Environment Setup

If you are having trouble running the following line of code, try adding the group project folder into your "My Drive" by navigating to the ITCS-6156-project folder, right clicking and selecting "Add shortcut to My Drive" in Google Drive.

```
!pip install PyDrive
```

```
Requirement already satisfied: PyDrive in /usr/local/lib/python3.6/dist-packages (1.3.1)
Requirement already satisfied: PyYAML>=3.0 in /usr/local/lib/python3.6/dist-packages (fro
Requirement already satisfied: oauth2client>=4.0.0 in /usr/local/lib/python3.6/dist-packa
Requirement already satisfied: google-api-python-client>=1.2 in /usr/local/lib/python3.6/
Requirement already satisfied: pyasn1-modules>=0.0.5 in /usr/local/lib/python3.6/dist-pac
Requirement already satisfied: rsa>=3.1.4 in /usr/local/lib/python3.6/dist-packages (fro
Requirement already satisfied: pyasn1>=0.1.7 in /usr/local/lib/python3.6/dist-packages (
Requirement already satisfied: httplib2>=0.9.1 in /usr/local/lib/python3.6/dist-packages
Requirement already satisfied: six>=1.6.1 in /usr/local/lib/python3.6/dist-packages (fro
Requirement already satisfied: google-auth>=1.4.1 in /usr/local/lib/python3.6/dist-packag
Requirement already satisfied: uritemplate<4dev,>=3.0.0 in /usr/local/lib/python3.6/dist-
Requirement already satisfied: google-auth-httplib2>=0.0.3 in /usr/local/lib/python3.6/di
Requirement already satisfied: setuptools>=40.3.0 in /usr/local/lib/python3.6/dist-packag
Requirement already satisfied: cachetools<5.0,>=2.0.0 in /usr/local/lib/python3.6/dist-pa
```

```
!pip install -U spacy
```

```
Collecting spacy
  Downloading https://files.pythonhosted.org/packages/e5/bf/ca7bb25edd21f1cf9d498d0023808
     |████████████████████████████████| 10.4MB 14.4MB/s
Requirement already satisfied, skipping upgrade: blis<0.8.0,>=0.4.0 in /usr/local/lib/pyt
Requirement already satisfied, skipping upgrade: tqdm<5.0.0,>=4.38.0 in /usr/local/lib/py
Requirement already satisfied, skipping upgrade: requests<3.0.0,>=2.13.0 in /usr/local/li
Requirement already satisfied, skipping upgrade: preshed<3.1.0,>=3.0.2 in /usr/local/lib/
Requirement already satisfied, skipping upgrade: srsly<1.1.0,>=1.0.2 in /usr/local/lib/py
Requirement already satisfied, skipping upgrade: catalogue<1.1.0,>=0.0.7 in /usr/local/li
Requirement already satisfied, skipping upgrade: murmurhash<1.1.0,>=0.28.0 in /usr/local/
Requirement already satisfied, skipping upgrade: setuptools in /usr/local/lib/python3.6/d
Requirement already satisfied, skipping upgrade: numpy>=1.15.0 in /usr/local/lib/python3
Requirement already satisfied, skipping upgrade: plac<1.2.0,>=0.9.6 in /usr/local/lib/pyt
Collecting thinc<7.5.0,>=7.4.1
  Downloading https://files.pythonhosted.org/packages/c0/1a/c3e4ab982214c63d743fad57c45c5
     |████████████████████████████████| 1.1MB 58.5MB/s
Requirement already satisfied, skipping upgrade: wasabi<1.1.0,>=0.4.0 in /usr/local/lib/p
Requirement already satisfied, skipping upgrade: cymem<2.1.0,>=2.0.2 in /usr/local/lib/py
Requirement already satisfied, skipping upgrade: certifi>=2017.4.17 in /usr/local/lib/pyt
Requirement already satisfied, skipping upgrade: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1
Requirement already satisfied, skipping upgrade: chardet<4,>=3.0.2 in /usr/local/lib/pyth
Requirement already satisfied, skipping upgrade: idna<3,>=2.5 in /usr/local/lib/python3.6
Requirement already satisfied, skipping upgrade: importlib-metadata>=0.20; python_versio
Requirement already satisfied, skipping upgrade: zipp>=0.5 in /usr/local/lib/python3.6/di
Installing collected packages: thinc, spacy
  Found existing installation: thinc 7.4.0
    Uninstalling thinc-7.4.0:
      Successfully uninstalled thinc-7.4.0
  Found existing installation: spacy 2.2.4
    Uninstalling spacy-2.2.4:
```

```
      Successfully uninstalled spacy-2.2.4
Successfully installed spacy-2.3.5 thinc-7.4.5


!python -m spacy download en

Collecting en_core_web_sm==2.3.1
  Downloading https://github.com/explosion/spacy-models/releases/download/en_core_web_sm-
     |████████████████████████████████| 12.1MB 602kB/s
Requirement already satisfied: spacy<2.4.0,>=2.3.0 in /usr/local/lib/python3.6/dist-packa
Requirement already satisfied: setuptools in /usr/local/lib/python3.6/dist-packages (fro
Requirement already satisfied: catalogue<1.1.0,>=0.0.7 in /usr/local/lib/python3.6/dist-p
Requirement already satisfied: thinc<7.5.0,>=7.4.1 in /usr/local/lib/python3.6/dist-packa
Requirement already satisfied: preshed<3.1.0,>=3.0.2 in /usr/local/lib/python3.6/dist-pac
Requirement already satisfied: requests<3.0.0,>=2.13.0 in /usr/local/lib/python3.6/dist-p
Requirement already satisfied: tqdm<5.0.0,>=4.38.0 in /usr/local/lib/python3.6/dist-packa
Requirement already satisfied: plac<1.2.0,>=0.9.6 in /usr/local/lib/python3.6/dist-packag
Requirement already satisfied: blis<0.8.0,>=0.4.0 in /usr/local/lib/python3.6/dist-packag
Requirement already satisfied: cymem<2.1.0,>=2.0.2 in /usr/local/lib/python3.6/dist-packa
Requirement already satisfied: wasabi<1.1.0,>=0.4.0 in /usr/local/lib/python3.6/dist-pacl
Requirement already satisfied: srsly<1.1.0,>=1.0.2 in /usr/local/lib/python3.6/dist-packa
Requirement already satisfied: numpy>=1.15.0 in /usr/local/lib/python3.6/dist-packages (
Requirement already satisfied: murmurhash<1.1.0,>=0.28.0 in /usr/local/lib/python3.6/dist
Requirement already satisfied: importlib-metadata>=0.20; python_version < "3.8" in /usr/l
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.6/dist-packages (fr
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.6/dist-package
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/local/lib/
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.6/dist-packag
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.6/dist-packages (from
Building wheels for collected packages: en-core-web-sm
  Building wheel for en-core-web-sm (setup.py) ... done
  Created wheel for en-core-web-sm: filename=en_core_web_sm-2.3.1-cp36-none-any.whl size=
  Stored in directory: /tmp/pip-ephem-wheel-cache-itx2lkec/wheels/2b/3f/41/f0b92863355c3b
Successfully built en-core-web-sm
Installing collected packages: en-core-web-sm
  Found existing installation: en-core-web-sm 2.2.5
    Uninstalling en-core-web-sm-2.2.5:
      Successfully uninstalled en-core-web-sm-2.2.5
Successfully installed en-core-web-sm-2.3.1
✔ Download and installation successful
You can now load the model via spacy.load('en_core_web_sm')
✔ Linking successful
/usr/local/lib/python3.6/dist-packages/en_core_web_sm -->
/usr/local/lib/python3.6/dist-packages/spacy/data/en
You can now load the model via spacy.load('en')


!sudo python3 -m spacy download en_core_web_lg

Collecting en_core_web_lg==2.3.1
  Downloading https://github.com/explosion/spacy-models/releases/download/en_core_web_lg-
     |████████████████████████████████| 782.7MB 1.7MB/s
Requirement already satisfied: spacy<2.4.0,>=2.3.0 in /usr/local/lib/python3.6/dist-packa
Requirement already satisfied: setuptools in /usr/local/lib/python3.6/dist-packages (fro
Requirement already satisfied: tqdm<5.0.0,>=4.38.0 in /usr/local/lib/python3.6/dist-packa
Requirement already satisfied: blis<0.8.0,>=0.4.0 in /usr/local/lib/python3.6/dist-packag
Requirement already satisfied: plac<1.2.0,>=0.9.6 in /usr/local/lib/python3.6/dist-packag
Requirement already satisfied: requests<3.0.0,>=2.13.0 in /usr/local/lib/python3.6/dist-p
Requirement already satisfied: cymem<2.1.0,>=2.0.2 in /usr/local/lib/python3.6/dist-packa
Requirement already satisfied: catalogue<1.1.0,>=0.0.7 in /usr/local/lib/python3.6/dist-p
Requirement already satisfied: numpy>=1.15.0 in /usr/local/lib/python3.6/dist-packages (
```

```
    Requirement already satisfied: preshed<3.1.0,>=3.0.2 in /usr/local/lib/python3.6/dist-pac
    Requirement already satisfied: srsly<1.1.0,>=1.0.2 in /usr/local/lib/python3.6/dist-packa
    Requirement already satisfied: thinc<7.5.0,>=7.4.1 in /usr/local/lib/python3.6/dist-packa
    Requirement already satisfied: wasabi<1.1.0,>=0.4.0 in /usr/local/lib/python3.6/dist-pacl
    Requirement already satisfied: murmurhash<1.1.0,>=0.28.0 in /usr/local/lib/python3.6/dist
    Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.6/dist-package
    Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/local/lib/
    Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.6/dist-packages (fr
    Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.6/dist-packag
    Requirement already satisfied: importlib-metadata>=0.20; python_version < "3.8" in /usr/l
    Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.6/dist-packages (from
    Building wheels for collected packages: en-core-web-lg
      Building wheel for en-core-web-lg (setup.py) ... done
      Created wheel for en-core-web-lg: filename=en_core_web_lg-2.3.1-cp36-none-any.whl size=
      Stored in directory: /tmp/pip-ephem-wheel-cache-zswoa135/wheels/ce/4d/1b/bc6cabb6df139c
    Successfully built en-core-web-lg
    Installing collected packages: en-core-web-lg
    Successfully installed en-core-web-lg-2.3.1
    ✔ Download and installation successful
    You can now load the model via spacy.load('en_core_web_lg')
```

```
%load_ext tensorboard
%matplotlib inline


import os
from pydrive.auth import GoogleAuth
from pydrive.drive import GoogleDrive
from google.colab import auth
from google.colab import drive as dv
from oauth2client.client import GoogleCredentials

import tensorflow as tf
import keras
from keras.preprocessing import image
from keras.applications.resnet50 import ResNet50
from keras.applications.resnet50 import preprocess_input
from keras.models import Model
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences
from keras.utils import to_categorical
from tensorflow.keras.models import model_from_json

import pandas as pd
import json

import collections
from collections import defaultdict
import numpy as np
import spacy
from datetime import datetime
from packaging import version
import matplotlib.pyplot as plt
import seaborn as sns
from PIL import Image


auth.authenticate_user()
```

```
auth.authenticate_user()
gauth = GoogleAuth()
gauth.credentials = GoogleCredentials.get_application_default()
drive = GoogleDrive(gauth)
```

```
dv.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
!ls "/content/drive/My Drive/ITCS-6156-Project/data"
```

```
annotations                      image_feature_dictionary4.json
annotations_trainval2017.zip     image_feature_dictionary5.json
filteredOutDoor.json             image_feature_dictionary6.json
filteredOutDoor.txt              image_feature_dictionary7.json
image_feature_dictionary0.json   image_feature_dictionary8.json
image_feature_dictionary10.json  image_feature_dictionary9.json
image_feature_dictionary11.json  imgs
image_feature_dictionary12.json  logs
image_feature_dictionary13.json  model.h5
image_feature_dictionary14.json  model.json
image_feature_dictionary1.json   test2017
image_feature_dictionary2.json   val2017
image_feature_dictionary3.json
```

```
IMG_LOCATION = '/content/drive/My Drive/ITCS-6156-Project/data/imgs'
```

```
#Configure GPU
tf.config.experimental.list_physical_devices('GPU')
```

```
[PhysicalDevice(name='/physical_device:GPU:0', device_type='GPU')]
```

```
#Configure TPU
'''
try:
  tpu = tf.distribute.cluster_resolver.TPUClusterResolver()  # TPU detection
  print('Running on TPU ', tpu.cluster_spec().as_dict()['worker'])
except ValueError:
  raise BaseException('ERROR: Not connected to a TPU runtime; please see the previous cell in

tf.config.experimental_connect_to_cluster(tpu)
tf.tpu.experimental.initialize_tpu_system(tpu)
tpu_strategy = tf.distribute.experimental.TPUStrategy(tpu)
'''
```

```
'\ntry:\n  tpu = tf.distribute.cluster_resolver.TPUClusterResolver()  # TPU detection\n
print('Running on TPU ', tpu.cluster_spec().as_dict()['worker'])\nexcept ValueError:\n
raise BaseException('ERROR: Not connected to a TPU runtime; please see the previous cell
in this notebook for instructions!')\n\ntf.config.experimental_connect_to_cluster(tpu)\n
tf.tpu.experimental.initialize_tpu_system(tpu)\ntpu_strategy = tf.distribute.experimenta
```

Image Feature Extraction

```
'''
# load the ResNet50 Model
feature_extractor = ResNet50(weights='imagenet', include_top=False)
feature_extractor_new = Model(feature_extractor.input, feature_extractor.layers[-2].output)
feature_extractor_new.summary()

image_feature_dictionary = {}

image_path = IMG_LOCATION

num_completed_items = 0

for file in os.listdir(image_path):
    path = image_path + "/" + file
    img = image.load_img(path, target_size=(90, 90))
    img_data = image.img_to_array(img)
    img_data = np.expand_dims(img_data, axis=0)
    img_data = preprocess_input(img_data)

    feature = feature_extractor_new.predict(img_data)
    feature_reshaped = np.array(feature).flatten()
    image_feature_dictionary[file] = feature_reshaped
    num_completed_items += 1
    print(num_completed_items, end=', ')
  '''
```


```
'\n# load the ResNet50 Model\nfeature_extractor = ResNet50(weights=\'imagenet\', include
_top=False)\nfeature_extractor_new = Model(feature_extractor.input, feature_extractor.la
yers[-2].output)\nfeature_extractor_new.summary()\n\nimage_feature_dictionary = {}\n\nim
age_path = IMG_LOCATION\n\nnum_completed_items = 0\n\nfor file in os.listdir(image_pat
h):\n    path = image_path + "/" + file\n    img = image.load_img(path, target_size=(90,
90))\n    img_data = image.img_to_array(img)\n    img_data = np.expand_dims(img_data, ax
is=0)\n    img_data = preprocess_input(img_data)\n    \n    feature = feature_extractor_
new.predict(img_data)\n    feature_reshaped = np.array(feature).flatten()\n    image_fea
```


```
#len(image_feature_dictionary)
```

Save Results

The following code saves the output of the above image feature extraction into multiple different JSON files in order to prevent Colab from crashing due to RAM usage. The above image feature extraction takes a few hours to run so saving the output will save some time. All files labeled image_feature_dictionaryX.json correspond to the results.

```
class NumpyEncoder(json.JSONEncoder):
    def default(self, obj):
        if isinstance(obj, np.ndarray):
            return obj.tolist()
        return json.JSONEncoder.default(self, obj)


def save_results():
  counter = 0
```

```
  keys_to_extract = []
  temp_buffer = []

  for key, value in image_feature_dictionary.items():
    if counter % 1057 == 0 and len(temp_buffer) > 0:
      keys_to_extract.append(temp_buffer)
      temp_buffer = []

    temp_buffer.append(key)
    counter += 1
  if len(temp_buffer) > 0:
    keys_to_extract.append(temp_buffer)

  file_count = 0
  for set_of_keys in keys_to_extract:
    a_subset = {key: image_feature_dictionary[key] for key in set_of_keys}
    json_dump = json.dumps(a_subset, cls=NumpyEncoder)
    f = open("/content/drive/My Drive/ITCS-6156-Project/data/image_feature_dictionary" + str(f
    f.write(json_dump)
    f.close()
    file_count += 1


#WARNING---DO NOT UNCOMMENT THIS UNLESS YOU HAVE NEW RESULTS TO SAVE
#save_results()
```

Read Prior Image Feature Extraction Results

Running this section of code is only necessary if you have not ran the model above or recently loaded in the data

```
def load_results():
  json_list = []
  for file_count in range(15):
    json_list.append("/content/drive/My Drive/ITCS-6156-Project/data/image_feature_dictionary"

  all_data = {}
  for i in range(len(json_list)):
      f = open(json_list[i],"r")
      file = f.read()
      f.close()
      data = json.loads(file)
      all_data = {**all_data, **data}

  return all_data


images_segmented = load_results()
print(len(images_segmented))
```

```
15854
```

Caption Extraction

```
# load captions from captions file
# loading filteredOutDoor.txt
# Reading the json as a dict
with open('/content/drive/MyDrive/ITCS-6156-Project/data/filteredOutDoor.json') as json_data:
    data = json.load(json_data)

captions_df = pd.DataFrame.from_dict(data, orient='index').T
captions_df.head(10)
```

	Segments	captions	images
0	{'segmentation': [[390.95, 131.85, 392.68, 128...	{'image_id': 203564, 'id': 37, 'caption': 'A b...	{'license': 3, 'file_name': '000000574769.jpg'...
1	{'segmentation': [[564.89, 150.38, 565.83, 128...	{'image_id': 322141, 'id': 49, 'caption': 'A r...	{'license': 1, 'file_name': '000000118113.jpg'...
2	{'segmentation': [[135.37, 77.49, 140.94, 78.2...	{'image_id': 322141, 'id': 109, 'caption': 'Bl...	{'license': 2, 'file_name': '000000374628.jpg'...
3	{'segmentation': [[253.33, 246.45, 254.81, 252...	{'image_id': 322141, 'id': 121, 'caption': 'Th...	{'license': 3, 'file_name': '000000384213.jpg'...
4	{'segmentation': [[1.31, 29.86, 8.22, 29.86, 1...	{'image_id': 322141, 'id': 163, 'caption': 'A ...	{'license': 3, 'file_name': '000000223648.jpg'...
5	{'segmentation': [[570.1, 132.76, 572.44, 133....	{'image_id': 203564, 'id': 181, 'caption': 'Th...	{'license': 2, 'file_name': '000000337264.jpg'...
6	{'segmentation': [[156.28, 217.59,	{'image_id': 322141, 'id': 250,	{'license': 2, 'file_name':

```
captions_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 79277 entries, 0 to 79276
Data columns (total 3 columns):
 #   Column    Non-Null Count  Dtype 
---  ------    --------------  ----- 
 0   Segments  45310 non-null  object
 1   captions  79277 non-null  object
 2   images    15847 non-null  object
dtypes: object(3)
memory usage: 1.8+ MB
```

```
captions_df.images[0]
```

```
{'coco_url': 'http://images.cocodataset.org/train2017/000000574769.jpg',
 'date_captured': '2013-11-14 17:07:59',
 'file_name': '000000574769.jpg',
 'flickr_url': 'http://farm8.staticflickr.com/7010/6728227647_3d5a0d55ee_z.jpg',
 'height': 640,
 'id': 574769,
 'license': 3,
 'width': 480}
```

```
captions_df.captions[0]
```

```
    {'caption': 'A bicycle replica with a clock as the front wheel.',
     'id': 37,
     'image_id': 203564}
```

```
for item_idx in range(0,5):
  sample_dict = captions_df.captions[item_idx]
  caption = sample_dict['caption']
  id = sample_dict['image_id']
  print("Picture ID: {}\nCaption: {}\n-------------------------------------".format(id,caption
```

```
    Picture ID: 203564
    Caption: A bicycle replica with a clock as the front wheel.
    -------------------------------------
    Picture ID: 322141
    Caption: A room with blue walls and a white sink and door.
    -------------------------------------
    Picture ID: 322141
    Caption: Blue and white color scheme in a small bathroom.
    -------------------------------------
    Picture ID: 322141
    Caption: This is a blue and white bathroom with a wall sink and a lifesaver on the wall.
    -------------------------------------
    Picture ID: 322141
    Caption: A blue boat themed bathroom with a life preserver on the wall
    -------------------------------------
```

```
captions_df.captions.count()
```

```
    79277
```

```
image_lookup_dict = {}
final_dict = defaultdict(set)
all_captions = []

for image_idx in range(captions_df.images.count()):
  temp_dict = captions_df.images[image_idx]
  name = temp_dict['file_name']
  image_id = temp_dict['id']
  image_lookup_dict[image_id] = name

for caption_idx in range(captions_df.captions.count()):
  temp_dict = captions_df.captions[caption_idx]
  caption = temp_dict['caption']

  caption = caption.replace(".","")
  caption = caption.replace("\n","")
  caption = "<start> " + caption + " <end>"


  image_name = image_lookup_dict[temp_dict['image_id']]
  final_dict[image_name].add(caption)
  all_captions.append(caption)
```

```
final_dict.keys()
```

```
dict_keys(['000000203564.jpg', '000000322141.jpg', '000000571635.jpg', '000000301837.jpg
```

```
final_dict['000000203564.jpg']
```

```
{'<start> A bicycle figurine in which the front wheel is replaced with a clock <end>',
 '<start> A bicycle replica with a clock as the front wheel <end>',
 '<start> A black metal bicycle with a clock inside the front wheel <end>',
 '<start> A clock with the appearance of the wheel of a bicycle  <end>',
 '<start> The bike has a clock as a tire <end>'}
```

```
all_captions
```

```
 '<start> A home kitchen with wood cabinets and a long table <end>',
 '<start> A large mirror with black framing on the wall of a bathroom above the sink <en
 '<start> A bathroom toilet sitting next to a counter and a roll of toilet paper <end>',
 '<start> A kitchen with black countertops, chrome appliances, and wooden cabinets <end>
 '<start> A lot of people that are in a kitchen <end>',
 '<start> A kitchen that is very nice and clean <end>',
 '<start> The interior of a kitchen with brown wooden finishes <end>',
 '<start> A woman stands in the kitchen with a vest draped over a chair <end>',
 '<start> A beautiful and spacious kitchen with stainless steel appliances <end>',
 '<start> A Winnie The Pooh and a turtle stuffed animals sitting on a towel rack <end>',
 '<start> A bathroom features a large mirror and toiletries next to the sink <end>',
 '<start> A woman preparing food for cooking in her kitchen <end>',
 '<start> A kitchen scene complete with a sink, refrigerator and an oven <end>',
 '<start> A flat screen TV next to a fire place in a living room <end>',
 '<start> A woman riding a green bike with a small white dog on the back <end>',
 '<start> A kitchen that has many wooden cabinets and cupboards <end>',
 '<start> A computer desk with highlighted papers all over it <end>',
 '<start> A dinning room table sitting next to a small fireplace <end>',
 '<start> A cop riding a motorcycle next to a  white van <end>',
 '<start> A cat perching on top of an open door <end>',
 '<start> A desk with paper, a monitor and keyboard <end>',
 '<start> A building with a tall clock tower with a green top <end>',
 "<start> A woman in a fur coat on a bike with a dog in the bike's basket <end>",
 '<start> A woman in a kitchen preparing a meal <end>',
 '<start> A woman wearing a coat rides a bicycle with a dog  <end>',
 '<start> A woman standing in front of a kitchen sink on display <end>',
 '<start> The bathroom scene with focus on the sink and the mirror <end>',
 '<start> A large teddy bear attached to the roof of an SUV <end>',
 '<start> A desktop computer sitting on top of an office desk <end>',
 '<start> A tall green building with a massive tall brown clock tower behind it <end>',
 "<start> A toilet, with a roll of toilet paper and a children's Hanukkah book on the co
 '<start> A woman looks upon a science museum exhibit <end>',
 '<start> A couple of men standing next to each other at a ribbon cutting ceremony <end>
 '<start> A computer on a desk covered in papers <end>',
 '<start> People walking around and near a city church <end>',
 '<start> A blurry image of a man in a room full of pots on tables <end>',
 '<start> A city scape, the main brick building has a small clock tower on top  <end>',
 '<start> a yellow and black bathroom with a mirror and sink <end>',
 '<start> A computer on a desk covered with many papers <end>',
 '<start> Three jovial women at a kitchen island interacting <end>',
 '<start> A man in a large room with baskets and pottery <end>',
 '<start> A woman standing in front of a wall lined with sinks <end>',
 '<start> an image of a cat perched on top of a wooden ledge <end>',
 '<start> A computer screen open to someone running on video <end>',
 '<start> THERE ARE A LOT OF PEOPLE IN THE KITCHEN SMILING  <end>',
 '<start> Dining table for six in a wood paneled room with a fireplace <end>',
```

```
      '<start> A WOMAN IS LOOKING AT A SINK IN A CROWD <end>',
      '<start> An Italian Neoclassic church amongst Renaissance era architecture <end>',
      '<start> A STUFF ANIMAL IS UNDER A BASKET  <end>',
      '<start> THERE IS A TOILET AND SINK IN THE BATHROOM <end>',
      '<start> Several people in a kitchen and one is standing at a sink <end>',
      '<start> A large building in the background with a clock and tower on the top of it and
      '<start> an image of a man that is going in a restaurant  <end>',
      '<start> Someone had the foresight to place a roll of toilet paper on the cupboard by th
      '<start> an image of a woman in the kitchen cooking <end>',
      '<start> The woman is riding her bike with her dog on the back <end>',
      '<start> Two stuffed animals stare at each other while perched on a towel rack <end>',
      '<start> A street with people walking about near a large ornate building <end>',
      ...]


train_image_captions = {}
test_image_captions = {}

for key in final_dict:
  if(len(train_image_captions) < 12000):
    train_image_captions[key] = final_dict[key]
  elif(len(test_image_captions) < 3000):
    test_image_captions[key] = final_dict[key]


print(train_image_captions['000000203564.jpg'])
print(test_image_captions['000000113132.jpg'])

     {'<start> The bike has a clock as a tire <end>', '<start> A black metal bicycle with a cl
     {'<start> A computer screen sits on next to pencils and a teddy bear <end>', '<start> The


nlp = spacy.load('en', disable=['tagger', 'parser', 'ner'])

# tokenize every caption, remove punctuations, lowercase everything
for key, value in train_image_captions.items():
    ls = []
    for v in value:
        doc = nlp(v)
        new_v = " "
        for token in doc:
            if not token.is_punct:
                if token.text not in [" ", "\n", "\n\n"]:
                    new_v = new_v + " " + token.text.lower()

        new_v = new_v.strip()
        ls.append(new_v)
    train_image_captions[key] = ls


# create a vocabulary of all the unique words present in captions
# flatten the list
#all_captions = [caption for list_of_captions in all_captions for caption in list_of_captions]

# use spacy to convert to lowercase and reject any special characters
tokens = []
for captions in all_captions:
    doc = nlp(captions)
```

```
    for token in doc:
        if not token.is_punct:
            if token.text not in [" ", "\n", "\n\n"]:
                tokens.append(token.text.lower())

# get tokens with frequency less than 10
word_count_dict = collections.Counter(tokens)
reject_words = []
for key, value in word_count_dict.items():
    if value < 10:
        reject_words.append(key)

reject_words.append("<")
reject_words.append(">")

 # remove tokens that are in reject words
tokens = [x for x in tokens if x not in reject_words]

# convert the token to equivalent index using Tokenizer class of Keras
tokenizer = Tokenizer()
tokenizer.fit_on_texts(tokens)


tokenizer.word_index
```

```
      'counters': 688,
      'body': 689,
      'cathedral': 690,
      'desks': 691,
      'right': 692,
      'bunk': 693,
      'hold': 694,
      'lighting': 695,
      'uses': 696,
      'design': 697,
      'hardwood': 698,
      'someones': 699,
      'adults': 700,
      'kitten': 701,
      'taken': 702,
      'pens': 703,
      'writing': 704,
      'way': 705,
      'den': 706,
      'balcony': 707,
      'umbrellas': 708,
      'breakfast': 709,
      'fan': 710,
      'shopping': 711,
      'not': 712,
      'multi': 713,
      'pottery': 714,
      'numerals': 715,
      'crib': 716,
      'banana': 717,
      'includes': 718,
      'neck': 719,
      'fresh': 720,
      'closed': 721,
      'intersection': 722,
```

```
    'stop': 723,
    'fabric': 724,
    'enjoying': 725,
    'pitcher': 726,
    'fake': 727,
    'backpack': 728,
    'blankets': 729,
    'wide': 730,
    'friends': 731,
    'type': 732,
    'rock': 733,
    'where': 734,
    'unmade': 735,
    'school': 736,
    'parking': 737,
    'tiny': 738,
    'shoes': 739,
    'poses': 740,
    'towers': 741,
    'held': 742,
    'marble': 743,
    'before': 744,
    'opened': 745,
    'sewing': 746,
    'himself': 747,
```

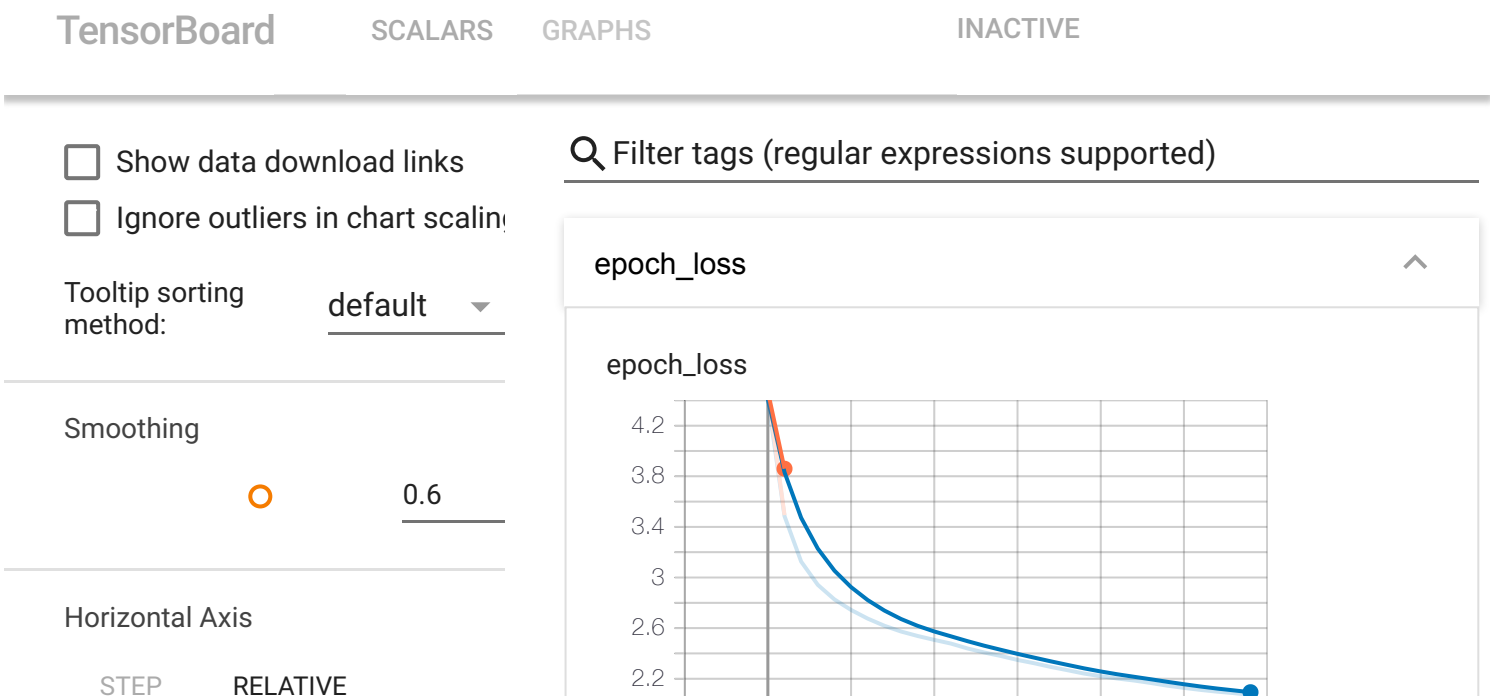
```
# compute length of vocabulary and maximum length of a caption (for padding)
vocab_len = len(tokenizer.word_counts) + 1
print(f"Vocabulary length - {vocab_len}")

max_caption_len = max([len(x.split(" ")) for x in all_captions])
print(f"Maximum length of caption - {max_caption_len}")
```

```
Vocabulary length - 2590
Maximum length of caption - 51
```

```
# generator function to generate inputs for model
def create_training_data(captions, images, tokenizer, max_caption_length, vocab_len, photos_pe

    X1, X2, y = list(), list(), list()
    n=0

    # loop through every image
    while 1:
        for key, cap in captions.items():
            n+=1
            # retrieve the photo feature
            image = images[key]

            for c in cap:
                # encode the sequence
                sequence = [tokenizer.word_index[word] for word in c.split(' ') if word in lis

                # split one sequence into multiple X, y pairs

                for i in range(1, len(sequence)):
                    # creating input, output
                    inp, out = sequence[:i], sequence[i]
```

```
                    inp, out = sequence[:i], sequence[i]
                    # padding input
                    input_seq = pad_sequences([inp], maxlen=max_caption_length)[0]
                    # encode output sequence
                    output_seq = to_categorical([out], num_classes=vocab_len)[0]
                    # store
                    X1.append(image)
                    X2.append(input_seq)
                    y.append(output_seq)

            # yield the batch data
            if n==photos_per_batch:
                yield ([np.array(X1), np.array(X2)], np.array(y))
                X1, X2, y = list(), list(), list()
                n=0


def create_model(max_caption_length, vocab_length):

    # sub network for handling the image feature part
    input_layer1 = keras.Input(shape=(18432))
    feature1 = keras.layers.Dropout(0.2)(input_layer1)
    feature2 = keras.layers.Dense(max_caption_length*4, activation='relu')(feature1)
    feature3 = keras.layers.Dense(max_caption_length*4, activation='relu')(feature2)
    feature4 = keras.layers.Dense(max_caption_length*4, activation='relu')(feature3)
    feature5 = keras.layers.Dense(max_caption_length*4, activation='relu')(feature4)

    # sub network for handling the text generation part
    input_layer2 = keras.Input(shape=(max_caption_length,))
    cap_layer1 = keras.layers.Embedding(vocab_length, 300, input_length=max_caption_length)(in
    cap_layer2 = keras.layers.Dropout(0.2)(cap_layer1)
    cap_layer3 = keras.layers.LSTM(max_caption_length*4, activation='relu', return_sequences=T
    cap_layer4 = keras.layers.LSTM(max_caption_length*4, activation='relu', return_sequences=T
    cap_layer5 = keras.layers.LSTM(max_caption_length*4, activation='relu', return_sequences=T
    cap_layer6 = keras.layers.LSTM(max_caption_length*4, activation='relu')(cap_layer5)

    # merging the two sub network
    decoder1 = keras.layers.merge.add([feature5, cap_layer6])
    decoder2 = keras.layers.Dense(256, activation='relu')(decoder1)
    decoder3 = keras.layers.Dense(256, activation='relu')(decoder2)

    # output is the next word in sequence
    output_layer = keras.layers.Dense(vocab_length, activation='softmax')(decoder3)
    model = keras.models.Model(inputs=[input_layer1, input_layer2], outputs=output_layer)

    model.summary()

    return model


import en_core_web_lg
# create word embeddings
nlp = en_core_web_lg.load()

# create word embeddings
embedding_dimension = 300
embedding_matrix = np.zeros((vocab_len, embedding_dimension))
```

```
# travel through every word in vocabulary and get its corresponding vector
for word, index in tokenizer.word_index.items():
    doc = nlp(word)
    embedding_vector = np.array(doc.vector)
    embedding_matrix[index] = embedding_vector


train_image_features = {key: images_segmented[key] for key in train_image_captions}
# get training data
train_data = create_training_data(train_image_captions, train_image_features, tokenizer, max_c

# initialize model
model = create_model(max_caption_len, vocab_len)

logdir = "/content/drive/My Drive/ITCS-6156-Project/data/logs/scalars/" + datetime.now().strft
tensorboard_callback = keras.callbacks.TensorBoard(log_dir=logdir)

# adding embeddings to model
model.layers[2]
model.layers[2].set_weights([embedding_matrix])
model.layers[2].trainable = False

steps_per_epochs = len(train_image_captions)//32

# compile model
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics='accuracy')
model.fit_generator(train_data, epochs=30, steps_per_epoch=steps_per_epochs, callbacks=tensorb


%tensorboard --logdir=/content/drive/MyDrive/ITCS-6156-Project/data/logs/scalars
```



Save Neural Network

This model took roughly 6 hours to train. Saving the trained model will allow us to open it at a later time rather than having to retrain every time. ***WARNING:*** Run this cell ONLY if you are updating the saved file.

Write a regex to filter runs

```
'''
# serialize model to JSON
model_json = model.to_json()
with open("/content/drive/My Drive/ITCS-6156-Project/data/model.json", "w") as json_file:
    json_file.write(model_json)
# serialize weights to HDF5
model.save_weights("/content/drive/My Drive/ITCS-6156-Project/data/model.h5")
print("Saved model to disk")
'''
```

```
Saved model to disk
```

Load Neural Network From File

```
# load json and create model
json_file = open('/content/drive/My Drive/ITCS-6156-Project/data/model.json', 'r')
loaded_model_json = json_file.read()
json_file.close()
loaded_model = model_from_json(loaded_model_json)

# load weights into new model
loaded_model.load_weights("/content/drive/My Drive/ITCS-6156-Project/data/model.h5")
print("Loaded model from disk")
```

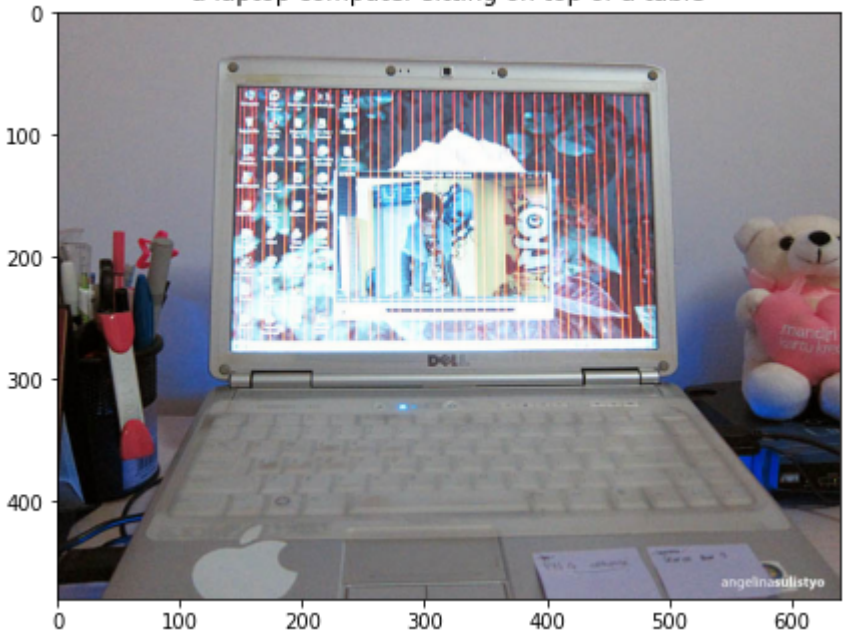
```
WARNING:tensorflow:Layer lstm will not use cuDNN kernel since it doesn't meet the cuDNN k
```

```
WARNING:tensorflow:Layer lstm_1 will not use cuDNN kernel since it doesn't meet the cuDNN
WARNING:tensorflow:Layer lstm_2 will not use cuDNN kernel since it doesn't meet the cuDNN
WARNING:tensorflow:Layer lstm_3 will not use cuDNN kernel since it doesn't meet the cuDNN
Loaded model from disk
```

Testing Neural Network:

```
# method for generating captions
def generate_captions(model, image, word_index, max_caption_length, index_word):

    # input is <start>
    input_text = '<start>'

    # keep generating words till we have encountered <end>
    for i in range(max_caption_length):
        seq = [word_index[w] for w in input_text.split() if w in list(word_index.keys())]
        seq = pad_sequences([seq], maxlen=max_caption_length)
        prediction = model.predict([image,seq], verbose=0)
        prediction = np.argmax(prediction)
        word = index_word[prediction]
        input_text += ' ' + word
        if word == 'end':
            break

    # remove <start> and <end> from output and return string
    output = input_text.split()
    output = output[1:-1]
    output = ' '.join(output)
    return output

# traverse through testing images to generate captions
test_image_features = {key: images_segmented[key] for key in test_image_captions}
image_path = IMG_LOCATION
count = 0

for key, value in test_image_features.items():
    test_image = test_image_features[key]
    test_image = np.expand_dims(test_image, axis=0)
    final_caption = generate_captions(loaded_model, test_image, tokenizer.word_index, max_capt

    plt.figure(figsize=(7,7))
    image = Image.open(image_path + "/" + key)
    plt.imshow(image)
    plt.title(final_caption)
    '''print("Picture: " + key)
    for value in test_image_captions[key]:
      print(value)
    print("----------------------")
    '''

    count = count + 1
    if count == 20:
        break
```

a laptop computer sitting on top of a table

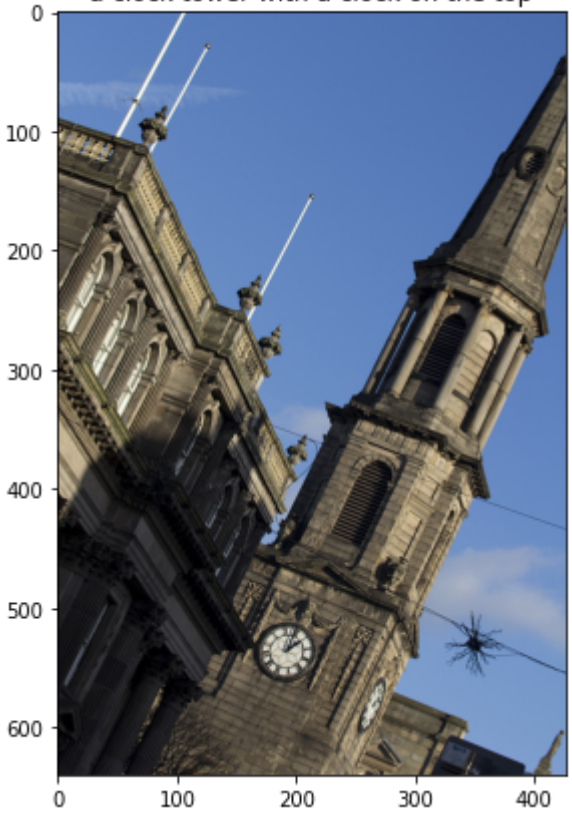


a laptop computer sitting on top of a desk

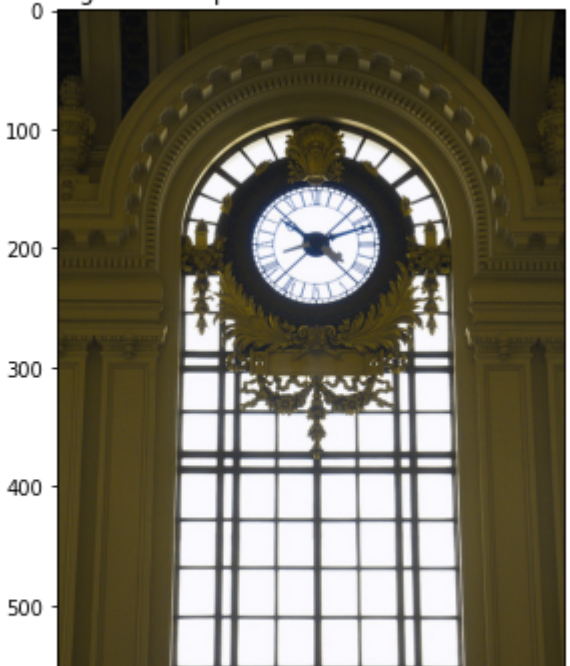
a stuffed teddy bear sitting next to each other

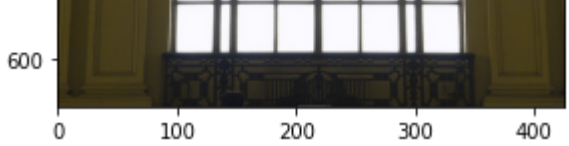
a laptop computer sitting on top of a table

a clock tower with a clock on the top

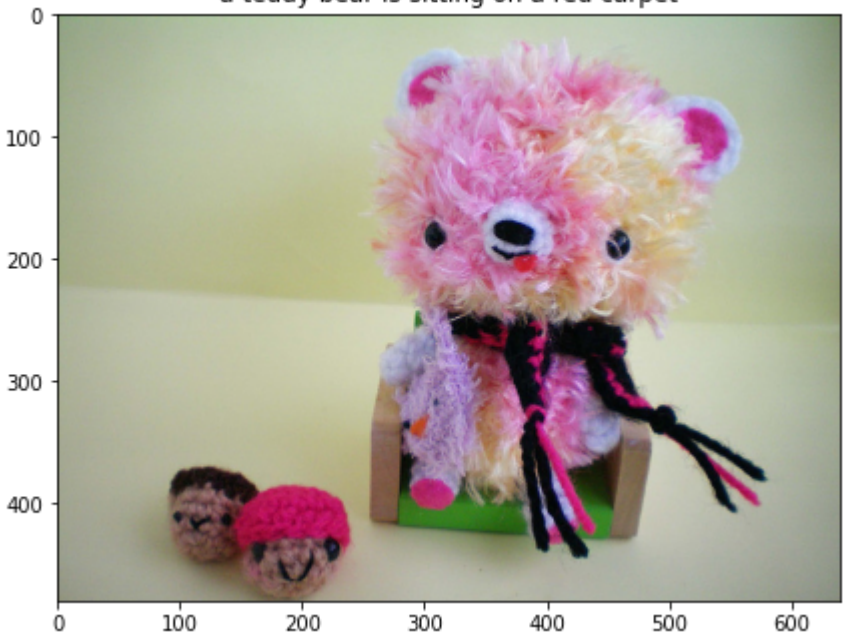


a large clock is posted on the side of a building

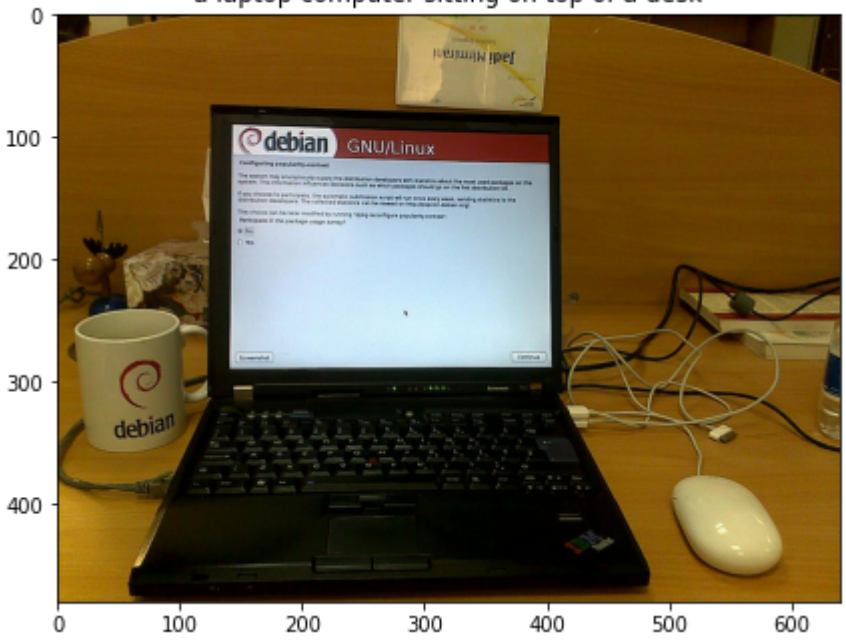




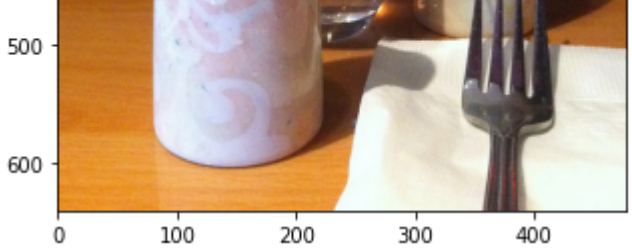
a teddy bear is sitting on a red carpet



a laptop computer sitting on top of a desk

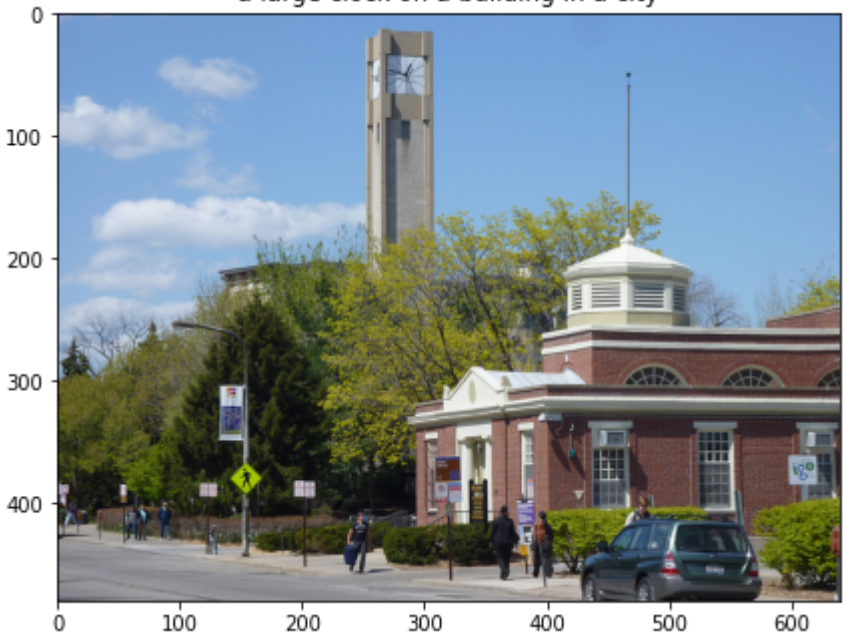


a toothbrush in a vase sitting on a table

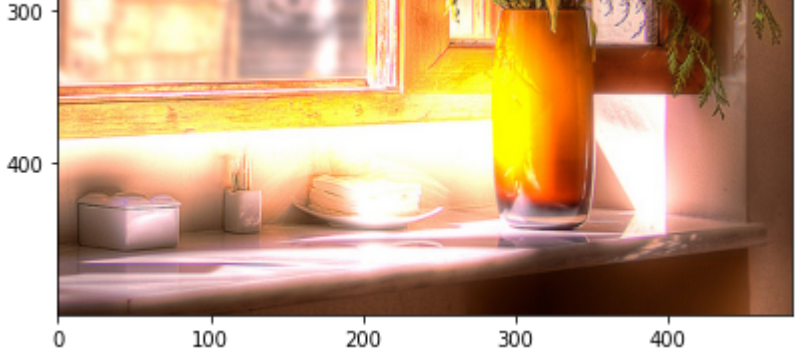


a keyboard and keyboard sitting on a desk

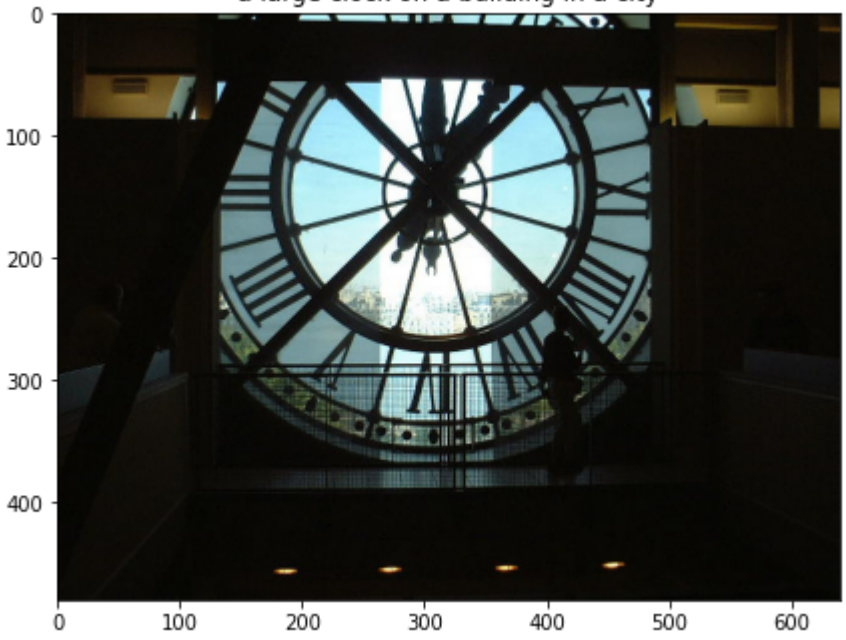
a large clock on a building in a city



a vase of flowers sitting on a table



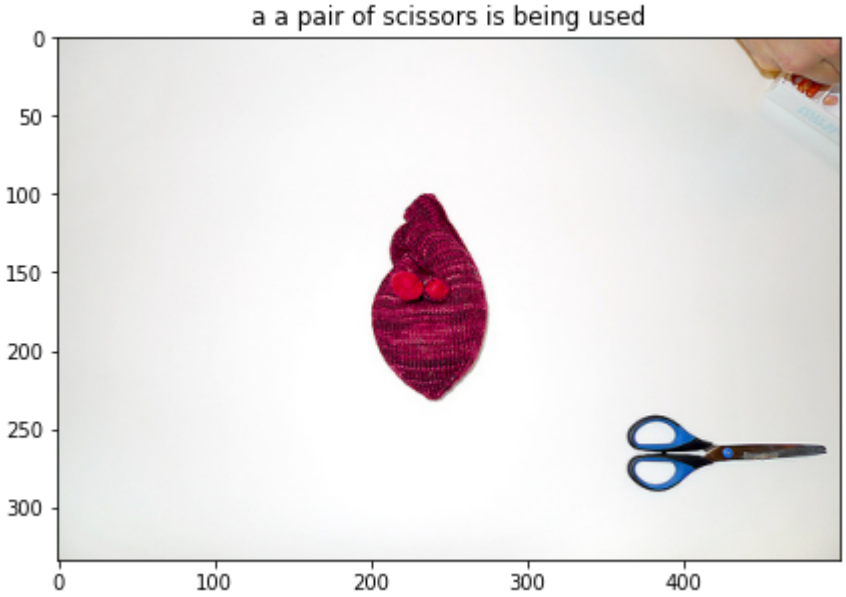
a large clock on a building in a city



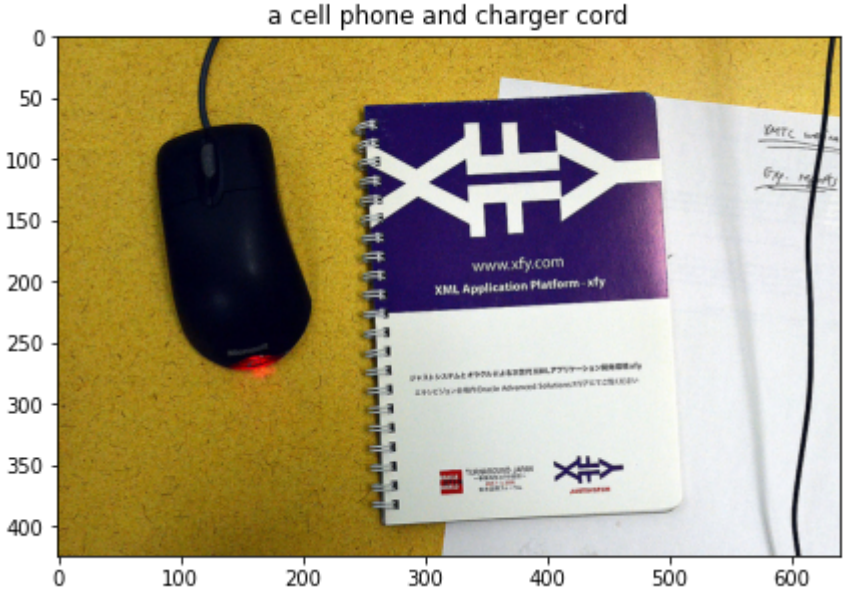
a man and a woman are playing a video game

a group of children sitting on a bench

a a pair of scissors is being used



a cell phone and charger cord



a clock tower rises high high high at dusk

```
#test_image_captions = {}
test_image_features = {key: images_segmented[key] for key in test_image_captions}
#image_lookup_dict[image_id] = name
test_image_features.keys()
```

```
dict_keys(['000000113132.jpg', '000000224056.jpg', '000000452783.jpg', '000000208815.jpg
```

```
test_image_captions['000000113132.jpg']
```

```
{'<start> A computer screen sits on next to pencils and a teddy bear <end>',
 '<start> A laptop computer sitting on a desk with a mini teddy bear and a container of p
 '<start> An Apple laptop is sitting with its desktop open near a small teddy bear and wi
 '<start> An open laptop computer sitting on a desk <end>',
 '<start> The laptop has a striped background on the desktop <end>'}
```

Custom Loss Function

Here, our aim is to more accurately predict specifically keywords in a sentence to ensure that the caption is relevant to the picture itself regardless of the organization of other words (ie. verbs, adjectives, etc).

```
import nltk
nltk.download('wordnet')
```

```
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Unzipping corpora/wordnet.zip.
True
```

```
from nltk.corpus import wordnet as wn
nouns = {x.name().split('.', 1)[0] for x in wn.all_synsets('n')}
#remove some nouns that are more commonly used as prepositions
nouns.remove("a")
nouns.remove("at")
nouns.remove("are")
nouns.remove("while")
```

```
nouns
```

```
 'genus_eucalyptus',
 'bb',
 'tundra',
 'sunroof',
 'neckerchief',
 'lateran',
 'rite',
 'wave_number',
 'marsilea',
 'heat',
 'triostium',
 'genus_kalmia',
 'search',
 'girdle',
 'lateralization',
 'oligoporus_leucospongia',
 'neurotransmitter',
 'tammany_hall',
 'cherokee',
 'commando',
 'net_melon',
 'amazon',
 'bluegill',
 'dipsomania',
 'nuclear_chemist',
```

```
      'self-hypnosis',
      'qassam_brigades',
      'displacement',
      'blastopore',
      'bedroom_community',
      'point_after',
      'spicule',
      'fourth_estate',
      'romanticist',
      'armet',
      'close-order_drill',
      'tael',
      'raw_wood',
      'halobacteria',
      'chabad',
      'off-season',
      'gelding',
      'sea_bream',
      'rodeo',
      'quartz',
      'furnishing',
      'snellen_chart',
      'count_palatine',
      'posthypnotic_suggestion',
      'wanton',
      'gingiva',
      'pyralidae',
      'country_doctor',
      'man-made_fiber',
      'traveling_salesman',
      'labiodental_consonant',
      'turdus',
      'reorder',
      'llud',
      'smooth_dogfish',

print(f"Example Caption to observe extracted key words (nouns): \n{all_captions[70]}\n")
word_list = all_captions[70].split()
for word in word_list:
  result = word in nouns
  print(f"{word} : {result}")

    Example Caption to observe extracted key words (nouns):
    <start> A man plays the violin in a home <end>

    <start> : False
    A : False
    man : True
    plays : False
    the : False
    violin : True
    in : False
    a : False
    home : True
    <end> : False


embedded_nouns = {}
dont_include = ["start","end"]
for key, value in tokenizer.index_word.items():
  if value in nouns and value not in dont_include:
```

```
embedded_nouns[key] = value
print(f"{value} is a noun. added key: {key}")
```

```
greeting is a noun. added key: 2462
fairy is a noun. added key: 2466
exit is a noun. added key: 2467
enclosure is a noun. added key: 2468
give is a noun. added key: 2469
boardwalk is a noun. added key: 2470
bib is a noun. added key: 2471
violin is a noun. added key: 2472
pouch is a noun. added key: 2476
shrimp is a noun. added key: 2481
environment is a noun. added key: 2488
tell is a noun. added key: 2490
official is a noun. added key: 2500
regular is a noun. added key: 2502
burner is a noun. added key: 2506
establishment is a noun. added key: 2507
factory is a noun. added key: 2512
mass is a noun. added key: 2513
hour is a noun. added key: 2514
festival is a noun. added key: 2515
california is a noun. added key: 2516
rainbow is a noun. added key: 2517
pug is a noun. added key: 2518
cutter is a noun. added key: 2522
pond is a noun. added key: 2523
trip is a noun. added key: 2525
direction is a noun. added key: 2527
medium is a noun. added key: 2529
leopard is a noun. added key: 2534
ham is a noun. added key: 2536
mustard is a noun. added key: 2537
eclectic is a noun. added key: 2538
brand is a noun. added key: 2539
indian is a noun. added key: 2540
nail is a noun. added key: 2541
hairbrush is a noun. added key: 2542
must is a noun. added key: 2543
laugh is a noun. added key: 2544
banquet is a noun. added key: 2553
smoke is a noun. added key: 2558
tone is a noun. added key: 2560
bloom is a noun. added key: 2561
campus is a noun. added key: 2562
doily is a noun. added key: 2564
jewelry is a noun. added key: 2566
golf is a noun. added key: 2567
crying is a noun. added key: 2569
noon is a noun. added key: 2570
thames is a noun. added key: 2572
tunnel is a noun. added key: 2573
daisy is a noun. added key: 2574
hammer is a noun. added key: 2577
seal is a noun. added key: 2579
border is a noun. added key: 2580
blank is a noun. added key: 2582
vent is a noun. added key: 2584
breath is a noun. added key: 2585
charger is a noun. added key: 2588
register is a noun. added key: 2589
```

```
loss_object = tf.keras.losses.CategoricalCrossentropy(
    from_logits=False, reduction='none')

def keras_custom_loss_function(real, pred):

  loss = loss_object(real, pred)

  return loss


# generator function to generate inputs for model
def custom_loss_helper(captions, images, tokenizer, max_caption_length, vocab_len, photos_per_

    X1, X2, y = list(), list(), list()
    n=0

    # loop through every image
    while 1:
        for key, cap in captions.items():
            n+=1
            # retrieve the photo feature
            image = images[key]

            for c in cap:
                # encode the sequence
                sequence = [tokenizer.word_index[word] for word in c.split(' ') if word in lis

                # split one sequence into multiple X, y pairs

                for i in range(1, len(sequence)):
                    # creating input, output
                    inp, out = sequence[:i], sequence[i]
                    # padding input
                    input_seq = pad_sequences([inp], maxlen=max_caption_length)[0]
                    if sequence[i] in embedded_nouns:
                      noun_to_check = sequence[i]
                      # encode output sequence
                      output_seq = to_categorical([out], num_classes=vocab_len)[0]
                      output_seq[(sequence[i])] = 1.1
                    else:
                      output_seq = to_categorical([out], num_classes=vocab_len)[0]
                    # store
                    X1.append(image)
                    X2.append(input_seq)
                    y.append(output_seq)

            # yield the batch data
            if n==photos_per_batch:
                yield ([np.array(X1), np.array(X2)], np.array(y))
                X1, X2, y = list(), list(), list()
                n=0


train_image_features = {key: images_segmented[key] for key in train_image_captions}
# get training data
```

```
# get training data
train_data = custom_loss_helper(train_image_captions, train_image_features, tokenizer, max_cap


'''sample_pred = np.zeros((2590,))
sample_pred[1] = 1.0
sample_pred[3] = 0.0
sample_pred[4] = 0.0
sample_pred[5] = 0.0
sample_pred[6] = 0.0
sample_pred[7] = 0.0

keras_custom_loss_function(next(train_data), sample_pred)
'''
```

```
'sample_pred = np.zeros((2590,))\nsample_pred[1] = 1.0\nsample_pred[3] = 0.0\nsample_pre
d[4] = 0.0\nsample_pred[5] = 0.0\nsample_pred[6] = 0.0\nsample_pred[7] = 0.0\n\nkeras_cu
stom_loss_function(next(train_data), sample_pred)\n'
```

```
# initialize model
model = create_model(max_caption_len, vocab_len)

logdir = "/content/drive/My Drive/ITCS-6156-Project/data/logs/scalars/" + datetime.now().strft
tensorboard_callback = keras.callbacks.TensorBoard(log_dir=logdir)

# adding embeddings to model
model.layers[2]
model.layers[2].set_weights([embedding_matrix])
model.layers[2].trainable = False

steps_per_epochs = len(train_image_captions)//32

# compile model
model.compile(optimizer='adam', loss=keras_custom_loss_function, metrics='accuracy')
model.fit_generator(train_data, epochs=30, steps_per_epoch=steps_per_epochs, callbacks=tensorb
```

```
WARNING:tensorflow:Layer lstm_4 will not use cuDNN kernel since it doesn't meet the cuDNN
WARNING:tensorflow:Layer lstm_5 will not use cuDNN kernel since it doesn't meet the cuDNN
WARNING:tensorflow:Layer lstm_6 will not use cuDNN kernel since it doesn't meet the cuDNN
WARNING:tensorflow:Layer lstm_7 will not use cuDNN kernel since it doesn't meet the cuDNN
Model: "functional_3"
```

Layer (type)	Output Shape	Param #	Connected to
input_4 (InputLayer)	[(None, 51)]	0	
input_3 (InputLayer)	[(None, 18432)]	0	
embedding_1 (Embedding)	(None, 51, 300)	777000	input_4[0][0]
dropout_2 (Dropout)	(None, 18432)	0	input_3[0][0]
dropout_3 (Dropout)	(None, 51, 300)	0	embedding_1[0][0]
dense_7 (Dense)	(None, 204)	3760332	dropout_2[0][0]
lstm_4 (LSTM)	(None, 51, 204)	412080	dropout_3[0][0]
dense_8 (Dense)	(None, 204)	41820	dense_7[0][0]

```
lstm_5 (LSTM)                   (None, 51, 204)      333744      lstm_4[0][0]
__________________________________________________________________________________________
dense_9 (Dense)                 (None, 204)          41820       dense_8[0][0]
__________________________________________________________________________________________
lstm_6 (LSTM)                   (None, 51, 204)      333744      lstm_5[0][0]
__________________________________________________________________________________________
dense_10 (Dense)                (None, 204)          41820       dense_9[0][0]
__________________________________________________________________________________________
lstm_7 (LSTM)                   (None, 204)          333744      lstm_6[0][0]
__________________________________________________________________________________________
add_1 (Add)                     (None, 204)          0           dense_10[0][0]
                                                                 lstm_7[0][0]
__________________________________________________________________________________________
dense_11 (Dense)                (None, 256)          52480       add_1[0][0]
__________________________________________________________________________________________
dense_12 (Dense)                (None, 256)          65792       dense_11[0][0]
__________________________________________________________________________________________
dense_13 (Dense)                (None, 2590)         665630      dense_12[0][0]
==========================================================================================
Total params: 6,860,006
Trainable params: 6,860,006
Non-trainable params: 0
__________________________________________________________________________________________
Epoch 1/30
375/375 [==============================] - 628s 2s/step - loss: 4.6150 - accuracy: 0.255
Epoch 2/30
375/375 [==============================] - 620s 2s/step - loss: 12.0887 - accuracy: 0.23
Epoch 3/30
375/375 [==============================] - 618s 2s/step - loss: 5.9719 - accuracy: 0.219
Epoch 4/30
375/375 [==============================] - 624s 2s/step - loss: 126.9714 - accuracy: 0.1
Epoch 5/30
375/375 [==============================] - 626s 2s/step - loss: 7.3377 - accuracy: 0.087
Epoch 6/30
375/375 [==============================] - 623s 2s/step - loss: 6.8165 - accuracy: 0.087
```

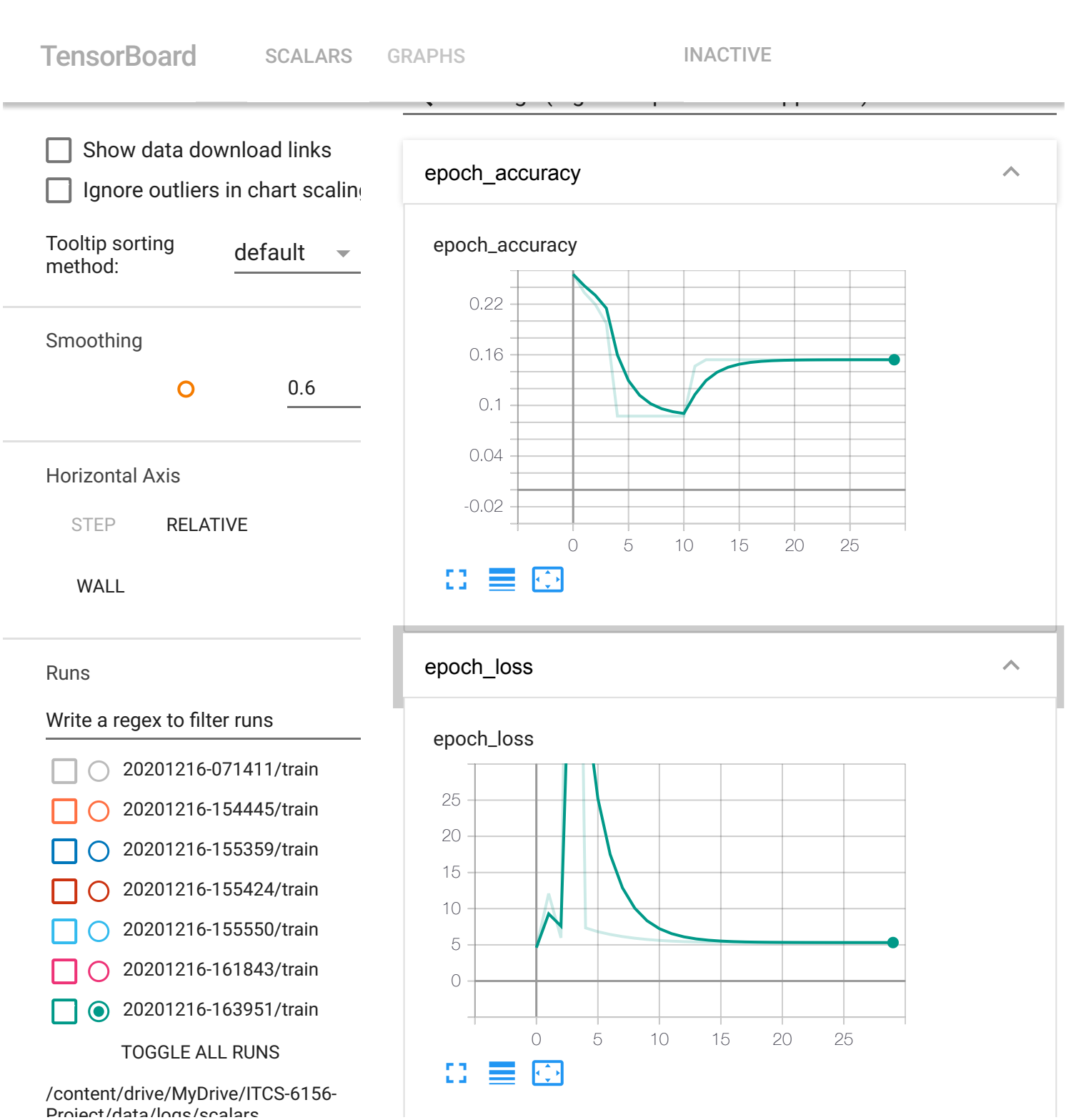
Save Neural Network

This model took roughly 6 hours to train. Saving the trained model will allow us to open it at a later time rather than having to retrain every time. ***WARNING:*** Run this cell ONLY if you are updating the saved file.

```
# serialize model to JSON
model_json = model.to_json()
with open("/content/drive/My Drive/ITCS-6156-Project/data/model2.json", "w") as json_file:
    json_file.write(model_json)
# serialize weights to HDF5
model.save_weights("/content/drive/My Drive/ITCS-6156-Project/data/model2.h5")
print("Saved model to disk")
```

```
Saved model to disk
```

```
%tensorboard --logdir=/content/drive/MyDrive/ITCS-6156-Project/data/logs/scalars
```



Testing New Neural Network

```
# method for generating captions
def generate_captions(model, image, word_index, max_caption_length, index_word):

    # input is <start>
    input_text = '<start>'

    # keep generating words till we have encountered <end>
    for i in range(max_caption_length):
        seq = [word_index[w] for w in input_text.split() if w in list(word_index.keys())]
```

```
        seq = pad_sequences([seq], maxlen=max_caption_length)
        prediction = model.predict([image,seq], verbose=0)
        print(prediction)
        print(prediction.shape)
        prediction = np.argmax(prediction)
        print(prediction)
        word = index_word[prediction]
        input_text += ' ' + word
        if word == 'end':
            break

    # remove <start> and <end> from output and return string
    output = input_text.split()
    output = output[1:-1]
    output = ' '.join(output)
    return output

# traverse through testing images to generate captions
test_image_features = {key: images_segmented[key] for key in test_image_captions}
image_path = IMG_LOCATION
count = 0

for key, value in test_image_features.items():
    test_image = test_image_features[key]
    test_image = np.expand_dims(test_image, axis=0)
    final_caption = generate_captions(model, test_image, tokenizer.word_index, max_caption_len

    plt.figure(figsize=(7,7))
    image = Image.open(image_path + "/" + key)
    plt.imshow(image)
    plt.title(final_caption)
    '''print("Picture: " + key)
    for value in test_image_captions[key]:
      print(value)
    print("----------------------")
    '''

    count = count + 1
    if count == 5:
        break
```

```
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
```

```
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
```

```
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
```

```
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
```

```
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
```

```
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
```

```
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
```

```
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
```

```
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
```

```
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
```

```
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
```

```
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
```

```
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
```

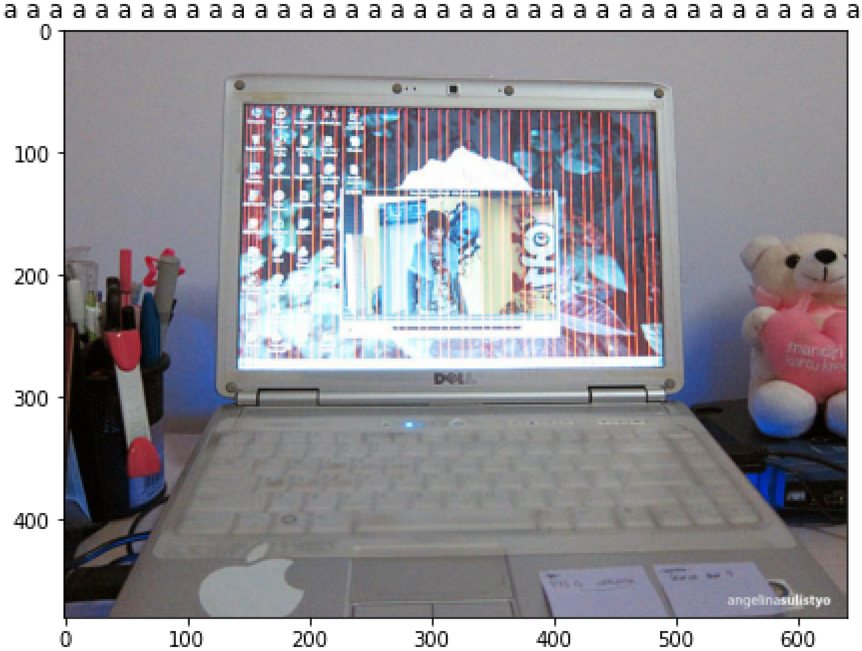
```
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
```

```
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
```

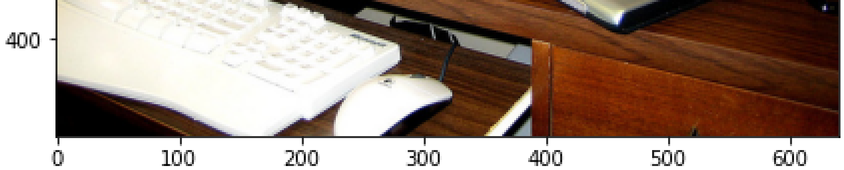
```
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
```

```
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
[[1.6628858e-06 1.5299521e-01 8.6731307e-02 ... 1.4509987e-05
  2.2088616e-05 2.2103639e-05]]
(1, 2590)
1
```

a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a



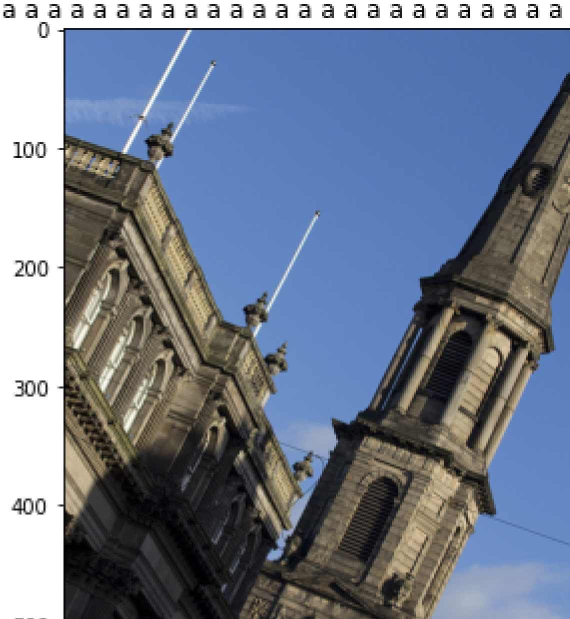
a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a

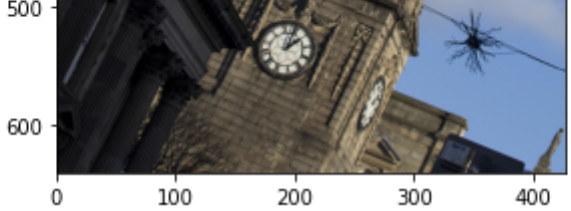


a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a

a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a

a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a




500
600
0
100
200
300
400