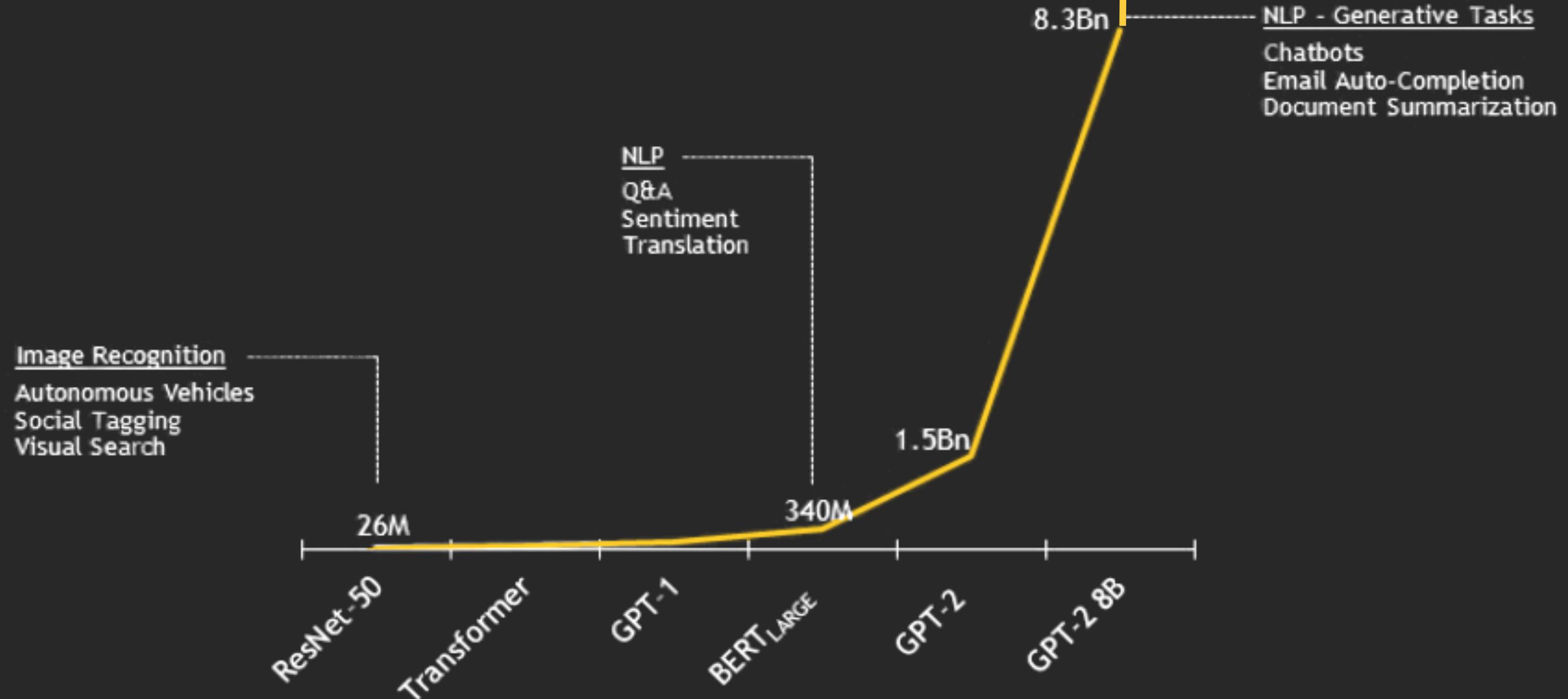
- Lab: Model Deployment

---

- Conclusion

# Exploding model complexity

## Number of parameters by network

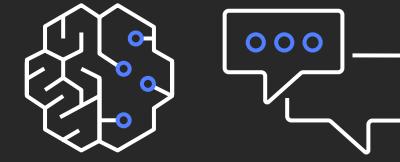


# Machine learning use cases (Day One)

Applications that benefit from accelerated compute

## Machine learning/AI

Natural language processing



Image/video analysis



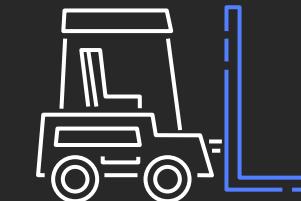
Financial services



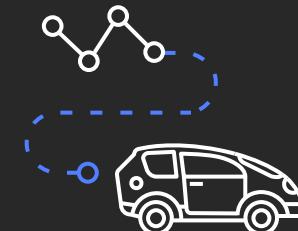
Healthcare & life sciences



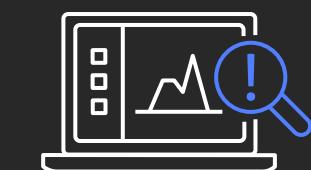
Manufacturing



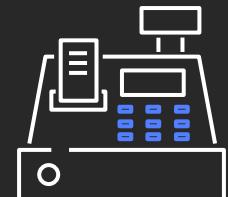
Autonomous vehicle systems



Recommendation systems



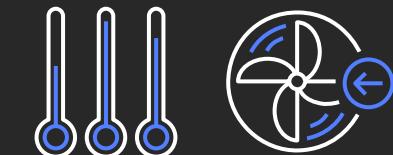
Retail



Travel and hospitality



Energy



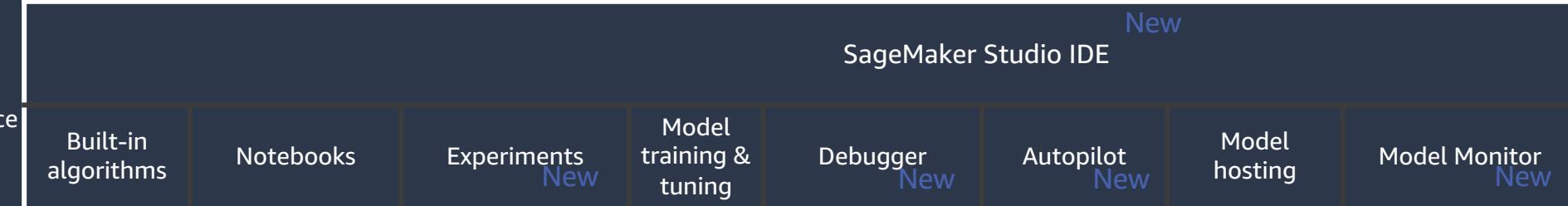
# The AWS ML Stack

Broadest and most complete set of Machine Learning capabilities

## AI SERVICES (Backed by our Open source Libraries)

VISION	SPEECH	TEXT/NLP	SEARCH	CHATBOTS	PERSONALIZATION	FORECASTING	FRAUD	DEVELOPMENT	CONTACT CENTERS		
 Amazon Rekognition	 Amazon Polly	 Amazon Transcribe <small>+Medical New</small>	 Amazon Comprehend <small>+Medical</small>	 Amazon Translate	 Amazon Kendra New	 Amazon Lex	 Amazon Personalize	 Amazon Forecast	 Amazon Fraud Detector New	 Amazon CodeGuru New	 Contact Lens <small>For Amazon Connect</small> New

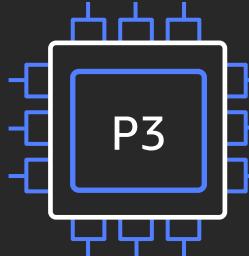
## ML SERVICES (Build your own)

 Amazon SageMaker	 Ground Truth	 ML Marketplace	 SageMaker Studio IDE <small>New</small> Built-in algorithms   Notebooks   Experiments <small>New</small>   Model training & tuning   Debugger <small>New</small>   Autopilot <small>New</small>   Model hosting   Model Monitor <small>New</small>							 Neo	 Augmented AI <small>New</small>
---	--	--	---	--	--	--	--	--	--	---	--

## ML FRAMEWORKS & INFRASTRUCTURE (Framework support and EC2 Accelerated Instance Types)



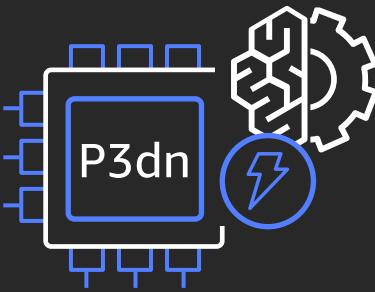
# Accelerated compute portfolio for machine learning



## ML training

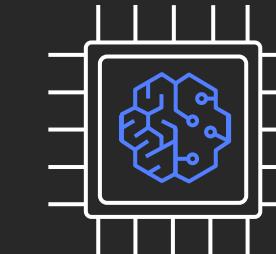
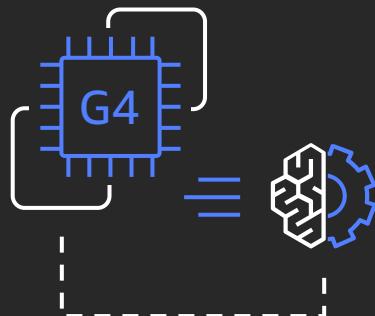
### P3/P3dn GPU compute instance

- Up to 1 PetaFLOP of compute with 8x NVIDIA V100 GPUs
- Up to 256 GB of GPU memory
- Up to 100 Gbps of networking
- Designed to handle large distributed training jobs for fastest time to train



### G4: GPU compute instance

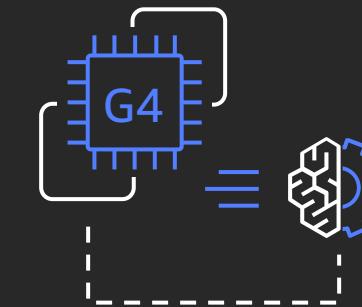
- Up to 520 TeraFLOPs of compute with 8x NVIDIA T4 GPUs
- Cost-effective, small-scale training jobs



## ML inference

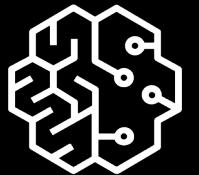
### AWS Inf1 instance

- Up to 2000 TOPs with 16x AWS-designed AWS Inferentia accelerators
- Lowest cost per inference in the cloud
- Designed for high throughput and low latency



### G4: GPU compute instance

- Up to 1030 TOPs of compute with 8x NVIDIA T4 GPUs
- Increased performance, lower latency and reduced cost per inference compared to previous GPU-based instances



# Amazon SageMaker

Machine learning for every developer & data scientist

Build, train, and deploy ML models at scale

<https://aws.amazon.com/sagemaker/>

# Amazon SageMaker

Bringing machine learning to all developers

Pre-built notebooks for common problems



Collect and prepare training data

Built-in, high performance algorithms



Choose and optimize your ML algorithm

One-click training



Set up and manage environments for training

Optimization



Train and tune model (trial and error)

One-click deployment



Deploy model in production

Fully managed with auto scaling, health checks, automatic handling of node failures, and security checks



Scale and manage the production environment

End-to-end machine learning platform

**Flexible model training**



**mxnet**

PyTorch

**GLUON**

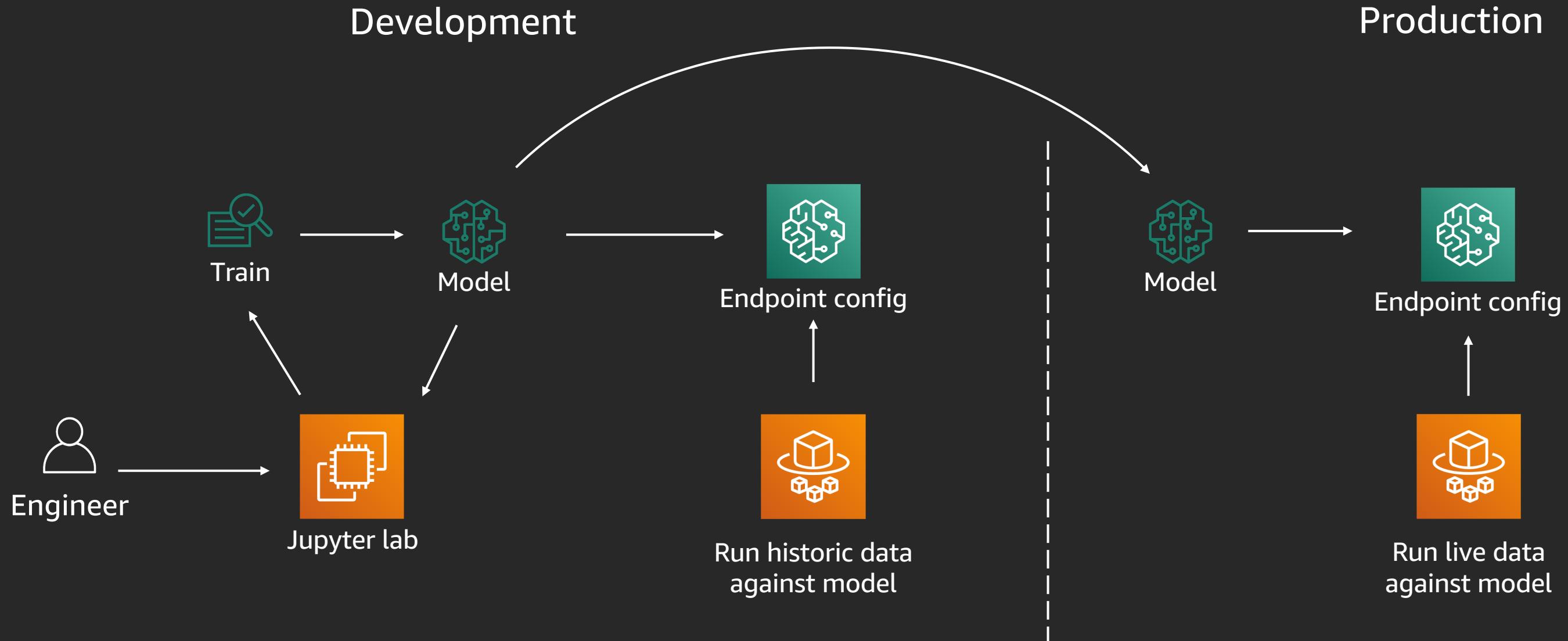
TensorFlow

**Chainer**



**Pay by the second**

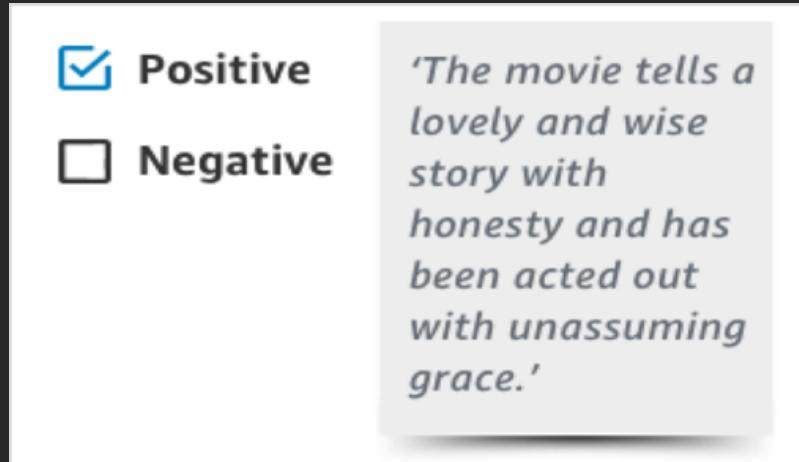
# Build, train, and test with Amazon SageMaker



# Use pre-built labeling workflows or set up your own



Bounding boxes



Text Classification



Image Classification



Semantic Segmentation



Custom and  
Active Learning

# Pools of Workforces



## Public

An on-demand 24 x7 workforce of over 500,000 independent Contractors worldwide, powered by Amazon Mechanical Turk



## Private

A team of workers that you have sourced yourself, including your own employees or contractors for handling data that needs to stay within your organization



## Vendors

A curated list of third party vendors that specialize in providing data labeling services, available via the AWS Marketplace

# 3 ways to train models with SageMaker

Built-in  
Algorithms

Bring-your-own script  
(TensorFlow, Scikit  
Learn, PyTorch, ...)

Bring-your-own  
container

# Amazon SageMaker has built-in algorithms or bring your own

## Classification

- Linear Learner
- XGBoost
- KNN

## Working with Text

- BlazingText
- Supervised
- Unsupervised

## Sequence Translation

- Seq2Seq

## Computer Vision

- Image Classification
- Object Detection
- Semantic Segmentation

## Recommendation

- Factorization Machines

## Anomaly Detection

- Random Cut Forests
- IP Insights

## Regression

- |                  |           |
|------------------|-----------|
| • Linear Learner | • XGBoost |
|                  | • KNN     |

## Topic Modeling

- LDA
- NTM

## Forecasting

- DeepAR

## Clustering

- KMeans

## Feature Reduction

- PCA
- Object2Vec

# Amazon SageMaker APIs

```
import boto3
import sagemaker

sess = sagemaker.Session()

pca = sagemaker.estimator.Estimator(containers[boto3.Session().region_name],
                                      role,
                                      train_instance_count=1,
                                      train_instance_type='ml.c4.xlarge',
                                      output_path=output_location,
                                      sagemaker_session=sess)
pca.set_hyperparameters(feature_dim=50000,
                        num_components=10,
                        subtract_mean=True,
                        algorithm_mode='randomized',
                        mini_batch_size=200)

pca.fit({'train': s3_train_data})
```

# Distributed training with Amazon SageMaker

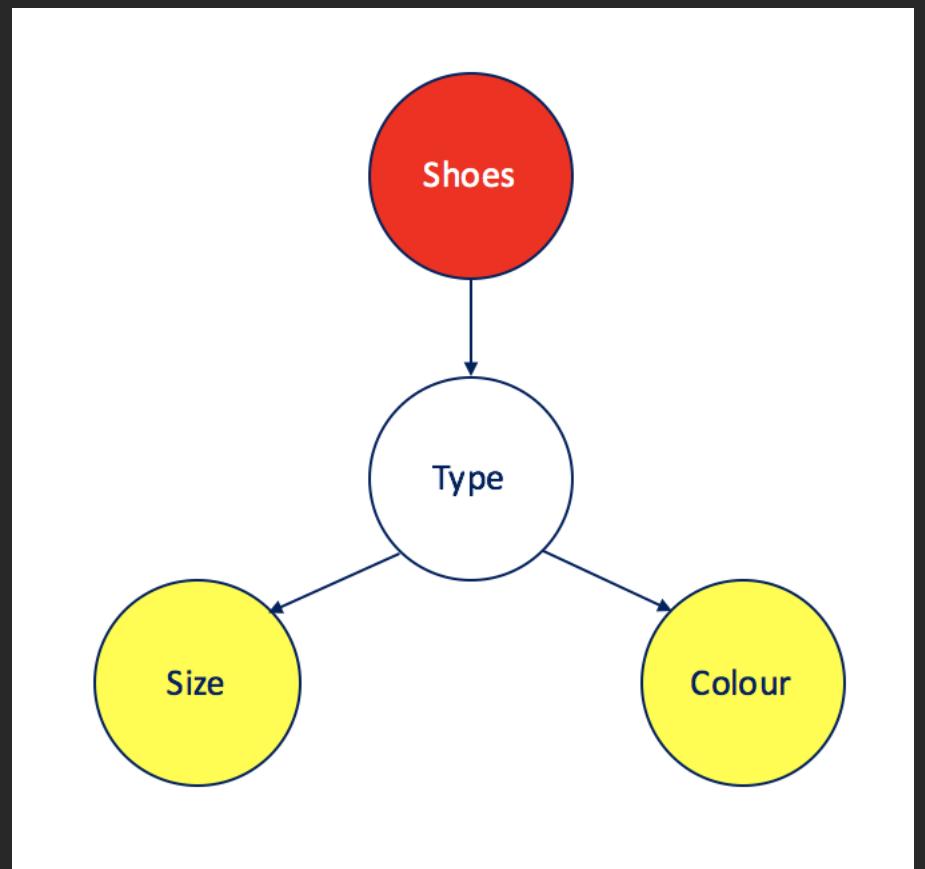
- ✓ Horovod (synchronous workers)
- ✓ Parameter servers (synchronous workers)

```
estimator = TensorFlow(entry_point='script.py', role=role,  
    train_instance_type='ml.p3.2xlarge',  
    train_instance_count=2,  
    distributions= {'parameter_server': {'enabled': True}} )
```

# Hyper-Parameter Optimization (HPO)

Hyper-parameters structure many every day processes but few people are aware of their existence. When you are looking for a pair of shoes to buy, you are solving a hyper-parameter optimization problem.

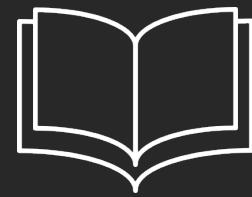
Once you have decided which type of shoes you are going to buy (hyper-parameter) you need to fit the parameters of the shoes to meet your requirements (size, color,...). Your available budget is also a hyper-parameter that will affect your choices.



# Introducing Amazon SageMaker Autopilot

## Automatic model creation with full visibility & control

---



### Quick to start

Provide your data in a tabular form & specify target prediction



### Automatic model creation

Get ML models with feature engineering & automatic model tuning automatically done



### Visibility & control

Get notebooks for your models with source code



### Recommendations & optimization

Get a leaderboard & continue to improve your model

# Deploy & Manage

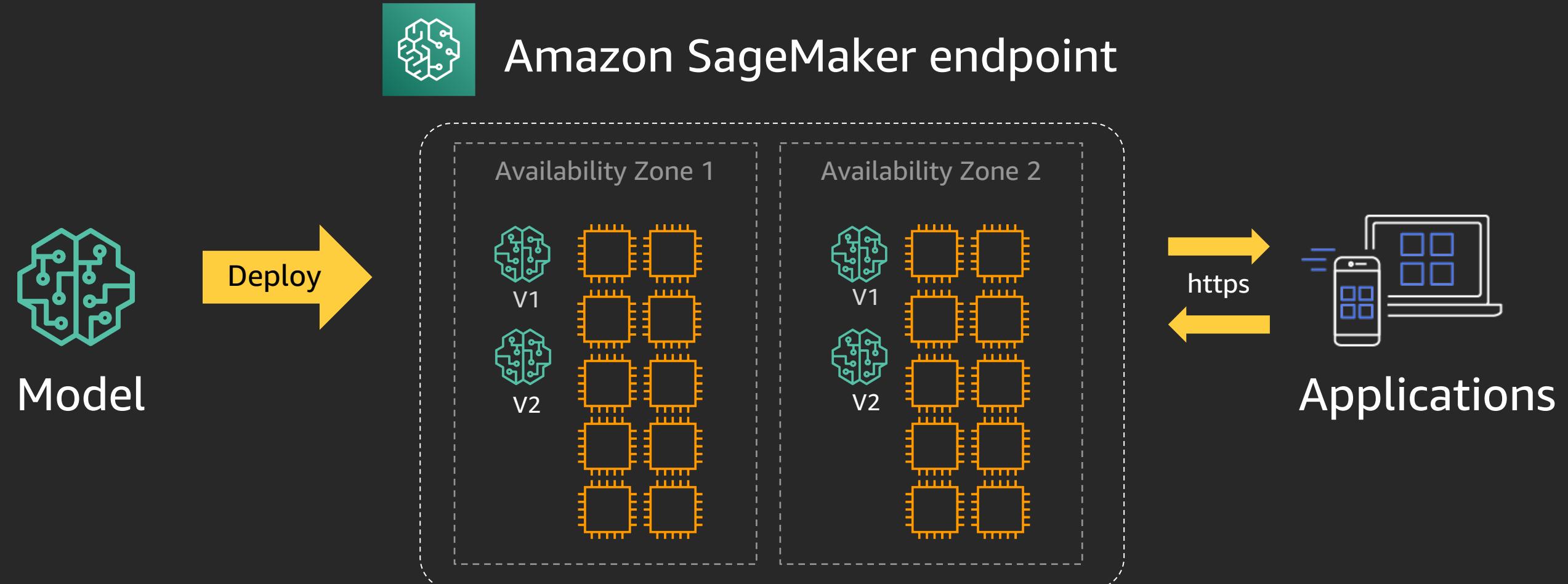
2 way to make predictions

Real-time,  
on-demand

Batch inference

# Fully managed model hosting at scale

Letting data scientists focus on models



- Autoscaling across multiple AZs for variable loads
- A/B testing support

# Use Amazon SageMaker Model Monitor to identify model drift and take action

Amazon SageMaker Studio File Edit View Run Kernel Git Tabs Settings Help

Amazon S UC-DEMO UC-DEMO model-mo SageMake model-mo DEMO-xgb model-mo X

Monitoring results Monitoring job history AWS settings

ENDPOINTS

Name	Endpoint status
AutoML-toms-super-notebo...	✓ InService
UC-DEMO-xgb-churn-pred...	✓ InService
demo-xgboost-customer-ch...	✓ InService
mengyw-test-form-1	✓ InService
mengyw-test-capture	✓ InService
my-sagemaker-ap1	✓ InService
mengyw-test-vpc-3	✗ Failed
DEMO-xgb-churn-pred-mod...	✓ InService
mengyw-test-vpc-2	✓ InService
mengyw-test-vpc	✗ Failed

AMAZON SAGEMAKER MODEL MONITOR

Amazon SageMaker Model Monitor detects data drift and other issues that can affect models in production and alerts you so you can take corrective action.

2 CHARTS Add chart

Int'l Plan\_yes: Sum

Sum

Date and time

source

Baseline Current

CHART PROPERTIES

Timeline

- 1 week
- 1 day
- 12 hours
- 6 hours
- 1 hour

Statistic

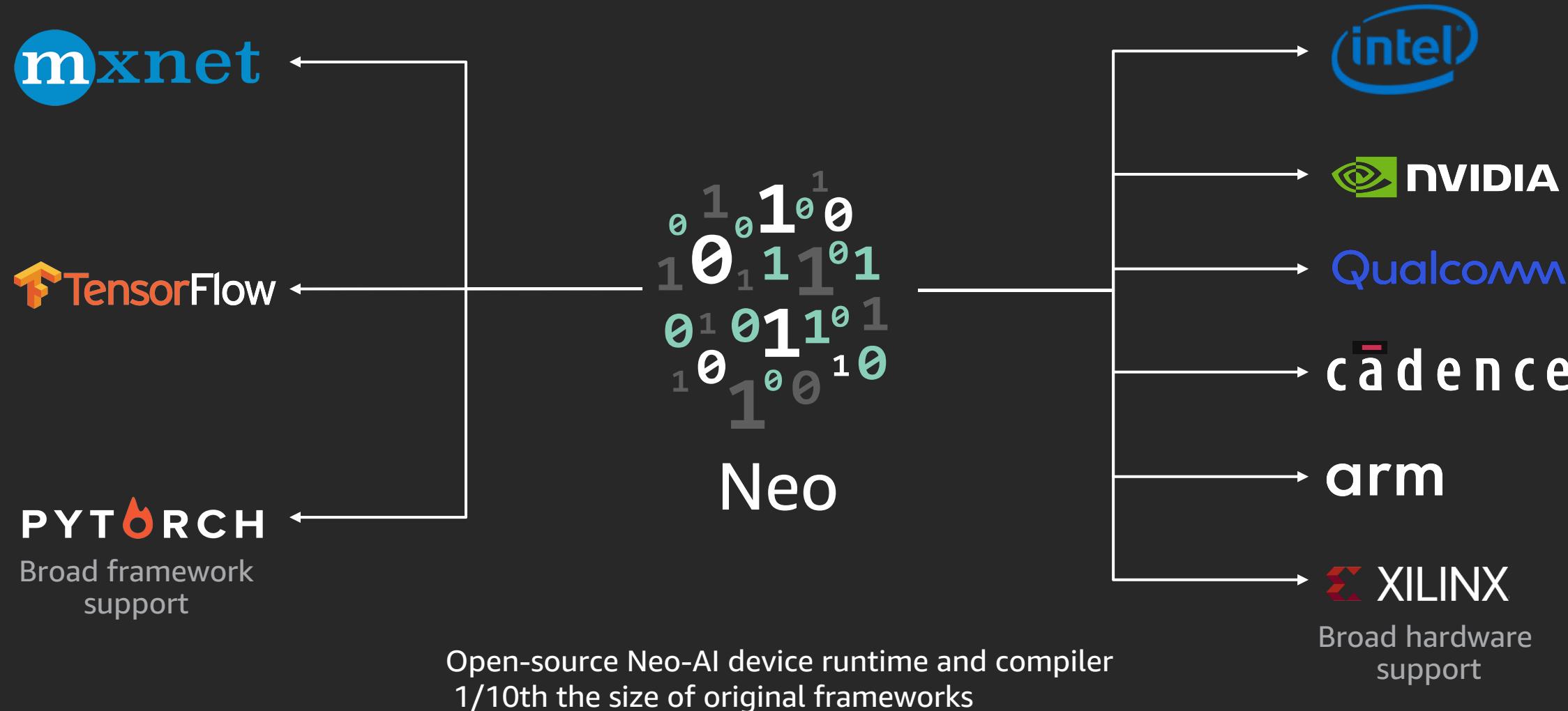
- Average
- SampleCount
- Sum
- Minimum
- Maximum

Feature

feature\_data\_Int'l Plan\_yes ▾

# Amazon SageMaker Neo

Train once and run anywhere with 2x performance



# Open Source Ecosystem

Services

TVM  
compiler

DGL  
graphs

Community

Jupyter  
notebooks

GluonTS  
time series

MXNet  
framework

Research

D2L  
book

AutoGluon  
AutoML

GluonCV  
computer vision

GluonNLP  
natural language

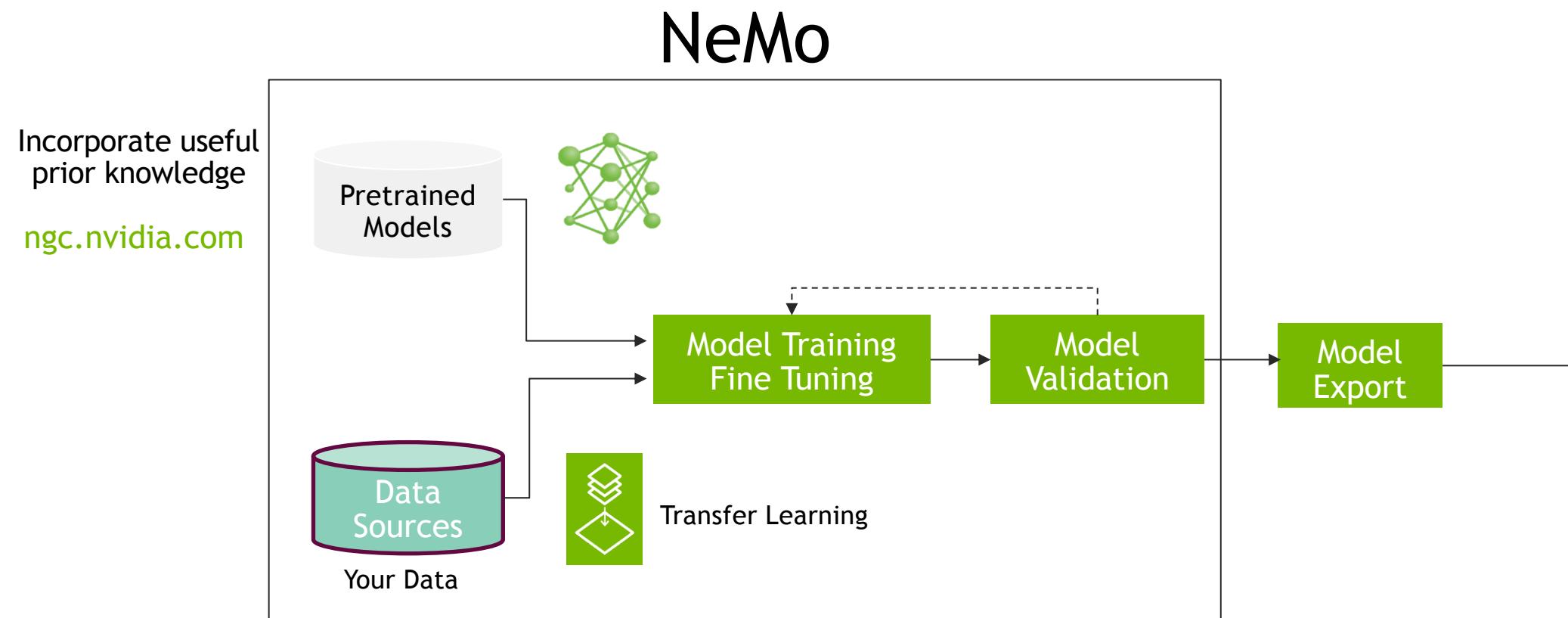
# Additional Resources

---

- MLU on YouTube: <https://bit.ly/31cBcf3>
- Gluon:
  - Deep Graph Library <https://www.dgl.ai/>
  - NLP <http://gluon-nlp.mxnet.io/>
  - Computer Vision <http://gluon-cv.mxnet.io/>
  - Time Series <https://gluon-ts.mxnet.io/>
- Dive into Deep Learning: <http://d2l.ai/>

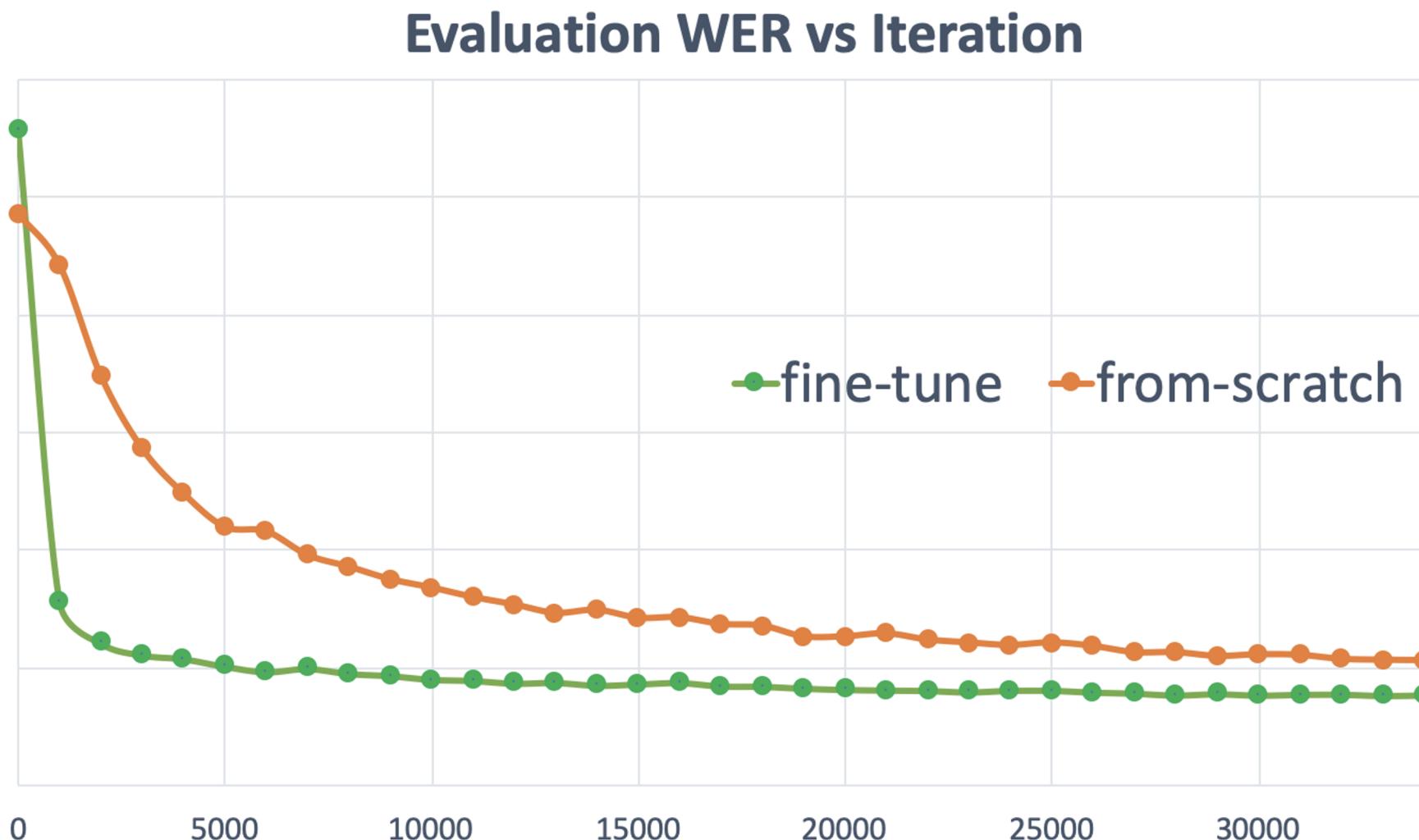
# Workflow to Build Domain Specific Models

Domain transfer leveraging pretrained models



# THE POWER OF TRANSFER LEARNING

- Domain Specific ASR - KENSHO & NVIDIA

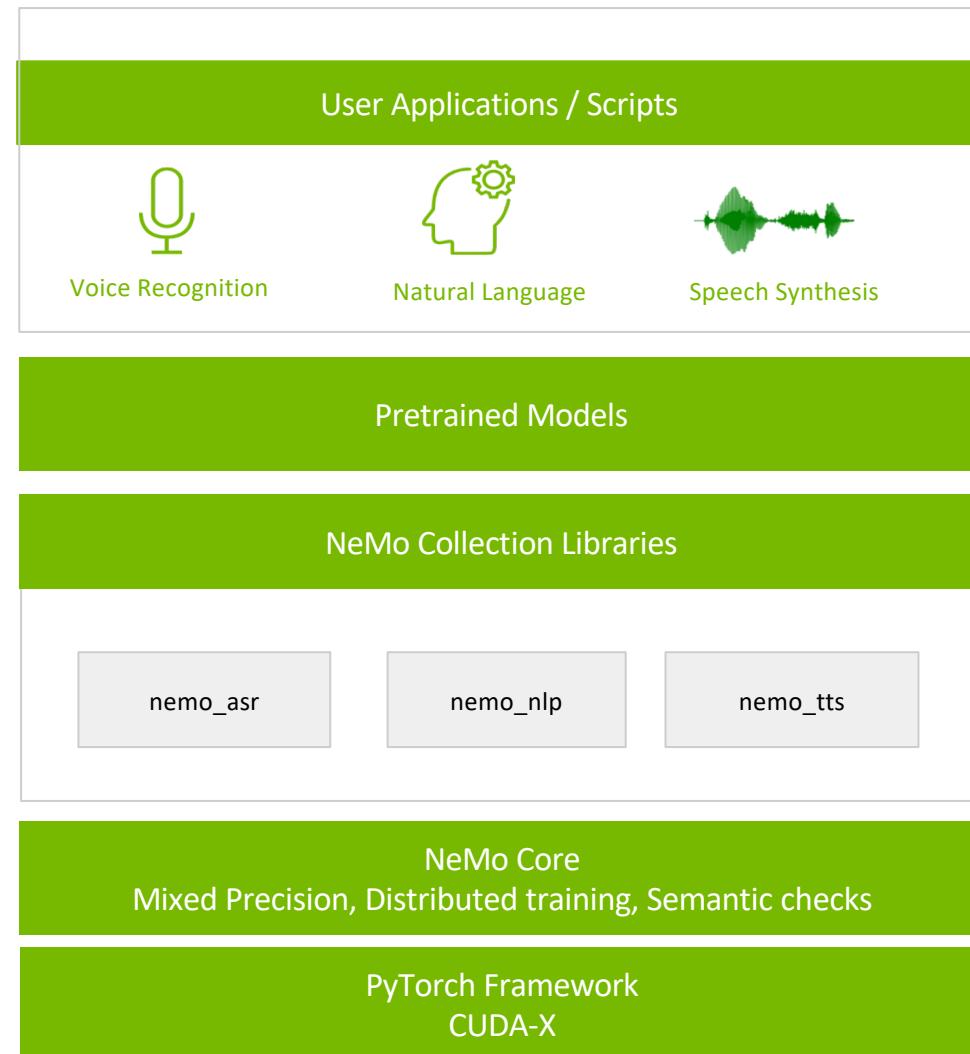


- Fine tuning was faster and had more accuracy than training from scratch
- Jasper trained on domain specific financial data outperformed all leading ASR models
- Enables quick start and has many benefits of reusability and scale

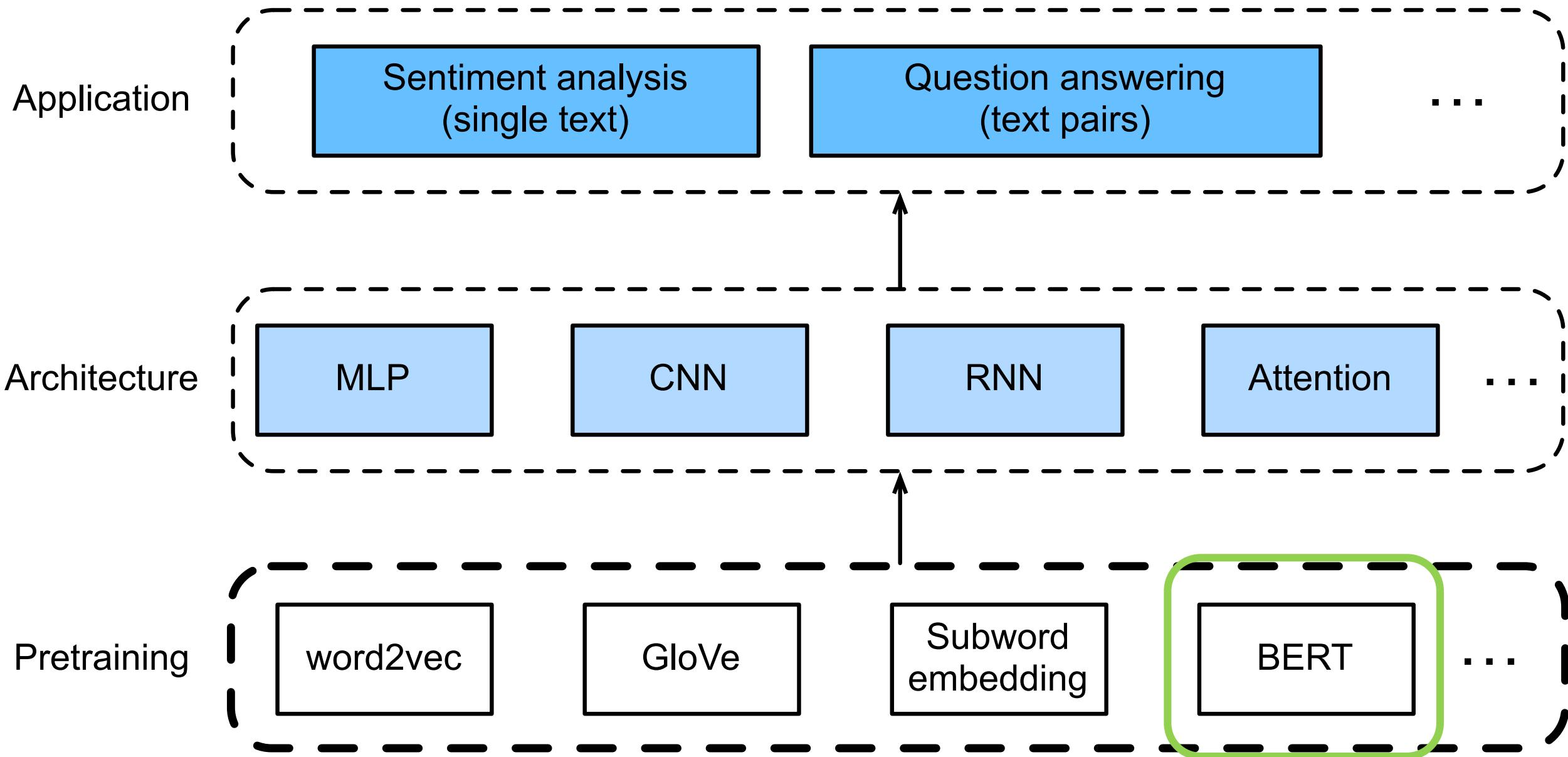
# NVIDIA NeMO Demo

Toolkit to Build State-Of-The-Art Conversational AI Models

- Develop SOTA DL-based speech and language models easily with collections of modules
- Pretrained models and scripts(modules) for building, transfer learning, adaptation or fine tuning to custom data
- ASR: Jasper, Quartznet
- NLP: Transformer, BERT
- TTS: Tacotron2, WaveGlow
- Mandarin language support
- Automatic semantic compatibility checking
- Built for high performance training on NVIDIA GPUs
- PyTorch backend
- Extend collection of modules, contribute or use out of the box



# NLP Workflow



# BERT

Traditional language modeling has problems ...

Amazon is on fire...



# BERT

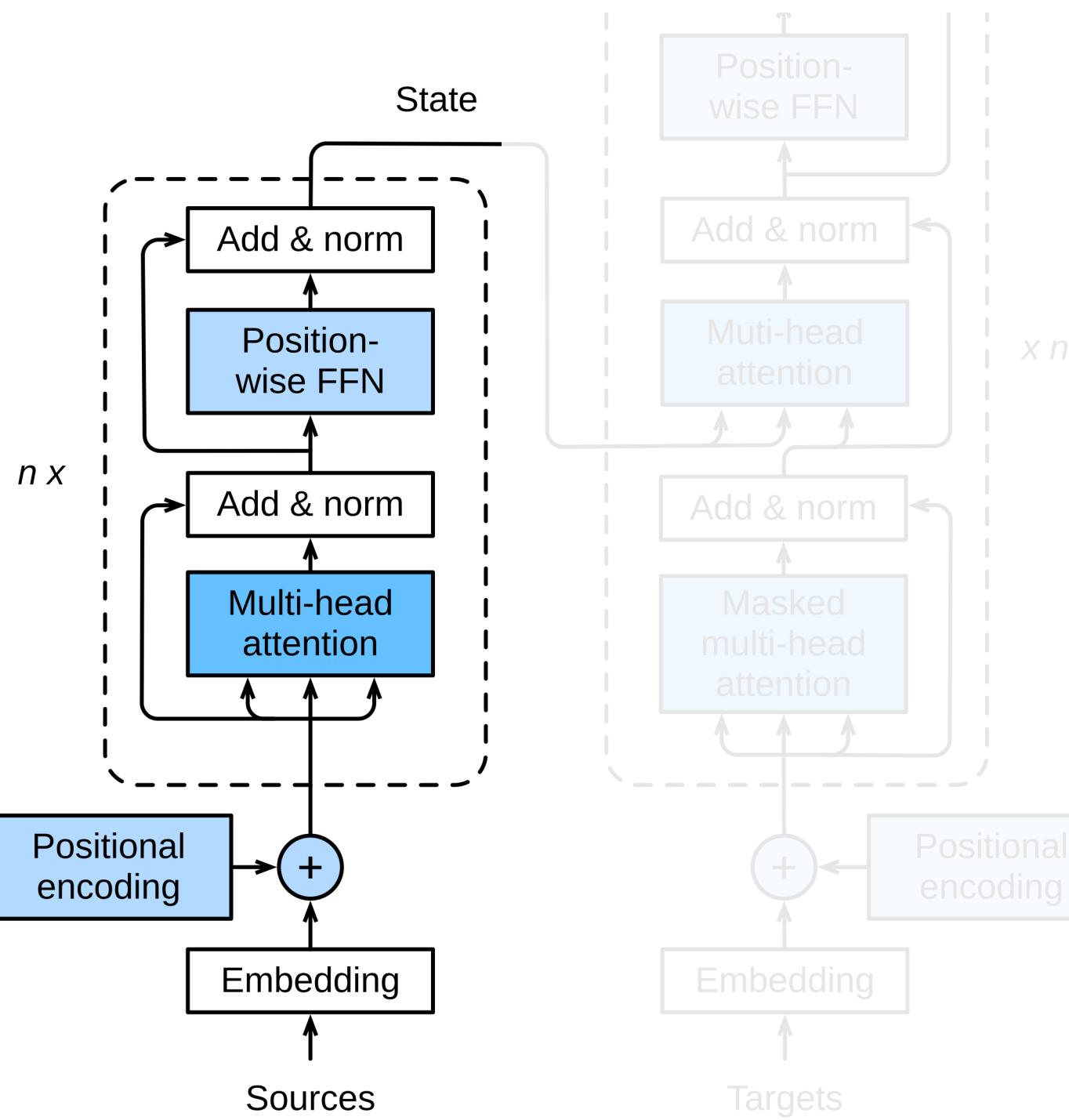
*BERT –*

*Bidirectional Encoder Representations from Transformers*

- Bidirectional Context
- Built on a Transformer encoder
- *Task-agnostic* model
- Improved 11 NLP tasks to SOTA



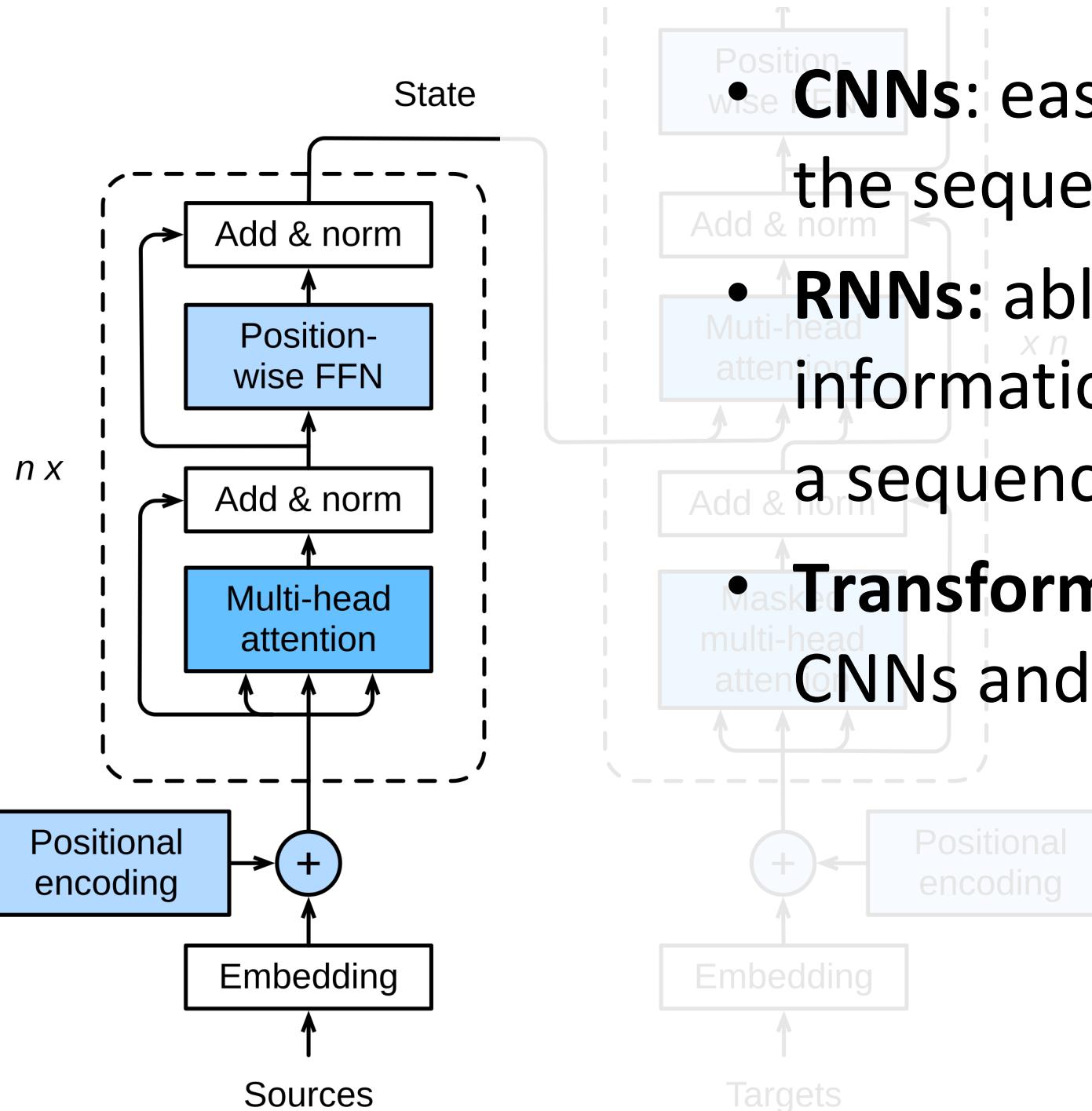
# BERT



Bidirectional  
Encoder  
Representations  
from Transformers

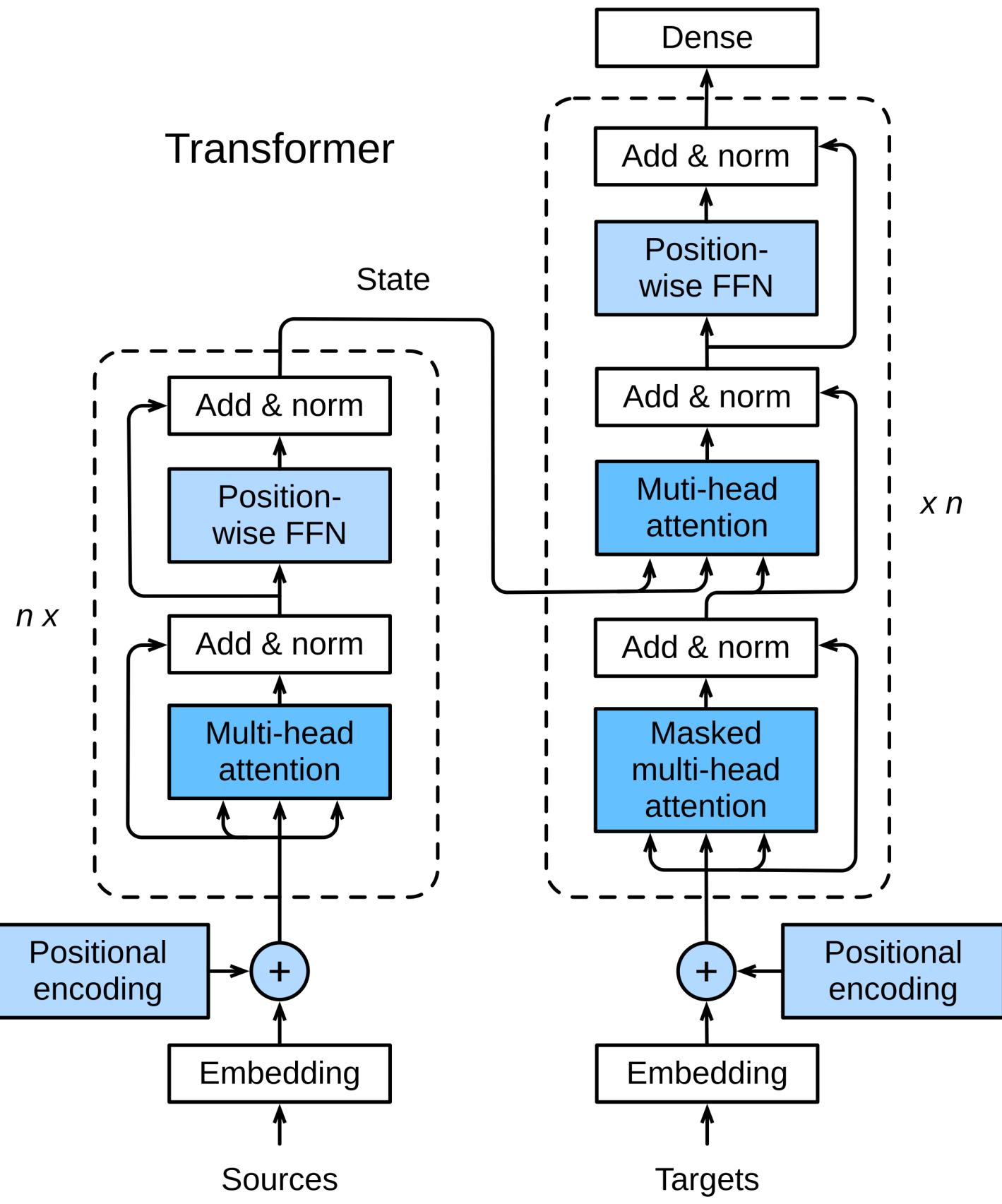


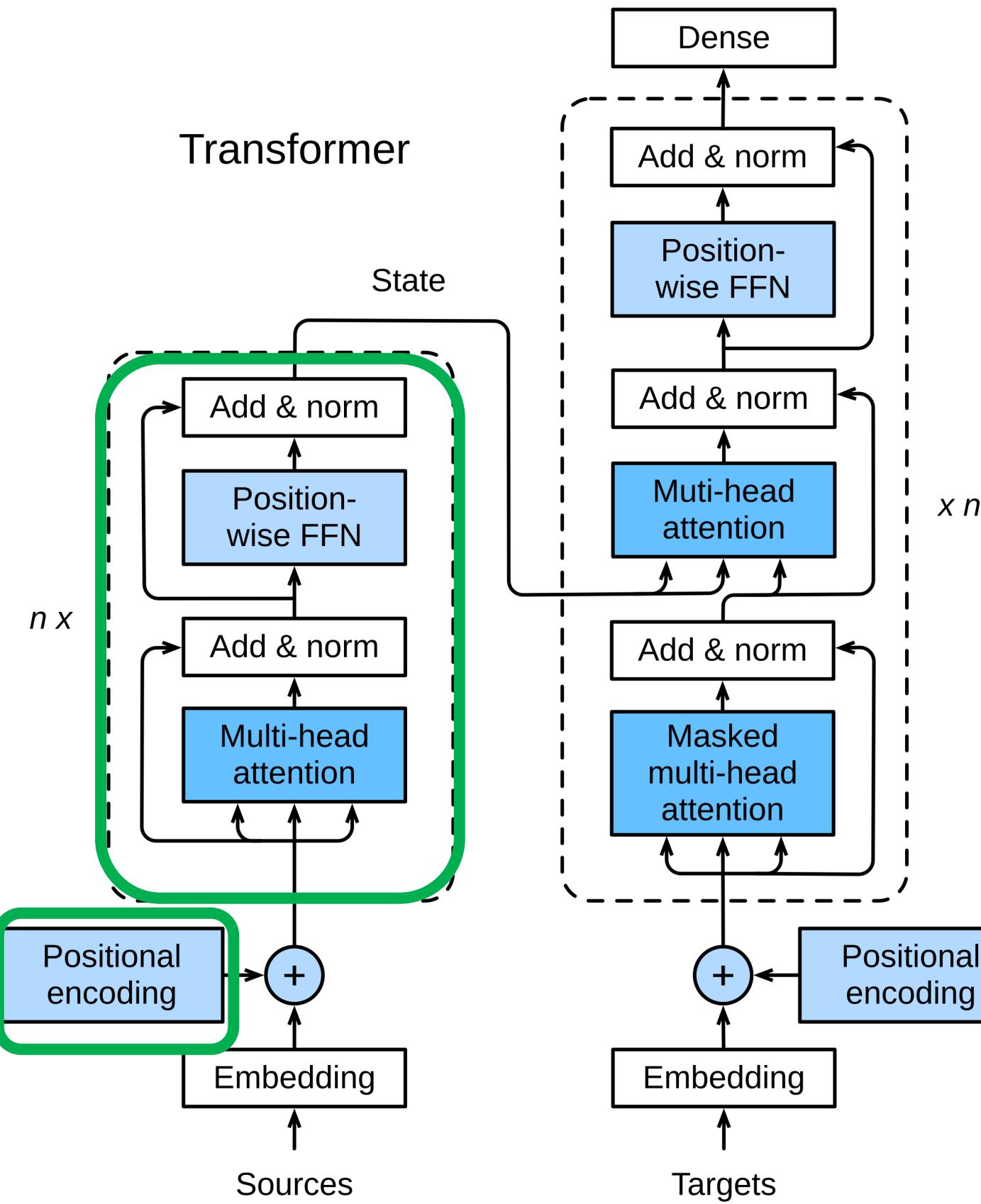
# Transformer\*



- **CNNs:** easy to parallelize, but cannot capture the sequential dependency
- **RNNs:** able to capture the sequential information, but unable to parallelize within a sequence
- **Transformer:** combine the advantages of CNNs and RNNs

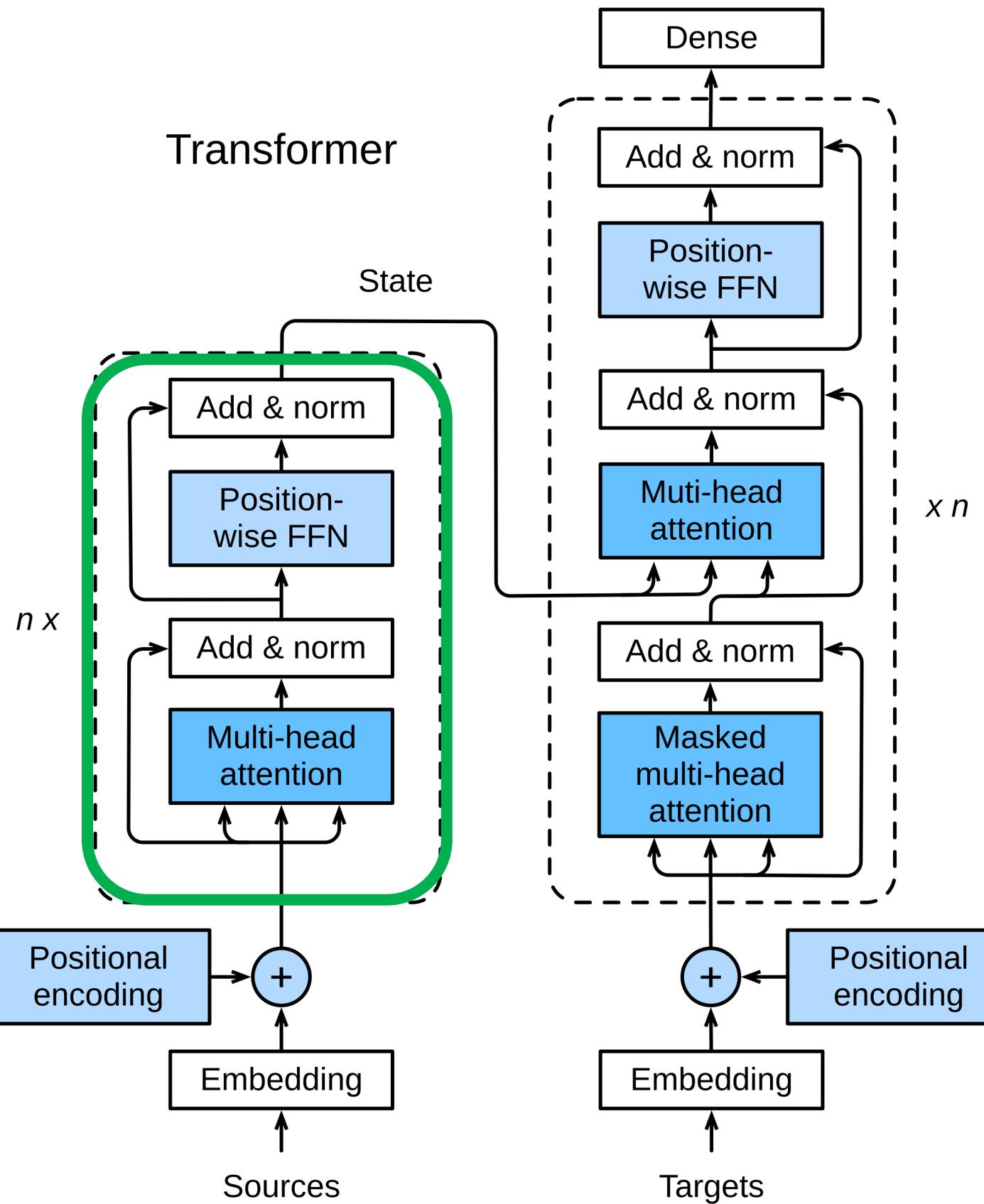
\*Vaswani, et al., 2017 - “attention is all you need”





# Transformer Block

## Positional Encoding



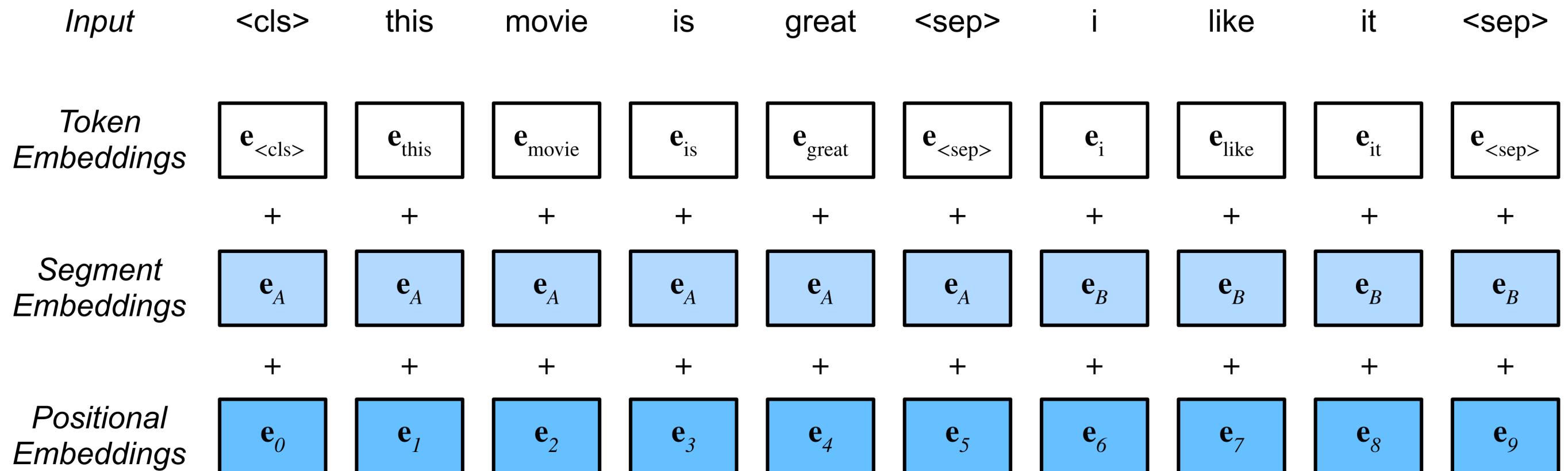
## Transformer Block:

- Multi-head attention
- Position-wise feed forward network (FFN)
- Layer normalization (“Add & Norm”)

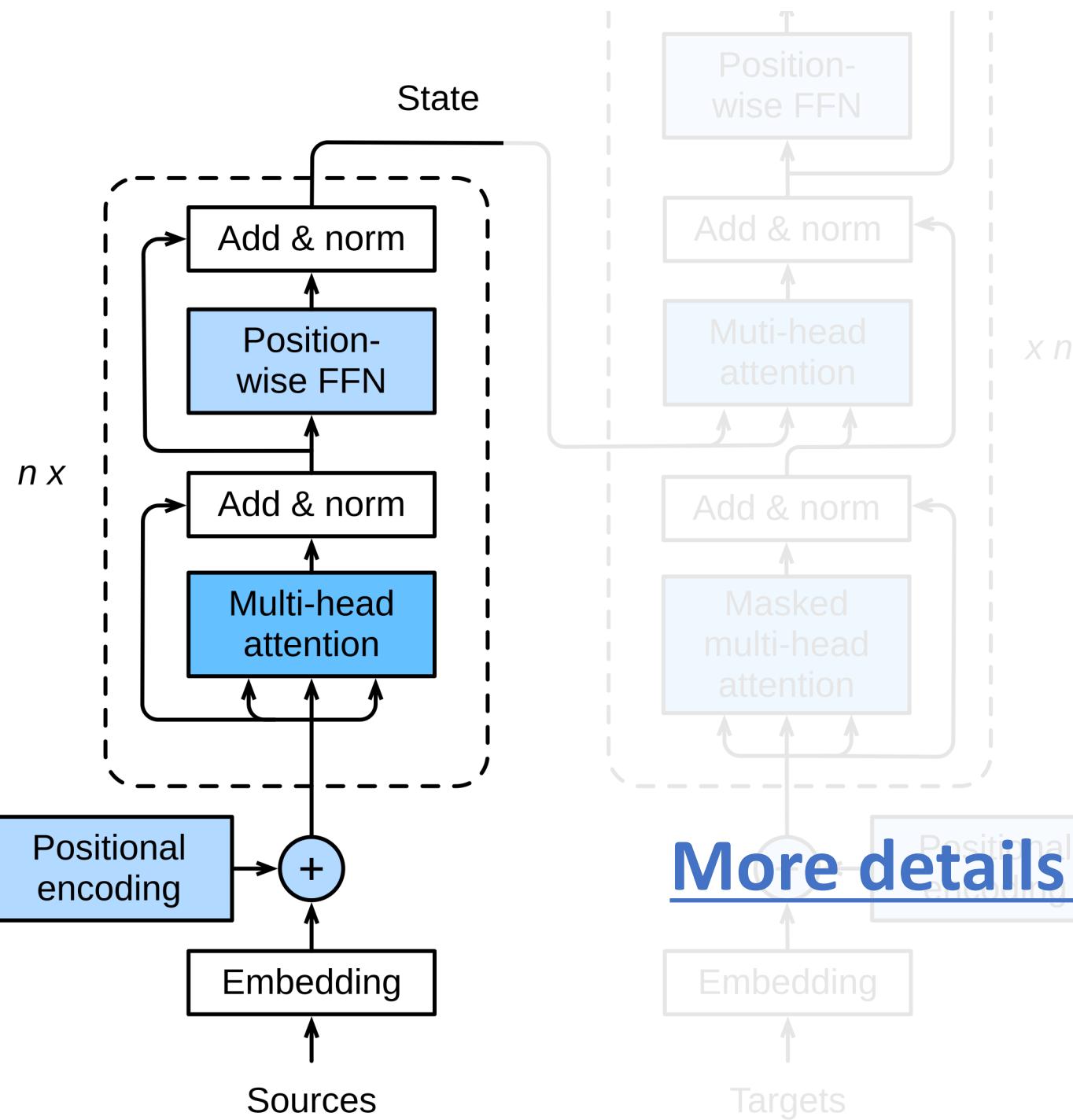
**More details on D2L**

# BERT

- BERT input sequence embeddings



# BERT



Bidirectional  
Encoder  
Representations  
from Transformers

[More details on D2L](#)

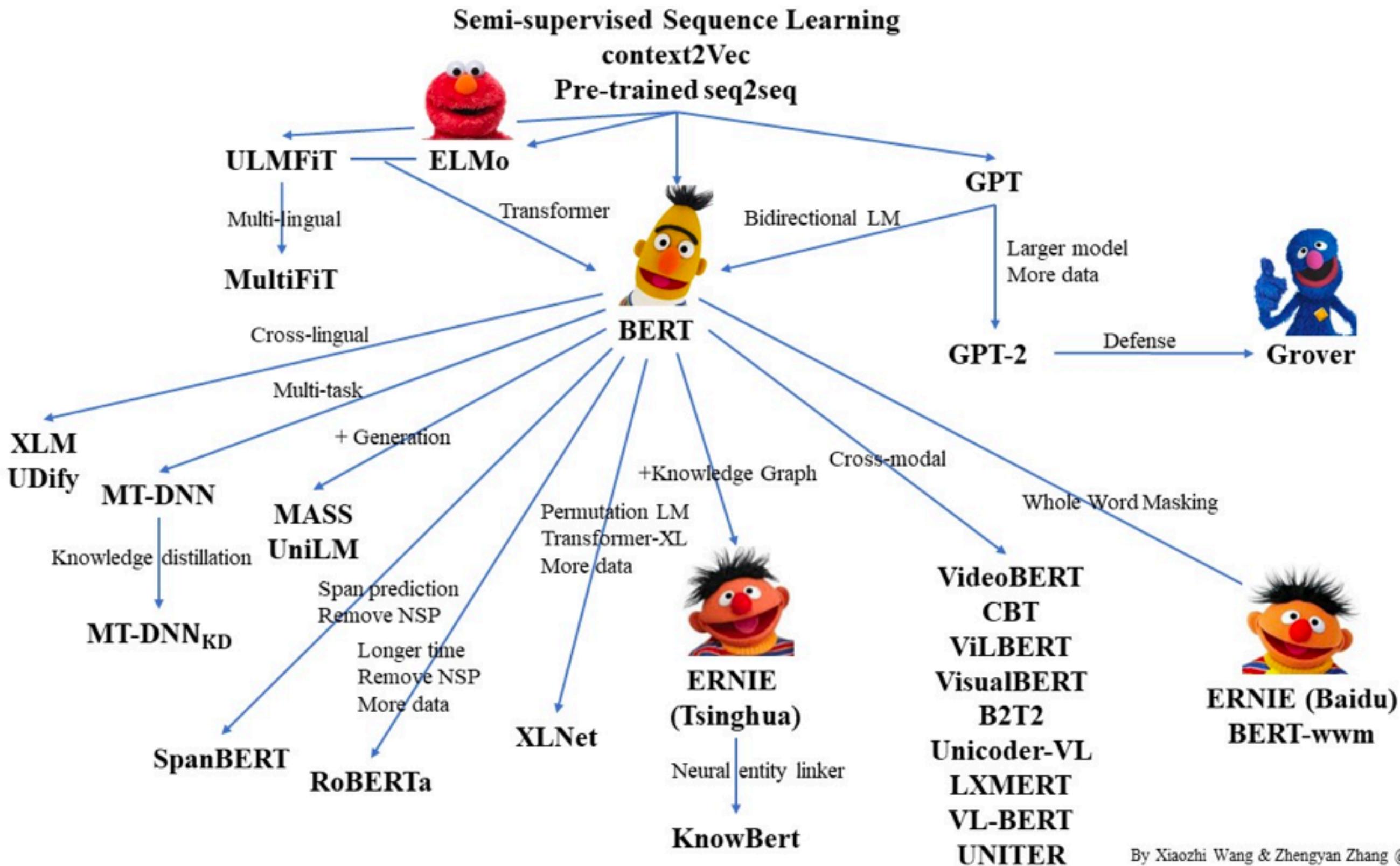


# BERT

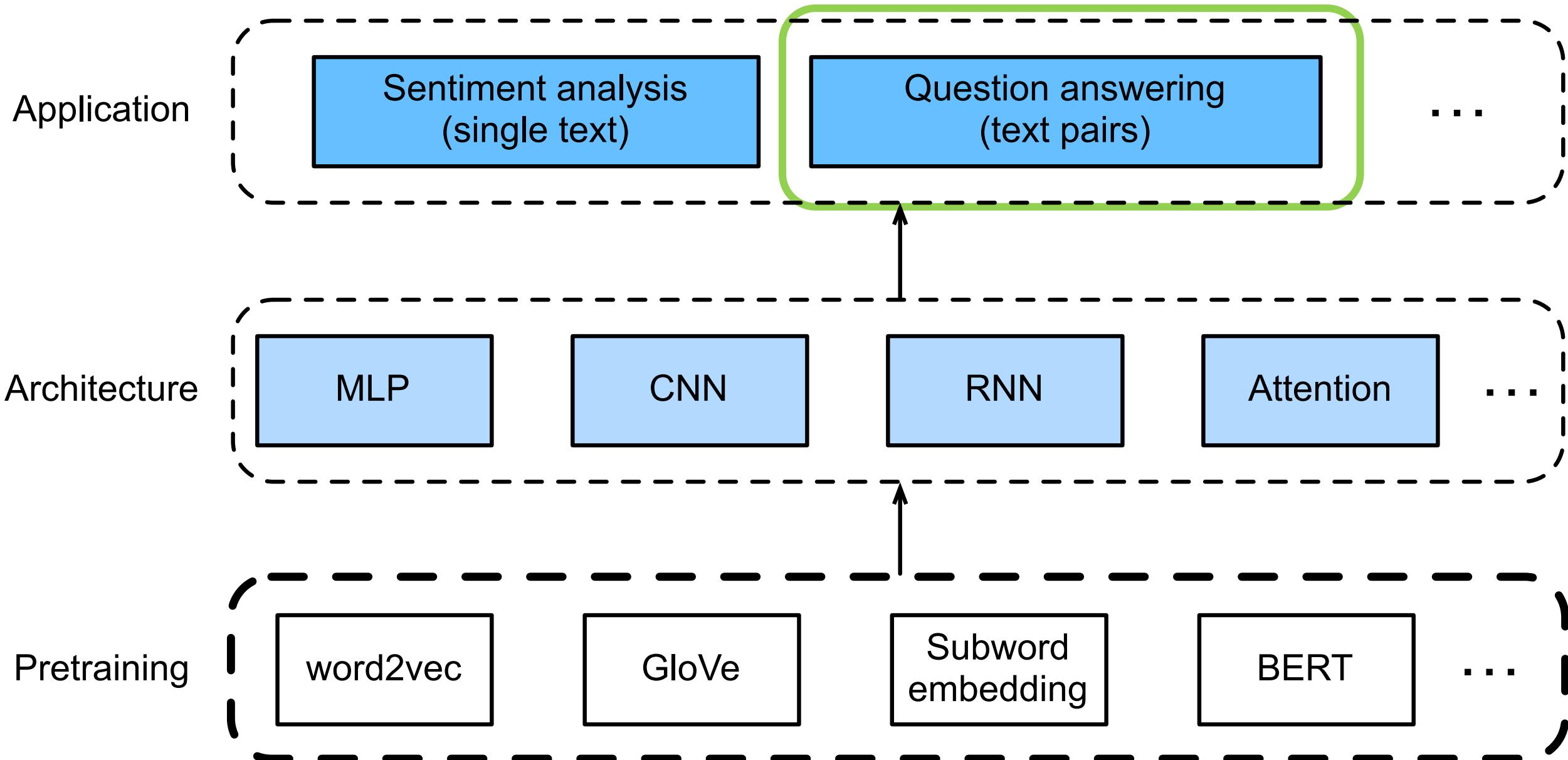
Two Methods of Usages:

- Pretrained:
  - Only train an additional output layer based on the NLP task
  - Minimal model architecture changes
- Finetuned:
  - Train an output layer
  - Tune all parameters based on a specific task and datasets





# NLP Workflow

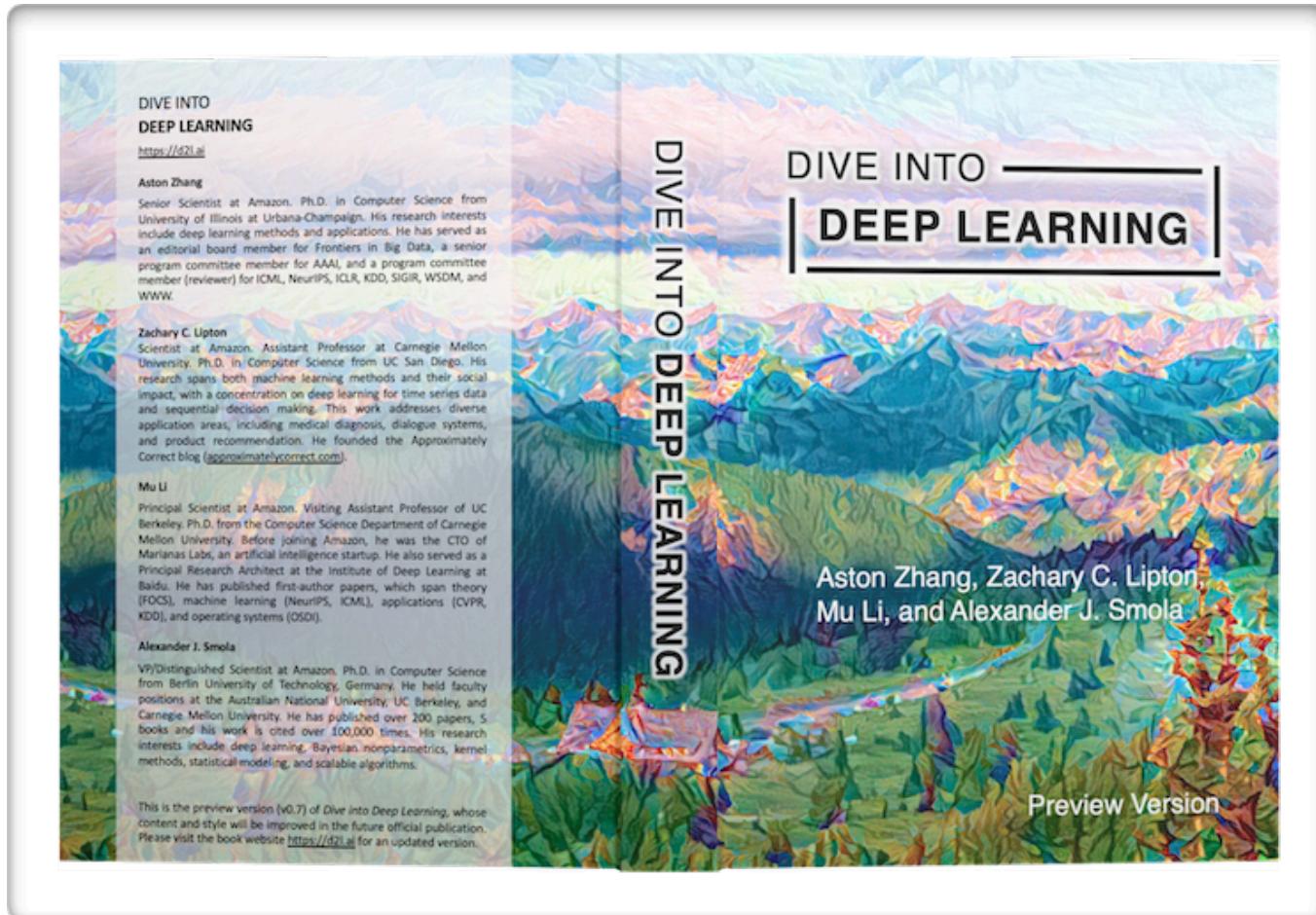


# NLP Application

## Question answering

- **Question:** Who shall use D2L?
- **Passage context:** *Dive into Deep Learning* (D2L, D2L.ai) is an interactive open-sourced deep learning online book with code, math, and discussions. It is designed for engineers, researchers and students to learn state of the art deep learning models with in-depth theory and fast prototype notebooks.

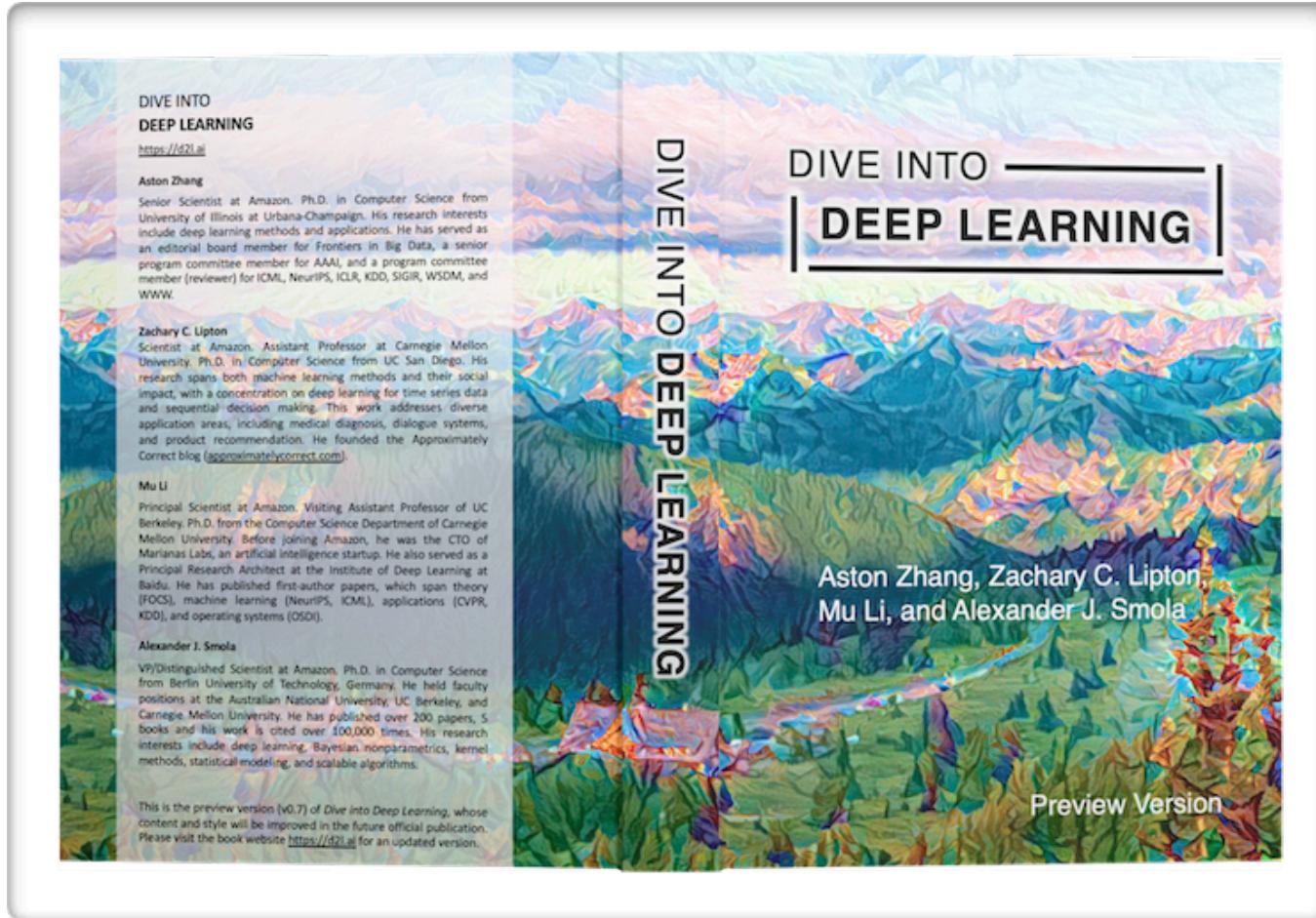
# D2L



[D2L.ai](https://d2l.ai)

- Adopted as a textbook or reference book at UC Berkeley, CMU, MIT, and 70+ universities worldwide
- Berkeley class (slides, videos at [courses.d2l.ai](https://courses.d2l.ai))
- Wide theoretical coverage: statistics, optimization, machine learning basics, GPU parallel training, etc.
- 150+ runnable Jupyter Notebooks from model architectures to applications (CV, NLP, etc.)
- Multilingual content EN, ZH (in progress: KO, JA, FR, TR)

# D2L



[D2L.ai](https://d2l.ai)

*Dive Into Deep Learning* is an excellent text on deep learning and deserves attention from anyone who wants to learn why deep learning has ignited the AI revolution – the most powerful technology force of our time.

— Jensen Huang,  
President and CEO of Nvidia

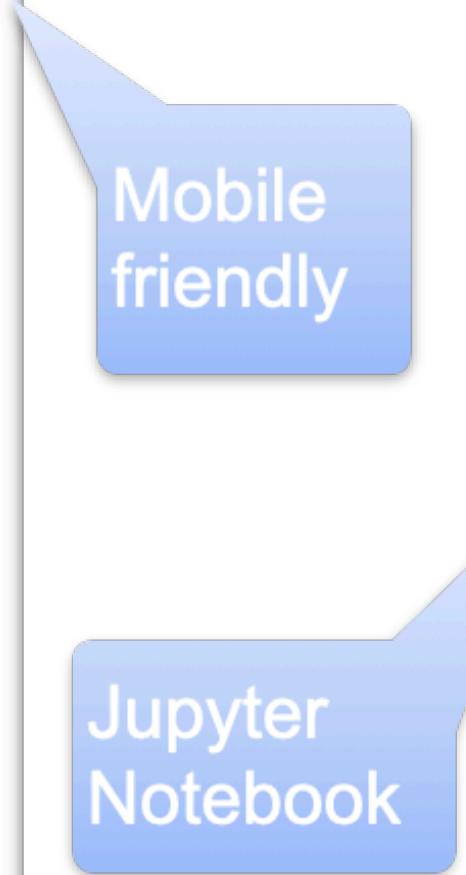
# D2L

A screenshot of a mobile phone screen showing a web page from d2l.ai. The page title is "7.1. Deep Convolutional ...". Below it, a section titled "7.1.2.3. Capacity Control and Preprocessing" is visible. The text discusses AlexNet's use of dropout for capacity control and various preprocessing techniques like flipping, clipping, and color changes. At the bottom, there is a code snippet in Python:

```
import sys
sys.path.insert(0, '...')

import d2l
from mxnet import gluon, init, nd
from mxnet.gluon import data as gdata, nn
import os
import sys

net = nn.Sequential()
```



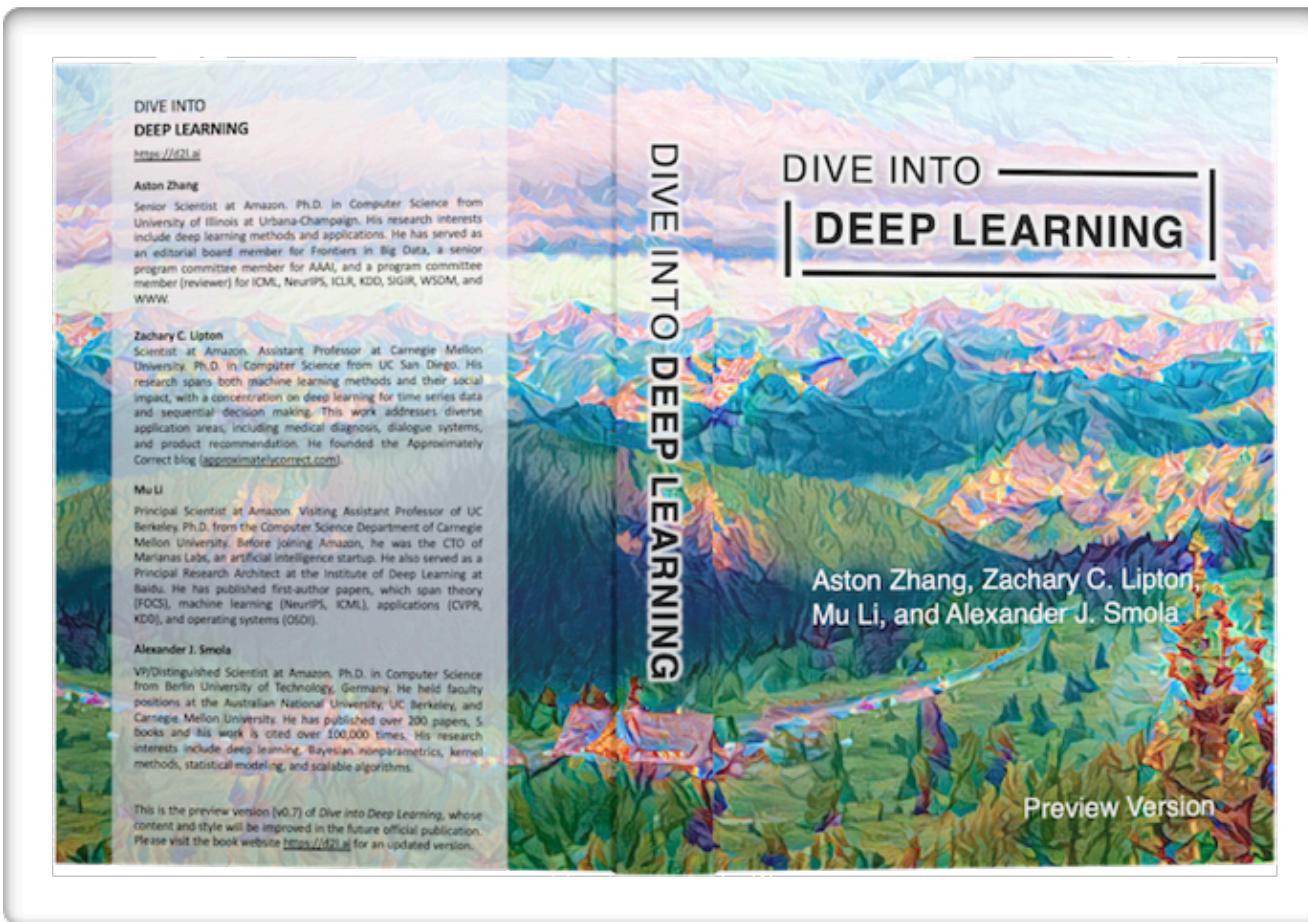
A screenshot of a Jupyter Notebook interface running on localhost. The notebook title is "jupyter alexnet (unsaved changes)". The content area shows two sections: "Activation Functions" and "Capacity Control and Preprocessing", which correspond to the sections in the mobile device screenshot. Below the sections, a code cell labeled "In [1]" contains the same Python code as the mobile device screenshot.

```
In [1]: import sys
sys.path.insert(0, '...')

import d2l
from mxnet import gluon, init, nd
from mxnet.gluon import data as gdata, nn
import os
import sys

net = nn.Sequential()
```

# Gluon Ecosystem



[GluonCV](#)  
Computer Vision

[GluonNLP](#)  
Natural Language

[GluonTS](#)  
Time Series Prediction

[DGL](#)  
Deep Learning on Graphs

[TVM](#)  
Deep Learning Compiler

[MXNet](#)  
Imperative & Symbolic

# Questions ?

## **Tim O'Brien (AWS)**

Sr. Solutions Architect AI/ML  
[tpobrien@amazon.com](mailto:tpobrien@amazon.com)

## **Rachel Hu (AWS)**

Applied Scientist  
[rlhu@amazon.com](mailto:rlhu@amazon.com)

## **Wen-ming Ye (AWS)**

Sr. Solutions Architect AI/ML  
[wye@amazon.com](mailto:wye@amazon.com)

