A Novel Solution for Data Augmentation in NLP using Tensorflow 2.0

K.C. Tung, PhD (kc.tung@microsoft.com)  
Cloud Solution Architect  
Microsoft US, Dallas TX

October 31, 2019

tensorflow.world  
#TFWorld
Goals for this talk

- Demonstrating a path to faster model development
- Bridging gap between development and real-time scoring in cloud
Outline

• Problem and motivation
• Proposed solutions and outcome
• Reference architecture for deployment
• Lesson learned
Aircraft Maintenance
Standard procedure

- Aircraft enters maintenance phase.
- Crews, inspectors, mechanics identify repairs needed.
- Current solution: rule-based model classifies repair types.
- Problem: intractable model update and misclassification.
- Solution required: machine learning model for text classification.
- Deployment requirement: cloud and at scale.
### Maintenance and repair works

<table>
<thead>
<tr>
<th>25</th>
<th>Equipment / Furnishings</th>
<th>32</th>
<th>Landing Gear</th>
<th>73</th>
<th>Engine Fuel &amp; Control</th>
<th>74</th>
<th>Ignition</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>00 General</td>
<td></td>
<td>00 General</td>
<td></td>
<td>00 General</td>
<td></td>
<td>00 General</td>
</tr>
<tr>
<td></td>
<td>10 Flight Compartment</td>
<td></td>
<td>10 Main Gear &amp; Doors</td>
<td></td>
<td>10 Distribution</td>
<td></td>
<td>10 Distribution</td>
</tr>
<tr>
<td></td>
<td>20 Passenger Compartment</td>
<td></td>
<td>20 Nose Gear / Tail Gear &amp; Doors</td>
<td></td>
<td>20 Controlling</td>
<td></td>
<td>20 Electrical Power Supply</td>
</tr>
<tr>
<td></td>
<td>30 Buffet / Galley</td>
<td></td>
<td>30 Extension &amp; Retraction</td>
<td></td>
<td>30 Indicating</td>
<td></td>
<td>20 Distribution</td>
</tr>
<tr>
<td></td>
<td>40 Lavatories</td>
<td></td>
<td>40 Wheels &amp; Brakes</td>
<td></td>
<td>30 Switching</td>
<td></td>
<td>20 Distribution</td>
</tr>
<tr>
<td></td>
<td>50 Cargo Compartments</td>
<td></td>
<td>50 Steering</td>
<td></td>
<td></td>
<td></td>
<td>30 Switching</td>
</tr>
<tr>
<td></td>
<td>60 Emergency</td>
<td></td>
<td>60 Position &amp; Warning, and Ground Safety Switch</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>70 Accessory Compartments</td>
<td></td>
<td>70 Supplementary Gear - Skis, Floats</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>80 Insulation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Works are classified by Air Transport Association code (ATA code)
- 1,091 unique four-digits codes.

[aviationmaintenancejobs.aero/aircraft-ata-chapters-list](http://aviationmaintenancejobs.aero/aircraft-ata-chapters-list)
<table>
<thead>
<tr>
<th><strong>Emergency Equipment</strong></th>
<th><strong>Engine Air Inlet</strong></th>
<th><strong>Engine Compressor</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>'kit used mel authorized 540954 a', 'seal broken first aid kit used mel', 'mel authorized 53923949 nee', 'authorized 047502 used 1 overhead bin', 'missing fire extinguisher due replace', 'used benadrill mel 25 18 authorize', 'broken aft f j door 41ant1 2 authori', 'ref 2521b one first aid kit unusa', 'medical kit unusable flight medical m', 'aid kit lifefest &lt;UNUSED&gt; 7f miss', 'flashlight seal broken', 'aft left dog house operative lh o', 'ng mask door 2 req', 'door action holder broken missing ', 'first aid kit tamper seal broken ', 'cabin first aid kit missing broken ', 'emergency equipment found flashlight in', 'contents used 20', 'flight medical kit used mel authoriz',</td>
<td>'lower 1230 position exact fac', 'areasures 5 3' width 35 16liner 3 o', 'uppered 1 4 inches leading edge ma', 'plastic wire mesh damage 13024 34 11', 'hours e metal missing &lt;NNK&gt; found o', 'wide 13 o'clock 2 e guide', 'finlet cowl anbleel skin 5 danel', 'card fack p sh tagh missing 12 ocl', '7169682 time 52593 14 cycles 100', 'filler missing seterial 4 crack crew', 'due jclock pos missing requires i', '20862865 recairs fircraters 8 o''', 'barrel wire mesh delt 10 0 1 rub 6', 'screwers 2 lighe shroud senures eng', 'lining 1 8lonoer 5 oclock position 1', 'installed servicevalue ncr 2 engine bet', 'needs fan shred sh wilh around foun', 'reported struck tower 123clock positio', 'lh 4 10 rea 5 44 13 cycles 2',</td>
<td>'engine wash 72 96 11 engine esn 7334', 'engine lan 71 ref amm 71 21 148ar', '1489 queshed 291', '2168201 b repoved eng acoust', 'cycles 7042 reviewed rhy fwd spless', 'time 15576 49 cycles 9401 j1 55874', 'wash pormance require complet pane', 'position 13 th hest pelt engine esn 73', '00218snox s43 ongen coridin', '7230 &gt; 13 found crack befinle', 'compressor fright cycles 7894 blende 82', 'requires compressors ritem ch ipectio', 'fan crack 1 ea damaged', 'amm unateryll peened fan bact bin', 'ref's isc empower wash required', 'card 4909 eng 4 bow damage', '679866 28 cycles 7239 amm 72 31', 'rhing misson rh eng fan blade', 'dueshor 2 eng spinner 5410 nt 501',</td>
</tr>
</tbody>
</table>
Distribution of repair categories

In-Flight Entertainment System
Original corpus specs

- File size: 56MB
- \( n: 883,651 \)
- Target: 1091 classes, most frequent repair is In-Flight Entertainment System (\( n=59930 \))
- Class of interest: emergency equipment (\( n=7778 \)), engine air inlet (\( n=699 \)), engine compressor (\( n=546 \))
- Training size: 565536
- Validation size: 141384
- Holdout size: 176731
2. Proposed solution and outcome
Options for classifying imbalanced corpus

- Sampling techniques – requires little data science endeavor but was not effective.
- Cost function modification* – requires deep data science endeavor (see 2018 publication for focal loss), effectiveness unsure for text, long development and test time.
- Synthesize new corpus – requires some data science endeavor, short development and test time (‘fail fast’), many examples available.

Train a model to generate new corpus

Corpus of a minority category i.e., engine air inlet

Rank each unique word as seed

Build and train model

Model

New corpus

tensorflow.org/tutorials/text/text_generation
From old corpus to new corpus

TensorFlow.org/tutorials/text/text_generation
Text generation training design

Lorem ipsum dolor sit amet,

<table>
<thead>
<tr>
<th>Layer (type)</th>
<th>Output Shape</th>
<th>Param #</th>
</tr>
</thead>
<tbody>
<tr>
<td>embedding_2 (Embedding)</td>
<td>(64, None, 256)</td>
<td>12032</td>
</tr>
<tr>
<td>gru_2 (GRU)</td>
<td>(64, None, 1024)</td>
<td>3935232</td>
</tr>
<tr>
<td>dense_2 (Dense)</td>
<td>(64, None, 47)</td>
<td>48175</td>
</tr>
</tbody>
</table>

Total params: 3,995,439
Trainable params: 3,995,439
Non-trainable params: 0

orem ipsum dolor sit amet, c
CNN model for text classification

```
MAX_SENTENCE_LENGTH = 200
VOCAB_SIZE = 150000
EMBEDDING_SIZE = 128

model = tf.keras.Sequential()
model.add(tf.keras.layers.Embedding(VOCAB_SIZE, EMBEDDING_SIZE, input_length=MAX_SENTENCE_LENGTH))
model.add(tf.keras.layers.Conv1D(filters=128, kernel_size=5, strides=1, padding='same', activation='relu'))
model.add(tf.keras.layers.MaxPooling1D(pool_size=2, strides=2, padding='same'))
model.add(tf.keras.layers.Conv1D(filters=256, kernel_size=5, strides=1, padding='same', activation='relu'))
model.add(tf.keras.layers.MaxPooling1D(pool_size=2, strides=2, padding='same'))
model.add(tf.keras.layers.Conv1D(filters=512, kernel_size=5, strides=1, padding='same', activation='relu'))
model.add(tf.keras.layers.MaxPooling1D(pool_size=2, strides=2, padding='same'))
model.add(tf.keras.layers.Flatten())
model.add(tf.keras.layers.Dense(1091, activation='softmax'))
```
### Classification model

<table>
<thead>
<tr>
<th>Layer (type)</th>
<th>Output Shape</th>
<th>Param #</th>
</tr>
</thead>
<tbody>
<tr>
<td>embedding_1 (Embedding)</td>
<td>(None, 200, 128)</td>
<td>19200256</td>
</tr>
<tr>
<td>conv1d_1 (Conv1D)</td>
<td>(None, 200, 128)</td>
<td>82048</td>
</tr>
<tr>
<td>max_pooling1d_1 (MaxPooling1)</td>
<td>(None, 100, 128)</td>
<td>0</td>
</tr>
<tr>
<td>conv1d_2 (Conv1D)</td>
<td>(None, 100, 256)</td>
<td>164096</td>
</tr>
<tr>
<td>max_pooling1d_2 (MaxPooling1)</td>
<td>(None, 50, 256)</td>
<td>0</td>
</tr>
<tr>
<td>conv1d_3 (Conv1D)</td>
<td>(None, 50, 512)</td>
<td>655872</td>
</tr>
<tr>
<td>max_pooling1d_3 (MaxPooling1)</td>
<td>(None, 25, 512)</td>
<td>0</td>
</tr>
<tr>
<td>flatten_1 (Flatten)</td>
<td>(None, 12800)</td>
<td>0</td>
</tr>
<tr>
<td>dense_1 (Dense)</td>
<td>(None, 1091)</td>
<td>13965891</td>
</tr>
</tbody>
</table>

Total params: 34,068,163  
Trainable params: 34,068,163  
Non-trainable params: 0
Workflow from classification to deployment

Data transformation

Model building and training

Save model

Containerization

Inferencing Deployment

tf.data.Dataset

tf.keras
tf.distribute

tf.keras.models.save_model(.h5)

azureml.core.image

azureml.core.compute

azureml.core.webservice
## Data Augmentation Results

<table>
<thead>
<tr>
<th>ATA Code</th>
<th>Training sample size</th>
<th>Precision</th>
<th>Recall</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Original</td>
<td>Augmented</td>
<td>Original</td>
<td>Augmented</td>
</tr>
<tr>
<td>2560 Emergency equipment</td>
<td>3889</td>
<td>+3274</td>
<td>0.64</td>
<td>0.75</td>
</tr>
<tr>
<td>7220 Engine air inlet</td>
<td>699</td>
<td>+1583</td>
<td>0.2</td>
<td>0.67</td>
</tr>
<tr>
<td>7230 Engine compressor</td>
<td>546</td>
<td>+1616</td>
<td>0.47</td>
<td>0.68</td>
</tr>
</tbody>
</table>
3. Reference architecture for deployment
Enablement in Azure ML Service

Build, test, save model

Create Azure ML Service resources

Create image

Deploy container

Create Kubernetes clusters
Real Time Scoring with ML model in Azure

- Docker Image
  - Azure Container Registry

- Model Registry
  - Azure Machine Learning

- Input Data

- Request With Payload
  - Microsoft Data Science Virtual Machine

- HTTP Request
- HTTP Response

- Authenticate and Distribute Requests to Pods
  - Azure Machine Learning

- Pod
- Web Service
  - Model Scoring Script
  - Serve Scoring Requests
    - Azure Kubernetes Services

- tiny.cc/v52hfz
Real Time Scoring with ML model in Azure

1. Input Data
2. Docker Image: Azure Container Registry
3. Request With Payload: Microsoft Data Science Virtual Machine
4. Azure Machine Learning
5. Model Registry: Azure Machine Learning
6. Authenticate and Distribute Requests to Pods: Azure Machine Learning
7. Pod: Web Service
   - Model Scoring Script
8. Serve Scoring Requests: Azure Kubernetes Services

---

# TensorFlow World

# TFWorld
Registering model with Azure ML Service

```python
from azureml.core.model import Model

SHARE_ROOT = "/databricks/driver/tmp/"
MODEL = 'cnn_20191013-021616_ckpt.h5'
MODEL_PATH = SHARE_ROOT + MODEL
WORD2INDEX = 'word2index.pickle'
WORD2INDEX_PATH = SHARE_ROOT + WORD2INDEX

model = Model.register(
    model_path=MODEL_PATH, model_name=MODEL, tags = {'type': "cnn", 'target': "ATA_CODE"},
    description = "CNN model to predict ATA_CODE", workspace = ws)

word2index_model = Model.register(
    model_path = WORD2INDEX_PATH, model_name = WORD2INDEX, tags = {'type': "dict", 'target': "idx"},
    description = "word2index dictionary", workspace = ws)
```

Purpose: Push model and assets from Databricks to Azure estate.

Authentication: Azure Active Directory
Real Time Scoring with ML model in Azure
# Conda environment specification. The dependencies defined in this file will be automatically provisioned for runs with userManagedDependencies=False.

# Details about the Conda environment file format:
# https://conda.io/docs/user-guide/tasks/manage-environments.html#create-env-file-manually

name: project_environment dependencies:
# The python interpreter version.
# Currently Azure ML only supports 3.5.2 and later.
python=3.6.2
- pip:
# Required packages for AzureML execution, history, and data preparation.
- azureml-defaults
- tensorflow==2.0.0
- nltk
channels:
- conda-forge

import <libraries>
from azureml.core.model import Model

init(
    model_path = Model.get_model_path(MODEL)
    model = tf.keras.models.load_model(model_path)
)

run(
    handle JSON input
data extraction and feature engineering
    predict and return output as JSON payload
)
import <libraries>
from azureml.core.model import Model

init(
    model_path = Model.get_model_path(<MODEL>)
    model = tf.keras.models.load_model(model_path)
)

run(
    data = json.loads(raw_data)["data"]
    dataf = pd.read_json(data, orient='records')
    normalize input data
    predicted = model.predict(<normalized_input>)
    append prediction to dataframe
    df2json = df.to_json(orient='records')
    return df2json
)
# use azureml to create yml.

```python
from azureml.core.conda_dependencies import CondaDependencies

myenv = CondaDependencies()
myenv.add_tensorflow_pip_package(core_type='cpu', version='2.0.0')
myenv.add_conda_package("nltk")

env_file = "env_tf20.yml"
```
Build Docker image

```python
from azureml.core.image import ContainerImage

image_config = ContainerImage.image_configuration(
    execution_script="score.py",
    runtime="python",
    conda_file="tf20model.yml",
    description="Container for text classification",
    tags={"data": "texts", "type": "classification"}
)

image = ContainerImage.create(
    name="classification.image",
    models=[model,
        word2index_model,
        index2word_model,
        stopwords_set_model,
        index2tgt_model],
    image_config=image_config,
    workspace=ws)
```
Real Time Scoring with ML model in Azure

Diagram showing the process:
- **Input Data**
- **Docker Image**
  - Azure Container Registry
- **Request With Payload**
  - Microsoft Data Science Virtual Machine
- **HTTP Request**
- **HTTP Response**
- **Azure Machine Learning**
- **Model Registry**
- **Model Scoring Script**
- **Pod**
- **Web Service**
- **Serve Scoring Requests**
  - Azure Kubernetes Services
- **Authenticate and Distribute Requests to Pods**
  - Azure Machine Learning
Provision Kubernetes cluster

```python
from azureml.core.compute import AksCompute, ComputeTarget

prov_config = AksCompute.provisioning_configuration(
    agent_count=6,
    vm_size="Standard_DS2_v2",
    sslcname=None,
    ssl_cert_pem_file=None,
    ssl_key_pem_file=None,
    location="EastUs2")

# Create the cluster
aks_target = ComputeTarget.create(
    workspace=ws,
    name=aks_name,
    provisioning_configuration=prov_config)
```
Real Time Scoring with ML model in Azure
from azureml.core.webservice import AksWebservice

# Set the web service configuration
aks_config = AksWebservice.deploy_configuration(
    autoscale_enabled=True, autoscale_min_replicas=3,
    autoscale_max_replicas=10, autoscale_refresh_seconds=None,
    autoscale_target_utilization=80, collect_model_data=None,
    cpu_cores=None, memory_gb=None, enable_app_insights=True,
    scoring_timeout_ms=None, replica_max_concurrent_requests=None,
    num_replicas=None, primary_key=None, secondary_key=None,
    tags={'type': 'cnn', 'target': 'ATA_CODE'},
    description='AKS Service')

# Create the Web Service
aks_service = AksWebservice.deploy_from_image(
    workspace=ws,
    name=aks_service_name,
    image=image,
    deployment_config=aks_config,
    deployment_target=aks_target)
Real Time Scoring with ML model in AKS

- Input Data
- Docker Image
- Azure Container Registry
- Model Registry
- Azure Machine Learning

- Request With Payload
  Microsoft Data Science Virtual Machine

- HTTP Request
- HTTP Response

- Authenticate and Distribute Requests to Pods
  Azure Machine Learning

- Pod
- Web Service
- Model Scoring Script
- Serve Scoring Requests
  Azure Kubernetes Services
REST request

AKS URI

POST → http://<redacted>/api/v1/service/datacode-aks/service/score

Bearer token

Params

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Authorization</td>
<td>Bearer mWnxBMaHlc9mfrdwS...</td>
</tr>
<tr>
<td>Content-Type</td>
<td>application/json</td>
</tr>
</tbody>
</table>

Body

```
1  
2
```

Tools

- [O'Reilly TensorFlow World](https://oreilly.com)
- [TensorFlow](https://tensorflow.org)
- [tensorflow.world](https://tensorflow.world)
- [#TFWorld](https://tfworld.org)
<table>
<thead>
<tr>
<th>Body</th>
<th>Cookies</th>
<th>Headers (8)</th>
<th>Test Results</th>
<th>Status: 200 OK</th>
<th>Time: 1160 ms</th>
<th>Size: 1012 B</th>
<th>Save</th>
<th>Download</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

JSON

```
{
    "OPEN_FMR_TXT": "5-983NEF-CREF.2530DN *NEF* GALLEY CART DOORS///HANDLES///DIVIDERS AFT GALLEY RAIL NEEDS TO BE REATTACHED. MEL AUTHORIZED BY 317887",
    "PREDICTED_OPEN_ORIGINALATA": "2530",
    "OPEN_FMR_TXT": "A///C REQUIRES ETOPS PRE-DEPARTURE CK",
    "PREDICTED_OPEN_ORIGINALATA": "1210",
    "OPEN_FMR_TXT": "ETOPS PDC REQUIRED",
    "PREDICTED_OPEN_ORIGINALATA": "1210",
    "OPEN_FMR_TXT": "ZIPPER ON CARGO LINER BROKEN ON P9 6R",
    "PREDICTED_OPEN_ORIGINALATA": "5010",
    "OPEN_FMR_TXT": "6-60INEF-CREF.4402AN *NEF* PSGR AUDIO VIDEO SYSTEM ZONE IFE SYSTEM INOP SCREEN STUCK ON INITIALIZING MEL AUTHORIZED BY 190545",
    "PREDICTED_OPEN_ORIGINALATA": "4420"
}
```
4. Lesson learned
Train with dataset iterators instead of numpy

```python
# Prepare the training dataset
train_dataset = tf.data.Dataset.from_tensor_slices((x_train, y_train))
train_dataset = train_dataset.shuffle(buffer_size=BUFFER_SIZE).batch(BATCH_SIZE)

# Prepare the validation dataset
val_dataset = tf.data.Dataset.from_tensor_slices((x_val, y_val))
val_dataset = val_dataset.batch(BATCH_SIZE)

model.fit(train_dataset, validation_data=val_dataset, epochs=NUM_EPOCHS,
          validation_steps=VAL_STEPS)
```

- Efficiency; iterate through data instead of loading entire numpy array.
- For data larger than memory, use TFRecordDataset.
- Does not affect serving. Model can score both numpy and Dataset.
Distributed training (if necessary)

```python
strategy = tf.distribute.MirroredStrategy()
with strategy.scope():

    my_train_dataset = train_dataset.shuffle(...).batch(...)
    my_test_dataset = test_dataset.batch(...)
    my_val_dataset = val_dataset.shuffle(...).batch(...)

    model = tf.keras.Sequential(...)
    model.compile(...)

    mymodel = model.fit(...)
```

- Minimal modification to local training code.
- Tensorflow distributed training API takes care of data sharding.
- All-Reduce algorithm implemented by Tensorflow.
- This example is for one machine, multiple GPU, synchronous gradient update.

tensorflow.org/guide/distributed_training
Model checkpoint and callback flexibility

def decay(epoch):
    if epoch < 3:
        return 1e-3
    else:
        return 1e-4

callbacks = [
    tf.keras.callbacks.ModelCheckpoint(filepath=CHKPT, save_best_only=False,
                                         monitor='val_accuracy', mode='max',
                                         save_weights_only=True),
    tf.keras.callbacks.LearningRateScheduler(decay)
]
Fast and easy model testing

```python
model = tf.keras.Sequential()
model.add(tf.keras.layers.Embedding(VOCAB_SIZE, EMBEDDING_SIZE, input_length=MAX_SENTENCE_LENGTH))
model.add(tf.keras.layers.Conv1D(filters=128, kernel_size=5, strides=1, padding='same', activation='relu'))
model.add(tf.keras.layers.MaxPooling1D(pool_size=2, strides=2, padding='same'))
model.add(tf.keras.layers.Conv1D(filters=256, kernel_size=5, strides=1, padding='same', activation='relu'))
model.add(tf.keras.layers.MaxPooling1D(pool_size=2, strides=2, padding='same'))
model.add(tf.keras.layers.Conv1D(filters=512, kernel_size=5, strides=1, padding='same', activation='relu'))
model.add(tf.keras.layers.MaxPooling1D(pool_size=2, strides=2, padding='same'))
model.add(tf.keras.layers.Flatten())
model.add(tf.keras.layers.Dense(1091, activation='softmax'))
```

```python
model = tf.keras.Sequential()
model.add(tf.keras.layers.Embedding(VOCAB_SIZE, EMBEDDING_SIZE, input_length=MAX_SENTENCE_LENGTH))
model.add(tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(HIDDEN_LAYER_SIZE, dropout=0.2, recurrent_dropout=0.2)))
model.add(tf.keras.layers.Dense(1091, activation='softmax'))
```
Recognize different misclassification risk at each class

<table>
<thead>
<tr>
<th>ATA Code</th>
<th>Training sample size</th>
<th>Precision</th>
<th>Recall</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATA2560 Emergency equipment</td>
<td>3889 +3274</td>
<td>0.64 0.75</td>
<td>0.47</td>
<td>0.76 0.76</td>
</tr>
<tr>
<td>ATA7220 Engine air inlet</td>
<td>699 +1583</td>
<td>0.2 0.67</td>
<td>0.15</td>
<td>0.54 0.17 0.6</td>
</tr>
<tr>
<td>ATA7230 Engine compressor</td>
<td>546 +1616</td>
<td>0.47 0.68</td>
<td>0.07</td>
<td>0.56 0.13 0.61</td>
</tr>
</tbody>
</table>

Without data augmentation, over half of actual emergency equipment repair were misclassified.

\[
\text{precision} = \frac{TP}{TP + FP} \\
\text{recall} = \frac{TP}{TP + FN} \\
F1 = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}
\]
References

• Reference architecture for deployment:
  • tiny.cc/mhyhfz

• Kubernetes cluster configuration:
  • tiny.cc/zlyhfz

• Deployment through Azure ML Python API:
  • tiny.cc/9nyhfz

• Github
  • tiny.cc/jeyhfz
Thank you

kc.tung@microsoft.com

linkedin.com/in/kc-tung