Lessons learned deploying a deep learning visual search service at scale

Scott Cronin / Senior Data Scientist
ShopRunner

ODSC


IT'S EXHAUSTING
#HOLLYWOODDARLINGS
Pop

/ ShopRunner

DATA PLATFORM

Saks Fifth Avenue

BERGDORF GOODMAN

Neiman Marcus

TORY BURCH

Lord & Taylor

bloomingdale's

/ Visual Search Definition

/ Scale of the Problem

1. Search space: Millions of products
2. Filter results to
   a. Items that are available
   b. Items within a reasonable price range
3. Search time: < 100ms

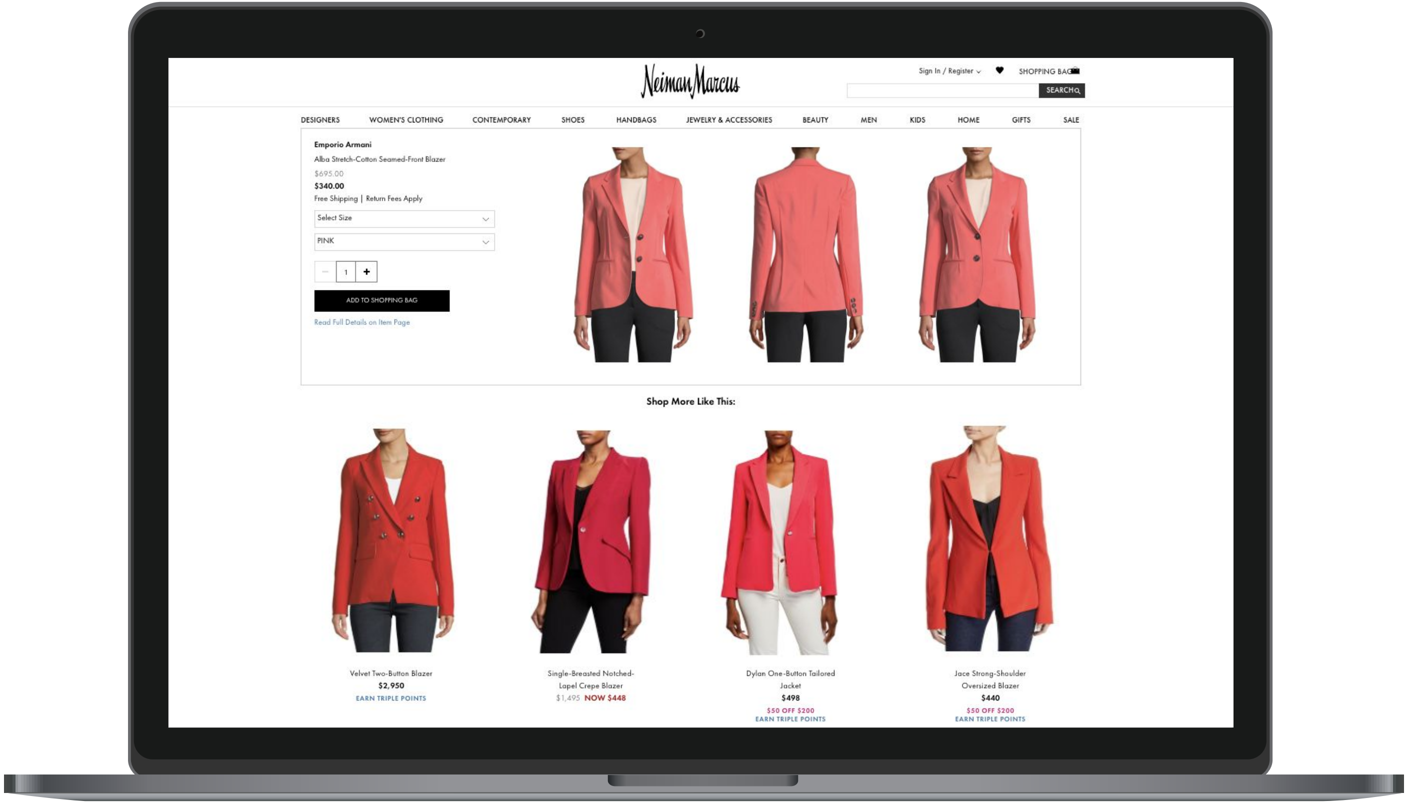
Development

Deep Learning Model

[0.75, -1.40, 0.22, -0.03, …]

Similarity Algorithm

/ Deployment



Neiman Marcus
Sign In / Register
SHOPPING BAG
SEARCH
DESIGNERS
WOMEN'S CLOTHING
CONTEMPORARY
SHOES
HANDBAGS
JEWELRY & ACCESSORIES
BEAUTY
MEN
KIDS
HOME
GIFTS
SALE
Emporio Armani
Alba Stretch-Cotton Seamed-Front Blazer
$695.00
$340.00
Free Shipping | Return Fees Apply
Select Size
PINK
ADD TO SHOPPING BAG
Read Full Details on Item Page
Shop More Like This:
Velvet Two-Button Blazer
$2,950
EARN TRIPLE POINTS
Single-Breasted Notched-Lapel Crepe Blazer
$1,495 NOW $448
Dylan One-Button Tailored Jacket
$498
$50 OFF $200
EARN TRIPLE POINTS
Jace Strong-Shoulder Oversized Blazer
$440
$50 OFF $200
EARN TRIPLE POINTS

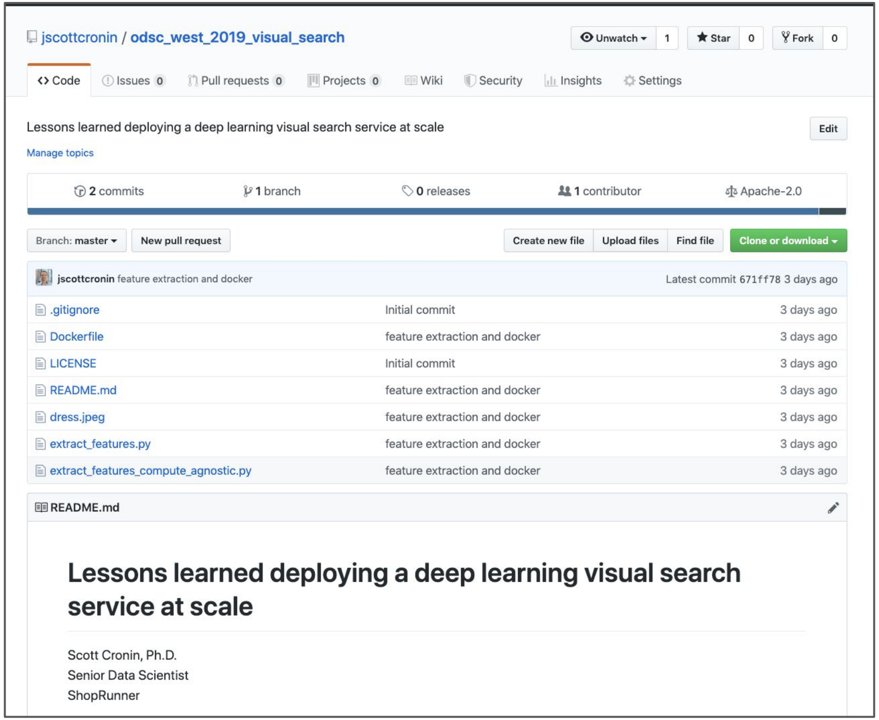
/ Agenda

1. Development
    a. Feature Vector Extraction
    b. Algorithm Development + Evaluation

2. Deployment
    a. In-memory RESTful Microservice
    b. Database-backed RESTful Microservice

/ Resources

https://github.com/jscottcronin/odsc_west_2019_visual_search



I

Development

/ Feature Vector Extraction

Feature Vector Extraction

→ Deep Learning Model → [0.75, -1.40, 0.22, -0.03, …]

1. Want to develop code locally, run on cloud compute

2. Need to scale to millions of images. We'll need to write CPU / GPU agnostic code.

How difficult?

```
import torch
import torch.nn as nn
import torchvision.models as models
import torchvision.transforms as transforms
from torch.autograd import Variable
from PIL import Image
```

```
get_vector('dress.jpeg')
```

```
array([2.78678685e-01, 1.66654658e+00,
       7.94355094e-01, 2.87524796e+00,
       1.90596068e+00, 7.01830804e-01,
       4.53295678e-01, 5.12652397e-01,
       1.08691239e+00, 3.67466509e-01,
       1.01419389e+00, 1.38786221e+00,
       5.25947332e-01, 4.33954932e-02,
       3.92269678e-02, 3.49154502e-01,
       2.38699257e-01, 1.32591993e-01,
       6.23517394e-01, 4.36226547e-01,
       8.99461806e-01, 2.94402272e-01,
       2.48889685e+00, 9.02168266e-03,
       3.20008636e+00, 4.21325922e-01,
       3.07740378e+00, 2.24915370e-01,
       5.18537052e-02, 1.78528237e+00,
       2.62373894e-01, 3.87807220e-01,
       1.65249109e+00, 4.57018092e-02,
       7.07945004e-02, 8.22626412e-01,
```

/ Lesson 1 - Utilize Docker, pin dependencies

https://github.com/jscottcronin/odsc_west_2019_visual_search

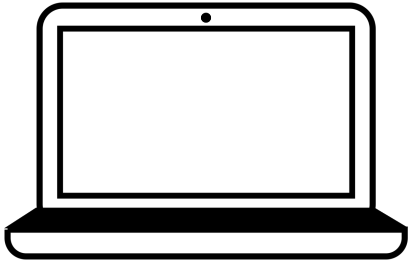
Dockerfile

```
FROM nvcr.io/nvidia/pytorch:19.09-py3
WORKDIR /visual_search
COPY . .
```

scronin@m-scronin-sr10134: ~ (zsh)

```
~ docker build -t visual_search .
~ docker run -it --rm -p 8888:8888
  -v $PWD/visual_search
  visual_search bash
~ jupyter lab --ip 0.0.0.0
  --no-browser --allow-root
~
```

/ Lesson 1 - Utilize Docker, pin dependencies



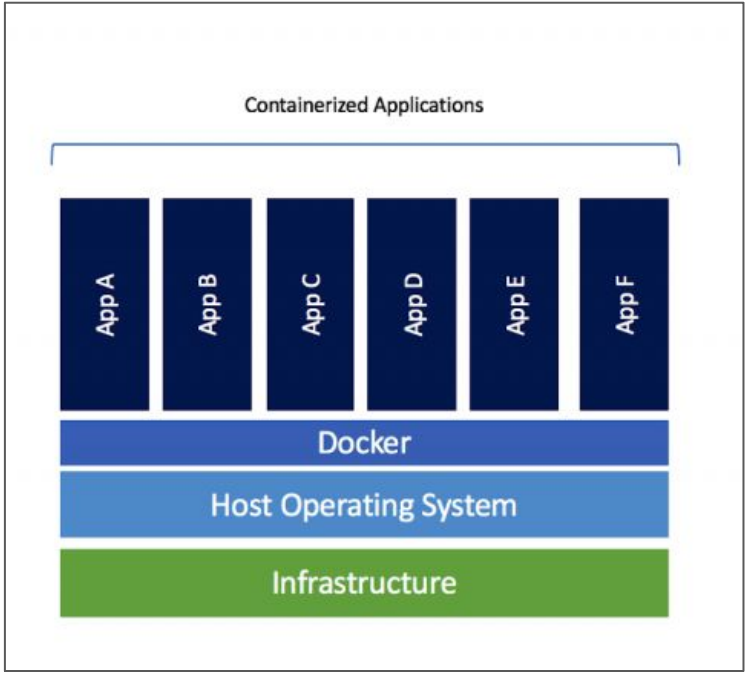
```
from extract_features import get_vector

vec = get_vector('dress.jpeg')

vec[:2]

array([0.27841428, 1.6626568 ], dtype=float32)

vec.shape

(512,)
```


```
from extract_features import get_vector

vec = get_vector('dress.jpeg')

vec[:2]

array([0.27841428, 1.6626568 ], dtype=float32)

vec.shape

(512,)
```

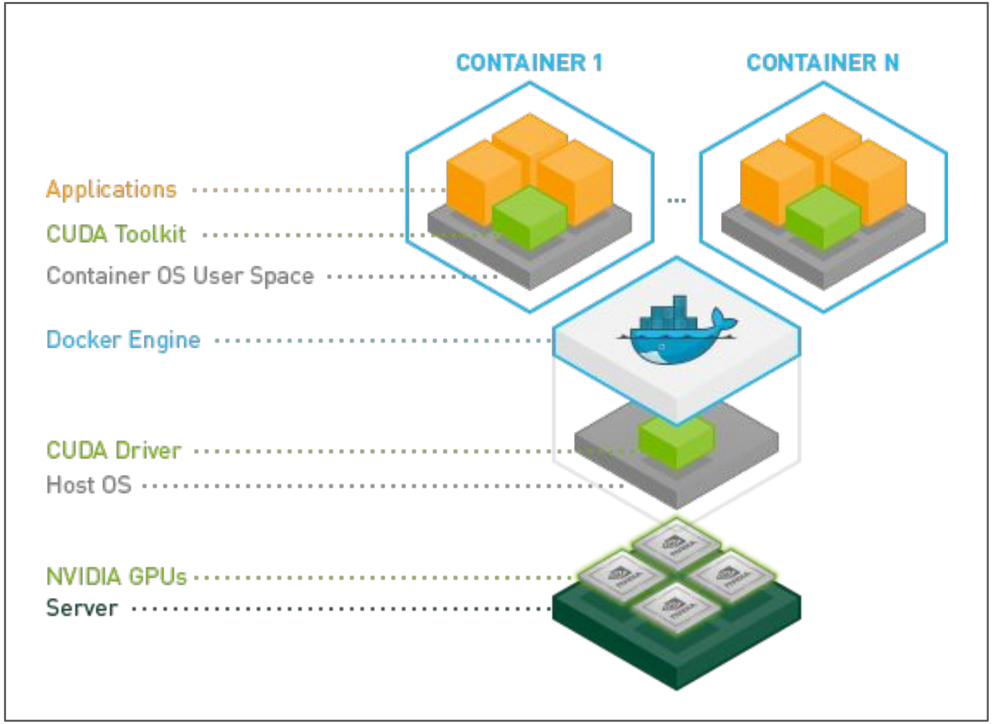

SUCCESS

Lesson 1.5: Use Docker w/ Integrated Cuda Toolkit

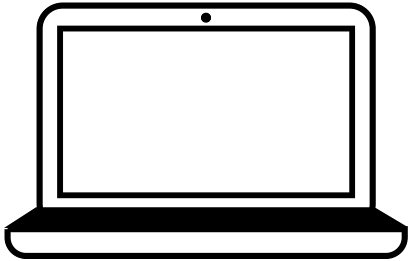
Traditional Docker Architecture



Nvidia Docker Architecture



Lesson 1.5: Use Docker w/ Integrated Cuda Toolkit



kubernetes

aws

scronin@m-scronin-sr10134: ~ (zsh)

```
~ docker run -it --rm -p 8888:8888 -v $PWD/visual_search visual_search bash
```

scronin@m-scronin-sr10134: ~ (zsh)

```
~ nvidia-docker run -it --rm -p 8888:8888 -v $PWD/visual_search visual_search bash
```

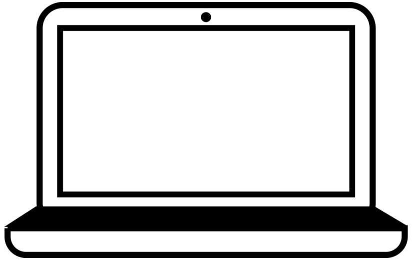
```
[1]: import torch
     torch.cuda.is_available()

[1]: False
```

```
[1]: import torch
     torch.cuda.is_available()

[1]: True
```

/ Lesson 2: Write CPU / GPU Agnostic Code

```
import torch
import torch.nn as nn
import torchvision.models as models
import torchvision.transforms as transforms
from torch.autograd import Variable
from PIL import Image

model = models.resnet18(pretrained=True)

device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
model.to(device).eval()

layer = model._modules.get('avgpool')

scaler = transforms.Scale((224, 224))
normalize = transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])
to_tensor = transforms.ToTensor()

def get_vector(image_name):
    img = Image.open(image_name)
    t_img = Variable(normalize(to_tensor(scaler(img))).unsqueeze(0))
    t_img = t_img.to(device)

    # Create hook to extract vector
    my_embedding = torch.zeros(512)
    def copy_data(m, i, o):
        if device == 'cuda':
            my_embedding.copy_(o.data.resize(512).cpu())
        else:
            my_embedding.copy_(o.data.resize(512))
    h = layer.register_forward_hook(copy_data)

    model(t_img)
    h.remove()

    return my_embedding.numpy()
```

/ Lesson 2: Write CPU / GPU Agnostic Code



kubernetes aws

```
from extract_features_compute_agnostic import get_vector
```

```
%timeit get_vector('dress.jpeg')
```

```
29.8 ms ± 2.63 ms per loop (mean ± std. dev.
of 7 runs, 10 loops each)
```

```
from extract_features_compute_agnostic import get_vector
```

```
%timeit get_vector('dress.jpeg')
```

```
4 ms ± 57.2 µs per loop (mean ± std.
dev. of 7 runs, 100 loops each)
```


gifbin.com

/ Lesson 3: Don’t repeat the hard stuff

Feature Extraction History

Row	IMAGE_URL_SKU_472_472	MODEL_NAME	MODEL_ID	SNAPSHOT_DATE
1	http://net.shoprunner.prd.m...	tax_image_model	tax_image_model__20190501_130940	2019-08-12-17:37:56
2	http://net.shoprunner.prd.m...	tax_image_model	tax_image_model__20190501_130940	2019-08-12-17:37:56
3	http://net.shoprunner.prd.m...	tax_image_model	tax_image_model__20190501_130940	2019-08-12-17:37:56
4	http://net.shoprunner.prd.m...	tax_image_model	tax_image_model__20190501_130940	2019-08-12-17:37:56
5	http://net.shoprunner.prd.m...	tax_image_model	tax_image_model__20190501_130940	2019-08-12-17:37:56
6	http://net.shoprunner.prd.m...	tax_image_model	tax_image_model__20190501_130940	2019-08-12-17:37:56
7	http://net.shoprunner.prd.m...	tax_image_model	tax_image_model__20190501_130940	2019-08-12-17:37:56
8	http://net.shoprunner.prd.m...	tax_image_model	tax_image_model__20190501_130940	2019-08-12-17:37:56

/ Lesson 4: Isolate work in pipelines

Model 1

Model 2

Model 3

model_names

```
{
    'vgg16',
    'resnet50',
    'inception_v3',
    'attribute_image',
    'tax_image_model',
    ...
}
```

for model_name in model_names:

1. Load deep learning model
2. Find all non-processed product image urls
3. Download images
4. Pass images through deep learning model
5. Persist feature vectors on cloud storage
6. Update database with completed work

II

Development

/ Algorithm Development + Evaluation

/ Lesson 5: Define a Similarity Algorithm

[0.32, 0.79, -0.03, …]

[0.36, 0.71, -0.12, …]

Spotify

Annoy
Approximate Nearest
Neighbors (Oh Yeah!)

/ Use the same similarity algo through testing

Lesson 6: Be scrappy; find a way to evaluate

Challenges we faced:

a. Still developing the app that this tool would go in
b. Wanted to validate algo before showing to customers

/ Lesson 6: Be scrappy; find a way to evaluate

cloudfactory

HIVE

figure eight

amazon mechanical turk

samasource

Lesson 6: Be scrappy; find a way to evaluate

Lesson 6: Be scrappy; find a way to evaluate

/ Lesson 6: Be scrappy; find a way to evaluate

1. 2 day turnaround
2. $100 to get significance
3. No impact to our customers

Lesson 7: Iterate

1. Try off-the-shelf deep learning models
2. Try internal deep learning models
3. Hybridize multiple models together

Lesson 7: Iterate

Off the Shelf Deep Learning Models

/ Lesson 7: Iterate

Internal Color / Pattern Model

Lesson 7: Iterate

Internal Taxonomy Model

Lesson 7: Iterate

Hybridized Color/Pattern + Taxonomy Model

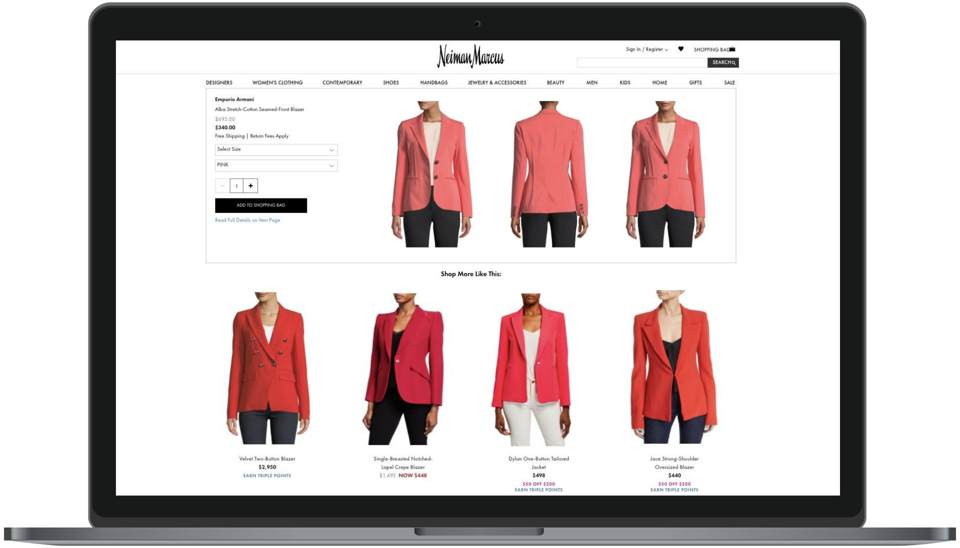
III

Deployment

/ In-memory RESTful Microservice

/ Deployment Requirements

1. RESTful API to abstract complexity
2. Scalable to millions of images
3. Low latency < 100 ms
4. Filter to relevant products
   - Available
   - Right size
   - Right price point
5. Easy to add new products
6. SIMPLICITY

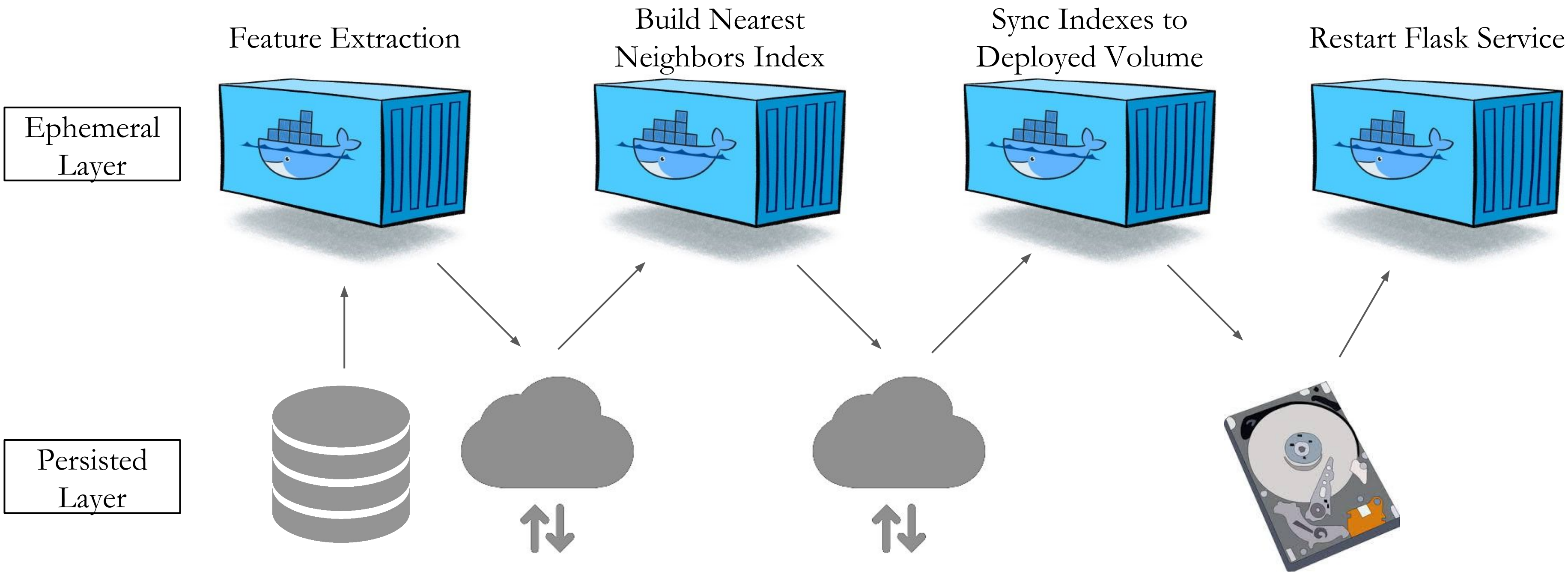


/ In-memory RESTful Microservice

1. Need job manager such as Kubernetes, Airflow, Argo to build Nearest Neighbor Index, and load into Flask App

2. Need all components of code to be dockerized

Lesson 8 - Utilize Dockerized DAG Pipelines




gifak.net

Pros and Cons of Deployment Architectures

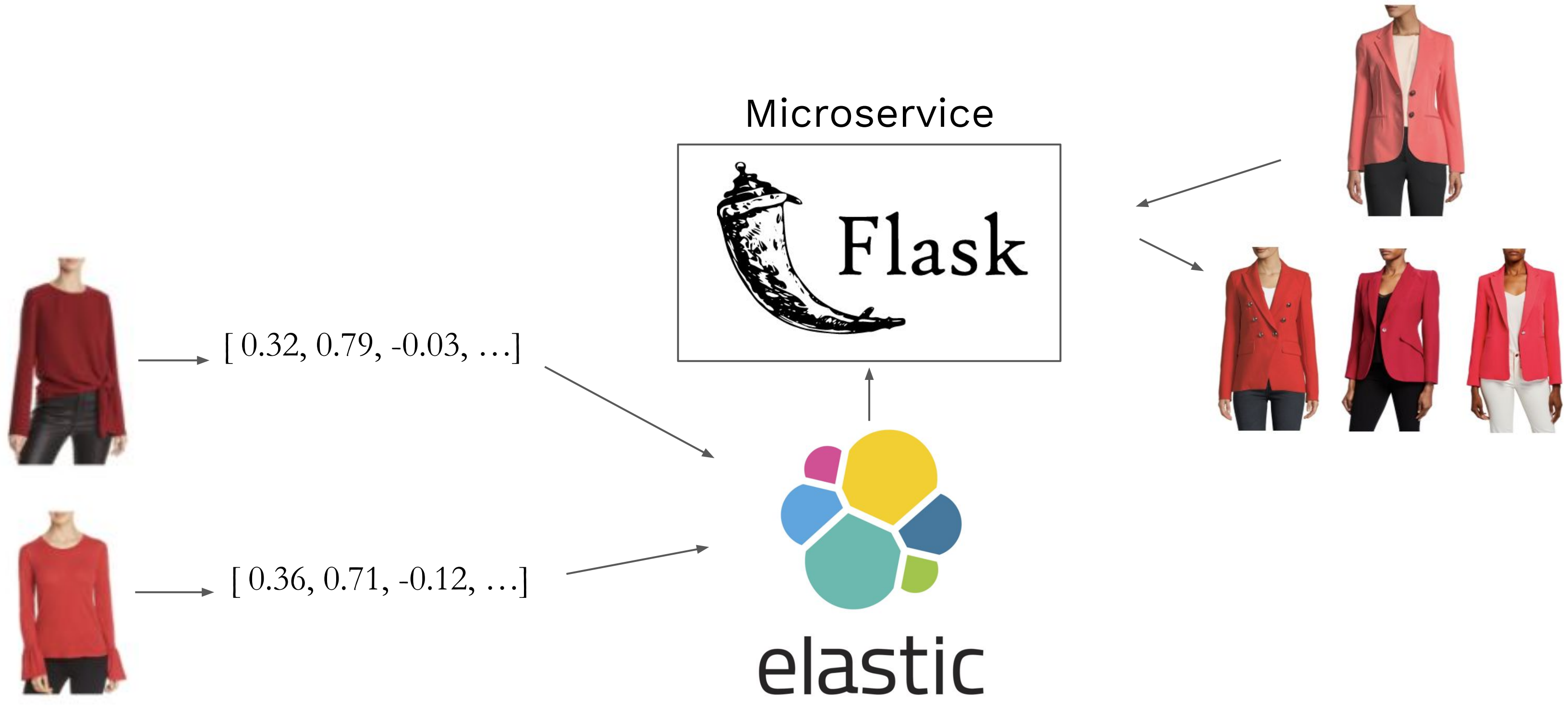
	In-Memory Flask App (using annoy library)
RESTful API to abstract complexity	
Scalable to millions of images	
Latency < 100ms	
Product Filtering on results	
Add new products to search in real time?	
Simplicity	

IV

Deployment

/ Database-backed RESTful Microservice

/ Database-backed RESTful Microservice



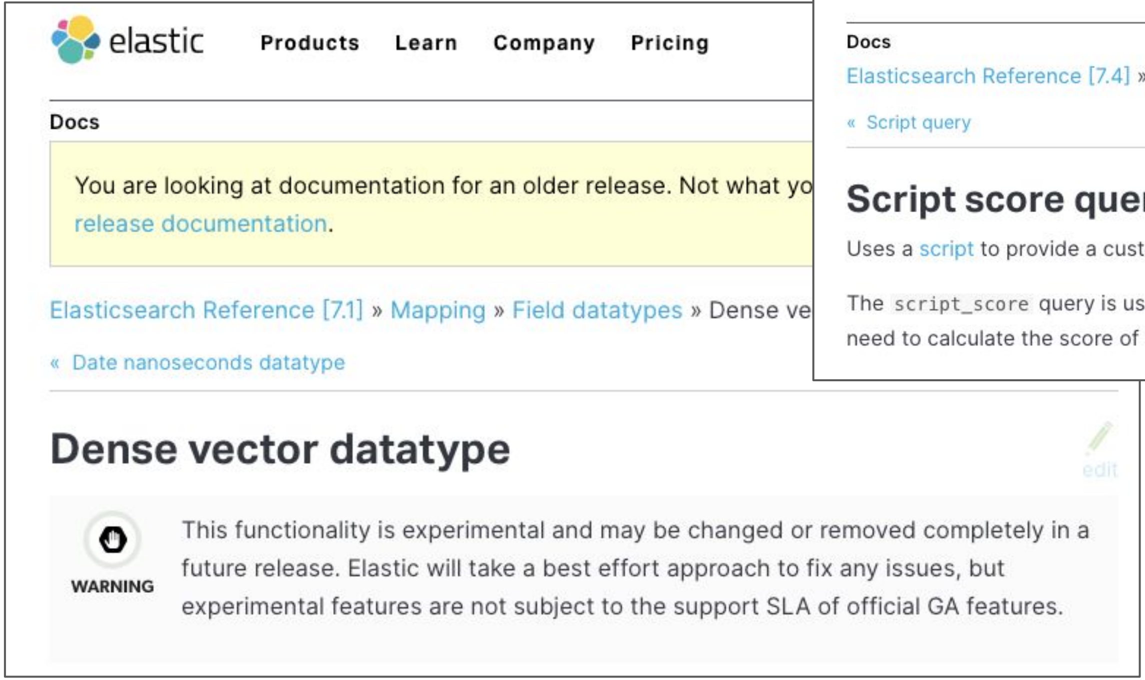
Why Elastic?

1. Our product catalog is hosted there
    - Source of truth for all product information
    - Utilize any filter: price, availability, sizing, ...
2. Horizontally Scalable and battle tested
    - Database for wikipedia, github, ...
3. Easy to add new products as they come

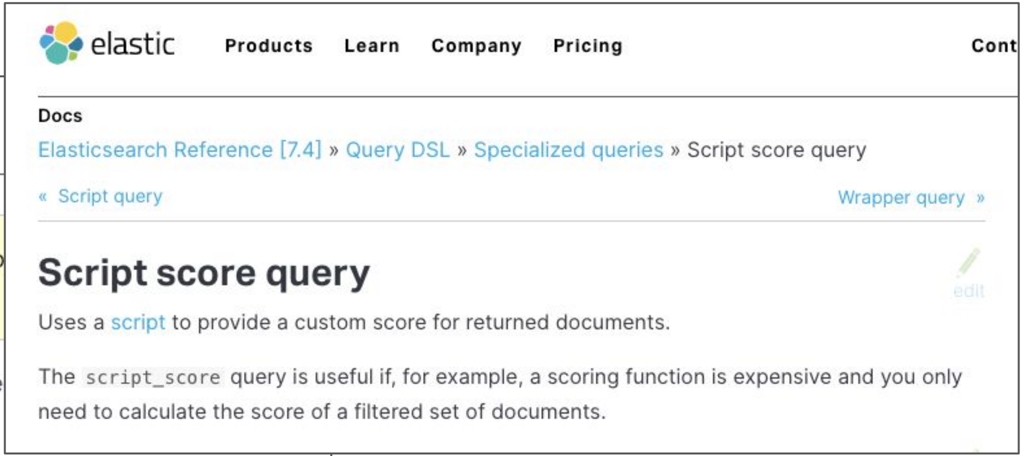
Can Cosine Similarity be applied in real time with low latency?

/ Lesson 9: Keep up with literature / newsletters / blogs / Industry

Elastic 7.1, May 2019



Elastic 7.4, October 2019



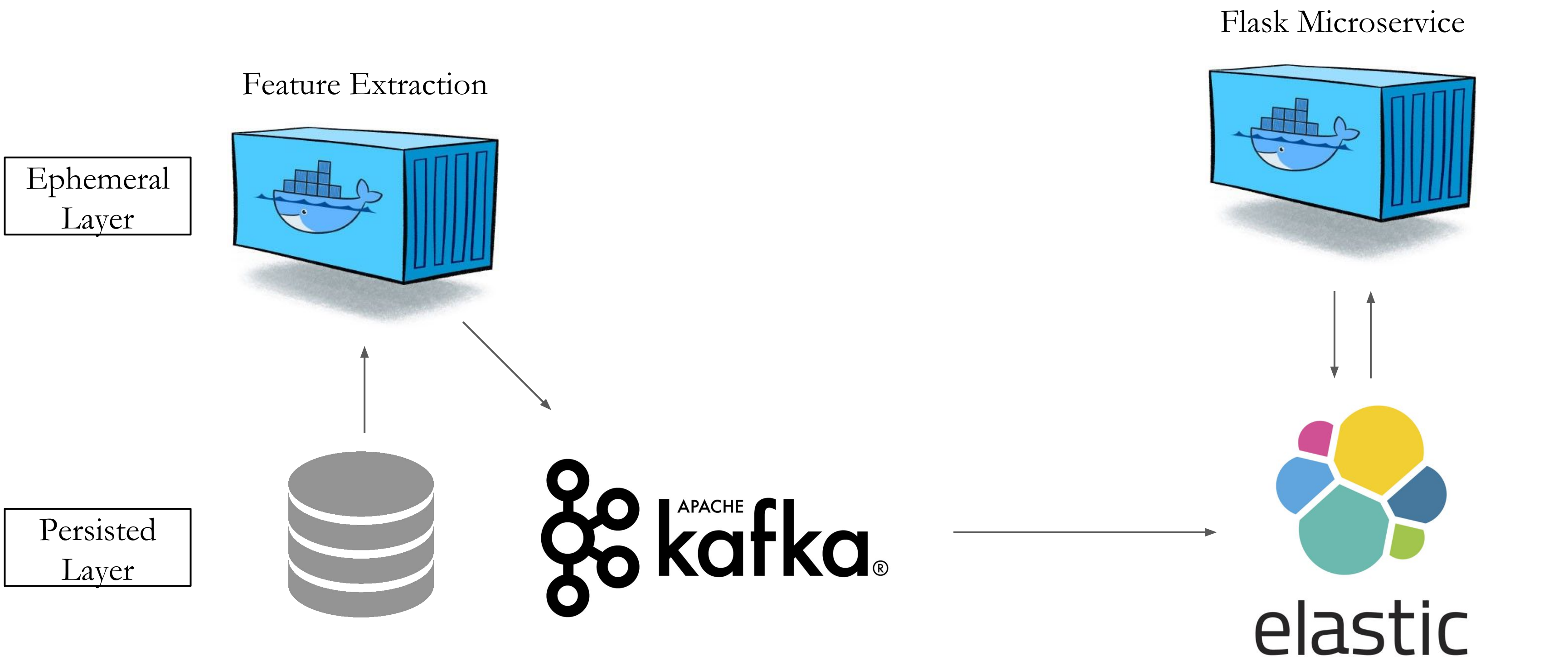
/ Lesson 10: POC before committing

/ Lesson 11: Load test to quantify scalability

Image Search Space	Latency
10K images	10ms
100k images	20ms
1M images	320ms

** Based on embedding vector length 40

/ Lesson 12: Separate ETL from Deployed Microservice

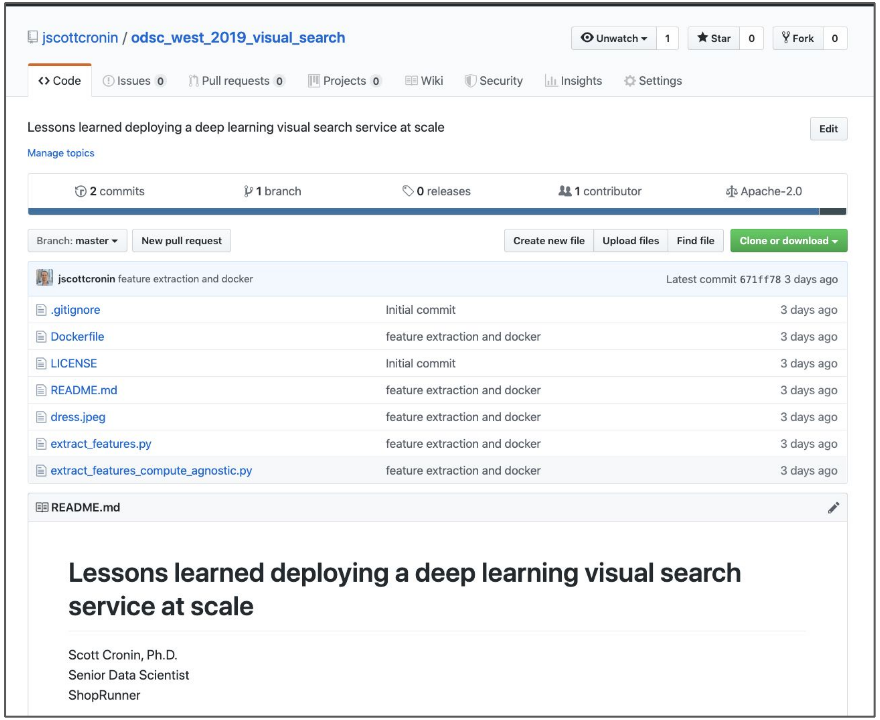


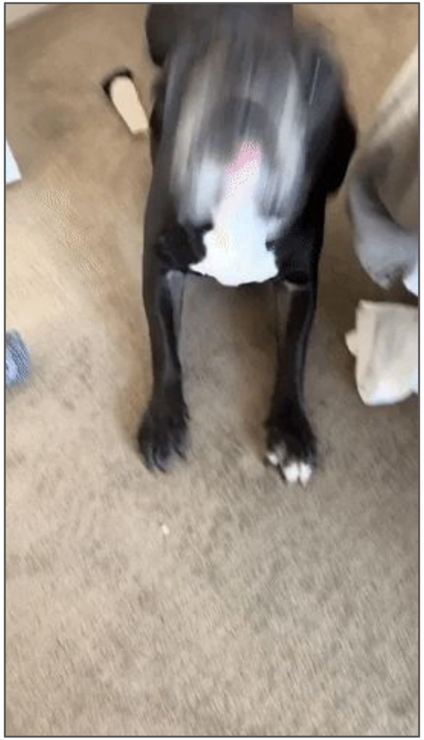
Pros and Cons of Deployment Architectures

	In-Memory Flask App (using annoy library)	Database-backed Flask App (using Elastic)
RESTful API to abstract complexity	✓	
Scalable to millions of images	✓	
Latency < 100ms	✓	
Product Filtering on results	• Filtering is post recommendations • Requires hops to other services	
Add new products to search in real time?	✗	
Simplicity	✗	

Thank You!

https://github.com/jscottcronin/odsc_west_2019_visual_search





Thank You